«به نام خالق آرامش»

نام کتاب: اصول امنیت شبکه های کامپیوتری (بخش دوم)

نام نویسنده: ویلیام استالینگز

نام مترجم: مسعود موحد

تعداد صفحات: ۲۴۰ صفحه

تاریخ انتشار: ــــــــــــــــ





I → R: CKY_I, OK_KEYX, GRP, g^X, EHAO, NIDP, ID_I, ID_R, N_I, $S_{KI}[ID_I \| ID_R \| N_I \| GRP \| g^X \| EHAO]$

R → I: CKY_R, CKY_I, OK_KEYX, GRP, g^Y, EHAS, NIDP, ID_R, ID_I, N_R, N_I, $S_{KR}[ID_R \| ID_I \| N_R \| N_I \| GRP \| g^Y \| g^X \| EHAS]$

I → R: CKY_I, CKY_R, OK_KEYX, GRP, g^X, EHAS, NIDP, ID_I, ID_R, N_I, N_R, $S_{KI}[ID_I \| ID_R \| N_I \| N_R \| GRP \| g^X \| g^Y \| EHAS]$

Notation:

I	=	Initiator
R	=	Responder
CKYI, CKYR	=	Initiator, responder cookies
OK_KEYX	=	Key exchange message type
GRP	=	Name of Diffie-Hellman group for this exchange
g^X, g^Y	=	Public key of initiator, responder; g^{XY} = session key from this exchange
EHAO, EHAS	=	Encryption, hash, authentication functions, offered and selected
NIDP	=	Indicates encryption is not used for remainder of this message
ID_I, ID_R	=	Identifier for initiator, responder
N_I, N_R	=	Random nonce supplied by initiator, responder for this exchange
$S_{KI}[X]$, $S_{KR}[X]$	=	Indicates the signature over X using the private key (signing key) of initiator, responder

شکل ۱۱-۶ مثال مبادله کلید Aggressive Oakley

مثالی از مبادلهٔ Oakley

مشخصه‌های Oakley شامل مثال‌هائی از مبادلاتی است که تحت این پروتکل قابل اجرا هستند. برای اینکه درک بهتری از Oakley وجود داشته باشد، یکی از این مثال‌ها که در مشخصات، aggressive key exchange نامیده می‌شود را ارائه می‌کنیم. دلیل این نام این است که تنها سه پیام ردوبدل می‌شود.

شکل ۱۱-۶ پروتکل aggressive key exchange را نشان می‌دهد. در قدم اول آغازگر (I) یک cookie، گروهی که بکار خواهد رفت، و کلید عمومی Diffie-Hellman آغازگر I برای این تبادل را ارسال می‌کند. I همچنین روش رمزنگاری کلید- عمومی، تابع درهم‌ساز و الگوریتم اعتبارسنجی پیشنهاد شده که در این مبادله بکار خواهد رفت را مشخص می‌سازد. در این پیام همچنین شناسه‌های I و R (پاسخ دهنده) و nonce مربوط به I وجود دارد. بالاخره I یک امضاء که با استفاده از کلید خصوصی I روی دو شناسه، nonce، گروه، کلید عمومی Diffie-Hellman و الگوریتم‌های پیشنهادی انجام شده است را به انتهای پیام وصل می‌کند.

وقتی R پیام را دریافت می‌کند، R امضاء را با استفاده از کلید عمومی I تأئید می‌نماید. R تأئید خود را با پس فرستادن cookie متعلق به I، شناسه، nonce و همچنین گروه به I انجام می‌دهد. R همچنین در پیام خود یک cookie، کلید عمومی Diffie-Hellman خود، الگوریتم‌های انتخاب شده (که بایستی از میان الگوریتم‌های ارسالی انتخاب شده باشد)، شناسهٔ R و nonce خود را می‌گنجاند. بالاخره R یک امضاء که دو شناسه، دو nonce، گروه، دو کلید عمومی Diffie-Hellman و الگوریتم‌های انتخاب شده را با کلید خصوصی R امضاء کرده است به پیام وصل می‌کند.

وقتی I پیام دوم را دریافت می‌کند، I امضاء را با استفاده از کلید عمومی R باز می‌کند. انـدازه‌هـای nonce در پیـام اطمینان می‌دهند که این بازخوانی یک پیام کهنه نیست. برای کامل کردن این مبادله، I بایستی پیام دیگری را برای R فرستاده و دریافت کلید عمومی R را اعلام نماید.

ISAKMP

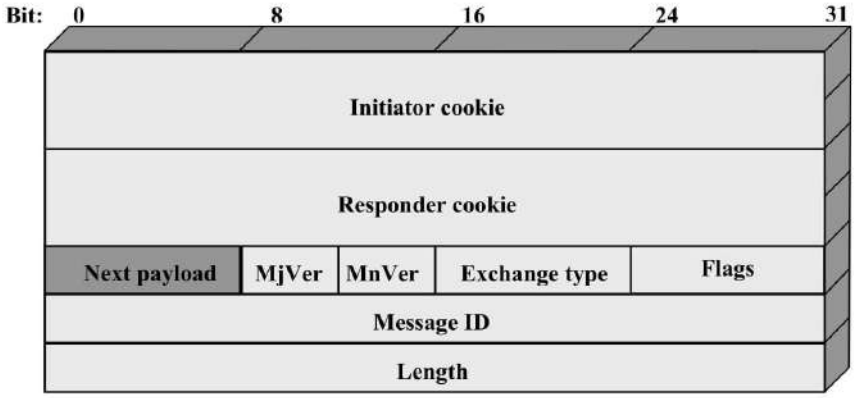
یـک پیـام ISAKMP (Internet Security Association and Key Management Protocol)، رویـه‌هـا و فرمـت بسته‌ها را برای برقراری، توافق، جرح و تعدیل و حذف اتحادهای امنیتی تعریف مـی‌کنـد. بعنـوان مرحلـه‌ای از برقـراری SA، ISAKMP محموله‌های مربوط به مبادلهٔ تولید کلید و داده‌های اعتبارسنجی را تعریف می‌کند. ایـن فرمـت محمولـه‌هـا یـک چهارچوب مستقل از پروتکل خاص مبادلهٔ کلید، الگوریتم رمزنگاری و مکانیسم اعتبارسنجی را فراهم می‌آورد.

فرمت سرآیند ISAKMP

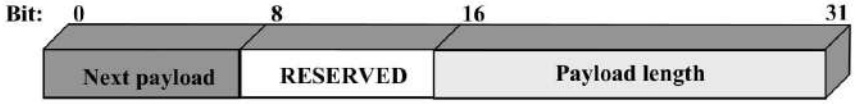
یک پیام ISAKMP شامل یک سرآیند ISAKMP است که بتوسط یک یا چند محموله دنبال می‌شود. تمـام اینهـا در یک پروتکل حمل‌ونقل، حمل می‌شوند. مشخصه، حمایت از UDP بعنوان پروتکل حمل‌ونقـل در پیـاده‌سـازی‌هـا را اجبـاری می‌داند.

شکل ۱۲-۶ فرمت سرآیند یک پیام ISAKMP را نشان می‌دهد. این سرآیند شامل میدان‌های زیر است:

- **Initiator Cookie (64 bits)**: cookie واحدی که برای برقراری SA، تعیین SA و یا حذف SA اقدام کرده است.
- **Responder Cookie (64 bits)**: cookie واحد پاسخ‌دهنده که در اولین پیام از طرف آغازگر، خـالی خواهـد بود.
- **Next Payload (8 bits)**: نشان‌دهندهٔ نوع اولین محموله در پیام است. محموله‌ها در بخش بعد تعریف خواهنـد شد.
- **Major Version (4 bits)**: نسخهٔ اصلی ISAKMP مورد استفاده را نشان می‌دهد.
- **Minor Version (4 bits)**: نسخهٔ فرعی ISAKMP مورد استفاده را نشان می‌دهد.
- **Exchange Type (8 bits)**: نوع مبادله را نشان می‌دهد. بعداً در همین بخش به آن اشاره خواهد شد.
- **Flags (8 bits)**: موارد مختص به این مبادلهٔ ISAKMP را نشان می‌دهد. تا کنون دو بیت از این میدان تعریف شده است. بیت Encryption که در صورتی 1 است که تمام محموله‌های بعد از سرآیند با الگـوریتم رمزنگـاری مرتبط به این SA رمزنگاری شده باشند. بیت Commit برای اطمینان از این است که مواد رمزنگاری شـده، قبـل از کامل شدن برقراری SA دریافت نشده باشند.
- **Message ID (32 bits)**: ID یکتای مختص این پیام.
- **Length (32 bits)**: طول کل پیام (سرآیند بعلاوهٔ تمام محموله‌ها) بر حسب اُکتت.



(الف) سرآیند ISAKMP



(ب) سرآیند عمومی محموله‌ها (payload)

شکل ۱۲-۶ فرمت ISAKMP

انواع محموله‌های ISAKMP

تمام محموله‌های ISAKMP با یک سرآیند عمومی که در شکل ۱۲-۶ب نشان داده شده است، شروع می‌شوند. میدان Next Payload، اگر این آخرین محمولهٔ پیام باشد دارای اندازهٔ 0 و در غیر اینصورت اندازهٔ نوع محمولهٔ بعد را نشان می‌دهد. میدان Payload Length نشان‌دهندهٔ طول این محموله بر حسب اُکتت بوده که شامل سرآیند عمومی محموله نیز می‌گردد.

جدول ۳-۶ انواع محموله‌های تعریف شده برای ISAKMP را نشان داده و میدان‌ها یا پارامترهائی که بخشی از هر محموله هستند را نیز مشخص می‌نماید. **SA payload** برای شروع استقرار SA بکار می‌رود. در این محموله، پارامتر Domain of Interpretation نمایش‌دهندهٔ DOI است که توافق تحت نظر آن صورت می‌پذیرد. IPSec DOI یک مثال آن است ولی ISAKMP می‌تواند در مقوله‌های دیگر نیز بکار رود. پارامتر Situation، خط‌مشی امنیتی این توافق را تعریف می‌کند. در واقع سطوح امنیتی لازم برای رمزنگاری و محرمانگی مشخص می‌شوند (مثلاً سطح حساسیت، بخش امنیتی).

Proposal payload شامل اطلاعاتی است که در خلال توافق SA بکار گرفته می‌شود. محموله، نمایش‌دهندهٔ پروتکل این SA (ESP یا AH) است که برای آن سرویس‌ها و مکانیسم‌ها مورد توافق قرار می‌گیرند. محموله همچنین شامل SPI واحد فرستنده و تعداد تبدیل‌هاست. هر تبدیل در یک محمولهٔ تبدیل قرار دارد. استفاده از محموله‌های با چند تبدیل، آغازگر را قادر می‌سازد تا حالات ممکن متعددی را پیشنهاد نماید که از بین آنها پاسخ‌دهنده بایستی یکی را انتخاب کرده و یا پاسخ منفی دهد.

Transform payload یک تبدیل امنیتی را تعریف می‌کند که از آن برای امن کردن کانال ارتباطی برای پروتکل مشخص شده استفاده می‌شود. پارامتر # Transform بمنظور شناسائی این محمولهٔ مخصوص بکار می‌رود تا پاسخ‌دهنده بتواند از آن برای موافقت با این تبدیل استفاده کند (مثلاً 3DES برای ESP، HMAC-SHA-1-96 برای AH) که ملحقات مربوطه نیز در آن وجود دارد (مثلاً طول hash).

Key Exchange payload می‌تواند برای تکنیک‌های متنوع مبادلهٔ کلید بکار رود که شامل Oakley، Diffie-Hellman و مبادلهٔ کلید RSA-based برای PGP است. میدان دیتای Key Exchange شامل دیتای مورد نیاز برای تولید یک کلید اجلاس بوده و مستقل از الگوریتم مبادلهٔ کلید بکار رفته است.

Identification payload برای تعیین هویت طرفین ارتباط بکار رفته و ممکن است برای تعیین اعتبار اطلاعات استفاده شود. معمولاً میدان ID Data شامل آدرس‌های IPv4 یا IPv6 است.

Certificate payload یک گواهی‌نامهٔ کلید- عمومی را منتقل می‌کند. میدان Certificate Encoding نمایش‌دهندهٔ نوع گواهی‌نامه و یا اطلاعات مربوط به گواهی‌نامه است که ممکن است شامل موارد زیر باشد:

- PKCS#7 wrapped X.509 certificate
- PGP certificate
- DNS signed key
- X.509 certificate-signature
- X.509 certificate-key exchange
- Kerberos tokens
- Certificate Revocation List (CRL)
- Authority Revocation List (ARL)
- SPKI certificate

در هر نقطه از مبادلهٔ ISAKMP، فرستنده ممکن است یک محمولهٔ **Certificate Request** برای درخواست گواهی‌نامهٔ واحد مرتبط ارسال کند. محموله ممکن است بیش از یک نوع گواهی‌نامهٔ قابل قبول و یا بیش از یک مسئول صدور گواهی قابل قبول را تعیین نماید.

Hash Payload شامل دیتای تولیدشده بتوسط تابع درهم‌ساز در بخشی از پیام و/ یا حالت ISAKMP است. این محموله برای تأئید صحت داده‌ها در یک پیام و یا برای احراز هویت واحدهای نظیر بکار رود.

جدول ۳-۶ انواع محموله‌های ISAKMP

Type	Parameters	Description
Security Association (SA)	Domain of Interpretation, Situation	Used to negotiate security attributes and indicate the DOI and Situation under which negotiation is taking place.
Proposal (P)	Proposal #, Protocol-ID, SPI Size, # of Transforms, SPI	Used during SA negotiation; indicates protocol to be used and number of transforms.
Transform (T)	Transform #, Transform-ID, SA Attributes	Used during SA negotiation; indicates transform and related SA attributes.
Key Exchange (KE)	Key Exchange Data	Supports a variety of key exchange techniques.
Identification (ID)	ID Type, ID Data	Used to exchange identification information.
Certificate (CERT)	Cert Encoding, Certificate Data	Used to transport certificates and other certificate- related information.
Certificate Request (CR)	# Cert Types, Certificate Types, # Cert Auths, Certificate Authorities	Used to request certificates; indicates the types of certificates requested and the acceptable certificate authorities.
Hash (HASH)	Hash Data	Contains data generated by a hash function.
Signature (SIG)	Signature Data	Contains data generated by a digital signa ture function.
Nonce (NONCE)	Nonce Data	Contains a nonce.
Notification (N)	DOI, Protocol-ID, SPI Size, Notify Message Type, SPI, Notification Data	Used to transmit notification data, such as an error condition.
Delete (D)	DOI, Protocol-ID, SPI Size, # of SPIs, SPI (one or more)	Indicates an SA that is no longer valid.

Signature payload شامل دیتای تولیدشده بتوسط یک تابع امضاء دیجیتال روی بخشی از پیام و/ یا حالت ISAKMP است. این محموله برای تأئید صحت دیتا در یک پیام بکار رفته و ممکن است برای سرویس‌های عدم انکار نیز مورد استفاده قرار گیرد.

Nonce payload شامل یک سری داده‌های تصادفی است که از آنها برای بهنگام بودن و جلوگیری از حملات بازخوانی استفاده می‌شود.

Notification payload شامل اطلاعات خط و یا حالت مرتبط با این SA و یا توافقات این SA است. پیام‌های خطا در ISKAMP شامل موارد تعریف شدهٔ زیراند:

Invalid Payload Type	Invalid Protocol ID	Invalid Cert Encoding
DOI Not Supported	Invalid SPI	Invalid Certificate
Situation Not Supported	Invalid Transform ID	Bad Cert Request Syntax
Invalid Cookie	Attributes Not Supported	Invalid Cert Authority
Invalid Major Version	No Proposal Chosen	Invalid Hash Information
Invalid Minor Version	Bad Proposal Syntax	Authentication Failed
Invalid Exchange Type	Payload Malformed	Invalid Signature
Invalid Flags	Invalid Key Information	Address Notification
Invalid Message ID		

تنها پیام حالت که تاکنون تعریف شده است، Connected است. علاوه براین یادآوری‌های ISAKMP، یادآوری‌های مختص به DOI نیز مورد استفاده قرار می‌گیرند. برای IPSec پیام‌های حالت اضافی زیر تعریف شده‌اند:

- **Responder-Lifetime**: زمان حیات SA که بتوسط پاسخ‌دهنده انتخاب شده است را نشان می‌دهد.
- **Replay-Status**: برای پاسخ مثبت پاسخ‌دهنده به این سؤال که آیا او عملیات تشخیص anti-replay را انجام خواهد داد یا نه مورد استفاده است.
- **Initial-Contact**: طرف دیگر را از اینکه آیا این اولین SA برقرار شده با سیستم دور است مطلع می‌سازد. گیرندهٔ این یادآوری آنگاه بایستی هر SA ای که برای سیستم فرستنده دارد را، با فرض اینکه سیستم فرستنده reboot کرده و دیگر دسترسی به این SAها ندارد، حذف نماید.

Delete payload یک یا چند SA که فرستنده از پایگاه دادهٔ خود حذف کرده و دیگر معتبر نیستند را نشان می‌دهد.

جدول ۴-۶ انواع مبادله‌های ISAKMP

مبادله	توضیح
(a) Base Exchange	
(1) **I → R:** SA; NONCE	Begin ISAKMP-SA negotiation
(2) **R → I:** SA; NONCE	Basic SA agreed upon
(3) **I → R:** KE; ID_I; AUTH	Key generated; Initiator identity verified by responder
(4) **R → I:** KE; ID_R; AUTH	Responder identity verified by initiator; Key generated; SA established
(b) Identity Protection Exchange	
(1) **I → R:** SA	Begin ISAKMP-SA negotiation
(2) **R → I:** SA	Basic SA agreed upon
(3) **I → R:** KE; NONCE	Key generated
(4) **R → I:** KE; NONCE	Key generated
(5)* **I → R:** ID_I; AUTH	Initiator identity verified by responder
(6)* **R → I:** ID_R; AUTH	Responder identity verified by initiator; SA established
(c) Authentication Only Exchange	
(1) **I → R:** SA; NONCE	Begin ISAKMP-SA negotiation
(2) **R → I:** SA; NONCE; ID_R; AUTH	Basic SA agreed upon; Responder identity verified by initiator
(3) **I → R:** ID_I; AUTH	Initiator identity verified by responder; SA established
(d) Aggressive Exchange	
(1) **I → R:** SA; KE; NONCE; ID_I	Begin ISAKMP-SA negotiation and key Exchange
(2) **R → I:** SA; KE; NONCE; ID_R; AUTH	Initiator identity verified by responder; Key generated; Basic SA agreed upon
(3)* **I → R:** AUTH	Responder identity verified by initiator; SA established
(e) Informational Exchange	
(1)* **I → R:** N/D	Error or status notification, or deletion

علائم اختصاری: I = آغازگر(Initiator) R = پاسخ‌دهنده (Responder)
* = رمزنگاری محموله بعد از سرآیند ISAKMP واقع می‌شود
AUTH = از مکانیسم اعتبارسنجی استفاده شده است.

مبادله‌های ISAKMP

ISAKMP یک چهارچوب برای مبادلهٔ پیام را فراهم می‌سازد که انواع محموله‌ها، عوامل تشکیل‌دهندهٔ آنها هستند. مشخصه پنج نوع مبادلهٔ پیش‌فرض را که بایستی حمایت گردند تعیین کرده است که در جدول ۴-۶ خلاصه شده‌اند. در این جدول، SA به یک محمولهٔ SA با محموله‌های نظیر Protocol و Transform اشاره می‌نماید.

Base Exchange اجازه می‌دهد تا مبادلهٔ کلید و مواد اعتبارسنجی با هم انتقال یابند. این امر تعداد تبادل‌ها را به حداقل می‌رساند ولی البته هویت‌ها مورد حفاظت قرار نمی‌گیرند. اولین دو پیام، cookieها را تولید کرده و یک SA با پروتکل و تبدیل‌های توافق شده را فراهم می‌آورد. هر دو طرف از یک nonce برای اطمینان‌بخشی در برابر حملات بازخوانی استفاده می‌کنند. آخرین دو پیام، مواد کلید و ID کاربران را مبادله کرده و از یک مکانیسم اعتبارسنجی برای تأئیدکلیدها، هویت‌ها و nonceهای دو پیام اول استفاده می‌کند.

Identity Protection Exchange برای محافظت از هویت کاربران، Base Exchange را بسط می‌دهد. دو پیام اول، SA را مستقر می‌کنند. دو پیام بعدی مبادلهٔ کلید را انجام می‌دهند و از nonceها برای محافظت جواب استفاده می‌شود. بمحض اینکه کلید اجلاس محاسبه گردید، دو طرف ارتباط به مبادلهٔ پیام‌های رمزنگاری شده که شامل اطلاعات اعتبارسنجی، همانند امضاءهای دیجیتال و احیاناً گواهی‌نامه‌های تأئیدکنندهٔ کلیدهای عمومی، است اقدام می‌کنند.

Authentication Only Exchange برای انجام اعتبارسنجی دوطرفه، بدون یک مبادلهٔ کلید بکار می‌رود. دو پیام اول، SA را مستقر می‌کنند. علاوه براین پاسخ‌دهنده از پیام دوم برای رساندن ID خود استفاده کرده و اعتبارسنجی را برای حفاظت پیام بکار می‌برد. آغازکننده، پیام سوم را ارسال نموده تا ID اعتبارسنجی شده را منتقل کند.

Aggressive Exchange تعداد مبادلات را به قیمت عدم حفاظت از هویت‌ها می‌نیمم می‌کند. در اولین پیام، آغازگر، یک SA با پروتکل پیشنهادی و تبدیل‌های ممکن را ارسال می‌کند. شروع‌کننده همچنین یک مبادلهٔ کلید را آغاز کرده و ID آن را فراهم می‌سازد. در پیام دوم، پاسخ‌دهنده، پذیرش این SA را با یک پروتکل و تبدیل بخصوص نشان‌داده، مبادلهٔ کلید را کامل ساخته و اطلاعات انتقال یافته را اعتبارسنجی می‌نماید. در پیام سوم، آغازکننده نتیجهٔ اعتبارسنجی بر روی اطلاعات قبلی، که با استفاده از کلید سّری باشتراک گذاشته شده رمزنگاری شده است را می‌فرستد.

از **Information Exchange** برای انتقال یکطرفهٔ اطلاعات برای مدیریت SA استفاده می‌شود.

۶-۷ منابع مطالعاتی

IPv6 و IPv4 بطور مفصل‌تری در [STAL04] پوشش داده شده‌اند. [CHEN98] بحث مفیدی در مورد طراحی IPSec دارد. [FRAN01] و [DORA03] پوشش تفصیلی‌تری از IPSec دارند.

CHEN98 Cheng, P., et al. "A Security Architecture for the Internet Protocol." *IBM Systems Journal*, Number 1, 1998.
DORA03 Doraswamy, N., and Harkins, D.*IPSec*.Upper Saddle River, NJ: Prentice Hall, 2003.
FRAN01 Frankel, S. *Demystifying the IPSec Puzzle*. Boston: Artech House, 2001.
STAL04 Stallings, W. *Computer Networking with Internet Protocols and Technology*. Upper Saddle River, NJ: Prentice Hall, 2004.

وب سایت‌های مفید

- **NIST IPSEC Project**: شامل مقالات، ارائهٔ مطالب و پیاده‌سازی‌های مرجع است.

۶-۸ واژه‌های کلیدی، سؤالات مرورکنندهٔ بحث و مسائل

واژه‌های کلیدی

نسخهٔ چهارم IP	IPv4
نسخهٔ ششم IP	IPv6
پروتکل تعیین کلید اُکلی	Oakley key determination protocol
حملهٔ بازخوانی	replay attack
اتحاد امنیتی	security association (SA)
مُود حمل‌ونقل	transport mode
مُود تونل	tunnel mode
سرویس ضد- بازخوانی	anti-replay service
سرآیند اعتبارسنجی	authentication header (AH)
کپسولی کردن محمولهٔ امنیتی	encapsulating security payload (ESP)
پروتکل اتحاد امنیتی و مدیریت کلید اینترنت	Internet Security Association and Key Management Protocol (ISAKMP)
امنیت IP	IP Security (IPSec)

سؤالات مرورکننده بحث

۶-۱ مثال‌هائی از کاربرد IPSec را بیان کنید.

۶-۲ چه سرویس‌هائی بتوسط IPSec فراهم می‌گردند؟

۶-۳ چه پارامترهائی یک SA را معرفی و چه پارامترهائی ماهیت یک SA بخصوص را تعیین می‌کنند؟

۶-۴ تفاوت بین مُود حمل‌ونقل و مُود تونل چیست؟

۶-۵ حملهٔ بازخوانی کدام است؟

۶-۶ چرا ESP دارای یک میدان padding است؟

۶-۷ روش‌های اصلی ترکیب SAها کدامند؟

۶-۸ نقش پروتکل‌های تعیین کلید Oakley و ISAKMP در IPSec چیست؟

مسائل

۶-۱ در بحث پردازش AH، خاطرنشان شده بود که تمام میدان‌های سرآیند IP در محاسبات MAC وارد نمی‌شوند.

الف- برای هریک از میدان‌های سرآیند IPv4، نشان دهید که آیا آن میدان تغییرناپذیر، تغییرپذیر ولی قابل پیش‌بینی و یا تغییرپذیر (که قبل از محاسبات ICV باید صفر شوند) است.

ب- همین کار را برای IPv6 انجام دهید.

ج - همین کار را برای سرآیندهای الحاقی IPv6 انجام دهید.

در هر مورد دلیل خود برای هر میدان را توجیه کنید.

۶-۲ وقتی از مُود تونل استفاده می‌شود، یک سرآیند IP بیرونی ساخته می‌شود. برای هر دو نسخهٔ IPv4 و IPv6 رابطهٔ بین میدان‌های سرآیند IP بیرونی و هر سرآیند الحاقی در بستهٔ بیرونی را با میدان و یا سرآیند الحاقی بستهٔ درونی نشان دهید. یعنی نشان دهید که کدام اندازه‌های بیرونی از مقادیر درونی مشتق شده و کدام اندازه‌های بیرونی مستقل از مقادیر درونی ساخته می‌شوند.

۶-۳ رمزنگاری و اعتبارسنجی سر- به- سر بین دو میزبان کاری مطلوب است. شکل‌هائی شبیه به شکل‌های ۶-۶ و ۶-۹ کشیده که نشان دهد:

الف- مجاورت مُودهای حمل‌ونقل با رمزنگاری قبل از اعتبارسنجی.

ب- یک SA حمل‌ونقل در داخل یک SA تونل که در آن رمزنگاری قبل از اعتبارسنجی انجام شود.

ج - یک SA حمل‌ونقل در داخل یک SA تونل که در آن اعتبارسنجی قبل از رمزنگاری انجام شود.

۶-۴ اسناد معماری IPSec بیان می‌کنند که وقتی دو SA مُود حمل‌ونقل با هم ترکیب شده تا هم پروتکل AH و هم پروتکل ESP را روی یک جریان سر- به- سر ایجاد کنند تنها یک روش مناسب بنظر می‌رسد که آنهم اجرای پروتکل ESP قبل از اجرای AH است. چرا این روش پیشنهاد شده و اعتبارسنجی قبل از رمزنگاری پیشنهاد نگردیده است؟

۶-۵ **الف-** کدامیک از انواع مبادلات ISAKMP (جدول ۴-۶) نظیر مبادلهٔ کلید aggressive Oakley است(شکل ۱۱-۶)؟

ب- برای مبادلهٔ کلید aggressive Oakley نشان دهید که کدام پارامترها در هر پیام، در کدام نوع محمولهٔ ISAKMP حمل می‌شوند.

ضمیمهٔ ۶- الف عملیات بین‌شبکه‌ای و پروتکل‌های اینترنت

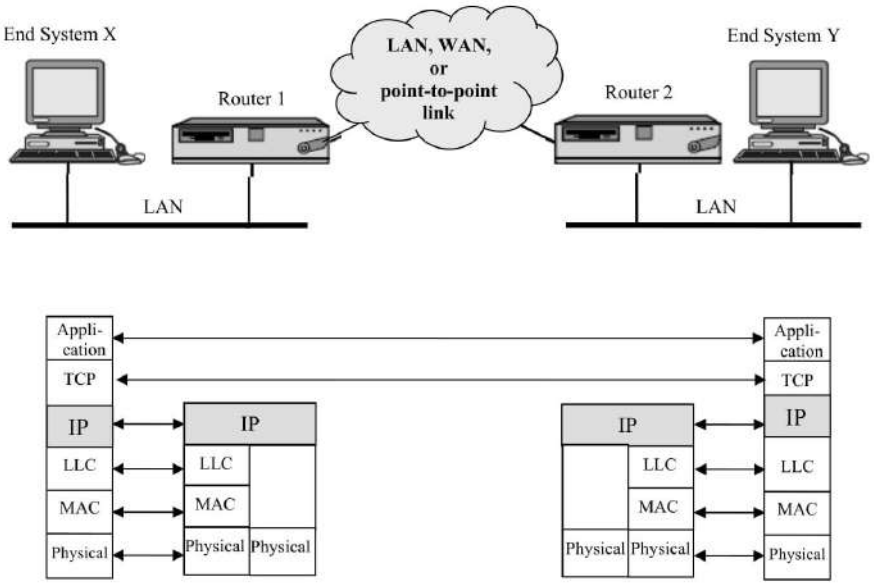
این ضمیمه، مروری بر پروتکل‌های بین‌شبکه‌ای دارد. بحث را با بیان خلاصهٔ نقش یک پروتکل بین‌شبکه‌ای در فراهم‌آوردن عملیات بین‌شبکه‌ای آغاز می‌کنیم. آنگاه دو پروتکل بین‌شبکه‌ای اصلی یعنی IPv4 و IPv6 را معرفی می‌نمائیم.

نقش یک پروتکل بین‌شبکه‌ای

یک پروتکل بین‌شبکه‌ای (IP)، نیازهای مربوط به اتصال دو سیستم انتهائی در عرض شبکه‌های متعدد را برآورده می‌سازد. برای این منظور، IP در هر سیستم انتهائی و همچنین در مسیریاب‌ها که دستگاه‌هائی برای ایجاد اتصال بین شبکه‌ها هستند پیاده‌سازی می‌شود. داده‌های لایهٔ بالاتر در یک سیستم انتهائی، برای انتقال، در یک واحد پروتکلی دیتای IP (IP PDU) کپسولی می‌شود. این PDU آنگاه از یک یا چند شبکه و همچنین مسیریاب‌های ارتباط‌دهنده عبور کرده تا به سیستم انتهائی مقصد برسد.

مسیریاب بایستی بتواند خود را با تنوعی که بین شبکه‌های مختلف وجود دارد وفق دهد. برخی از تفاوت‌های بین شبکه‌ها بقرار زیر است:

- **روش‌های آدرس‌دهی**: شبکه‌ها ممکن است روش‌های متفاوتی را برای تخصیص آدرس‌ها به دستگاه‌ها بکار گیرند. مثلاً یک IEEE802 LAN برای هر دستگاه شبکه یک آدرس ۱۶- بیتی و یا ۴۸- بیتی را بکار می‌برد. یک شبکهٔ سوئیچینگ بسته‌ای X.25 از آدرس‌های ۱۲ رقمی اعشاری (۴ بیت برای هر رقم و در جمع ۴۸ بیت) استفاده می‌کند. نوعی آدرس‌دهی عمومی بعلاوهٔ فهرستی از آدرس‌ها بایستی فراهم شود.
- **اندازهٔ ماکزیمم بسته‌ها**: بسته‌های یک شبکه ممکن است برای عبور از شبکه دیگر نیاز به قطعه‌قطعه شدن داشته باشند که این عمل را fragmentation گویند. بعنوان مثال Ethernet اندازهٔ ماکزیمم بسته‌ها را ۱,۵۰۰ بایت قرار داده است در حالی‌که اندازهٔ ماکزیمم بسته‌ها در شبکه‌های X.25 برابر ۱,۰۰۰ بایت است. یک بسته که روی سیستم Ethernet انتقال یافته و بمنظور عبور به یک شبکهٔ X.25 از یک مسیریاب عبور می‌کند، ممکن است نیاز به تبدیل به دو بستهٔ کوچک‌تر داشته باشد.
- **واسط‌ها**: واسط‌های (interfaces) سخت‌افزاری و نرم‌افزاری در شبکه‌های مختلف متفاوت‌اند. ماهیت عمل مسیریاب بایستی مستقل از چنین تفاوت‌هائی باشد.
- **قابلیت اعتماد**: سرویس‌های مختلف شبکه می‌توانند از یک مدار مجازی سر- به- سر تا یک سرویس غیرقابل اعتماد متغیر باشند. عمل مسیریاب‌ها بایستی مستقل از فرض مورد اعتماد بودن شبکه و یا خلاف آن باشد.



شکل ۱۳-۶ پیکربندی برای مثال TCP/IP

عمل مسیریاب همانطور که در شکل ۱۳-۶ نشان داده شده است، وابسته به یک پروتکل بین شبکه‌ای است. در این مثال، پروتکل اینترنت (IP) از مجموعهٔ پروتکلی TCP/IP این عمل را انجام می‌دهد. IP بایستی در تمام سیستم‌های انتهائی، روی تمام شبکه‌ها و همچنین در مسیریاب‌ها، تعبیه شود. علاوه برآن، هر سیستم انتهائی بایستی پروتکل‌های سازگاری در بالای IP داشته تا ارتباط بصورت موفق انجام پذیرد. مسیریاب‌های بین راه تنها کافی است که تا سطح IP را حمایت نمایند.

انتقال یک بلوک دیتا از سیستم انتهائی X به سیستم انتهائی Y در شکل ۱۳-۶ را در نظر بگیرید. لایهٔ IP در X بلوک‌های دیتا که بایستی برای Y ارسال شوند را از لایهٔ TCP سیستم X تحویل می‌گیرد. لایهٔ IP، یک سرآیند که آدرس جهانی Y را مشخص می‌کند به دیتا اضافه می‌نماید. این آدرس دارای دو قسمت شناسهٔ شبکه و شناسهٔ سیستم انتهائی است. اجازه دهید که این بلوک را یک بستهٔ IP بنامیم. در مرحلهٔ بعد IP درمی‌یابد که مقصد (Y) روی زیرشبکهٔ دیگری است. بنابراین اولین قدم این است که بسته را به یک مسیریاب، که در این مورد مسیریاب 1 است، ارسال کند. برای انجام این امر، IP واحد دیتای خود را با اطلاعات کامل آدرس‌دهی به لایهٔ LLC در قسمت پائین‌تر می‌دهد. LLC واحد دیتای LLC PDU را خلق کرده که در مرحلهٔ بعد به لایهٔ MAC داده می‌شود. لایهٔ MAC یک بستهٔ MAC که سرآیند آن شامل آدرس مسیریاب 1 است را می‌سازد.

سپس بسته از درون شبکهٔ LAN به مسیریاب 1 می‌رود. مسیریاب، سرآیندها و ته‌آیندهای بسته و LLC را کنده و سرآیند IP را بررسی می‌کند تا مقصد نهائی دیتا که در این مورد Y است را تعیین کند. مسیریاب در اینجا بایستی نسبت به مسیریابی تصمیم‌گیری نماید. دو امکان وجود دارد:

۱- سیستم انتهائی مقصد (Y) مستقیماً به یکی از زیرشبکه‌هائی متصل است که مسیریاب نیز در آنها قرار دارد.
۲- برای رسیدن به مقصد، بایستی از یک یا چند مسیریاب دیگر نیز عبور کرد.

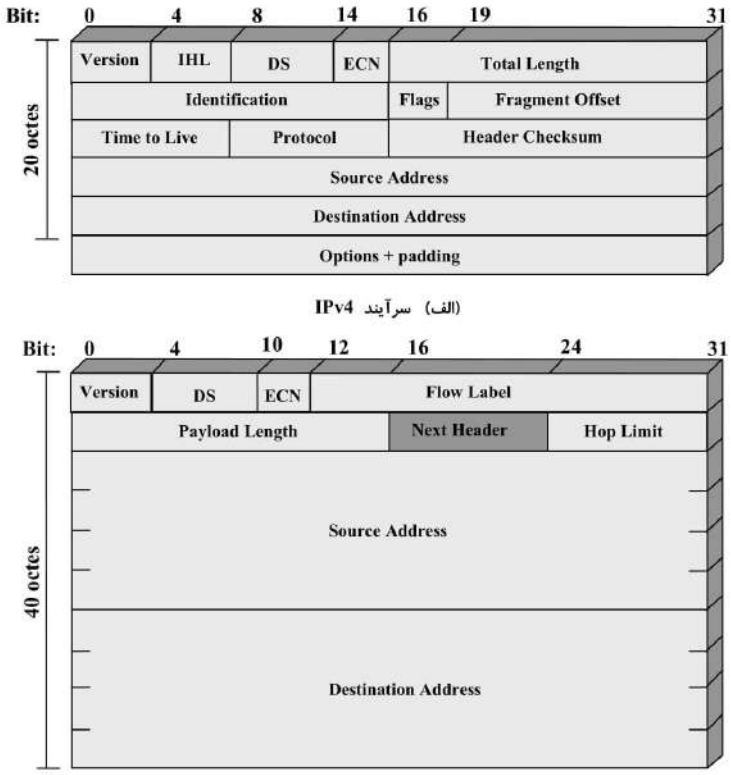
در این مثال، بسته بایستی قبل از رسیدن به مقصد از مسیریاب 2 عبور کند. بنابراین مسیریاب 1 بستهٔ IP را از طریق شبکهٔ میانی به مسیریاب 2 می‌فرستد. برای این مقصود، پروتکل‌های آن شبکه بکار گرفته می‌شوند. مثلاً اگر شبکهٔ میانی یک شبکهٔ X.25 است، واحد دیتای IP، به همراه اطلاعات آدرس‌دهی مرتبط برای رسیدن به مسیریاب 2 در یک بستهٔ X.25 پیچیده می‌شود. وقتی این بسته به مسیریاب 2 وارد می‌شود، سرآیند بسته کنده می‌شود. مسیریاب تعیین می‌کند که این بسته IP به مقصد Y است که مستقیماً روی زیرشبکه‌ای که مسیریاب به آن متصل است قرار دارد. در نتیجه مسیریاب یک بسته با آدرس مقصد Y را خلق کرده و آن را روی شبکهٔ LAN می‌فرستد. نهایتاً دیتا وارد Y میگردد که در آنجا سرآیندها و ته‌آیندهای بسته، LLC و اینترنت می‌توانند از آن جدا شوند.

سرویسی که بتوسط IP فراهم می‌شود یک سرویس غیرقابل اعتماد است. یعنی IP تضمین نمی‌کند که تمام دیتا به مقصد تحویل شده و یا اینکه دیتا با نظم اولیه وارد مقصد گردد. این وظیفهٔ یک لایهٔ بالاتر، در این مورد TCP، است که هر خطائی را که ممکن است واقع شود تصحیح نماید. این روش انعطاف‌پذیری زیادی را به ارمغان می‌آورد. چون تحویل بسته‌ها تضمین شده نیست، نیازی به اعتماد به زیرشبکه‌ها وجود ندارد. در نتیجه پروتکل با هر ترکیبی از انواع زیرشبکه‌ها کار می‌کند. چون تضمینی برای تحویل منظم بسته‌ها وجود ندارد، بسته‌های پشت سرهم می‌توانند مسیرهای متفاوتی را از درون اینترنت طی کنند. این امر به پروتکل اجازه می‌دهد تا در صورت مواجهه با تراکم و یا خرابی در شبکه، مسیر بستهٔ دیتا را عوض کند.

IPv4

برای دهه‌های متوالی سنگ‌بنای معماری پروتکل TCP/IP، پروتکل اینترنت (IP) نسخهٔ ۴ بوده است. شکل ۱۴- ۶الف فرمت سرآیند IP را نشان می‌دهد که حداقل دارای ۲۰ اُکتت و یا ۱۶۰ بیت است. میدان‌ها بقرار زیراند:

- **Version (4 bits)**: نشان‌دهندهٔ شمارهٔ نسخهٔ پروتکل است تا بعداً بتوان نسخ تکامل‌یافته‌تر را با شمارهٔ جدیدی نشان داد. در اینجا اندازهٔ آن ۴ است.
- **Internet Header Length (IHL) (4 bits)**: طول سرآیند بر حسب کلمات ۳۲- بیتی است. حداقل آن ۵ است که طول می‌نیمم سرآیند یعنی ۲۰ اُکتت را نشان می‌دهد.



(الف) سرآیند IPv4

(ب) سرآیند IPv6

DS = Differentiated services field
ECN = Explicit congestion notification field

توجه : میدان هشت بیتی DS/ECN قبلاً در سرآیند IPv4 به نام میدان Type of Service و درسرآیند IPv6 به نام Traffic Class خوانده می‌شد

شکل ۱۴-۶ سرآیندهای IP

- **DS/ECN (8 bits)**: قبل از معرفی سرویس‌های مشتق شده، این میدان بنام میدان **Type of Service** خوانده می‌شد و قابلیت اعتماد، اولویت، تأخیر و پارامترهای مربوط به توان عملیاتی را مشخص می‌نمود. این تعبیر اکنون کنار گذاشته شده است. اولین ۶ بیت میدان TOS اکنون با نام میدان DS (Differentiated Services) خوانده می‌شود. ۲ بیت باقیمانده برای میدان ECN (Explicit Congestion Notification) رزرو شده‌اند.
- **Total Length (16 bits)**: طول کلّ بستهٔ IP بر حسب اُکتت.
- **Identification (16 bits)**: یک شمارهٔ ردیف که به همراه آدرس منبع، آدرس مقصد و پروتکل کاربر یک بسته را بطور یکتا مشخص می‌سازد. بنابراین این عدد برای آدرس منبع بسته، آدرس مقصد بسته و پروتکل کاربر در خلال مدتی که بسته در اینترنت می‌ماند بایستی یکتا باشد.
- **Flags (3 bits)**: در حال حاضر تنها دو بیت آن تعریف شده است. وقتی بسته‌ای قطعه‌قطعه می‌گردد (fragmentation)، بیت More نشان می‌دهد که آیا این بسته آخرین قطعهٔ بستهٔ اولیه است. بیت Don't Fragment اگر set باشد به مفهوم این است که این بسته نبایستی قطعه‌قطعه گردد. این بیت زمانی ممکن است مفید واقع شود که بدانیم مقصد قابلیت دوباره سرهم‌کردن (reassembly) قطعه‌ها را نخواهد داشت. از طرفی وقتی این بیت set باشد، اگر اندازهٔ بسته از اندازهٔ ماکزیمم مجاز در زیرشبکه‌های مسیر بیشتر شود، بسته معدوم خواهد شد. بنابراین اگر این بیت set است، عاقلانه‌تر است که از مسیریابی منبع استفاده شود تا از عبور بسته از زیرشبکه‌هائی که اندازهٔ ماکزیمم بسته در آنها کوچک است اجتناب گردد.
- **Fragment Offset (13 bits)**: نشان می‌دهد که این قطعه به کجای بستهٔ اصلی تعلق دارد و اندازهٔ آن بر حسب واحدهای ۶۴- بیتی مشخص می‌گردد. این امر لازم می‌دارد که قطعه‌ها بجز قطعهٔ آخر بایستی دارای میدان دیتائی باشند که طول آن مضربی از ۶۴ بیت باشد.
- **Time to Live (8 bits)**: تعیین می‌کند که این بسته برای چه مدتی، بر حسب ثانیه، مجاز به ماندن در اینترنت است. هر مسیریابی که یک بسته را پردازش می‌کند بایستی TTL را حداقل 1 واحد کاهش دهد. بنابراین TTL تا حدودی شبیه شمارش‌گر پرش‌ها (hops) در مسیر است.
- **Protocol (8 bits)**: پروتکل لایهٔ بالاترکه بایستی میدان دیتا در مقصد را تحویل بگیرد نشان می‌دهد. بنابراین این میدان نوع سرآیند بعدی در بسته، بعد از سرآیند IP، را تعیین می‌کند.
- **Header Checksum (16 bits)**: یک کُد تشخیص خطاست که تنها به سرآیند اعمال می‌گردد. چون برخی از میدان‌های سرآیند ممکن است در زمان ترانزیت تغییر کنند (مثلاً Time to Live یا میدان‌های مربوط به سگمنت)، این کُد در هر مسیریاب کنترل و مجدداً محاسبه می‌گردد. میدان checksum یک جمع متمم یک ۱۶- بیتی از تمام کلمات ۱۶- بیتی سرآیند است. در محاسبات، میدان checksum در ابتدا برابر صفر قرار داده می‌شود.
- **Source Address (32 bits)**: بیت‌های کُد شده‌ای است که بمنظور تعیین یک شبکه و سیستم انتهائی متصل به آن بکار می‌رود (۷ و ۲۴ بیت، ۱۴ و ۱۶ بیت یا ۲۱ و ۸ بیت).
- **Destination Address (32 bits)**: همان مشخصه‌های آدرس منبع را داراست.

- **Options (variable)**: مقوله‌های اختیاری تقاضاشده بتوسط کاربر ارسال‌کننده را نشان می‌دهد. اینها می‌توانند شامل برچسب امنیتی، مسیریابی منبع، ثبت مسیریابی و برچسب زمانی باشند.
- **Padding (variable)**: برای تکمیل طول سرآیند بسته به مضربی از ۳۲ بیت بکار می‌رود.

IPv6

در سال ۱۹۹۵ میلادی، IETF (Internet Engineering Task Force) که استانداردهای پروتکلی اینترنت را فراهم می‌آورد، مشخصه‌ای برای IP نسل بعد را انتشار داد که در آن زمان به IPng معروف شد. این مشخصه در سال ۱۹۹۶ بصورت استاندارد درآمده و IPv6 نام گرفت. IPv6 نسبت به IP فعلی (IPv4)، تعدادی عملیات اضافی در بر دارد که بمنظور کارآئی بیشتر در شبکه‌های پرسرعت امروزی و اختلاط جریان‌های دیتا که شامل گرافیک و ویدئو بوده و پیوسته خواستاران بیشتری دارد طراحی شده است. ولی انگیزهٔ اصلی در فراهم آوردن پروتکل جدید، نیاز به آدرس‌های بیشتر بود. IPv4 از یک آدرس ۳۲- بیتی برای مشخص کردن منبع و مقصد استفاده می‌کند. با رشد انفجاری اینترنت و شبکه‌های خصوصی متصل به آن، این طول آدرس برای برآوردن نیازهای تمام سیستم‌های نیازمند به آدرس کافی نیست. همانطور که شکل ۱۴-۶ب نشان می‌دهد، IPv6 شامل آدرس‌های ۱۲۸- بیتی منبع و مقصد در میدان آدرس خود است. در نهایت تمام پیاده‌سازی‌های TCP/IP از IP کنونی به سمت IPv6 سوق داده خواهند شد که البته این امر ممکن است سال‌ها و بلکه ده‌ها سال بطول انجامد.

سرآیند IPv6

سرآیند IPv6 دارای طول ثابت ۴۰ اُکتت است که شامل میدان‌های زیر است (شکل ۱۴-۶ب):

- **Version (4 bits)**: شمارهٔ نسخهٔ پروتکل اینترنت. این اندازه برابر ۶ است.
- **DS/ECN (8 bits)**: قبل از معرفی سرویس‌های مشتق شده، این میدان بنام میدان **Traffic Class** خوانده می‌شد و برای استفادهٔ گره‌های آغازگر و/ یا مسیریاب‌های جلوبرنده بمنظور تشخیص و تمایز بین کلاس‌های مختلف و یا اولویت‌های مختلف بسته‌های IPv6 رزرو شده بود. اولین ۶ بیت میدان Traffic Class اکنون بنام میدان DS (Differentiated Services) خوانده می‌شود. دو بیت باقیمانده برای میدان ECN (Explicit Congestion Notification) رزرو شده‌اند.
- **Flow Label (20 bits)**: می‌تواند بتوسط یک میزبان برای تعیین آن بسته‌هائی بکار رودکه سرویس‌های خاصی را از مسیریاب‌های بین راه طلب می‌کنند. تعیین برچسب برای جریان ترافیک می‌تواند در رزرو کردن منبع و پردازش بلادرنگ ترافیک مؤثر باشد.
- **Payload Length (16 bits)**: اندازهٔ بقیهٔ بسته IPv6 بر حسب اُکتت که پس از سرآیند قرار می‌گیرد. بعبارت دیگر، این طول کل تمام سرآیندهای الحاقی باضافهٔ طول PDU سطح حمل‌ونقل است.
- **Next Header (8 bits)**: نوع سرآیندی که بلافاصله بعد از سرآیند IPv6 قرار می‌گیرد را تعیین می‌کند. این یا یک سرآیند الحاقی IPv6 و یا یک سرآیند لایهٔ بالاتر مانند TCP و یا UDP است.

- **Hop Limit (8 bits)**: تعداد باقیماندهٔ پرش‌های مجاز (hops) این بسته است. حد پرش بتوسط منبع به یک مقدار ماکزیمم دلخواه تنظیم شده و پس از عبور از هر گره که بسته را جلو می‌راند، یک واحد کم می‌شود. در صورتی که Hop Limit به صفر تقلیل یابد، این بسته معدوم خواهد شد.
- **Source Address (128 bits)**: آدرس منبع آغازگر این بسته است.
- **Destination Address (128 bits)**: آدرس گیرندهٔ بستهٔ مورد نظر است. اگر یک سرآیند الحاقی Routing وجود داشته باشد، این آدرس ممکن است آدرس مقصد نهائی نباشد.

اگرچه سرآیند IPv6 طولانی‌تر از بخش اجباری سرآیند IPv4 است (۴۰ اُکتت در برابر ۲۰ اُکتت)، ولی دارای میدان‌های کمتری است (۸ در برابر ۱۲). بنابراین مسیریاب‌ها برای هر سرآیند پردازش کمتری انجام می‌دهند که این امر مسیریابی را سرعت می‌بخشد.

سرآیندهای الحاقی IPv6

یک بستهٔ IPv6، شامل سرآیند IPv6 که هم اکنون تشریح گردید، بعلاوهٔ هیچ یا چند سرآیند الحاقی دیگر است. خارج از IPSec، سرآیندهای الحاقی زیر تعریف شده‌اند:

- **Hop-by-Hop Options Header**: موارد اختیاری بخصوصی را تعریف می‌کند که بتوسط پردازش hop-by-hop مورد نیاز است.
- **Routing Header**: مسیریابی گسترده‌تری همانند مسیریابی منبع در IPv4 را فراهم می‌آورد.
- **Fragment Header**: اطلاعات fragmentation و reassembly را شامل می‌شود.
- **Authentication Header**: مکانیسم اعتبارسنجی و سنجش صحت بسته را فراهم می‌کند.
- **Encapsulating Security Payload Header**: خصوصی ماندن بسته را تأمین می‌کند.
- **Destination Options Header**: اطلاعات اختیاری که بتوسط گرهٔ مقصد مورد مداقّه قرار می‌گیرد را فراهم می‌سازد.

استاندارد IPv6 توصیه می‌کند که وقتی سرآیندهای الحاقی متعددی بکار گرفته می‌شوند، سرآیندهای IPv6 دارای نظم زیر باشند:

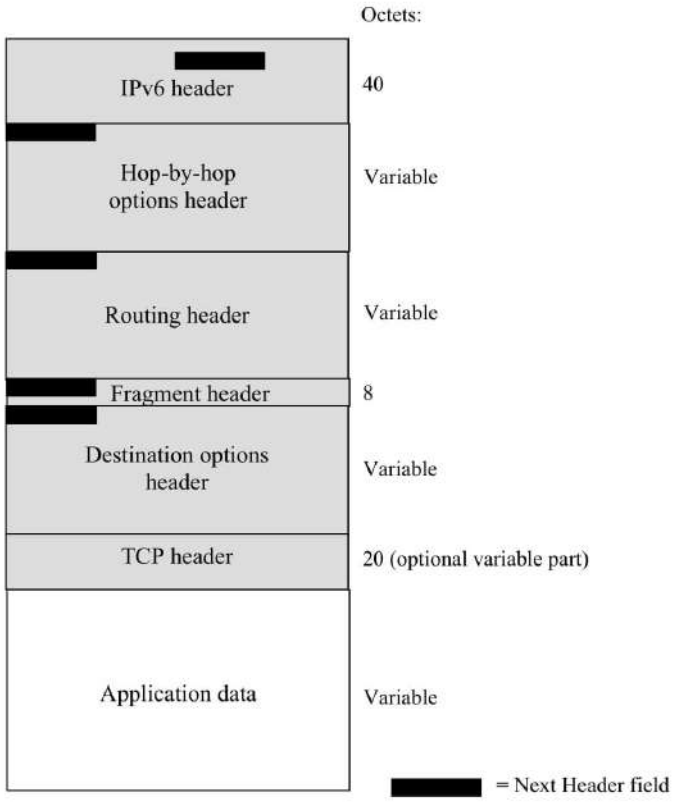
۱- سرآیند IPv6: اجباری و همیشه در ابتدا
۲- سرآیند Hop-by-Hop Options
۳- سرآیند Destination Options: برای موارد اختیاری که بایستی بتوسط اولین مقصدی که در میدان IPv6 Destination Address مشخص شده و مقصدهائی که بعد از آن در سرآیند Routing آمده است، پردازش شود
۴- سرآیند Routing
۵- سرآیند Fragment

۶- سرآیند Authentication

۷- سرآیند Encapsulating Security Payload

۸- سرآیند Destination Options: برای موارد اختیاری که تنها بتوسط مقصد انتهائی بسته پردازش می‌شود

شکل ۱۵-۶ مثالی از یک بستهٔ IP را نشان می‌دهد که شامل موردی از هر سرآیند غیرامنیتی است. توجه کنید که سرآیند IPv6 و هر سرآیند الحاقی، دارای یک میدان Next Header است. این میدان نوع سرآیندی را که بلافاصله بعد از آن قرار دارد، تعیین می‌کند. در غیر اینصورت این میدان شامل شناسهٔ پروتکل لایهٔ بالاتری است که از IPv6 استفاده می‌کند. در شکل، پروتکل لایهٔ بالاتر TCP بوده و بنابراین دیتای لایهٔ بالاتر که بتوسط بستهٔ IPv6 حمل می‌گردد شامل یک سرآیند TCP است که به دنبال آن یک بلوک از دیتای کاربردی قرار دارد.



شکل ۱۵-۶ بستهٔ IPv6 با سرآیندهای الحاقی (شامل یک سِگمنت TCP)

سرآیند Hop-by-Hop Options: اطلاعات اختیاری را حمل می‌کند که اگر وجود داشته باشد بایستی بتوسط هر مسیریاب در طول مسیر مورد بازبینی قرار گیرد. این سرآیند شامل میدان‌های زیر است:

- **Header Extension Length (8 bits)**: طول این سرآیند بر حسب واحدهای ۶۴- بیتی را مشخص می‌کند که شامل اولین ۶۴ بیت نیست.
- **Options**: شامل یک یا چند مورد اختیاری است. هر مورد اختیاری از سه زیرمیدان تشکیل می‌شود: یک tag که نمایش‌دهندهٔ نوع مورد اختیاری، یک طول و یک اندازه است.

تا کنون تنها یک مورد اختیاری تعریف شده است: مورد Jumbo Payload که برای ارسال بسته‌های IPv6 با محموله‌هائی طولانی‌تر از ۶۵,۵۳۵ - ۱-$۲^{۶۱}$ اُکتت بکار می‌رود. میدان Option Data در اینجا ۳۲ بیت طول داشته و اندازهٔ طول بسته را، بجز سرآیند IPv6، بر حسب اُکتت مشخص می‌کند. برای هر بسته، میدان Payload Length در سرآیند IPv6 بایستی برابر صفر قرار داده شود و بایستی هیچ سرآیند Fragment وجود نداشته باشد. با این اختیار، IPv6 بسته‌های دیتائی تا طول چهار میلیارد اُکتت را حمایت می‌کند. این امر انتقال بسته‌های بزرگ ویدئو را تسهیل نموده و IPv6 را قادر می‌سازد تا از ظرفیت موجود محیط انتقال حداکثر استفاده را بنماید.

سرآیند Routing شامل لیستی از یک یا چند گرهٔ میانی است که بایستی در مسیر بسته تا مقصد ملاقات شوند. تمام سرآیندهای مسیریابی با یک بلوک ۳۲- بیتی که شامل چهار میدان ۸- بیتی است آغاز می‌شوند و به دنبال آن داده‌های مسیریابی متعلق به روش مسیریابی خاصی قرار می‌گیرد. چهار میدان ۸ - بیتی عبارت از Next Header، Extension Length Header و دو میدان زیراند:

- **Routing Type**: یک سرآیند Routing خاص را تعیین می‌نماید. اگر مسیریابی اندازهٔ Routing Type را شناسائی نکند، بایستی بسته را معدوم سازد.
- **Segments Left**: تعداد صریح گره‌های میانی که بایستی قبل از رسیدن به مقصد نهائی ملاقات شوند.

علاوه براین تعریف سرآیند عمومی، مشخصه‌های IPv6 سرآیند Type 0 Routing را تعریف می‌کند. وقتی از سرآیند Type 0 Routing استفاده می‌شود، گرهٔ منبع، نهائی‌ترین آدرس مقصد را در سرآیند IPv6 قرار نمی‌دهد. در عوض آن آدرس، آخرین آدرس لیست شده در سرآیند Routing بوده و سرآیند IPv6 شامل آدرس مقصد اولین مسیریاب موجود در مسیر خواهد بود. سرآیند Routing تا زمانی که بسته به گرهٔ مقصد سرآیند IPv6 نرسد، بررسی نخواهد شد. در آن نقطه، محتویات سرآیند IPv6 و Routing به روزرسانی شده و بسته مجدداً به جلو رانده می‌شود. به روزرسانی شامل قرار دادن آدرس بعدی در سرآیند IPv6 و کاهش دادن میدان Segments Left در سرآیند Routing است.

در IPv6 لازم است تا یک گره IPv6، مسیرها را در یک بستهٔ دریافت شدهٔ شامل سرآیند Routing معکوس کرده تا بسته بتواند به فرستنده برگردد.

سرآیند Fragment وقتی بتوسط یک منبع بکار می‌رود که قطعه قطعه کردن دیتا مورد نیاز باشد. در IPv6 قطعه‌قطعه کردن دیتا تنها می‌تواند در گره‌های مبدأ انجام شود و نه بتوسط مسیریاب‌هائی که در مسیر تحویل بسته واقع‌اند. برای استفادهٔ کامل از امتیازات محیط بین شبکه‌ها، یک گره بایستی یک الگوریتم کشف مسیر را به اجرا گذاشته که او را قادر

می‌سازد تا کوچک‌ترین واحد انتقال ماکزیمم (MTU) که بتوسط هر زیرشبکه مسیر حمایت می‌شود را پیدا کند. بعبارت دیگر، الگوریتم کشف مسیر، یک گره را قادر می‌سازد تا MTU زیرشبکهٔ «گلوگاه» در روی مسیر را کشف نماید. با این معلومات، گرهٔ منبع، در صورت نیاز، برای هر آدرس مقصد داده شده دیتا را قطعه‌قطعه خواهد کرد. در غیراینصورت منبع بایستی تمام بسته‌ها را به ۱,۲۸۰ اُکتت محدود سازد که حداقل MTU ای است که بایستی بتوسط هر زیرشبکه حمایت گردد.

علاوه بر میدان Next Header، سرآیند Fragment شامل میدان‌های زیر است:

- **Fragment Offset (13 bits)**: نشان می‌دهد که محمولهٔ این fragment به کجای بستهٔ اصلی تعلق دارد. این برحسب واحدهای ۶۴- بیتی محاسبه می‌شود. این امر بطور ضمنی شامل این مطلب است که قطعه‌ها (بغیر از آخرین قطعه) بایستی دارای میدان دیتائی باشند که مضربی از ۶۴ بیت است.
- **Res (2 bits)**: برای مصارف آتی رزرو شده است.
- **M Flag (1 bit)**: اگر مساوی 1 باشد یعنی قطعه‌های دیگری وجود دارند و اگر مساوی 0 باشد یعنی آخرین قطعه است.
- **Identification (32 bits)**: هدف آن شناسائی بستهٔ اولیه بطور یکتاست. این شناسهٔ بایستی برای آدرس منبع و آدرس مقصد بسته، برای مدت زمانی که بسته در اینترنت خواهد ماند، یکتا باشد. تمام قطعه‌هائی که دارای شناسهٔ یکسان، آدرس منبع یکسان و آدرس مقصد یکسان می‌باشند دوباره بهم پیوسته و بستهٔ اولیه را درست خواهند کرد.

سرآیند Destination Options اطلاعات اختیاری را حمل می‌کنند که اگر وجود داشته باشد تنها بتوسط گرهٔ مقصد مورد بررسی قرار می‌گیرد. فرمت این بستهٔ سرآیند همانند فرمت سرآیند Hop-by-Hop Options است.

فصل ۷

امنیت WEB

۷-۱ **ملاحظات امنیت وِب**

تهدیدهای امنیتی وب
روش‌های برخورد با امنیت ترافیک وِب

۷-۲ **لایهٔ سوکت آمن و امنیت لایهٔ حمل و نقل (SSL/TLS)**

معماری SSL
پروتکل SSL Record
پروتکل Change Cipher Spec
پروتکل Alert
پروتکل Handshake
محاسبات رمزنگاری
امنیت لایهٔ حمل و نقل

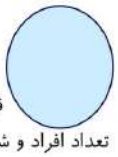
۷-۳ **معاملهٔ الکترونیکی آمن (SET)**

مروری بر SET
امضاء دوگانه
عملیات پرداخت

۷-۴ **منابع مطالعاتی**

۷-۵ **واژه‌های کلیدی، سؤالات مرورکنندهٔ بحث و مسائل**

واژه‌های کلیدی
سؤالات مرورکنندهٔ بحث
مسائل



قریباً تمام کسب و کارها، بیشتر سازمان‌های دولتی و شمار بسیاری از افراد، امروزه صاحب وِب سایت هستند. تعداد افراد و شرکت‌هائی که دسترسی به اینترنت دارند بسرعت افزایش یافته و تمام آنها به مرورگرهای گرافیکی وِب مجهزاند. در همین رابطه بازرگانان علاقه‌مندند تا بمنظور تجارت الکترونیک، تسهیلاتی را روی وِب فراهم نمایند. اما واقعیت این است که اینترنت و وِب شدیداً در مقابل دست‌یابی‌های غیرمجاز از انواع مختلف آسیب‌پذیرند. همین‌طور که کار و پیشه به این امر وقوف بیشتری پیدا می‌کند، تقاضا برای سرویس‌های امن وِب بیشتر می‌شود.

مقولهٔ امنیت وِب بسیار وسیع بوده و خود می‌تواند به‌تنهائی موضوع یک کتاب باشد. در این فصل ابتدا نیازهای عمومی امنیت وِب را بررسی کرده و آنگاه روی دو روش استاندارد SSL/TLS و SET که در بحث تجارت وِب اهمیت فزاینده‌ای پیدا کرده‌اند، متمرکز می‌شویم.

۷-۱ ملاحظات امنیت وِب

World Wide Web یا تار جهان‌گستر اصولاً یک کاربرد کلاینت/سِرور بوده که روی اینترنت و اینترانت‌های TCP/IP کار می‌کند. با چنین نگرشی، ابزارهای امنیتی و روش‌هائی که تا کنون در این کتاب مورد بحث قرار گرفته‌اند به مسألهٔ امنیت وِب هم مربوط می‌شوند. اما همانطور که در [GARF97] خاطرنشان شده است، وِب چالش‌های جدیدی را بوجود می‌آورد که معمولاً در مقولهٔ امنیت کامپیوتر و امنیت شبکهٔ نمی‌گنجند:

- اینترنت دوطرفه است. برخلاف محیط‌های انتشاراتی سنّتی و حتی سیستم‌های انتشاراتی الکترونیک که شامل تله‌تکست، پاسخ صوتی و یا فاکس‌برگردان می‌باشند، وِب نسبت به حملاتی که از طریق اینترنت روی سِرورهای وِب می‌شود آسیب‌پذیر است.
- وِب بصورت فزاینده‌ای بعنوان یک خروجی بسیار مرئی برای اطلاعات مربوط به سازمان‌ها و محصولات مختلف بکار گرفته شده و پایگاهی برای اسناد تجاری محسوب می‌گردد. اگر سِرورهای وِب مورد تهاجم قرار گیرند، شهرت و اعتبار و سرمایهٔ شرکت‌ها مورد تهدید قرار می‌گیرند.
- اگرچه مرورگرهای وِب به آسانی مورد استفاده قرار می‌گیرند، مدیریت و پیکربندی سِرورهای وِب نسبتاً آسان است و تهیهٔ محتویات وِب روز به روز سهل‌تر می‌شود، ولی نرم‌افزار زیربنای وِب بطور فوق‌العاده‌ای پیچیده است. این نرم‌افزار پیچیده ممکن است بسیاری از نقصان‌های امنیتی وِب را پنهان سازد. تاریخچهٔ کوتاه وِب پر از مثال‌هائی از سیستم‌های جدید، به روزرسانی شده و صحیح نصب شده است که در مقابل حملات امنیتی متعددی آسیب‌پذیر بوده‌اند.

- یک سِرور وِب می‌تواند بعنوان یک پایگاه عملیاتی مورد سوءاستفادهٔ مهاجمین قرار گرفته تا به تمام سیستم کامپیوتری یک سازمان حمله نمایند.
- کاربران متنوع و بی‌اطلاع (از دید مسائل امنیتی)، معمولاً کلاینت‌های سِرورهای وِب را تشکیل می‌دهند. این کاربران الزاماً از ریسک‌های موجود اطلاع نداشته و ابزار یا معلومات لازم برای مقابلهٔ مؤثر با این تهدیدها را ندارند.

تهدیدهای امنیتی وِب

جدول ۱-۷ خلاصه‌ای از انواع تهدیدهای امنیتی که در استفاده از وِب با آنها مواجهیم را نشان می‌دهد. یک روش برای دسته‌بندی این تهدیدها این است که آنها را بر اساس حملات غیرفعال و فعال دسته‌بندی کنیم. حملات غیرفعال شامل استراق‌سمع ترافیک شبکه بین مرورگر و سِرور و دست‌یابی به اطلاعات یک سایت وِب است که قرار بوده است محرمانه باشد. حملات فعال شامل جا زدن خود بجای شخص دیگر، تغییر پیام‌های در حال ترانزیت بین کلاینت و سِرور و همچنین تغییردادن اطلاعات یک وِب سایت هستند.

راه دیگری برای طبقه‌بندی تهدیدهای وِب این است که آنها را بر حسب محل تهدید طبقه‌بندی نمائیم: سِرور وِب، مرورگر وِب و ترافیک شبکه‌ای بین مرورگر و سِرور.

مقوله‌های امنیت سِرور و امنیت مرورگر در شاخهٔ امنیت سیستم‌های کامپیوتری جای دارند. در قسمت‌های بعدی این کتاب، مقولهٔ امنیت سیستم بطور کلی مورد بحث قرار می‌گیرد که البته قابل اعمال به امنیت سیستم‌های وِب نیز هست. مقولهٔ مربوط به امنیت ترافیک در طبقه‌بندی امنیت شبکه بوده و در این فصل به آن اشاره می‌شود.

جدول ۱-۷ یک مقایسه از تهدیدهای امنیتی وِب [RUBI97]

	تهدیدها	پیامد تهدیدها	روش‌های مقابله
صحت	• دستکاری در داده‌های کاربر • مرورگر اسب تروا • دستکاری حافظه • دستکاری ترافیک پیام در حال ترانزیت	• از بین رفتن اطلاعات • لورفتن ماشین • آسیب‌پذیری به انواع تهدیدهای دیگر	استفاده از جمع‌های کنترلی مرتبط با رمزنگاری
محرمانگی	• استراق سمع روی اینترنت • دزدی اطلاعات از سِرور • کسب اطلاعات در مورد پیکربندی شبکه • کسب اطلاعات در مورد این‌که کدام کلاینت با سِرور در تماس است.	• از بین رفتن اطلاعات • از بین رفتن محرمانگی	رمزنگاری، استفاده از پروکسی‌های وِب
انکار سرویس	• قطع رشته‌های ارتباطی کاربر • ایجاد سیلابی از تهدیدهای ساختگی • پرکردن حافظه‌های کامپیوتر • ایزوله کردن ماشین با حملات DNS	• ایجاد وقفه • ایجاد اذیت و آزار • بازماندن کاربر از انجام کارهای عادی	جلوگیری از این تهدیدها مشکل است
اعتبارسنجی	• جعل هویت کاربران قانونی • تقلب در داده‌ها	• معرفی غلط کاربر • ایجاد باور نسبت به صحت اطلاعات غلط	تکنیک‌های رمزنگاری

روش‌های برخورد با امنیت ترافیک وِب

روش‌های مختلفی برای تأمین امنیت وِب وجود دارد. روش‌های متفاوتی که مورد توجه قرار گرفته‌اند از نظر سرویس‌هائی که بوجود می‌آورند مشابهت داشته و حتی تا حد زیادی از مکانیسم‌های یکسانی استفاده می‌کنند. تفاوت عمدهٔ آنها در محدودهٔ عملیاتی روش‌ها و همچنین مکان آنها در پشتهٔ پروتکلی TCP/IP است.

شکل ۱-۷ تفاوت اخیر را نشان می‌دهد. یکی از روش‌های فراهم آوردن امنیت وِب، استفاده از امنیت IP است (شکل ۱-۷الف). مزیت استفاده از IPSec این است که برای کاربران و کاربردهای انتهائی نامرئی بوده و یک راه حل همه منظوره بشمار می‌آید. علاوه براین IPSec شامل یک قابلیت فیلترینگ بوده بطوری که تنها ترافیک انتخاب شده، لازم است از سربارهٔ پردازش IPSec استفاده کنند.

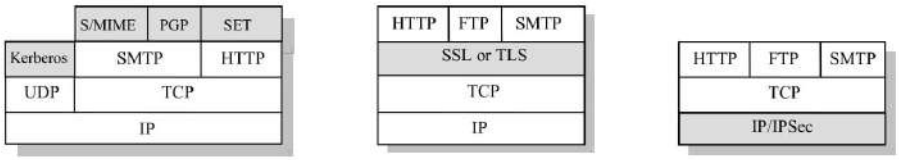
یک روش همه منظورهٔ دیگر این است که امنیت را درست در بالای لایهٔ TCP/IP پیاده‌سازی نمائیم (شکل ۱-۷ب). در رأس این روش لایهٔ سوکت امن Secure Socket Layer (SSL) و پس از آن استاندارد بعدی اینترنت بنام امنیت لایهٔ حمل‌ونقل Transport Layer Security (TLS) قرار دارد. در این سطح دو روش پیاده‌سازی مختلف وجود دارد. برای عمومیت کامل، SSL (یا TLS) می‌تواند بعنوان بخشی از پشتهٔ پروتکلی قلمداد شده و در نتیجه برای کاربردها نامرئی جلوه کند. روش دیگر این است که SSL در بسته‌های نرم‌افزاری مشخص جای داده شود. بعنوان مثال مرورگرهای Netscape و Microsoft Explorer مجهز به SSL به بازار می‌آیند. همچنین بیشتر سرورهای وِب این پروتکل را بکار گرفته‌اند.

سرویس‌های امنیتی مختص به کاربرد، در شکم کاربردهای مختلف جای دارند. شکل ۱-۷ج مثال‌هائی از این معماری را نشان می‌دهد. حسن این روش این است که سرویس را می‌توان بر حسب نیازهای مشخص یک کاربرد خاص دستکاری نمود. در محدودهٔ امنیت وِب، یک مثال مهم از این نوع برخورد، Secure Electronic Transaction (SET) است.

بقیهٔ این فصل به توصیف SSL/TLS و SET می‌پردازد.

۷-۲ لایهٔ سوکت اَمن و امنیت لایهٔ حمل و نقل (SSL/TLS)

SSL را Netscape پایه‌گذاری کرد. نسخهٔ سوم این پروتکل با نظرسنجی عمومی و بازخورد گرفته شده از صنعت، طراحی شد و به عنوان پیش‌نویس یک سند اینترنت منتشر گردید. بعد از آن وقتی یک توافق کلی برای تسلیم پروتکل بعنوان یک استاندارد اینترنت حاصل شد، گروه کاری TLS در درون IETF تشکیل گردید تا یک استاندارد مشترک را فراهم آورد. اولین نسخهٔ منتشرشده TLS را می‌توان SSLv3.1 دانست که تا حد زیادی با نسخهٔ قدیمی SSLv3 سازگار است.



(ج) سطح کاربرد (ب) سطح حمل و نقل (الف) سطح شبکه

شکل ۱-۷ موقعیت نسبی تسهیلات امنیتی در پشتهٔ پروتکلی TCP/IP

عمده مطالب این بخش به بحث در مورد SSLv3 اختصاص دارد. در انتهای بخش، تفاوت‌های بین SSLv3 و TLS توضیح داده شده است.

معماری SSL

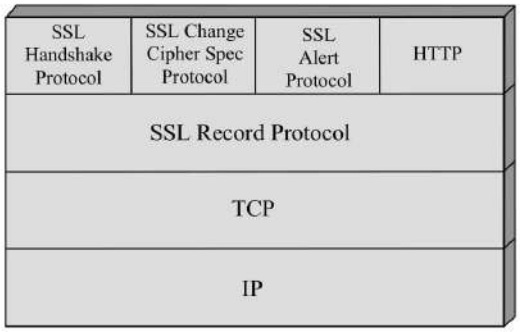
SSL برای فراهم آوردن یک سرویس امن سر- به- سر با استفاده از TCP/IP طراحی شده است. SSL یک پروتکل تنها نبوده بلکه همانطور که در شکل ۲-۷ نشان داده شده است، دو لایه از پروتکل‌ها آن را تشکیل می‌دهند.

پروتکل SSL Record سرویس‌های امنیتی پایه را برای پروتکل‌های مختلف لایهٔ بالاتر فراهم می‌آورد. علی‌الخصوص پروتکل http (hypertext transfer protocol) که سرویس انتقال برای تعامل کلاینت/سِرور روی وِب را بوجود می‌آورد می‌تواند در بالای SSL کار کند. سه پروتکل لایهٔ بالاتر به عنوان بخشی از SSL تعریف شده‌اند: پروتکل Handshake، پروتکل Change Cipher Spec، و پروتکل Alert. این پروتکل‌های مختص به SSL، در مدیریت ارتباطات SSL مورد استفاده قرار گرفته و بعداً در مورد آنها صحبت خواهد شد.

دو مفهوم مهم SSL، یکی اتصال SSL و دیگری اجلاس SSL است که در مشخصات به صورت زیر تعریف شده‌اند:

- **اتصال (Connection)**: یک اتصال، یک بستر حمل‌ونقل اطلاعات (در تعریف لایه‌های OSI) است که نوع مناسبی از سرویس را فراهم می‌آورد. برای SSL چنین اتصالاتی دارای رابطهٔ نظیر- به- نظیراند. اتصالات موقتی هستند. هر اتصال مرتبط با یک اجلاس است.
- **اجلاس (Session)**: یک اجلاس SSL، یک اتحاد بین یک کلاینت و یک سِرور است. اجلاس‌ها بتوسط پروتکل Handshake شکل می‌گیرند. اجلاس‌ها، مجموعه‌ای از پارامترهای امنیتی مرتبط با رمزنگاری را تعریف می‌کنند که می‌توانند بین اتصالات مختلف به اشتراک گذاشته شوند. از اجلاس‌ها بمنظور اجتناب از توافقات گران قیمت در مورد پارامترهای امنیتی جدید برای هر اتصال استفاده می‌شود.

بین دو زوج مرتبط (کاربردهائی همانند HTTP روی کلاینت و سِرور)، ممکن است اتصالات امن متعددی وجود داشته باشد. همچنین از نظر تئوری، ممکن است اجلاس‌های هم‌زمان متعددی نیز بین طرفین موجود باشد اما از این خصوصیت در عمل استفاده‌ای نمی‌شود.



شکل ۲-۷ پشتهٔ پروتکلی SSL

در حقیقت با هر اجلاس حالات متعددی مرتبط است. همین که اجلاسی برقرار شد، یک حالت عملیاتی جاری هم برای خواندن و هم برای نوشتن (یعنی دریافت و ارسال) وجود دارد. علاوه بر آن در خلال اجرای پروتکل Handshake حالات موقت خواندن و نوشتن ایجاد می‌شود. پس از پایان موفقیت‌آمیز پروتکل Handshake، حالات موقت به حالات جاری تبدیل می‌شوند.

حالت یک اجلاس بتوسط پارامترهای زیر تعریف می‌شود (تعاریف از مشخصه‌های SSL اقتباس شده‌اند):

- **Session identifier**: یک دنبالهٔ اختیاری از بایت‌ها که بتوسط سِرور انتخاب شده تا نشان دهد که حالت اجلاس فعال و یا قابل تداوم است.
- **Peer certificate**: یک گواهی X509.v3 واحد نظیراست. این عنصر حالت ممکن است خالی باشد.
- **Compression method**: الگوریتم استفاده شده برای فشرده‌سازی دیتا قبل از رمزنگاری است.
- **Cipher spec**: الگوریتم رمزنگاری دیتای اصلی (مانند null، DES و غیره)، الگوریتم hash (مانند SHA-1 یا MD5) که برای محاسبه MAC بکار می‌رود و همچنین سایر مشخصات رمزنگاری مثل اندازهٔ hash را مشخص می‌سازد.
- **Master secret**: ۴۸ بایت سرّی که بین کلاینت و سِرور به اشتراک گذاشته می‌شود.
- **Is resumable**: یک پرچم که مشخص می‌کند آیا اجلاس می‌تواند برای ایجاد اتصالات جدید بکار رود.

حالت یک اتصال با پارامترهای زیر تعریف می‌شود:

- **Server and client random**: دنباله‌ای از بایت‌ها که بتوسط سِرور و کلاینت برای هر اتصال انتخاب می‌شوند.
- **Server write MAC secret**: کلید سرّی بکار رفته در عملیات تولید MAC بر روی دیتائی که بتوسط سِرور ارسال شده است.
- **Client write MAC secret**: کلید سرّی بکار رفته در عملیات تولید MAC بر روی دیتائی که بتوسط کلاینت ارسال شده است.
- **Server write key**: کلید رمزنگاری سنّتی برای دیتائی که بتوسط سِرور رمزنگاری شده و بتوسط کلاینت رمزگشائی می‌شود.
- **Client write key**: کلید رمزنگاری سنّتی برای دیتائی که بتوسط کلاینت رمزنگاری شده و بتوسط سِرور رمزگشائی می‌شود.
- **Initialization vectors**: وقتی از یک رمز قالبی در مُود CBC استفاده می‌شود، یک بردار ابتدائی (IV) برای هر کلید وجود دارد. این میدان ابتدا بتوسط پروتکل SSL Handshake انتخاب می‌شود. بعد از آن، از آخرین قالب رمزشده هر رکورد، برای IV رکورد بعدی استفاده می‌شود.
- **Sequence numbers**: هر یک از طرفین ارتباط، شماره ردیف‌های متفاوتی را برای پیام‌های ارسال شده و دریافت شده برای هر اتصال نگهداری می‌کند. وقتی یک طرف ارتباط، یک پیام تغییر Cipher spec را ارسال می‌کند، شمارهٔ ردیف مرتبط صفر می‌شود. شماره ردیف‌ها نمی‌توانند از $۲^{۶۴}-۱$ تجاوز نمایند.

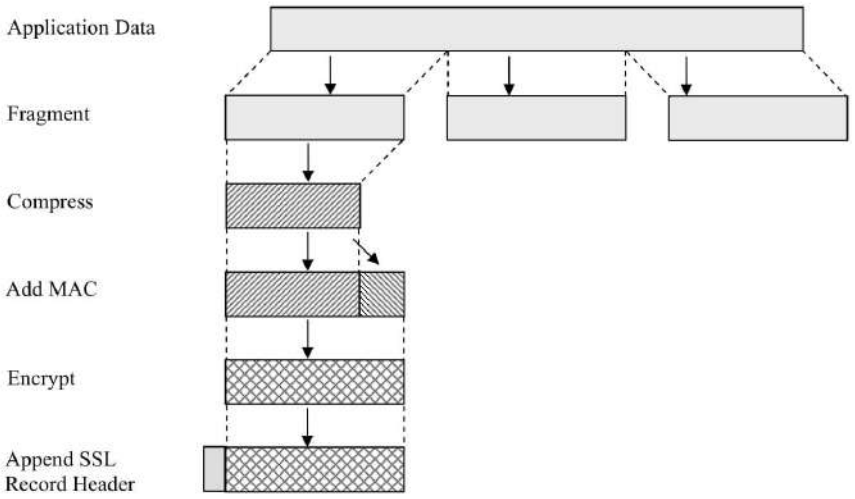
پروتکل SSL RECORD

پروتکل SSL Record دو سرویس را برای اتصالات SSL فراهم می‌آورد:

- **محرمانگی**: پروتکل Handshake، یک کلید سرّی مشترک را تعریف می‌کند که برای رمزنگاری سنّتی محمولهٔ دیتای SSL بکار می‌رود.
- **صحت پیام**: پروتکل Handshake، همچنین یک کلید سرّی مشترک دیگر را تعریف می‌کند که برای تشکیل کُد اعتبارسنجی پیام (MAC) مورد نیاز است.

شکل ۳-۷ عملیات کلی پروتکل SSL Record را نشان می‌دهد. این پروتکل یک پیام کاربردی را که قرار است ارسال شود اخذ کرده، دیتا را بصورت قالب‌های کوچکتر قابل پردازش درآورده، دیتا را بصورت اختیاری فشرده‌سازی نموده، یک سرآیند (header) به آن اضافه کرده و واحد نتیجه شده را در یک سِگمنت TCP انتقال می‌دهد. دیتای دریافت شده رمزگشائی شده، تأئید شده، از فشرده‌سازی درآمده و دوباره بهم می‌پیوندد و سپس به کاربران لایهٔ‌های بالاتر تحویل داده می‌شود.

اولین قدم، **قطعه قطعه کردن دیتا (fragmentation)** است. هر پیام لایهٔ بالاتر بصورت بلوک‌هائی با اندازهٔ 2^{14}=۱۶,۳۸۴ بایت یا کمتر در می‌آید. سپس **فشرده‌سازی (compression)** بصورت اختیاری و در صورت نیاز روی آن صورت می‌پذیرد. فشرده‌سازی بایستی بدون تلفات بوده و نبایستی طول محتویات را بیش از ۱,۰۲۴ بایت افزایش دهد. در SSLv3 (و همچنین در نسخهٔ فعلی TLS) نوع الگوریتم فشرده‌سازی مشخص نشده و بنابراین پیش فرض، عدم حضور آن است.



شکل ۳-۷ عملیات SSL Record Protocol

قدم بعدی پردازش، محاسبهٔ یک **کُد اعتبارسنجی پیام (MAC)** از دیتای فشرده شده است. برای این کار از یک کلید سرّی مشترک استفاده می‌شود. محاسبات چنین تعریف می‌شوند:

```
hash(MAC_write_secret || pad_2 ||
        hash(MAC_write_secret || pad_1 || seq_num || SSLCompressed.type ||
        SSLCompressed.length || SSLCompressed.fragment))
```

که درآن

||	= جمع رشته‌ای (concatenation)
MAC_write_secret	= کلید سرّی مشترک
hash	= الگوریتم رمزی hash که یا MD5 و یا SHA-1 است
pad_1	= بایت 0011 0110 (36 هکزادسیمال) که ۴۸ بار برای MD5 (۳۸۴ بیت) و ۴۰ بار برای SHA-1 (۳۲۰ بیت) تکرار می‌شود
pad_2	= بایت 1100 0101 (5C هکزادسیمال) که ۴۸ بار برای MD5 و ۴۰ بار برای SHA-1 تکرار می‌شود
seq _num	= شماره ردیف این پیام
SSLCompressed.type	= پروتکل لایهٔ بالاتر که برای پردازش این فِرگمنت بکار می‌رود
SSLCompressed.length	= طول فِرگمنت فشرده شده
SSLCompressed.fragment	= فِرگمنت فشرده شده (اگر از فشرده‌سازی استفاده نشده است، متن ساده)

توجه کنید که این خیلی شبیه الگوریتم HMAC است که در فصول قبل از آن یاد شد. تفاوت در این است که دو مقدار pad در SSLv3 باهم جمع رشته‌ای شده درحالی که در HMAC باهم XOR می‌شوند. الگوریتم MAC در SSLv3 مبتنی بر پیش نویس اینترنتی اولیه برای HMAC بوده که از جمع رشته‌ای استفاده می‌نماید. آخرین نسخهٔ HMAC که در RFC 2104 تعریف شده است از XOR استفاده می‌کند.

سپس پیام فشرده‌شده بعلاوهٔ MAC با استفاده از روش رمزنگاری متقارن **رمزنگاری** می‌شود. رمزنگاری نمی‌تواند طول محتوی را بیش از۱,۰۲۴ بایت افزایش دهد و بنابراین طول نهائی نبایستی از ۲,۰۴۸ + 2^{14} تجاوز کند. الگوریتم‌های رمزنگاری زیر الگوریتم‌های مجازند:

رمز قالبی		رمز دنباله‌ای	
الگوریتم	طول کلید	الگوریتم	طول کلید
AES	128,256	RC4-40	40
IDEA	128	RC4-128	128
RC2-40	40		
DES-40	40		
DES	56		
3DES	168		
Fortezza	80		

Fortezza می‌تواند در روش رمزنگاری یک کارت هوشمند بکار رود.

برای رمزنگاری دنباله‌ای، پیام فشرده شده باضافهٔ MAC رمزنگاری می‌شوند. توجه کنید که MAC قبل از اینکه رمزنگاری صورت پذیرد محاسبه می‌شود و آنگاه MAC باضافهٔ متن ساده یا متن سادهٔ فشرده شده رمزنگاری می‌گردد.

برای رمزنگاری قالبی، padding می‌تواند پس از محاسبهٔ MAC و قبل از رمزنگاری انجام شود. padding عبارت از تعدادی بایت لائی است که در انتهای آن یک بایت که اندازهٔ لائی را نشان می‌دهد قرار دارد. اندازهٔ کل لائی کوچکترین مقداری است که بتواند اندازهٔ کل دیتائی که باید رمزنگاری شود (متن ساده باضافهٔ MAC باضافهٔ padding) را به اندازهٔ مضربی از طول بلوک رمز قالبی درآورد. مثال این مورد یک متن سادهٔ ۵۸- بایتی (یا اگر فشرده‌سازی صورت پذیرفته است متن فشرده) با یک MAC با اندازهٔ ۲۰ بایت (با استفاده از SHA-1) است که با یک الگوریتم که طول قالب آن ۸ بایت است (مثل DES) رمزنگاری شود. با محاسبهٔ بایت مربوط به طول لائی این مجموعه ۷۹ بایت خواهد شد. برای اینکه این مقدار مضرب ۸ شود، ۱ بایت لائی به آن اضافه می‌شود.

قدم نهائی در پردازش پروتکل SSL Record این است که یک سرآیند(header) که شامل میدان‌های زیر است به ابتدای آن اضافه شود:

- **نوع محتوا (۸ بیت)**: پروتکل لایهٔ بالاتر که برای پردازش این فِرگمنت بکار می‌رود.
- **نسخهٔ اصلی (۸ بیت)**: نسخهٔ اصلی SSL مورد استفاده را نشان می‌دهد. برای SSLv3 این اندازه 3 خواهد بود.
- **نسخهٔ فرعی (۸ بیت)**: نسخهٔ فرعی مورد استفاده را نشان می‌دهد. برای SSLv3 این مقدار 0 است.
- **طول فشرده‌شده (۱۶ بیت)**: طول فِرگمنت متن ساده بر حسب بایت (در صورت فشرده‌سازی، طول فِرگمنت فشرده شده). اندازهٔ ماکزیمم ۲,۰۴۸ + $۲^{۱۴}$ است.

محتواهای تعریف شده عبارت از change_cipher_spec، alert، handshake و application_data هستند. سه‌تای اول پروتکل‌های مختص SSL بوده که موضوع بحث بعدی است. توجه کنیدکه هیچ تفاوتی بین انواع کاربردهائی (مثلاً http) که می‌توانند از SSL استفاده کنند وجود ندارد. محتوای داده‌های خلق شده بتوسط چنین کاربردهائی از دید SSL غیرقابل رؤیت‌اند.

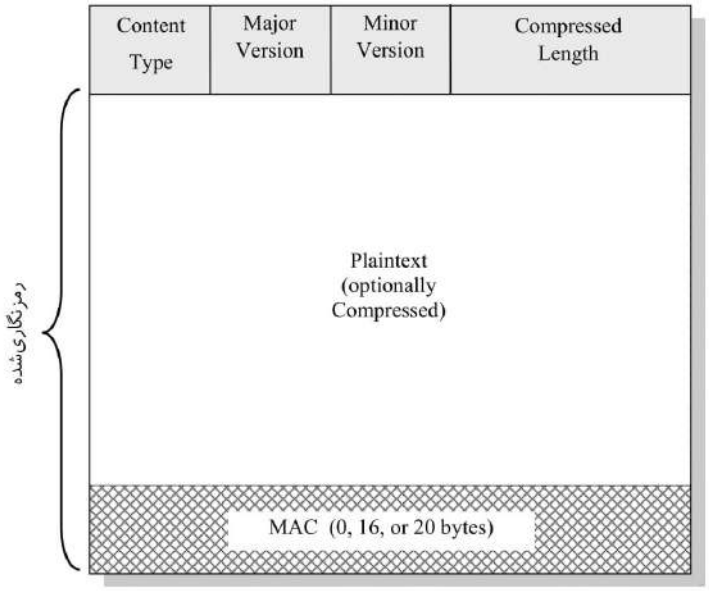
شکل ۴-۷ فرمت SSL Record را نشان می‌دهد.

پروتکل Change Cipher Spec

پروتکل Change Cipher Spec یکی از سه پروتکل مختص SSL است که از پروتکل SSL Record استفاده کرده و ساده‌ترین آنهاست. این پروتکل از یک پیام تنها تشکیل شده (شکل ۵-۷الف) که شامل یک بایت منفرد با مقدار 1 است. تنها هدف این پیام این است که باعث شود تا وضعیت موقت در وضعیت جاری کپی شده و مجموعهٔ رمزهائی که باید در این اتصال بکار روند بروز گردد.

پروتکل Alert

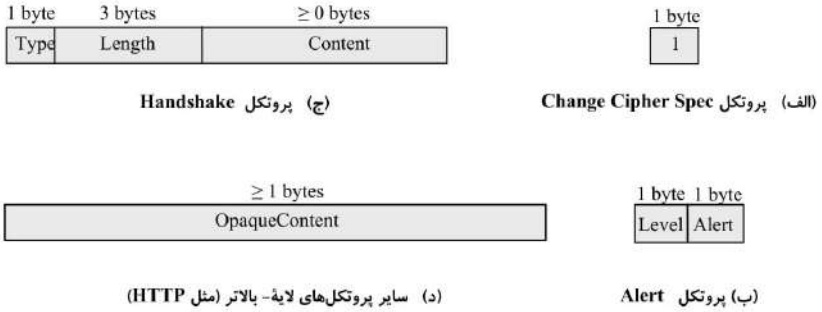
پروتکل Alert برای رساندن هشدارهای مرتبط با SSL به واحد نظیر بکار می‌رود. همانند سایر کاربردهائی که از SSL استفاده می‌کنند، پیام‌های هشدار فشرده‌سازی و رمزنگاری می‌شوند.



شکل ۴-۷ فرمت SSL Record

هر پیام در این پروتکل از دو بایت تشکیل می‌شود(شکل ۵-۷ب). بایت اول یا اندازهٔ 1 (warning) و یا اندازهٔ 2 (fatal) را داراست که اهمیت پیام را نشان می‌دهد. اگر اندازه برابر fatal باشد، SSL بلافاصله اتصال را خاتمه می‌دهد. سایر اتصالات مربوط به این اجلاس ممکن است ادامه یابند اما هیچ اتصال جدیدی در این اجلاس نمی‌تواند برقرار گردد. بایت دوم شامل یک کُد است که هشدار خاصی را مشخص می‌سازد. در ابتدا هشدارهائی که همیشه fatal هستند را بیان می‌کنیم (تعاریف از مشخصه‌های SSL اقتباس شده‌اند):

- **unexpected_message**: یک پیام نامربوط دریافت شده است.
- **bad_record_mac**: یک MAC ناصحیح دریافت شده است.
- **decompression_failure**: تابع معکوس فشرده‌سازی یک ورودی غیرصحیح را دریافت کرده است (مثلاً قادر نیست تا آنچه را دریافت کرده از فشرده‌سازی خارج کرده و یا در صورت این کار پیام از طول ماکزیمم مجاز بیشتر خواهد شد).
- **handshake_failure**: فرستنده قادر نبوده است تا روی یک مجموعهٔ قابل قبول از پارامترهای امنیتی با توجه به امکانات موجود توافق حاصل نماید.
- **illegal_parameter**: یک میدان در یک پیام handshake خارج از محدوده بوده و یا با سایر میدان‌های موجود همخوانی نداشته است.



شکل ۵-۷ محمولهٔ SSL Record Protocol

بقیهٔ هشدارها چنین‌اند:

- **close_notify**: به گیرنده اطلاع می‌دهد که فرستنده پیام‌های دیگری را روی این اتصال نخواهد فرستاد. هر یک از طرفین ارتباط موظف‌اند تا یک هشدار close_notify را قبل از بستن مرحلهٔ نوشتن پیام در یک اتصال ارسال دارند.
- **no_certificate**: ممکن است در پاسخ به درخواست یک گواهی، در صورتی که گواهی‌نامهٔ مطلوب در دسترس نباشد، ارسال شود.
- **bad_certificate**: گواهی دریافت شده مشکل داشته است (مثلاً شامل امضائی بوده است که مورد تأئید نیست).
- **unsupported_certificate**: نوع گواهی دریافت شده قابل قبول نیست.
- **certificate_revoked**: یک گواهی بتوسط امضاءکنندهٔ آن باطل شده است.
- **certificate_expired**: اعتبار یک گواهی تمام شده است.
- **certificate_unknown**: یک مسألهٔ غیرمشخص در مرحلهٔ پذیرش گواهی پیش آمده است که آن را غیرقابل پذیرش نموده است.

پروتکل Handshake

پیچیده‌ترین بخش SSL مربوط به پروتکل Handshake است. این پروتکل به سِرور و کلاینت اجازه می‌دهد تا هویت یکدیگر را تأئید نموده و راجع به یک الگوریتم رمزنگاری و MAC و همچنین کلیدهای رمزنگاری که قرار است دیتای ارسال شده در یک رکورد SSL را محافظت کنند، توافق نمایند. پروتکل Handshake قبل از اینکه هیچگونه دیتای کاربردی انتقال یابد، اجرا می‌شود.

پروتکل Handshake شامل یک سری پیام‌های ردوبدل شده بین کلاینت و سِرور است. تمام این پیام‌ها فرمت نشان داده شده در شکل ۵-۷ج را دارند. هر پیام شامل سه میدان است:

- **نوع (۱ بایت)**: نمایشگر یکی از ۱۰ نوع پیام متفاوت است. جدول ۲-۷ انواع پیام‌های مرتبط را نشان داده است.
- **طول (۳ بایت)**: طول پیام بر حسب بایت.
- **محتوا (0 بایت ≤)** : پارامترهای مرتبط با این پیام. این پارامترها در جدول ۲-۷ لیست شده‌اند.

شکل ۶-۷ مبادلات اولیه برای برقراری یک اتصال منطقی بین کلاینت و سرور را نشان می‌دهد. این مبادلات را می‌توان شامل چهار فاز دانست.

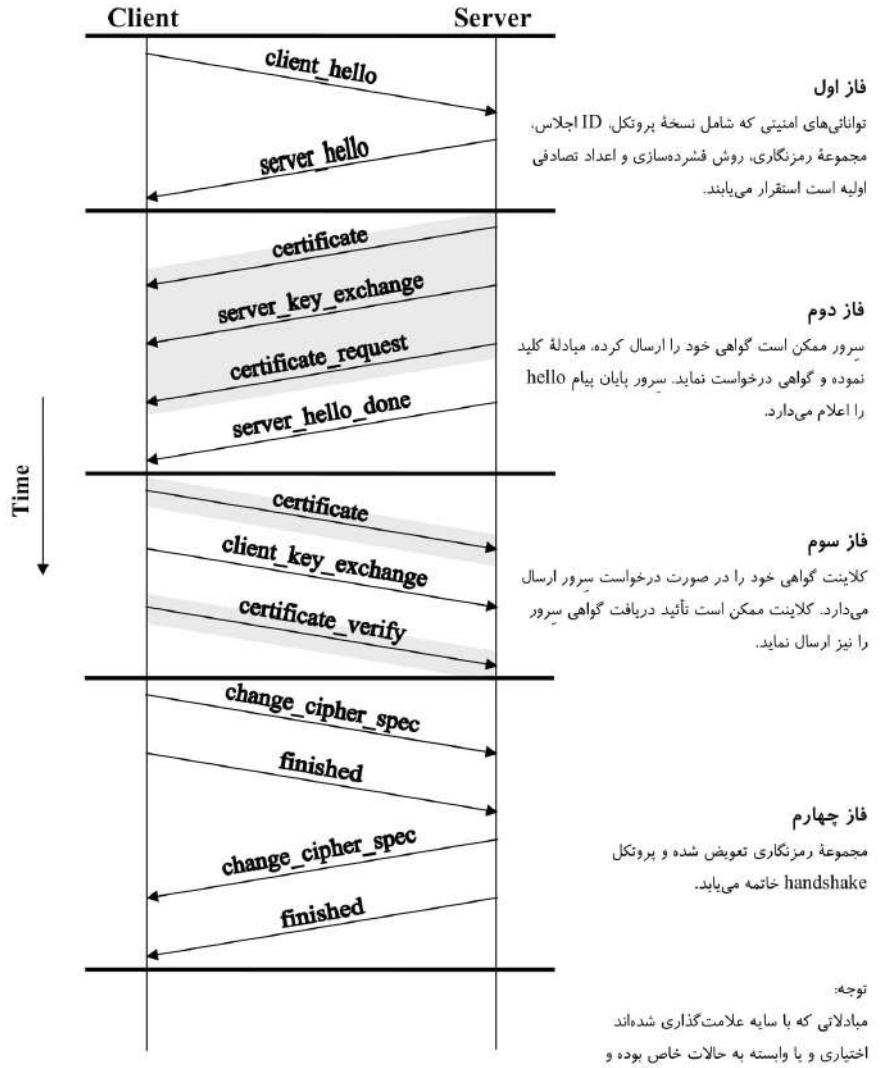
فاز اول- استقرار توانائی‌های امنیتی

این فاز برای شروع یک اتصال منطقی و استقرار توانائی‌های امنیتی مرتبط با آن بکار می‌رود. مبادلهٔ پیام‌ها از طرف کلاینت شروع می‌شود که یک **پیام client_hello** که شامل پارامترهای زیر است را ارسال می‌کند:

- **Version**: بالاترین نسخهٔ SSL که بتوسط کلاینت قابل فهم است.
- **Random**: یک ساختار تصادفی تولیدشده بتوسط کلاینت که شامل یک برچسب زمانی ۳۲- بیتی و همچنین ۲۸ بایت تولید شده بتوسط یک تولیدکنندهٔ اعداد تصادفی امن است. این مقادیر بعنوان nonceهای مختلف عمل نموده و در خلال مبادلهٔ کلید برای جلوگیری از حملات بازخوانی(replay) بکار می‌روند.
- **Session ID**: یک شناسهٔ اجلاس با طول متغیر. یک مقدار غیر صفر نشان می‌دهدکه کلاینت می‌خواهد پارامترهای موجود را بروزرسانی کرده و یا یک اتصال جدید در این اجلاس را برقرار نماید. یک مقدار صفر نمایش این است که کلاینت تمایل دارد تا یک اتصال جدید و یا یک اجلاس جدید را ایجاد کند.
- **CipherSuite**: این لیستی شامل مجموعه‌ای از الگوریتم‌های رمزنگاری است که مورد حمایت کلاینت بوده و بر حسب ترجیح نزولی مرتب شده است. هر عنصر لیست (هرمجموعهٔ رمز) هم الگوریتم تبادل کلید و هم یک CipherSpec را تعریف می‌کند که هردوی اینها بعداً توضیح داده خواهد شد.
- **Compression Method**: لیستی از روش‌های فشرده‌سازی مورد حمایت کلاینت است.

جدول ۲-۷ انواع پیام‌های پروتکل SSL Handshake

نوع پیام	پارامترها
hello_request	null
client_hello	version, random, session id, cipher suite, compression method
server_hello	version, random, session id, cipher suite, compression method
certificate	chain of X.509v3 certificates
server_key_exchange	parameters, signature
certificate_request	type, authorities
server_done	null
certificate_verify	signature
client_key_exchange	parameters, signature
finished	hash value



شکل ۶-۷ عملیات پروتکل Handshake

پس از ارسال پیام client_hello، کلاینت منتظر **پیام server_hello** می‌ماند که شامل همان پارامترهای پیام client_hello است. میدان Version شامل پائین‌ترین نسخهٔ پیشنهادشده بتوسط کلاینت و بالاترین نسخهٔ حمایت شده بتوسط سِرور است. میدان Random بتوسط سِرور تولید شده و مستقل از میدان Random کلاینت است. اگر میدان

SessionID کلاینت صفر نبوده است همان مقدار بتوسط سِرور نیز مورد استفاده قرار می‌گیرد. در غیر اینصورت میدان SessionID سِرور شامل مقدار تازه‌ای برای یک اجلاس جدید خواهد بود. میدان CipherSuite شامل یک مجموعهٔ رمزنگاری انتخاب شده بتوسط سِرور از بین آنهائی است که بتوسط کلاینت پیشنهاد شده‌اند. میدان Compression شامل متد فشرده‌سازی انتخاب شده بتوسط سِرور از بین آنهائی است که بتوسط کلاینت پیشنهاد شده‌اند.

اولین عنصر از پارامترهای CipherSuite متد مبادلهٔ کلید است (یعنی روشی که بتوسط آن کلیدهای رمزنگاری سنّتی و MAC مبادله می‌شوند). روش‌های زیر برای مبادلهٔ کلید مورد حمایت‌اند:

- **RSA**: کلید سرّی بتوسط کلید عمومی RSA گیرنده رمزنگاری می‌گردد. یک گواهی‌نامهٔ کلید- عمومی برای کلید گیرنده بایستی اخذ شود.
- **Fixed Diffie-Hellman**: این یک روش مبادلهٔ کلید Diffie-Hellman است که در آن گواهی‌نامهٔ سِرور شامل پارامترهای عمومی Diffie-Hellman بوده که بتوسط مسئول گواهی (CA) امضاء شده است. یعنی گواهی‌نامهٔ کلید- عمومی شامل پارامترهای کلید عمومی Diffie-Hellman است. کلاینت پارامترهای کلید عمومی Diffie-Hellman خود را یا در یک گواهی عرضه می‌نماید (اگر اعتبارسنجی کلاینت مورد نیاز باشد)، و یا اینکه آنها را در یک مبادلهٔ کلید عرضه می‌دارد. این روش به ایجاد یک کلید سرّی مشترک، مبتنی بر محاسبات Diffie-Hellman با استفاده از کلیدهای عمومی، بین دو واحد نظیر منجر می‌گردد.
- **Ephemeral Diffie-Hellman**: این تکنیک برای خلق کلیدهای سرّی ephemeral (موقت- یکبارمصرف) بکار می‌رود. در این مورد کلیدهای عمومی Diffie-Hellman مبادله می‌شوند که بتوسط کلیدهای خصوصی RSA فرستنده و یا کلید DSS امضاء شده‌اند. گیرنده می‌تواند از کلید عمومی متناظر با کلید خصوصی برای تأئید امضاء استفاده نماید. گواهی‌نامه‌ها برای اعتبارسنجی کلیدهای عمومی بکار می‌روند. بنظر می‌رسدکه این روش امن‌ترین روش از بین سه روش Diffie-Hellman باشد زیرا بواسطهٔ آن کلید موقت اعتبارسنجی شده ایجاد می‌گردد.
- **Anonymous Diffie-Hellman**: الگوریتم پایهٔ Diffie-Hellman بدون اعتبارسنجی مورد استفاده قرار می‌گیرد. یعنی هر طرف پارامترهای کلید عمومی Diffie-Hellman خود را برای دیگری ارسال کرده بدون اینکه هیچگونه احراز هویتی صورت پذیرد. این روش به حملات man-in-the-middle، که درآن حمله‌کننده بصورت ناشناس در وسط قرار گرفته و با طرفین ارتباط برقرار می‌کند، آسیب‌پذیر است.
- **Fortezza**: تکنیک تعیین شده در روش Fortezza.

بدنبال تعریف یک روش مبادلهٔ کلید، CipherSpec است که شامل میدان‌های زیر است:

- **CipherAlgorithm**: هر یک از الگوریتم‌هائی که قبلاً از آنها یاد شد مانند: Foretzza ,IDEA ,DES40 ,RC4 ,RC2 ,DES ,3DES
- **MACAlgorithm**: MD5 یا SHA-1
- **CipherType**: قالبی یا دنباله‌ای
- **IsExportable**: صحیح یا غلط
- **HashSize**: صفر یا ۱۶ بایت (برای MD5) و یا ۲۰ بایت (برای SHA-1).
- **Key Material**: دنباله‌ای از بایت‌ها که شامل داده‌های بکار رفته در تولید کلیدهای نوشتن است.
- **IV Size**: اندازهٔ Initialization Value برای رمزنگاری CBC (Cipher Block Chaining)

فاز دوم – تصدیق هویت سِرور و مبادلهٔ کلید

سِرور این فاز را، اگر لازم است که هویتش تصدیق شود، با ارسال گواهی‌نامهٔ خود آغاز می‌کند. پیام شامل یک یا زنجیره‌ای از گواهی‌های X.509 است. **پیام certificate** برای هریک از انواع روش‌های توافق شده مبادلهٔ کلید بجز anonymous Diffie-Hellman مورد نیاز است. توجه شود که اگر از fixed Diffie-Hellman استفاده شود این پیام گواهی بعنوان پیام مبادلهٔ کلید سِرور عمل می‌کند زیرا شامل پارامترهای عمومی Diffie-Hellman سِرور است.

سپس یک **پیام server_key_exchange** در صورت لزوم ارسال می‌شود. در دو حالت این پیام مورد لزوم نیست: (۱) سِرور یک گواهی با پارامترهای fixed Diffie-Hellman فرستاده است. (۲) از مبادلهٔ کلید RSA استفاده شود. پیام server_key_exchange برای موارد زیر مورد نیاز است:

- **Anonymous Diffie-Hellman**: محتوای پیام شامل دو مقدار عمومی Diffie-Hellman (یک عدد اول و ریشهٔ اولیهٔ آن عدد) و علاوه بر آن کلید عمومی Diffie-Hellman سِرور می‌باشد (به مبحث مربوط مراجعه شود).
- **Ephemeral Diffie-Hellman**: محتوای پیام شامل سه پارامتر Diffie-Hellman فراهم شده برای anonymous Diffie-Hellman بعلاوهٔ یک امضاء برای آن پارامترهاست.
- **RSA Key exchange که در آن سِرور از RSA استفاده می‌کند اما تنها یک کلید RSA برای امضاء دارد**: در اینجا کلاینت نمی‌تواند به تنهائی یک کلید سرّی که با کلید عمومی سِرور رمزنگاری شده است را ارسال نماید. در عوض سِرور بایستی یک جفت کلید RSA عمومی/خصوصی موقتی خلق کرده و از پیام server_key_exchange برای ارسال کلید عمومی استفاده نماید. محتوای پیام شامل دو پارامتر کلید عمومی موقت RSA (مدول و آرگومان) بعلاوهٔ امضاء آن پارامترهاست.
- **Foretzza**

جزئیات اضافی بیشتری در مورد امضاءها مورد تعهد است. همانند قبل یک امضاء بتوسط محاسبهٔ اندازهٔ hash یک پیام و رمزنگاری آن با کلید خصوصی فرستنده خلق می‌شود. در این مورد hash بصورت زیر تعریف می‌شود

hash(ClientHello.random || ServerHello.random || ServerParams)

بنابراین hash نه تنها پارامترهای Diffie-Hellman یا RSA را می‌پوشاند بلکه دو nonce از پیام‌های اولیهٔ hello را هم پوشش می‌دهد. این امر از حملات بازخوانی و یا نمایش غلط جلوگیری می‌کند. در مورد یک امضاء DSS، hash با استفاده از الگوریتم SHA-1 انجام می‌شود. در مورد یک امضاء RSA، هم hash مربوط به MD5 و هم hash مربوط به SHA-1 محاسبه می‌شود و جمع رشته‌ای این دو hash (۳۶ بایت) بتوسط کلید خصوصی سِرور رمزنگاری می‌شود.

در مرحلهٔ بعد یک سِرور غیرناشناس (سِروری که از anonymous Diffie-Hellman استفاده نمی‌کند)، می‌تواند از کلاینت درخواست یک گواهی‌نامه نماید. پیام certificate_request_message شامل دو پارامتر است: certificate_type و certificate_authorities. certificate_type نشان‌دهندهٔ الگوریتم رمزنگاری کلید- عمومی و کاربردهای آن است:

- RSA، فقط امضاء
- DSS، فقط امضاء
- RSA برای fixed Diffie-Hellman. در این مورد امضاء فقط برای اعتبارسنجی، با ارسال یک گواهی که بتوسط RSA امضاء شده است، بکار می‌رود
- DSS برای fixed Diffie-Hellman که باز هم فقط برای اعتبارسنجی استفاده می‌شود
- RSA برای ephemeral Diffie-Hellman
- DSS برای ephemeral Diffie-Hellman
- Fortezza

پارامتر دوم در پیام certificate_request یک لیست از نام‌های شناخته شده از CAهای مورد پذیرش است.

پیام آخر فاز دوم که همیشه مورد نیاز است، **پیام server_done** است که بتوسط سِرور ارسال می‌شود تا نشان دهد که پایان hello سِرور و پیام‌های مرتبط با آن است. پس از ارسال این پیام، سِرور منتظر پاسخ کلاینت می‌شود. پیام server_done شامل هیچ پارامتری نیست.

فاز سوم – تصدیق هویت کلاینت و مبادلهٔ کلید

پس از دریافت پیام server_done، کلاینت بایستی اگر لازم است تحقیق کند که سِرور یک گواهی معتبر را ارسال نموده و همچنین کنترل نماید که پارامترهای server_hello قابل قبول‌اند. اگر همه چیز بر وفق مراد باشد، کلاینت یک یا چند پیام را بسمت سِرور خواهد فرستاد.

اگر سِرور درخواست یک گواهی‌نامه نموده باشد، کلاینت این فاز را با ارسال یک **پیام certificate** آغاز می‌کند. اگر هیچ گواهی مناسبی وجود نداشته باشد، کلاینت بجای آن یک هشدار no_certificate را ارسال خواهد کرد.

سپس در این فاز **پیام client_key_exchange** بایستی ارسال گردد. محتوای پیام وابسته به نوع مبادلهٔ کلید بشرح زیر است:

- **RSA**: کلاینت یک دیتای ۴۸- بایتی *pre-master sercert* را تولید کرده و آن را با کلید عمومی گواهی‌نامهٔ سِرور و یا کلید موقت RSA از پیام server_key_exchange رمزنگاری می‌کند. کاربرد آن برای محاسبهٔ یک *master secret* بعداً توضیح داده خواهد شد.
- **Ephemeral or Anonymous Diffie-Hellman**: پارامترهای عمومی Diffie-Hellman کلاینت ارسال می‌شوند.
- **Fixed Diffie-Hellman**: پارامترهای عمومی Diffie-Hellman کلاینت در یک گواهی‌نامه ارسال شده بود و بنابراین محتویات این پیام خالی است.
- **Fortezza**: پارامترهای Fortezza کلاینت ارسال می‌شوند.

بالاخره در این فاز، کلاینت ممکن است یک **پیام certificate_verify** ارسال نماید تا تأئید صریح یک گواهی کلاینت را فراهم کند. این پیام تنها بدنبال هر گواهی‌نامهٔ کلاینت که دارای قابلیت امضاء باشد ارسال می‌شود (یعنی تمام گواهی‌نامه‌ها بجز آنهائی که دارای پارامترهای fixed Diffie-Hellman هستند). این پیام یک کُد hash را که مبتنی بر پیام‌های قبل است امضاء می‌کند و چنین تعریف می‌شود:

CertificateVerify.signature.md5_hash

MD5(master_secret || pad_2 || MD5(handshake_messages || master_secret || pad_1));

Certificate.signature.sha_hash

SHA(master_secret || pad_2 || SHA(handshake_messages || master_secret || pad_1));

که در آن pad_1 و pad_2 مقادیری هستند که قبلاً برای MAC تعریف شده‌اند. handshake_messages به تمام پیام‌های پروتکل Handshake که در شروع client_hello ارسال و دریافت شده‌اند (ولی شامل این پیام نیستند) اشاره می‌کند، و master_secret یک مقدار سرّی محاسبه شده است که نحوهٔ ساخت آن را بعداً در همین بخش خواهیم دید. اگر کلید خصوصی کاربر DSS باشد، آنگاه از آن برای رمزنگاری hash مربوط به SHA-1 استفاده خواهد شد. اگر کلید خصوصی کاربر RSA باشد، از آن برای رمزنگاری جمع رشته‌ای hashهای مربوط به MD5 و SHA-1 استفاده می‌شود. در هریک از دو مورد، مقصود تأئید مالکیت کلید خصوصی کلاینت در گواهی کلاینت است. حتی اگر کسی از گواهی‌نامهٔ کلاینت سوء استفاده نماید، او قادر نخواهد بود که این پیام را ارسال نماید.

فاز چهارم - خاتمه

این فاز برقراری یک اتصال امن را کامل می‌کند. کلاینت یک **پیام change_cipher_spec** را ارسال کرده و CipherSpec موقت را در وضع فعلی CipherSpec کپی می‌کند. توجه شود که این پیام بعنوان بخشی از پروتکل Handshake تلقی نشده بلکه با استفاده از پروتکل Change Cipher Spec ارسال می‌شود. بعد از آن کلاینت بدون فوت وقت **پیام finished** را تحت الگوریتم‌های جدید، کلیدها و مقادیر سرّی ارسال می‌کند. پیام finished تأئید می‌کند که مبادلهٔ کلید و مراحل اعتبارسنجی موفقیت‌آمیز بوده است. محتویات پیام finished جمع رشته‌ای دو مقدار hash زیر است:

MD5(master_secret || pad2 || MD5(handshake_messages|| Sender || master_secret || pad1))

SHA(master_secret || pad2 || SHA(handshake_messages || Sender || master_secret || pad1))

که در آن Sender کُدی است که مشخص می‌کند ارسال‌کننده کلاینت می‌باشد و handshake_messages کلیهٔ داده‌های مربوط به پیام‌های handshake تا این پیام، ولی نه شامل این پیام، است.

در پاسخ به این دو پیام، سرور پیام change_cipher_spec خود را ارسال کرده و مقادیر موقت را به CipherSpec انتقال می‌دهد. سرور سپس پیام finished را ارسال می‌کند. در این هنگام، دستداد کامل شده و کلاینت و سرور می‌توانند به مبادلهٔ داده‌های لایهٔ کاربرد اقدام نمایند.

محاسبات رمزنگاری

دو مقولهٔ جالب توجه دیگر وجود دارند: خلق یک master secret مشترک بتوسط مبادلهٔ کلید و تولید پارامترهای رمزنگاری از روی master secret.

خلق Master Secret

master secret مشترک، یک مقدار ۴۸- بایتی (۳۸۴ بیت) یکبار مصرف است که برای این اجلاس بتوسط مبادلهٔ امن کلید تولید شده است. خلق این مقدار در دو مرحله انجام می‌شود. در ابتدا یک pre_master_secret مبادله می‌شود. سپس یک master_secret بتوسط هر دو طرف محاسبه می‌گردد. برای مبادلهٔ pre_master_secret دو امکان وجود دارد

- **RSA**: یک pre_master_secret ۴۸- بایتی بتوسط کلاینت تولید شده، بتوسط کلید RSA سرور رمزنگاری شده و برای سرور ارسال می‌شود. سرور متن رمزشده را با استفاده از کلید خصوصی خود رمزگشائی کرده تا pre_master_secret را بازیابی نماید.
- **Diffie_Hellman**: هر دوی کلاینت و سرور یک کلید عمومی Diffie-Hellman تولید می‌کنند. پس از اینکه این کلیدهای عمومی رد و بدل شدند، هریک از دو طرف محاسبات Diffie-Hellman را برای خلق pre_master_secret انجام می‌دهند.

دو طرف master_secret را چنین محاسبه می‌کنند:

```
master_secret = MD5(pre_master_secret || SHA('A' || pre_master_secret ||
                    ClientHello.random || ServerHello.random)) ||
                MD5(pre_master_secret || SHA('BB' || pre_master_secret ||
                    ClientHello.random || ServerHello.random)) ||
                MD5(pre_master_secret || SHA('CCC' || pre_master_secret ||
                    ClientHello.random  || ServerHello.random))
```

که در آن ClientHello.random و ServerHello.random دو مقدار nonce هستند که در پیام‌های hello اولیهٔ مبادله شده‌اند.

تولید پارامترهای رمزنگاری

CipherSpec نیاز به یک client write MAC secret، یک server write MAC secret، یک client write key، یک server write key، یک client write IV و یک server write IV دارد که از روی master secret بهمان ترتیب محاسبه می‌شوند. این پارامترها از روی master secret و با محاسبهٔ hash آن در دنباله‌ای از بایت‌های امن با طول مناسب برای همهٔ پارامترهای لازم تولید می‌شوند.

تولید اقلام کلید از master secret از همان فرمت تولید master secret از pre-master secret تبعیت می‌کند:

```
Key_block = MD5(master_secret || SHA('A' || master_secret ||
                ServerHello.random || ClientHello.random)) ||
            MD5(master_secret || SHA('BB' || master_secret ||
                ServerHello.random || ClientHello.random)) ||
            MD5(master_secret  || SHA('CCC' || master_secret ||
                ServerHello.random || ClientHello.random)) || ...
```

تا اینکه خروجی کافی تولید شده باشد. نتیجهٔ این ساختار الگوریتمی، یک تابع شبه تصادفی است. master_secret را می‌توان بذر(seed) این تابع تولید اعداد شبه تصادفی دانست. اعداد تصادفی کلاینت و سِرور را می‌توان بعنوان مقادیر salt برای پیچیده‌سازی حملات ممکن به رمز در نظر گرفت (برای بحث در مورد استفاده از salt به فصل ۹ مراجعه شود).

امنیت لایهٔ حمل و نقل (TLS)

TLS برگرفته از استانداردسازی IETF است که هدف اولیهٔ آن تولید نسخهٔ استاندارد اینترنتی SSL بوده است. TLS بعنوان یک استاندارد پیشنهادی اینترنت در RFC 2246 تعریف شده است. RFC 2246 شباهت زیادی به SSLv3 دارد. در این بخش این اختلافات را مورد بررسی قرار می‌دهیم.

Version Number

TLS Record Format همانند SSL Record Format (شکل۴-۷) بوده و میدان‌های موجود در سرآیند همان مفاهیم را دارند. تنها اختلاف در اندازهٔ version می‌باشد. برای نسخهٔ جاری TLS، MajorVersion برابر 3 و MinorVersion برابر 1 است.

کُد اعتبارسنجی پیام (MAC)

دو اختلاف بین روش‌های SSLv3 و TLS MAC وجود دارد: الگوریتم واقعی و منظر محاسبات TLS MAC از الگوریتم HMAC که در RFC 2104 تعریف شده است استفاده می‌کند. از آنچه در فصول قبل گفته شد بخاطر آورید که HMAC چنین تعریف می‌شود:

$$HMAC_K(M) = H[(K^+ \oplus \text{opad}) \| H[(K^+ \oplus \text{ipad}) \| M]]$$

که در آن

H = تابع hash درونی (برای TLS یا MD5 و یا SHA-1 است)

M = پیام ورودی به HAMC

K^+ = کلید سرّی با 0 های اضافه شده به سمت چپ آن بطوری که نتیجه برابر طول بلوک کُد hash باشد (برای MD5 و SHA-1 طول بلوک = ۵۱۲ بیت)

ipad = بایت 0110 0011 (36 هگزادسیمال) که ۶۴ بار تکرار شود (۵۱۲ بیت)

opad = بایت 1100 0101 (5C هگزادسیمال) که ۶۴ بار تکرار شود (۵۱۲ بیت)

SSLv3 از همین الگوریتم استفاده می‌کند بجز اینکه بایت‌های لائی (padding) با کلید سرّی جمع رشته‌ای می‌شود نه اینکه با کلید سرّی که به اندازهٔ طول بلوک طویل شده است، XOR گردد. سطح امنیت هر دو روش تقریباً یکسان است.

برای TLS، محاسبات MAC میدان‌های نشان داده شده در عبارت زیر را شامل می‌شوند:

HMAC_hash(MAC_write_secret,seq_num || TLSCompressed.type ||
TLSCompressed.version || TLSCompressed.length || TLSCompressed.fragment))

محاسبهٔ MAC تمام میدان‌های پوشانیده شده بتوسط SSLv3 را پوشش داده و علاوه بر آن میدان TLSCompressed.version، که نسخهٔ پروتکل بکار گرفته شده است، را نیز می‌پوشاند.

تابع شبه‌تصادفی

TLS از یک تابع شبه‌تصادفی که PRF نامیده می‌شود برای بسط مقادیر سرّی به بلوک‌های دیتا، برای مقاصد تولید کلید یا تأئید کلید، استفاده می‌کند. هدف این است که از یک مقدار سرّی نسبتاً کوچک برای تولید بلوک‌های بزرگتر دیتا بنحوی استفاده شود که از انواع حملاتی که روی توابع hash و MAC می‌شوند در امان باشد. PRF مبتنی بر تابع بسط‌دهندهٔ دیتا بصورت زیر است (شکل ۷- ۷):

P_hash(secret,seed) = HMAC_hash(secret,A(1) || seed) ||
HMAC_hash(secret,A(2) || seed) ||
HMAC_hash(secret,A(3) || seed) || ...

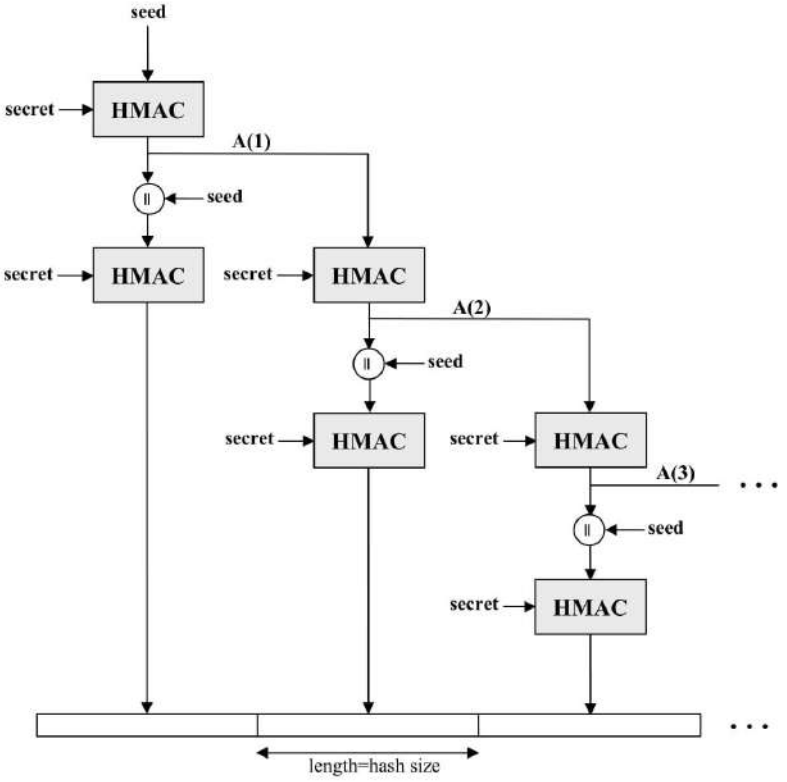
که در آن A( ) چنین تعریف می‌شود

A(0) = seed
A(i) = HMAC_hash(secret,A(i-1))

تابع بسط‌دهندهٔ دیتا از الگوریتم HMAC استفاده می‌کند که MD5 یا SHA-1 تابع hash درونی آن هستند. همانطور که می‌توان مشاهده کرد P_hash می‌تواند هرچند بار که لازم است تکرار گردد تا مقدار لازم دیتا تولید شود. برای مثال اگر از P_SHA-1 برای تولید ۶۴ بایت دیتا استفاده شود، لازم است تا چهاربار تکرار شده تا ۸۰ بایت دیتا را درست کرده که ۱۶ بایت آخر آن معدوم می‌شوند. در همین مثال P_MD5 بایستی چهار بار تکرار گردد که دقیقاً ۶۴ بایت را تولید نماید. توجه شود که هر تکرار شامل دوبار اجرای HMAC است که هرکدام بنوبهٔ خود شامل دوبار اجرای الگوریتم hash مورد استفاده است.

برای اینکه PRF تا حد امکان امن باشد از دو الگوریتم hash بنحوی استفاده می‌کند که در صورت امن ماندن هر یک از دو الگوریتم، امنیت آن تضمین شده باشد. PRF چنین تعریف می‌شود

PRF(secret, label, seed) = P_MD5(S1, label || seed) ⊕ P_SHA-1(S2, label || seed)



شکل ۷-۷ تابع P_hash (secret,seed) پروتکل TLS

PRF بعنوان ورودی یک مقدار سرّی، یک برچسب شناسائی و یک مقدار seed را گرفته و یک خروجی با طول دلخواه را تولید می‌کند. خروجی بتوسط نصف کردن مقدار سرّی به دو نیمهٔ (S1 و S2) و محاسبهٔ P_hash روی هر نیمه با استفاده از MD5 روی یک نیمه و SHA-1 روی نیمهٔ دیگر انجام می‌شود. دو نتیجه باهم XOR شده تا خروجی را درست کند. برای این منظور P_MD5 معمولاً باید بیشتر از P_SHA-1 تکرار شود تا مقدار برابری از دیتا برای ورودی تابع XOR را فراهم آورد.

کُدهای Alert

TLS تمام هشدارهای تعریف شده در SSLv3 بجز no_certificate را حمایت می‌کند. تعدادی هشدارهای اضافی نیز در TLS تعریف می‌شود که از میان آنها کُدهای زیر همیشه fatal هستند:

- **decryption_failed**: یک متن رمزشده که به روش غیرمعتبری رمزگشائی شود. یا طول متن رمزشده مضرب زوجی از طول بلوک نبوده و یا اندازهٔ لائی آن وقتی کنترل شده است صحیح نبوده است.
- **record_overflow**: یک رکورد TLS با بار آن (متن رمزشده)، دریافت شده که طول آن از ۲,۰۴۸ + $۲^{۱۴}$ بایت تجاوز کرده و یا یک متن رمزشده به طولی بیش از ۲,۰۴۸ + $۲^{۱۴}$ بایت رمزگشائی شده است.
- **unknown_ca**: یک زنجیره و یا بخشی از زنجیرهٔ گواهی‌نامه‌ها دریافت شده ولی پذیرفته نشده است، زیرا یا CA صادرکنندهٔ گواهی شناسائی نشده و یا گواهی با یک CA مورد اعتماد شناخته شده مرتبط نبوده است.
- **access_denied**: یک گواهی معتبر دریافت شده است ولی وقتی کنترل دست‌یابی به آن اعمال شده است فرستنده تصمیم گرفته است که به مذاکرات توافقی ادامه ندهد.
- **decode_error**: یک پیام نتوانسته است از کُد خارج شود زیرا یک میدان، از محدودهٔ تعیین شده خارج گشته و یا طول پیام ناصحیح بوده است.
- **export_restriction**: یک توافق که متناقض با محدودیت‌های طول کلید است، تشخیص داده شده است.
- **protocol_version**: نسخهٔ پروتکلی که کلاینت برای توافق روی آن تلاش می‌کند شناخته شده ولی مورد حمایت نیست.
- **insufficient_security**: این هشدار بجای handshake_failure وقتی ارسال می‌شود که سِرور به رمزهائی امن‌تر از آنچه بتوسط کلاینت حمایت می‌شود نیاز دارد.
- **internal_error**: یک خطای داخلی، غیرمرتبط با واحد نظیر و یا غیرمرتبط با صحت پروتکل، امکان ادامهٔ کار را نمی‌دهد.

بقیهٔ هشدارهای جدید چنین‌اند:

- **decrypt_error**: یکی از عملیات رمزنگاری دستداد موفق نبوده است که می‌تواند ناتوانی در تأئید یک امضاء، ناتوانی در رمزگشائی یک مبادلهٔ کلید و یا ناتوانی در تأئید اختتام موفقیت‌آمیز یک پیام باشد.
- **user_canceled**: این دستداد به دلیلی غیرمرتبط با مشکلات پروتکلی، ملغی شده است.
- **no_renegotiation**: بتوسط یک کلاینت در پاسخ به یک hello_request و یا بتوسط یک سِرور در پاسخ به client_hello پس از دستداد اولیه ارسال می‌شود. هریک از این دو پیام قاعدتاً منجر به توافق مجدد خواهد شد اما این هشدار نشان می‌دهد که فرستنده قادر به توافق مجدد نیست. این پیام همیشه یک اخطار است.

مجموعه‌های رمزنگاری

چند اختلاف کوچک بین مجموعه‌های رمزنگاری موجود تحت SSLv3 و TLS وجود دارد:

- **مبادلهٔ کلید**: TLS تمام تکنیک‌های مبادلهٔ کلید SSLv3 بجز Fortezza را می‌پذیرد.
- **الگوریتم‌های رمزنگاری متقارن**: TLS شامل تمام الگوریتم‌های رمزنگاری متقارن موجود در SSLv3 بجز Fortezza است.

انواع گواهی‌های کلاینت

TLS انواع گواهی‌های زیر که می‌تواند در یک پیام certificate_request تقاضا شود را تعریف می‌کند: rsa_sign، dss_sign، rsa_fixed_dh، dss_fixed_dh. همه اینها در SSLv3 تعریف شده‌اند. علاوه بر آن SSLv3 شامل rsa_ephemeral_dh، dss_ephemeral_dh و fortezza_kea می‌باشد. Ephemeral Diffie-Hellman امضاء پارامترهای Diffie-Hellman با RSA یا DSS است. در TLS از rsa_sign و dss_sign برای این منظور استفاده می‌شود و یک نوع امضاء مجزا برای امضاء پارامترهایDiffie-Hellman مورد نیاز نیست. TLS روش Fortezza را شامل نمی‌شود.

پیام‌های Certificate_Verify و Finished

در پیام certificate_verify مربوط به TLS، hashهای MD5 و SHA-1 تنها روی پیام‌های handshake_messages انجام می‌شود. بیاد آورید که برای SSLv3 محاسبات hash همچنین شامل master secret و padها هم بودند. احساس شده است که این میدان‌های اضافی چیزی به امنیت اضافه نمی‌کنند.

همانند پیام finished در SSLv3، پیام finished در TLS یک مقدار hash مبتنی بر master secret مشترک، پیام‌های قبلی handshake و یک برچسب است که کلاینت یا سِرور را مشخص می‌نمایند. محاسبات قدری متفاوت‌اند. برای TLS داریم:

PRF(master_secret,finished_label,MD5(handshake_messages) || SHA-1(handshake_messages))

که در آن finished_label دنبالهٔ "client finished" برای کلاینت و "server finished" برای سِرور است.

محاسبات رمزنگاری

pre_master_secret برای TLS بهمان صورت SSLv3 محاسبه می‌شود. همانند SSLv3، master_secret در TLS بصورت یک تابع hash با ورودی‌های pre_master_secret و دو عدد تصادفی hello محاسبه می‌گردد. فرم محاسبات TLS نسبت به SSLv3 متفاوت بوده و چنین تعریف می‌شود:

```
master_secret =
    PRF(pre_master_secret, "master secret ", ClientHello.random || ServerHello.random)
```

الگوریتم آنقدر اجرا می‌شود تا ۴۸ بایت شبه‌تصادفی خروجی تولید شود. محاسبات اقلام کلید (MAC secret keys، session encryption keys، و IVها) بصورت زیر تعریف می‌شوند:

```
key_block =
        PRF(master_secret, "key expansion",
            SecurityParameters.server_random  || SecurityParameters.client_random)
```

تا خروجی کافی تولید شده باشد. همانند SSLv3 ، key_block تابعی از master_secret و اعداد تصادفی کلاینت و سِرور است ولی برای TLS الگوریتم‌ها متفاوت‌اند.

لائی (padding)

در SSL، لائی اضافه شده قبل از رمزنگاری دیتای کاربر، حداقل مقدار لازم برای کامل کردن اندازهٔ کل دیتائی که باید رمز شود، به مضربی از طول بلوک رمز است. در TLS، لائی می‌تواند هر مقدار حداکثر تا ۲۵۵ بایت باشد که کل دیتا را مضربی از طول بلوک رمز نماید. بعنوان مثال اگر متن ساده (یا متن فشرده‌سازی شده در صورت استفاده از فشرده‌سازی) بعلاوه MAC بعلاوهٔ بایت مربوط به طول لائی، ۷۹ بایت باشد، آنگاه طول لائی می‌تواند ۱ و ۹ و ۱۷ و غیره تا ۲۴۹ بایت باشد. از یک طول متغیر لائی ممکن است برای پیچیده نمودن حملاتی که مبتنی بر تجزیه و تحلیل طول‌های پیام‌های مبادله شده است، استفاده شود.

۷-۳ معاملهٔ الکترونیکی آمن (Secure Electronic Transaction)

SET یک مدل رمزنگاری و امنیتی برای حفاظت معاملات مرتبط با کارت‌های اعتباری روی اینترنت است. نسخهٔ فعلی آن یعنی SETv1 به درخواست MasterCard و Visa برای ایجاد استانداردهای امنیتی، در فوریهٔ ۱۹۹۶ میلادی، بوجود آمد. شرکت‌های زیادی در تعیین مشخصه‌های اولیهٔ SET دخالت داشتندکه از آن جمله IBM، Microsoft، Netscape، RSA، Terisa و Verisign را می‌توان نام برد. فعالیت‌های مربوط که از سال ۱۹۹۶ آغاز شده بود پس از تست‌های فراوان باعث گردید تا در سال ۱۹۹۸ اولین موج محصولات مرتبط با SET به بازار عرضه گردد.

SET به خودی‌خود یک سیستم پرداخت نیست. بلکه SET مجموعه‌ای از پروتکل‌های امنیتی و فرمت‌هائی است که کاربران را قادر می‌سازد تا زیرساخت موجود پرداخت از طریق کارت‌های اعتباری را، از طریق یک شبکهٔ باز مثل اینترنت بصورت امن انجام دهند. فی‌الجمله SET سه سرویس را فراهم می‌آورد:

- یک کانال ارتباطی امن بین تمام طرفین درگیر در یک معامله ایجاد می‌کند.
- با استفاده از گواهی‌نامه‌های دیجیتال X.509v3، اعتماد را فراهم می‌سازد.
- محرمانگی را تضمین می‌نماید، زیرا اطلاعات تنها در زمان و مکان لازم در اختیار طرفین قرار می‌گیرد.

SET دارای مشخصه‌های پیچیده‌ای است که تعاریف آنها در ماه می ۱۹۹۷ در سه کتاب منتشر گردید:

- **کتاب اول** : توصیف کسب و کار (۸۰ صفحه)
- **کتاب دوم** : راهنمای برنامه‌نویسان (۶۲۹ صفحه)
- **کتاب سوم** : تعریف رسمی پروتکل : (۲۶۲ صفحه)

تمام اینها ۹۷۱ صفحه مشخصات را شامل می‌شود. در مقایسه، مشخصه‌های SSLv3 در ۶۳ صفحه و مشخصه‌های TLS در ۸۰ صفحه قرار دارند. بهمین مناسبت تنها خلاصه‌ای از این مشخصات همه جانبه در این بخش آورده شده است.

مروری بر SET

روش مناسبی برای شروع بحث در مورد SET این است که به انتظارات کسب و کار از SET، مشخصه‌های کلیدی آن و افراد درگیر در یک گردش کاری SET نگاهی بیندازیم.

انتظارات

کتاب اول از مشخصه‌های SET، نیازهای تجاری معاملات امن با کارت‌های اعتباری روی اینترنت و سایر شبکه‌ها را بصورت زیر به لیست درآورده است:

- **ایجاد محرمانگی در اطلاعات مربوط به سفارش کالا و پرداخت پول:** لازم است دارندگان کارت‌های اعتباری را مطمئن نمود که این نوع اطلاعات آنان محرمانه مانده و تنها در دسترس گیرنده معین خاص قرار می‌گیرد. محرمانه‌سازی اطلاعات همچنین خطر تقلب از سوی هر یک از طرفین معامله و یا شخص ثالث بداندیش را کاهش می‌دهد. برای محرمانه‌کردن اطلاعات، SET از رمزنگاری استفاده می‌کند.
- **اطمینان از صحت داده‌های انتقال یافته:** یعنی بایستی اطمینان داده شود که در خلال انتقال پیام‌های SET هیچگونه تغییری در محتوای داده‌ها بوجود نمی‌آید. برای حفظ صحت اطلاعات از امضاء دیجیتال استفاده می‌شود.
- **احراز هویت بمنظور اطمینان از اینکه یک دارندهٔ کارت اعتباری، صاحب قانونی آن است:** مکانیسمی که دارندهٔ کارت را به یک شماره حساب مشخص مرتبط می‌سازد، از موارد تقلب و هزینهٔ تمام شدهٔ پردازش پرداخت‌ها جلوگیری می‌کند. از امضاءهای دیجیتال و گواهی‌نامه‌ها برای تأئید این مطلب که دارندهٔ کارت همان صاحب قانونی یک حساب معتبر است استفاده می‌شود.
- **اعتبارسنجی یک فروشنده برای اینکه مشخص شود که او از طریق یک مؤسسهٔ مالی و اعتباری قادر به پذیرش معاملات کارت اعتباری است:** این مورد مکمل مورد قبل است. دارندگان کارت لازم است بتوانند بازرگانانی را که می‌خواهند با آنها معاملات امن داشته باشند شناسائی نمایند. بازهم در اینجا از امضاء دیجیتال و گواهی‌نامه‌ها استفاده می‌شود.
- **اطمینان استفاده از بهترین عملیات امنیتی و تکنیک‌های طراحی سیستم برای حفاظت همهٔ طرف‌های قانونی درگیر در یک معاملهٔ تجاری الکترونیک:** SET مجموعه‌ای است که بر مبنای الگوریتم‌ها و پروتکل‌های رمزنگاری بسیار امن بنا شده و امتحان خود را در این زمینه پس داده است.
- **خلق یک پروتکل که نه به مکانیسم‌های امنیت حمل و نقل وابسته بوده و نه از استفاده چنین مکانیسم‌هائی جلوگیری نماید:** SET می‌تواند بصورت امن روی یک پشته TCP/IP «خام» سوار شود. با وجود این SET در صورت استفاده از مکانیسم‌های امنیتی دیگر مثل IPSec و SSL/TLS دخالتی در کار آنها نمی‌کند.
- **تعامل بین نرم‌افزار و شبکه را تسهیل و تشویق نماید:** فرمت‌ها و پروتکل‌های SET مستقل از نوع سخت‌افزار، نوع سیستم عامل، و نرم‌افزار وِب می‌باشند.

مشخصه‌های کلیدی SET

برای رفع انتظاراتی که در بالا بیان گردید، SET تسهیلات زیر را فراهم کرده است:

- **محرمانگی اطلاعات:** اطلاعات مربوط به حساب و پرداخت‌های صاحب کارت در عبور از عرض شبکه امن می‌مانند. یک جنبهٔ قابل توجه و مهم SET این است که اجازه نمی‌دهد فروشنده از شمارهٔ کارت اعتباری خریدار مطلع گردد و این اطلاعات فقط به بانک صادرکنندهٔ کارت عرضه می‌شود. برای محرمانه‌سازی اطلاعات از رمزنگاری مرسوم DES استفاده می‌شود.

- **صحت داده‌ها**: اطلاعات پرداخت که از سوی خریدار برای فروشنده ارسال می‌شود شامل اطلاعات درخواست کالا، داده‌های شخصی و دستورات پرداخت است. SET تضمین می‌نماید که محتویات این پیام‌ها در عبور از شبکه عوض نشوند. امضاءهای دیجیتال RSA با استفاده از کُدهای hash نظیر SHA-1، صحت پیام را فراهم می‌آورد. علاوه بر آن برخی پیام‌ها بتوسط HMAC و با استفاده از SHA-1 نیز حفاظت می‌گردند.
- **اعتبارسنجی حساب دارندهٔ کارت**: SET فروشندگان را قادر می‌سازد تا تحقیق نمایند که دارندهٔ کارت، یک کاربر قانونی و صاحب یک حساب کارتی معتبر است. SET از گواهی‌های دیجیتال X.509v3 که برای این منظور دارای امضاءهای RSA هستند استفاده می‌کند.
- **تأئید هویت فروشنده**: SET دارندگان کارت‌های اعتباری را قادر می‌سازد تا هویت فروشنده را از این نظر که با یک مؤسسهٔ مالی مجاز به پذیرش پرداخت‌های کارتی در رابطه است، تعیین نمایند. SET برای این کار از گواهی‌های دیجیتال X.509v3 که برای این منظور دارای امضاءهای RSA هستند استفاده می‌کند.

توجه کنید که برخلاف IPSec و SSL/TLS، SET برای هر الگوریتم رمزنگاری فقط یک انتخاب دارد. این یک امر منطقی بوده زیرا SET یک کاربرد منفرد با مجموعهٔ خاصی از نیازهاست در حالی که IPSec و SSL/TLS برای حمایت از کاربردهای متعددی طراحی شده‌اند.

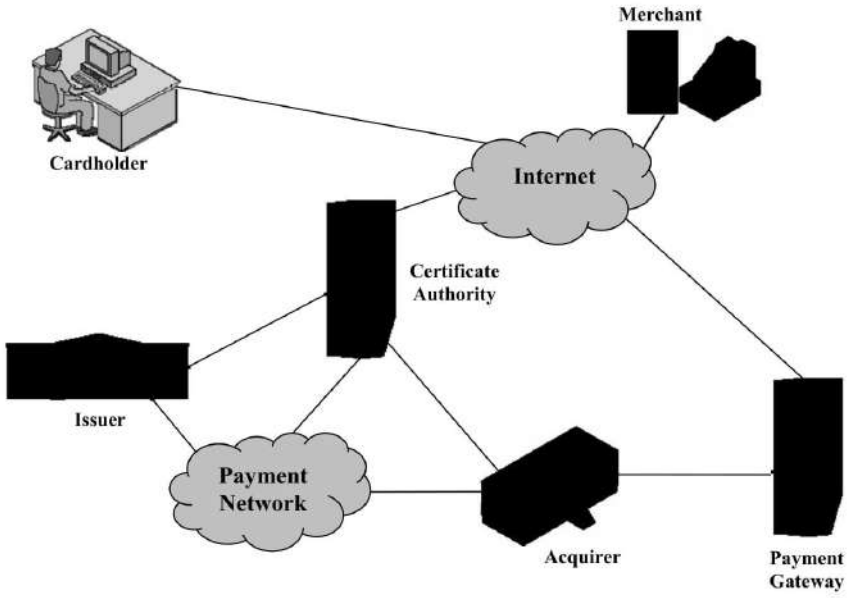
طرف‌های درگیر در SET

شکل ۸-۷ طرف‌های درگیر در یک سیستم SET را نشان می‌دهد که شامل عناصر زیراند:

- **دارندهٔ کارت**: در یک محیط الکترونیکی، مصرف‌کنندگان و خریداران با فروشندگان و تاجران از طریق کامپیوترهای شخصی متصل به اینترنت مرتبط‌اند. یک دارندهٔ کارت قاعدتاً صاحب قانونی یک کارت اعتباری (مثل Visa یا MasterCard) است که از طرف یک مؤسسهٔ مالی صادر شده است.
- **فروشنده یا تاجر**: فروشنده یک شخص و یا یک سازمان است که کالا و خدمات را برای فروش به دارندگان کارت عرضه می‌نماید. معمولاً این کالاها و سرویس‌ها از طریق یک وِب سایت و یا از طریق پست الکترونیک عرضه می‌شوند. فروشنده‌ای که پرداخت‌های کارتی را قبول می‌کند بایستی با یک مباشر ارتباط داشته باشد.
- **صادرکنندهٔ کارت**: این یک مؤسسهٔ مالی، مثل بانک، است که کارت اعتباری را برای دارندهٔ کارت صادر می‌کند. معمولاً تقاضای بازکردن چنین حساب‌های اعتباری یا حضوراً و یا از طریق پست صورت می‌پذیرد. در نهایت این صادرکنندهٔ کارت است که مسئول بازپرداخت دیون دارندهٔ کارت خواهد بود.
- **مباشر (Acquirer)**: این یک مؤسسهٔ مالی است که حسابی را با یک فروشنده برقرار کرده و مسئولیت پرداخت‌ها و تأئید حساب‌های اعتباری را بعهده دارد. تجار معمولاً بیش از یک نوع کارت اعتباری را می‌پذیرند ولی نمی‌خواهند تا با سازمان‌های متعدد بانکی و یا صادرکنندگان کارت‌های مختلف در تماس باشند. مباشر این اطمینان را برای فروشنده بوجود می‌آورد که حساب یک کارت عرضه شده معتبر و فعال بوده و میزان خرید صاحب کارت از اعتبار او تجاوز نمی‌نماید. مباشر همچنین موجبات انتقال الکترونیک وجوه پرداخت شده به حساب فروشنده را فراهم می‌آورد. به دنبال آن صادرکنندهٔ کارت مبلغ هزینه شده را بنحوی از طریق یک شبکهٔ پرداخت الکترونیک می‌پردازد.

- **دروازهٔ پرداخت**: این وظیفه‌ای است که بتوسط مباشر و یا شخص ثالث دیگری که پیام‌های مربوط به دریافت و پرداخت فروشنده را فراهم می‌آورد، انجام می‌شود. دروازهٔ پرداخت بین SET و شبکهٔ پرداخت کارتی موجود واسطه‌ای را ایجاد می‌کند که عملیات مجاز پرداخت‌ها را حمایت نماید. فروشنده پیام‌های SET را با دروازهٔ پرداخت رد و بدل کرده و دروازهٔ پرداخت از طریق یک لینک مستقیم و یا شبکه‌ای به سیستم پردازش مالی مباشر متصل است.
- **مقام مسئول گواهی‌کننده (CA)**: واحدی است که برای صدور گواهی‌نامه‌های کلید- عمومی X.509v3 برای دارندهٔ کارت، فروشنده و دروازهٔ پرداخت مورد اعتماد قرار گرفته است. موفقیت SET منوط به حضور یک زیرساخت CA برای این منظور است. همانطور که در فصول قبل بیان شد، یک ساختار سلسله مراتبی CA مورد استفاده قرار گرفته تا طرف‌های درگیر لزومی به تأئید از طرف بالاترین مقام CA را نداشته باشند.

حال بطور مختصر گردش کار یک معاملهٔ الکترونیک را بررسی می‌کنیم. سپس به برخی از جزئیات رمزنگاری SET نگاهی خواهیم انداخت.



شکل ۸-۷ عوامل معاملهٔ الکترونیکی امن

۱- **مشتری یک حساب باز می‌کند.** مشتری یک حساب کارت اعتباری، همانند Visa و یا MasterCard، در یک بانک که پرداخت الکترونیک و SET را حمایت می‌کند باز می‌کند.

۲- **مشتری یک گواهی‌نامه دریافت می‌کند.** پس از تصدیق هویت مشتری به روش مناسب، وی یک گواهی‌نامهٔ دیجیتال X.509v3 که بتوسط بانک امضاء شده است را دریافت می‌دارد. گواهی، کلید عمومی RSA مشتری و تاریخ انقضای آن را تعیین کرده است. این گواهی همچنین یک ارتباط تضمین شده بتوسط بانک بین جفت کلید مشتری و کارت اعتباری او برقرار می‌سازد.

۳- **فروشندگان گواهی‌نامه‌های خود را دارند.** فروشنده‌ای که نوع مخصوصی از کارت اعتباری را می‌پذیرد بایستی دو گواهی‌نامهٔ مختلف برای دو کلید عمومی که در اختیار اوست داشته باشد که یکی برای امضاء پیام‌ها و دیگری برای تبادل کلید است. فروشنده همچنین نیاز به یک نسخه از گواهی کلید- عمومی دروازهٔ پرداخت را دارد.

۴- **مشتری سفارش می‌دهد.** در این مرحله ممکن است لازم باشد که در ابتدا مشتری وِب سایت فروشنده را مرور نموده و قیمت کالا را بدست آورد. سپس او یک لیست از اقلامی که تمایل به خرید آنها را دارد برای فروشنده ارسال کرده و فروشنده در مقابل فرم سفارش کالا که شامل لیست اقلام، قیمت آنها، قیمت کل و شمارهٔ سفارش است را برای او برمی‌گرداند.

۵- **فروشنده تأئید می‌شود.** علاوه بر فرم سفارش، فروشنده یک نسخه از گواهی نامهٔ خود را برای خریدار ارسال می‌کند تا خریدار بتواند تأئید کند که با یک فروشندهٔ مجاز و معتبر روبروست.

۶- **سفارش و پرداخت آن تواماً ارسال می‌گردند.** مشتری سفارش خود و پرداخت مرتبط با آن را برای فروشنده ارسال می‌کند و بهمراه آن گواهی‌نامهٔ مشتری نیز فرستاده می‌شود. سفارش، خرید اقلامی را که در فرم ارسالی فروشنده آمده بود تأئید می‌کند. پرداخت شامل جزئیات مربوط به کارت اعتباری است. اطلاعات پرداخت طوری رمزنگاری می‌شودکه فروشنده قادر به خواندن آنها نباشد. گواهی‌نامهٔ مشتری، فروشنده را قادر می‌سازد تا اعتبار مشتری را تحقیق کند.

۷- **فروشنده تأئید پرداخت را درخواست می‌کند.** فروشنده اطلاعات پرداخت را به دروازهٔ پرداخت می‌فرستد و از او تأئید می‌گیرد که اعتبار مشتری برای خرید فعلی کافی است.

۸- **فروشنده سفارش را تأئید می‌کند.** فروشنده تأئید سفارش را برای مشتری ارسال می‌کند.

۹- **فروشنده محصول و یا سرویس را فراهم می‌آورد.** فروشنده،کالا را برای مشتری ارسال کرده و یا سرویس درخواستی او را فراهم می‌سازد.

۱۰- **فروشنده درخواست پرداخت می‌نماید.** این درخواست برای دروازهٔ پرداخت ارسال شده و تمام پردازش‌های مرتبط با پرداخت در آنجا انجام می‌شود.

امضاء دوگانه (Dual Signature)

قبل از اینکه به جزئیات پروتکل SET بپردازیم، اجازه دهید تا به یک نوآوری مهم که در SET فراهم آورده شده و به امضاء دوگانه معروف است اشاره نمائیم. هدف امضاء دوگانه این است که دو پیام که به مقصد دو دریافت کنندهٔ متفاوت ارسال

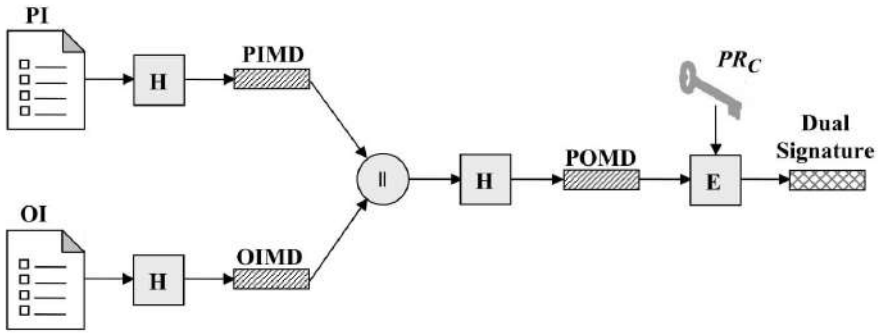
می‌شوند را بهم پیوند دهد. در این مورد، مشتری می‌خواهد که اطلاعات مربوط به سفارش کالا OI (Order Information) را برای فروشنده، و اطلاعات مربوط به قیمت آن کالا PI (Payment Information) را برای بانک ارسال نماید. فروشنده لازم نیست که شمارهٔ کارت اعتباری خریدار را بداند، و بانک هم نیازی به دانستن جزئیات سفارش کالای خریدار ندارد. از نظر خصوصی ماندن و حفاظت اطلاعات، بهتر است که این دو قلم از هم جدا باشند. از سوی دیگر این دو قلم اطلاعات بایستی طوری باهم مرتبط شوند که اگر لازم شد بتوان دعاوی آینده را پاسخ داد. این پیوند بایستی طوری باشد که مشتری بتواند اثبات کند که این پرداخت برای این سفارش، و نه سفارش یا سرویس دیگری بوده است.

برای ملاحظهٔ نیاز به این پیوند، فرض کنید که مشتریان دو پیام برای فروشنده ارسال نمایند- یک OI امضاء شده و یک PI امضاء شده - و فروشنده PI را برای بانک بفرستد. اگر فروشنده بتواند OI دیگری از خریدار دریافت کند، ممکن است ادعا نماید که این OI جدید به همراه PI بوده است و OI قدیم را نادیده انگارد. پیوند ذکرشده از این امر جلوگیری می‌کند.

شکل ۹-۷ استفاده از امضاء دوگانه (DS) برای رفع نیاز فوق‌الذکر را نشان می‌دهد. مشتری hash مرتبط با PI (با استفاده از SHA-1) و hash مرتبط با OI را استخراج می‌کند. این دو hash سپس با هم جمع رشته‌ای شده و hash نتیجهٔ آنها گرفته می‌شود. در انتها مشتری hash نهائی را با کلید خصوصی امضاء خود رمزنگاری کرده و امضاء دوگانه را تولید می‌کند. عملیات را می‌توان چنین خلاصه نمود

$$DS = E(PR_C, [H(H(PI) \| H(OI)])$$

که درآن PR_C کلید خصوصی امضاء مشتری است. حال فرض کنید که فروشنده امضاء دوگانه (DS)، OI و چکیدهٔ پیام برای PI (PIMD) را در اختیار دارد. فروشنده همچنین کلید عمومی مشتری را که در گواهی نامهٔ او ذکر شده است می‌داند. بنابراین فروشنده قادر است تا دو کمیت زیر را محاسبه نماید:



PI = Payment Information
OI = Order Information
H = Hash function (SHA-1)
|| = Concatenation

PIMD = PI message digest
OIMD = OI message digest
POMD = Payment Order message digest
E = Encryption (RSA)
PR_c = Customer's private signature key

شکل ۹-۷ نحوهٔ ساخت امضاء دوگانه

$$D(PU_C, DS) \text{ و } H(PIMD \,||\, H[OI])$$

که در آن PU_C کلید عمومی امضاء مشتری است. اگر این دو کمیت با هم برابر باشند، فروشنده امضاء مشتری را تأئید کرده است. بهمین ترتیب اگر بانک DS، PI، چکیدهٔ پیام برای OI (OIMD) و گواهی‌نامهٔ کلید- عمومی مشتری را داشته باشد آنگاه بانک می‌تواند دو کمیت زیر را حساب کند:

$$D(PU_C, DS) \text{ و } H(H[PI] \,||\, OIMD)$$

بازهم اگر این دو کمیت برابر باشد، آنگاه بانک امضاء مشتری را تأئید کرده است. بطور خلاصه،

۱- فروشنده OI را دریافت کرده و امضاء را گواهی نموده است.
۲- بانک PI را دریافت کرده و امضاء را گواهی نموده است.
۳- مشتری PI و OI را بهم پیوند داده و می‌تواند ارتباط این دو با هم را اثبات کند.

بعنوان مثال، فرض کنید که فروشنده می‌خواهد به نفع خود OI دیگری را برای این پرداخت جا بزند. در این صورت او مجبور خواهد بود تا OI دیگری که تابع hash آن با OIMD موجود یکسان باشد را پیدا کند. با استفاده از SHA-1 این کار امکان‌پذیر نخواهد بود. بنابراین فروشنده نمی‌تواند OI دیگری را به این PI مرتبط سازد.

عملیات پرداخت

جدول ۳-۷ انواع گردش اسناد مرتبط با SET را نشان می‌دهد. در آنچه ذیلاً خواهد آمد، به جزئیات سه گردش مالی زیر خواهیم پرداخت:

- درخواست خرید
- تأئید پرداخت
- برداشت پول

درخواست خرید

قبل از اینکه درخواست خرید ارسال شود، دارندهٔ کارت اعتباری در وب جستجو کرده، کالای مورد نظر خود را انتخاب نموده و سفارش را آماده کرده است. این فاز ابتدائی وقتی پایان می‌یابد که فروشنده یک فرم سفارش کالا (پیش فاکتور) را برای خریدار ارسال کرده باشد. تمام این مراحل بدون استفاده از SET صورت می‌پذیرد.

درخواست خرید شامل چهار پیام است: Initiate Request، Initiate Response، Purchase Request و Purchase Response.

برای ارسال پیام‌های SET به فروشنده، دارندهٔ کارت بایستی یک نسخه از گواهی‌نامه‌های فروشنده و دروازهٔ پرداخت را داشته باشد. خریدار در پیام **Initiate Request** که برای فروشنده ارسال می‌شود گواهی‌نامه‌ها را درخواست می‌نماید. در این پیام نوع کارت اعتباری که خریدار مایل به استفاده از آن است ذکر می‌شود. پیام همچنین شامل یک ID تخصیص داده شده برای این زوج درخواست/ پاسخ و یک nonce برای اطمینان از بهنگام بودن آن است.

جدول ۳-۷ انواع گردش اسناد SET

Cardholder registeration	دارندگان کارت قبل از این که بتوانند پیام‌های SET را برای فروشندگان بفرستند، بایستی در یک CA ثبت نام شده باشند.
Merchant registeration	فروشندگان قبل از این که بتوانند پیام‌های SET را با مشتریان و دروازه‌های پرداخت مبادله نمایند، بایستی در یک CA ثبت نام شده باشند.
Purchase Request	پیامی که از جانب مشتری برای فروشنده ارسال شده و شامل OI برای فروشنده و PI برای بانک است.
Payment authorization	مبادلهٔ بین فروشنده و دروازهٔ پرداخت تا مقدار مشخصی پول برای یک خرید را از طریق یک کارت اعتبار بخشد.
Payment capture	به فروشنده اجازه می‌دهد تا از دروازهٔ پرداخت تقاضای پول نماید.
Certificate inquiry and status	اگر CA نتواند یک درخواست گواهی را سریعاً پردازش نموده و پاسخ دهد، با ارسال یک پیام به دارندهٔ کارت اعتباری و یا فروشنده خبر می‌دهد که بعداً تماس بگیرند. دارندهٔ کارت یا فروشنده با ارسال پیام *Certificate Inquiry* از وضعیت تقاضای خود با خبر شده و اگر درخواست آنان تأئید شده باشد گواهی را دریافت خواهند کرد.
Purchase inquiry	به دارندهٔ کارت اجازه می‌دهد تا پس از دریافت پاسخ خرید، از وضعیت سفارش خود مطلع گردد. توجه شود که این پیام شامل اطلاعاتی همانند کالای فراهم شده نیست بلکه نشان‌دهندهٔ اعتبارسنجی، برداشت پول و پردازش‌های اعتباری است.
Authorization reversal	به یک فروشنده اجازه می‌دهد تا درخواست‌های قبلی خود را تصحیح کند. اگر سفارش کامل نشود، فروشنده تمام اعتبارسنجی را از سر خواهد گرفت. اگر بخشی از سفارش کامل نگردد، فروشنده آن بخش را از سر خواهد گرفت.
Capture reversal	به یک فروشنده اجازه می‌دهد تا اشتباهات احتمالی در یک درخواست پرداخت پول را که ممکن است ناشی از خطای یک منشی باشد تصحیح کند.
Credit	به یک فروشنده اجازه می‌دهد تا در صورت برگشت کالا و یا مثلاً فاسد شدن آن پولی را به حساب دارندهٔ کارت واریز نماید. توجه شود که پیام *Credit* در SET همیشه از سوی فروشنده و نه از سوی دارندهٔ کارت ارسال می‌شود. تمام ارتباطات بین دارندهٔ کارت و فروشنده که منجر به پردازش اعتبار می‌گردد خارج از SET واقع می‌شود.
Credit reversal	به یک فروشنده اجازه می‌دهد تا یک اعتبار درخواست شدهٔ قبلی را تصحیح نماید.
Payment gateway certificate request	به یک فروشنده اجازه می‌دهد تا از دروازهٔ پرداخت استعلام کرده و یک کپی از مبادلهٔ کلید جاری دروازه و گواهی امضاءها را دریافت کند.
Batch administration	به یک فروشنده اجازه می‌دهد تا اطلاعات مرتبط با معاملات او را به دروازهٔ پرداخت بفرستد.
Error message	نشان می‌دهد که یک پاسخ‌دهنده بعلت اشتباه در فرمت و یا اعتبار یک پیام از پاسخ به آن خودداری نموده است.

فروشنده یک پاسخ را تهیه کرده و آن را با کلید خصوصی امضاء خود امضاء می‌کند. پاسخ شامل nonce مشتری، nonce دیگری برای مشتری در ارسال پیام بعدی و یک ID معاملاتی مربوط به این خرید است. علاوه بر پاسخ امضاء شده، پیام **Initiate Response** شامل گواهی‌نامه امضاء فروشنده و گواهی‌نامهٔ مبادلهٔ کلید دروازهٔ پرداخت است.

دارندهٔ کارت، گواهی‌نامه‌های فروشنده و دروازه را بتوسط امضاءهای CA مرتبط با آنها تأئید کرده و آنگاه OI و PI را فراهم می‌آورد. ID معامله که بتوسط فروشنده به این معامله تخصیص یافته است، هم در OI و هم در PI منظور خواهند شد. OI بطور صریح داده‌های مربوط به سفارش مانند تعداد اقلام و قیمت هرقلم را ذکر نمی‌کند، بلکه به یک شمارهٔ سفارش اشاره می‌نماید که قبلاً در مبادلات بین خریدار و فروشنده (پیش فاکتور) ذکر شده است (قبل از اینکه وارد اولین پیام SET شویم). در مرحلهٔ بعد دارندهٔ کارت پیام **Purchase Request** (شکل ۱۰-۷) را تولید و ارسال می‌کند. برای این منظور دارندهٔ کارت یک کلید رمزنگاری متقارن یکبارمصرف K_S را تولید می‌کند. پیام شامل اطلاعات زیر است:

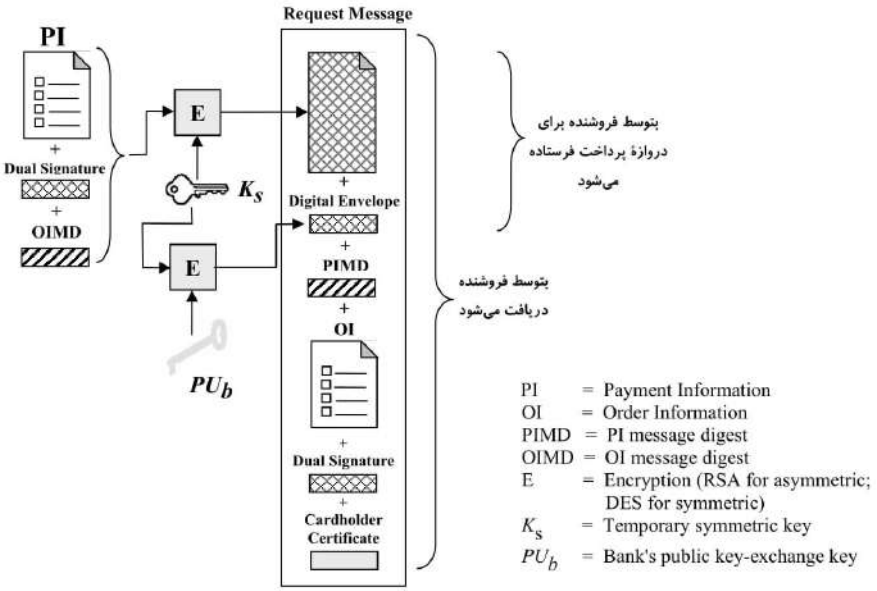
۱- **اطلاعات مرتبط با خرید**. این اطلاعات از طرف فروشنده به سمت دروازهٔ پرداخت ارسال شده و شامل اقلام زیر است

- PI
- امضاء دوگانه که از OI و PI محاسبه شده و بتوسط کلید خصوصی امضاء مشتری، امضاء شده است.
- چکیدهٔ پیام OI (OIMD)

OIMD بتوسط دروازهٔ پرداخت مورد نیاز بوده تا همانطور که قبلاً بیان شد امضاء دوگانه را تأئید نماید. تمام این اقلام با K_S رمزنگاری می‌شوند. آخرین قلم عبارت است از

- پاکت دیجیتال. این قلم با رمزنگاری K_S بتوسط کلید مبادلهٔ کلید- عمومی دروازهٔ پرداخت انجام می‌شود. آن را پاکت دیجیتال گویند زیرا قبل از اینکه سایر اقلامی که قبلاً به آنها اشاره شد خوانده شوند بایستی این پاکت باز شود (رمزگشائی شود).

اندازهٔ K_S در اختیار فروشنده قرار نمی‌گیرد. بنابراین فروشنده نمی‌تواند هیچ یک از اطلاعات مربوط به پرداخت را بخواند.



شکل ۱۰-۷ دارندهٔ کارت اعتباری درخواست خرید می‌کند

۲- **اطلاعات مرتبط با سفارش**. این اطلاعات مورد نیاز فروشنده بوده و شامل اقلام زیر است:
- OI
- امضاء دوگانه که از OI و PI محاسبه شده و بتوسط کلید خصوصی امضاء مشتری امضاء شده است.
- چکیدهٔ پیام PI (PIMD)

PIMD مورد نیاز فروشنده است تا بتواند امضاء دوگانه را تأئید نماید. توجه شود که OI بصورت رمزنشده ارسال می‌شود.

۳- **گواهی‌نامهٔ دارندهٔ کارت اعتباری**. این شامل کلید عمومی امضاء دارندهٔ کارت است که هم مورد نیاز فروشنده و هم مورد نیاز دروازهٔ پرداخت است.

وقتی فروشنده پیام Purchase Request را دریافت کرد عملیات زیر را انجام می‌دهد (شکل ۱۱-۷):

۱- گواهی‌نامه‌های دارندهٔ کارت را بتوسط امضاءهای CA تأئید می‌کند.
۲- امضاء دوگانه را با استفاده از کلید عمومی امضاء مشتری تأئید می‌کند. این مسأله این اطمینان را ایجاد می‌کند که سفارش در زمان ترانزیت دستکاری نشده و بتوسط کلید خصوصی امضاء دارندهٔ کارت امضاء شده است.
۳- پردازش مربوط به سفارش را انجام داده و اطلاعات پرداخت را جهت تأئید به دروازهٔ پرداخت می‌فرستد(بعداً توضیح داده خواهد شد).
۴- یک Purchase Response برای دارندهٔ کارت می‌فرستد.

پیام **Purchase Response** شامل یک بلوک پاسخ است که سفارش را تأئید نموده و به شمارهٔ مأخذ معامله اشاره می‌کند. این بلوک بتوسط فروشنده و با استفاده از کلید خصوصی او امضاء می‌شود. بلوک و امضاء آن به همراه گواهی امضاء فروشنده برای خریدار ارسال می‌شود.

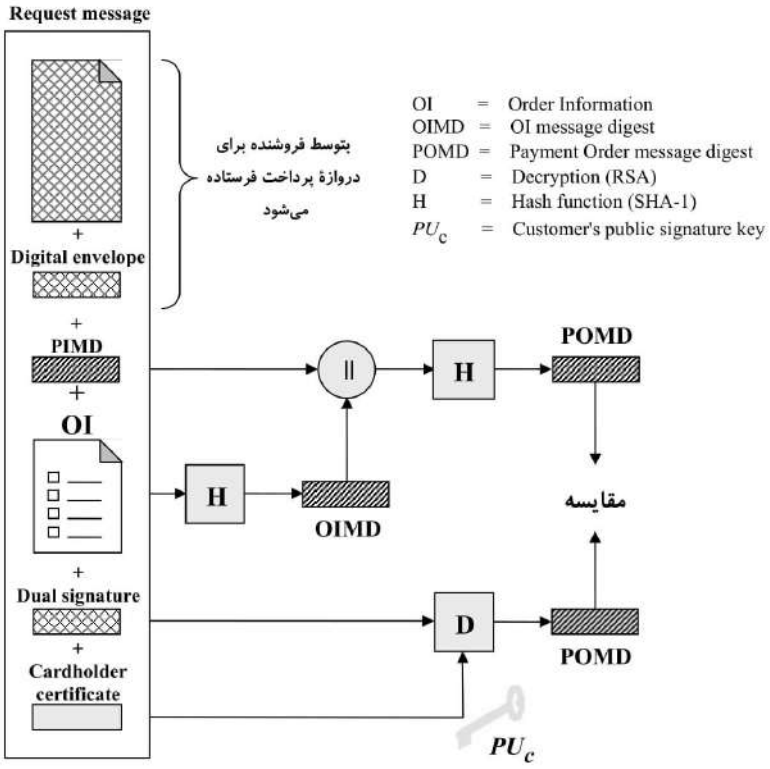
وقتی نرم‌افزار دارندهٔ کارت، پیام پاسخ خرید را دریافت کرد، گواهی‌نامهٔ فروشنده را تأئیدکرده و سپس امضاء بلوک پاسخ را تأئید می‌کند. بالاخره بر اساس پاسخ، عملیاتی همانند نمایش یک پیام برای کاربر و یا بروزرساندن یک پایگاه داده با وضعیت سفارش انجام می‌پذیرد.

تأئید پرداخت

در خلال انجام عملیات مربوط به سفارش کالای یک دارندهٔ کارت، فروشنده ضمن تماس با دروازهٔ پرداخت، معامله را تأئید می‌کند. تأئید پرداخت این اطمینان را ایجاد می‌کند که پرداخت بتوسط صادرکنندهٔ کارت مورد پذیرش است. این تأئید تضمین می‌نماید که فروشنده پول خود را دریافت خواهد کرد و بر این اساس سرویس و یا کالای لازم برای مشتری را فراهم می‌نماید. عملیات تأئید پرداخت شامل دو پیام است. درخواست تأئید و پاسخ تأئید.

فروشنده یک پیام **درخواست تأئید(Authorization Request)** را به دروازهٔ پرداخت می‌فرستدکه شامل اقلام زیر است:

۱- **اطلاعات مرتبط با خرید**. این اطلاعات از مشتری دریافت شده و شامل اقلام زیر است:
- PI
- امضاء دوگانه که از OI و PI محاسبه شده و بتوسط کلید خصوصی امضاء مشتری امضاء شده است.
- چکیدهٔ پیام OI (OIMD)
- پاکت دیجیتال



شکل ۱۱-۷ فروشنده درخواست خرید مشتری را تأئید می‌کند

۲- **اطلاعات مرتبط با تأئید**. این اطلاعات بتوسط فروشنده تولید شده و شامل:

- یک بلوک تأئید که شامل ID معامله است که بتوسط کلید خصوصی امضاء فروشنده امضاء شده و با یک کلید متقارن یکبارمصرف که بتوسط فروشنده تولید شده، رمزنگاری شده است.
- یک پاکت دیجیتال. این قلم با رمزنگاری کلید یکبارمصرف بتوسط کلید مبادلهٔ کلید عمومی دروازهٔ پرداخت تهیه شده است.

۳- **گواهی‌نامه‌ها**. فروشنده گواهی‌نامهٔ کلید امضاء دارندهٔ کارت (که برای امضاء دوگانه بکار می‌رود)، گواهی‌نامهٔ کلید امضاء فروشنده (که برای تأئید امضاء فروشنده بکار می‌رود) و گواهی‌نامهٔ مبادلهٔ کلید فروشنده (که در پاسخ دروازهٔ پرداخت لازم است) را بهمراه اقلام فوق الذکر ارسال می‌دارد.

دروازهٔ پرداخت وظایف زیر را انجام می‌دهد:

۱- همهٔ گواهی‌نامه‌ها تأئید می‌کند.
۲- پاکت دیجیتال بلوک تأئید را رمزگشائی کرده تا کلید متقارن را بدست آورده و آنگاه بلوک تأئید را رمزگشائی می‌نماید.
۳- امضاء فروشنده در بلوک تأئید را، تأئید می‌نماید.
۴- پاکت دیجیتال بلوک پرداخت را رمزگشائی نموده تا کلید متقارن را بدست آورده و آنگاه بلوک پرداخت را رمزگشائی می‌نماید.
۵- امضاء دوگانهٔ بلوک پرداخت را تأئید می‌نماید.
۶- ID معاملهٔ دریافت شده از فروشنده را با آنچه که از طرف مشتری (بطور غیرمستقیم) در PI است مقایسه و در صورت تطبیق تأئید می‌کند.
۷- از صادرکنندهٔ کارت تقاضای تأئید اعتبار نموده و آن را دریافت می‌دارد.

پس از دریافت تأئید از طرف صادرکنندهٔ کارت، دروازهٔ پرداخت یک پیام **پاسخ تأئید (Authorization Response)** را برای فروشنده ارسال می‌کند. این پیام شامل عناصر زیر است:

۱- **اطلاعات مرتبط با تأئید**. شامل یک بلوک تأئید است که بتوسط کلیدخصوصی امضاء دروازه، امضاء شده و بتوسط یک کلید متقارن یکبارمصرف که بتوسط دروازه تولید شده است، رمزنگاری شده است. این کلید همچنین شامل یک پاکت دیجیتال است که شامل کلید یکبارمصرف بوده و با کلید مبادلهٔ کلید عمومی فروشنده رمزنگاری شده است.
۲- **اطلاعات مرتبط با قبض پرداخت**. این اطلاعات در آینده برای امر پرداخت مورد استفاده قرار خواهد گرفت. این بلوک به همان فرم بند (۱) بوده یعنی یک قبض پرداخت شده رمزنگاری شدهٔ امضاءشده بهمراه یک پاکت دیجیتال است. این قبض بتوسط فروشنده مورد پردازش قرار نمی‌گیرد، بلکه بایستی با درخواست پرداخت برگردانده شود.
۳- **گواهی‌نامه**. گواهی‌نامهٔ کلید امضاء دروازهٔ پرداخت.

با تأئید موارد بتوسط دروازهٔ پرداخت، فروشنده می‌تواند کالا را برای خریدار فراهم نموده و یا سرویس مورد نیاز او را تأمین نماید.

برداشت پول

برای دریافت پول، فروشنده دروازهٔ پرداخت را در یک گردش برداشت پول درگیر کرده که شامل یک درخواست برداشت و یک پیام پاسخ برداشت است.

برای پیام **درخواست برداشت(Capture Request)**، فروشنده یک بلوک درخواست برداشت را تولید کرده، امضاء نموده و رمزنگاری می‌کند. این بلوک شامل مبلغ برداشت و ID معامله است. پیام همچنین شامل قبض برداشت رمزشده که قبلاً دریافت شده بود (در پاسخ تأئید) برای این معامله و همچنین کلید امضاء فروشنده و گواهی‌های مبادلهٔ کلید است.

وقتی دروازهٔ پرداخت، پیام درخواست برداشت را دریافت می‌کند، آن را رمزگشائی کرده و بلوک درخواست برداشت را تأئید نموده و بلوک قبض برداشت را رمزگشائی و تأئید می‌نماید. آنگاه تطابق بین درخواست برداشت با قبض برداشت را وارسی می‌نماید. سپس یک درخواست کلِر (اصطلاح بانکی معمول) تولید کرده که از طریق شبکهٔ خصوصی پرداخت، برای صادرکنندهٔ کارت ارسال می‌شود. این تقاضا باعث می‌شود که پول به حساب فروشنده واریز شود.

دروازه پرداخت آنگاه فروشنده را از پرداخت پول بتوسط یک پیام **پاسخ برداشت(Capture Response)** آگاه می‌سازد. پیام شامل یک بلوک پاسخ بوده که دروازه آن را امضاء نموده و رمزنگاری می‌کند. پیام همچنین شامل گواهی‌نامهٔ کلید امضاء دروازه می‌باشد. نرم‌افزار فروشنده، پاسخ برداشت را ذخیره کرده تا در رفع اختلاف با مباشر مورد استفاده قرار گیرد.

۷-۴ منابع مطالعاتی

[RESC01] مرور کاملی بر SSL و TLS را فراهم می‌آورد.

بهترین مأخذ مطالعهٔ جزئیات SET کتاب اول مشخصه‌ها است که در MasterCard SET Web site در دسترس است. [MACG97] نیز مطلب را بصورت عالی بررسی کرده است. [DREW99] نیز یک منبع مطالعاتی خوب است.

DREW99 Drew, G. *Using SET for Secure Electronic Commerce*. Upper Saddle River, NJ: Prentice Hall, 1999.

MACG97 Macgregor, R.; Ezvan, C.; Liguori, L.; and Han, J. *Secure Electronic Transactions: Credit Card Payment on the Web inTheory and Practice*. IBM RedBook SG244978-00, 1997. Available at **www.redbooks.ibm.com**.

RESC01 Rescorla, E. *SSL and TLS: Designing and Building Secure Systems*. Reading, MA: Addison-Wesely, 2001.

وِب سایت‌های مفید

- **Netscape's SSL Page**: شامل مشخصه‌های SSL است.
- **Transport Layer Security Charter**: آخرین RFC ها و پیش نویس‌های اینترنت در مورد TLS.
- **OpenSSL Project**: پروژهٔ تولید نرم‌افزارهای SSL و TLS. سایت شامل اسناد و لینک‌هائی در این مقوله است.

۷-۵ واژه‌های کلیدی، سؤالات مرورکنندهٔ بحث و مسائل

واژه‌های کلیدی

دروازهٔ پرداخت	payment gateway
معاملهٔ الکترونیکی امن	Secure Electronic Transaction (SET)
لایهٔ سوکت امن	Secure Socket Layer (SSL)
امنیت لایهٔ حمل و نقل	Transport Layer Security (TLS)

مباشر	acquirer
دارندهٔ کارت اعتباری	cardholder
مقام مسئول صدور گواهی‌نامه	certification authority (CA)
امضاء دوگانه	dual signature
صادرکننده کارت اعتباری	issuer
تاجر یا فروشندهٔ کالا	merchant

سؤالات مرورکننده بحث

۷-۱ مزایای هریک از سه روش نشان داده شده در شکل ۷-۱ کدامند؟

۷-۲ کدام پروتکل‌ها، SSL را می‌سازند؟

۷-۳ اختلاف بین یک اتصال SSL با یک اجلاس SSL چیست؟

۷-۴ پارامترهائی که حالت اجلاس SSL را تعریف می‌کنند لیست کرده و مختصراً توضیح دهید.

۷-۵ پارامترهائی که حالت اتصال SSL را تعریف می‌کنند لیست کرده و مختصراً توضیح دهید.

۷-۶ پروتکل SSL Record چه سرویس‌هائی را فراهم می‌آورد؟

۷-۷ انتقال پروتکل SSL Record شامل چه قدم‌هائی است؟

۷-۸ گروه‌های اصلی شرکت‌کننده در SET را لیست نموده و مختصراً تعریف کنید.

۷-۹ امضاء دوگانه چیست و اهداف آن کدام است؟

مسائل

۷-۱ در SSL و TLS چرا بجای اینکه تنها یک پیام change_cipher_spec در پروتکل Handshake وجود داشته باشد، یک پروتکل Change Cipher Spec جداگانه فراهم شده است؟

۷-۲ تهدیدهای زیر در مورد امنیت web را در نظر گرفته و توصیف کنید که چگونه با هر یک از آنها بتوسط یکی از مشخصه‌های SSL مقابله می‌شود.

الف- حملهٔ همه جانبهٔ شکستن رمز: یک جستجوی کامل فضای کلید برای یک الگوریتم رمزنگاری متقارن

ب- حملهٔ لغت نامه‌ای با دانستن متن ساده: بسیاری از پیام‌ها شامل متون سادهٔ قابل پیش‌بینی هستند (همانند فرمان HTTP GET). یک مهاجم یک لغت‌نامه که شامل متون رمزشدهٔ همه متن‌های سادهٔ ممکن است را تولید می‌کند. وقتی که یک پیام رمزشده مورد شنود قرار می‌گیرد، مهاجم بخشی را که شامل متن سادهٔ معلوم رمزنگاری شده است گرفته و در لغت نامه به دنبال متن رمزشده می‌گردد. متن رمزشده بایستی با یکی از اقلام موجود در لغت‌نامه که با همان کلید سّری رمزنگاری شده است تطبیق کند. اگر تطبیق در چند مورد صورت پذیرد، هر مورد با کل متن رمزشده مقایسه شده تا نتیجهٔ صحیح به دست آید. این حمله علی‌الخصوص در مورد اندازهٔ کوچک کلیدها (مثلاً کلید ۴۰- بیتی) مؤثر است.

ج- حملهٔ بازخوانی: پیام‌های handshake قدیمی دوباره اجرا شوند.

د - حملهٔ Man-in-the-Middle: یک مهاجم در هنگام مبادلهٔ کلید، خود را در مسیر ارتباطات قرار داده و برای سِرور بصورت کلاینت و برای کلاینت بصورت سِرور ظاهر می‌شود.

هـ- password sniffing: کلمات عبور در HTTP و یا سایر کاربردها مورد استراق سمع قرار می‌گیرد.

و- IP Spoofing: با استفاده از آدرس‌های IP تقلیدی یک میزبان را گول زده تا دیتای عوضی را بپذیرد.

ز- IP Hijacking: یک اتصال فعال معتبر بین دو میزبان، گسسته شده و حمله‌کننده جای یکی از طرفین را اشغال می‌کند.

ح- SYN Flooding: یک مهاجم پیام‌های TCP SYN را ارسال کرده تا یک اتصال را درخواست نماید ولی به پیام انتهائی جواب نمی‌دهد تا اتصال بطور کامل برقرار شود. مدول TCP مورد تهاجم معمولاً حدود چند دقیقه «اتصال نیمه باز» را نگاه می‌دارد. پیام‌های تکراری SYN می‌تواند مدول TCP را مسدود کند.

۷-۳ بر اساس آنچه در این فصل آموخته‌اید، آیا در SSL ممکن است که گیرنده، بلوک‌های SSL record را که خارج از نظم وارد می‌شوند به نظم درآورد. اگر چنین است توضیح دهید که چگونه چنین چیزی ممکن است؟ اگر نه چرا نه؟

فصل ۸

امنیت مدیریت شبکه

۸-۱ **مفاهیم اساسی SNMP**

معماری مدیریت شبکه
معماری پروتکل مدیریت شبکه
پروکسی‌ها
SNMPv2

۸-۲ **تسهیلات جامعه‌ای SNMPv1**

جوامع و نام‌های جوامع
سرویس اعتبارسنجی
خط‌مشی دست‌یابی
سرویس پروکسی

۸-۳ **SNMPv3**

معماری SNMP
پردازش پیام و مدل امنیتی کاربر
کنترل دست‌یابی مبتنی بر منظر

۸-۴ **منابع مطالعاتی**

۸-۵ **واژه‌های کلیدی، سؤالات مرورکنندهٔ بحث و مسائل**

واژه‌های کلیدی
سؤالات مرورکنندهٔ بحث
مسائل

بکه‌ها و سیستم‌های عملیاتی توزیع شده از اهمیت حیاتی و فزاینده‌ای در کسب و کار، دولت و سایر سازمان‌ها برخوردارند. در داخل یک سازمان بخصوص، رَوَند کار به سمت شبکه‌های پیچیده‌تر که کاربردها و کاربران بیشتری را پشتیبانی می‌کنند در حرکت است. همین‌طور که این شبکه‌ها رشد بیشتری می‌یابند، دو واقعیت ملموس‌تر میشود:

- شبکه و منابع مرتبط با آن و کاربردهای توزیع شده از ضروریات لاینفک سازمان می‌شوند.
- رویدادهای بیشتری ممکن است باعث خطا شده و شبکه یا بخشی از شبکه را از کار انداخته و یا عملکرد آن را به سطح غیرقابل قبولی تنزل دهد.

یک شبکهٔ بزرگ نمی‌تواند تنها بتوسط تلاش‌های انسانی، جمع و جور و مدیریت شود. پیچیدگی چنین سیستمی، استفاده از ابزارهای خودکار مدیریتی را ایجاب می‌کند. اگر شبکه شامل تجهیزاتی از سازندگان مختلف باشد، نیاز به چنین ابزارهائی افزایش یافته و تهیهٔ این ابزارها نیز مشکل می‌شود. در پاسخ به این نیازها، استانداردهائی که مرتبط با مدیریت شبکه هستند تهیه شده که سرویس‌ها، پروتکل‌ها و پایگاه‌های اطلاعات مدیریتی را پوشش می‌دهند. تا زمان حاضر پراستفاده‌ترین استانداردی از این نوع، پروتکل سادهٔ مدیریت شبکه SNMP (Simple Network Management Protocol) بوده است.

از زمان انتشار آن در سال ۱۹۸۸ میلادی، SNMP در تعداد روزافزونی از شبکه‌ها و محیط‌های پیچیده مورد استفاده واقع شده است. همین‌طور که استفاده از SNMP گسترش یافت، نیاز به کارآئی‌های جدید برای پوشش نیازهای جدید نیز هویدا گشت. همچنین اهمیت ایجاد یک قابلیت امنیتی بعنوان بخشی از مدیریت شبکه آشکارتر گردید. در جهت ایجاد کارآئی بیشتر، نسخهٔ دوم SNMP تعریف شد. قابلیت‌های امنیتی فراگیرتر در SNMPv3 فراهم آمد.

این فصل تسهیلات امنیتی مقدماتی در SNMPv1 را توصیف کرده و سپس به خصوصیات امنیتی بسیار گسترده‌تری که در SNMPv3 وجود دارد می‌پردازد.

۸-۱ مفاهیم اساسی SNMP

این بخش مروری بر چهارچوب اصلی SNMP دارد.

معماری مدیریت شبکه

یک سیستم مدیریت شبکه، مجموعه‌ای از ابزارها برای پائیدن و کنترل شبکه است که از جنبه‌های زیر یکپارچه‌اند:

- یک واسط اپراتوری منفرد، با مجموعه‌ای از فرامین قوی ولی آشنا با کاربر، برای انجام اکثر وظایف مدیریتی شبکه.
- میزان حداقلی از تجهیزات مجزا. یعنی بیشتر سخت‌افزارها و نرم‌افزارهای لازم برای مدیریت شبکه در داخل تجهیزات موجود کاربر جای داده شده‌اند.

یک سیستم مدیریت شبکه، شامل برخی سخت‌افزارها و نرم‌افزارهای اضافی است که به مؤلفه‌های موجود شبکه اضافه شده‌اند. نرم‌افزار استفاده شده برای انجام وظایف مدیریت شبکه، در دل کامپیوترهای میزبان و عوامل ارتباطی (مثل پردازشگرهای خط اول، کنترل‌کنندهٔ ترمینال‌ها) جای دارد. یک سیستم مدیریت شبکه طوری طراحی شده است که تمام شبکه را بصورت یکپارچه نگریسته، هر نقطهٔ از آن را با آدرس‌ها و برچسب‌هایش شناخته و صفات آن نقطه و ارتباطش با کل شبکه را درک کند. عناصر فعال شبکه، یک بازخورد منظم از اطلاعات مربوط به وضعیت شبکه را برای مرکز کنترل شبکه فراهم می‌آورند.

مدل مدیریت شبکه که در SNMP از آن استفاده می‌شود شامل عناصر کلیدی زیر است:

- ایستگاه مدیریت
- عامل مدیریت
- پایگاه اطلاعات مدیریت
- پروتکل مدیریت شبکه

ایستگاه مدیریت (management station) معمولاً یک دستگاه متکی به خود است، ولی ممکن است یک قابلیت پیاده‌سازی شده در یک سیستم اشتراکی باشد. در هر صورت، ایستگاه مدیریت واسط بین مدیریت انسانی شبکه با سیستم مدیریت شبکه است. ایستگاه مدیریت حداقل دارای مؤلفه‌های زیر است:

- یک مجموعه از برنامه‌های کاربردی مدیریتی برای تحلیل داده‌ها، بازیابی از خطا و غیره
- یک واسط که بتوسط آن مدیر شبکه بتواند شبکه را کنترل کرده و بپاید
- قابلیت ترجمه نیازهای مدیر شبکه به پایش واقعی و کنترل عناصر دور در شبکه
- یک پایگاه داده از اطلاعات استخراج شده تمام واحدهای مدیریت شده در شبکه

تنها دو مؤلفهٔ آخر، موضوع استانداردسازی SNMP را تشکیل می‌دهند.

عنصر فعال دیگر در سیستم مدیریت شبکه، **عامل مدیریت** (management agent) است. تجهیزات کلیدی مانند میزبان‌ها، پل‌ها، مسیریاب‌ها و هاب‌ها ممکن است با SNMP طوری تجهیز شوند که بتوان از یک ایستگاه مدیریت آنها را اداره نمود. عامل مدیریت به سؤالات اطلاعاتی یک ایستگاه مدیریت جواب داده، به درخواست‌های عملیاتی ایستگاه مدیریت واکنش نشان داده و ممکن است بطور غیرهمزمان اطلاعات درخواست نشده ولی مهم را برای ایستگاه مدیریت ارسال نماید.

برای مدیریت منابع در شبکه، هر منبع بصورت یک موضوع (object) نشان داده می‌شود. یک موضوع نوعاً یک متغیر اطلاعاتی است که وجهی از عامل مدیریت شده را نشان می‌دهد. به مجموعهٔ موضوعات، **پایگاه اطلاعات مدیریت** management information base (MIB) می‌گویند. MIB بصورت مجموعه‌ای از نقاط دست‌یابی در عامل مدیریت، برای ایستگاه مدیریت عمل می‌کند. موضوعات در عرض سیستم‌های یک کلاس بخصوص استاندارد شده‌اند (مثلاً همهٔ پل‌ها یک نوع موضوعات مدیریتی را حمایت می‌کنند). یک ایستگاه مدیریت، عمل پائیدن شبکه را با اخذ موضوعات MIB انجام می‌دهد. یک ایستگاه مدیریت می‌تواند باعث شود که عملی در یک عامل مدیریت صورت پذیرفته و یا می‌تواند پیکربندی یک عامل را، با تغییر اندازهٔ موضوعات مشخص، تغییر دهد.

ایستگاه مدیریت و عوامل مدیریت بتوسط **پروتکل مدیریت شبکه** (network management protocol) بهم پیوند می‌خورند. پروتکل استفاده شده برای مدیریت شبکه‌های TCP/IP، پروتکل سادهٔ مدیریت شبکه SNMP است. این پروتکل شامل قابلیت‌های کلیدی زیر است:

- **Get**: ایستگاه مدیریت را قادر می‌سازد تا اندازه‌های موضوعات در عامل مدیریت را بدست آورد.
- **Set**: ایستگاه مدیریت را قادر می‌سازد تا اندازه‌های موضوعات در عامل مدیریت را تنظیم کند.
- **Notify**: یک عامل مدیریت را قادر می‌سازد تا ایستگاه مدیریت را از پیشامدهای قابل توجه خبردار سازد.

معماری پروتکل مدیریت شبکه

در سال ۱۹۸۸ میلادی، مشخصه‌هایSNMP منتشر گردید و بسرعت بصورت استاندارد غالب مدیریت شبکه درآمد. تعدادی از فروشندگان، ایستگاه‌های کاری منفرد مدیریت شبکه مبتنی بر SNMP را عرضه نموده و بیشتر فروشندگان پل‌ها، مسیریاب‌ها، ایستگاه‌های کاری و PCها، بسته‌های نرم‌افزاری عامل SNMP که محصولات آنان را قادر به اعمال مدیریت از سوی ایستگاه مدیریت می‌کند، به مشتریان عرضه می‌دارند.

همانطور که از نام آن پیداست، SNMP یک ابزار ساده برای مدیریت شبکه است. این ابزار یک پایگاه اطلاعاتی مدیریت(MIB) از متغیرهای عددی و جداول دوبعدی که محدود و بسهولت قابل پیاده‌سازی است را تعریف می‌کند و همچنین یک پروتکل روان برای اینکه یک مدیر بتواند متغیرهای MIB را بدست آورده و تنظیم کند، و همچنین یک عامل بتواند یادآوری‌های درخواست نشده بنام *traps* صادر کند را بوجود می‌آورد. قدرت SNMP در این سهولت نهفته است. SNMP بسهولت پیاده‌سازی شده و از منابع شبکه و پردازش‌گر در حد متوسط استفاده می‌کند. همچنین ساختار پروتکل و MIB بحد کافی سرراست بوده و باعث می‌شود که بین ایستگاه‌های مدیریت و نرم‌افزارهای عامل سازندگان مختلف، تعامل خوبی برقرار باشد.

سه مشخصهٔ زیربنائی عبارتند از:

- **ساختار و شناسائی اطلاعات مدیریت برای شبکه‌های مبتنی برTCP/IP (RFC 1155)**: نحوهٔ تعریف موضوعات مدیریت شده در MIB را توصیف می‌کند.
- **پایگاه اطلاعات مدیریت برای اینترنت‌های مبتنی بر TCP/IP: MIB-II (RFC 1213)**: موضوعات مدیریت شده در MIB را توصیف می‌کند.
- **پروتکل سادهٔ مدیریت شبکه (RFC 1157)**: پروتکل استفاده شده برای مدیریت این موضوعات را تعریف می‌کند.

SNMP بصورت یک پروتکل سطح کاربرد طراحی گردیده که بخشی از مجموعهٔ پروتکلی TCP/IP است. هدف از طراحی، عملکرد SNMP در بالای پروتکل UDP (User Datagram Protocol) بوده است که در RFC 768 تعریف شده است. برای یک ایستگاه مدیریتی منفرد، یک پروسهٔ مدیریتی، دست‌یابی به MIB مرکزی در ایستگاه مدیریت را کنترل کرده و یک واسط برای مدیریت شبکه را فراهم می‌آورد. پروسهٔ مدیریتی، مدیریت شبکه را با استفاده از SNMP که در بالای UDP، IP و پروتکل‌های نظیر وابسته به شبکه (مثل Ethernet, FDDI, X.25) هستند انجام می‌دهد.

هر عامل نیز بایستی SNMP، UDP و IP را پیاده‌سازی نماید. علاوه برآن یک پروسهٔ عامل وجود دارد که پیام‌های SNMP را تعبیر نموده و MIB عامل را کنترل می‌نماید. برای هر دستگاه عامل که سایر کاربردها همانند FTP را پشتیبانی می‌کند، هم TCP و هم UDP مورد نیاز است.

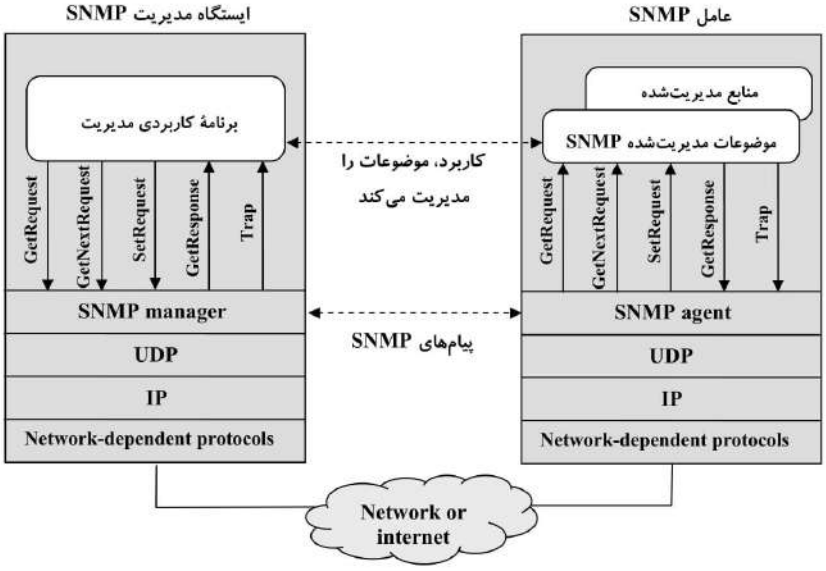
شکل ۱-۸ بستر پروتکل SNMP را نشان می‌دهد. از یک ایستگاه مدیریت، سه نوع پیام SNMP از جانب کاربردهای مدیریتی صادر می‌شود: GetRequest، GetNextRequest و SetRequest. دو پیام اول نوعی از تابع get هستند. هرسه نوع پیام بتوسط عامل و با پیام GetResponse تأئید می‌گردند که به کاربرد مدیریتی بالاتر ارجاع می‌شود. علاوه برآن، یک عامل ممکن است که یک پیام trap در پاسخ به پیشامدی که MIB و منابع مدیریت شده زیرمجموعه را تحت تأثیر قرار می‌دهد صادر کند.

چون SNMP به UDP که یک پروتکل غیراتصالی است متکی است، خود SNMP نیز غیراتصالی است. هیچ اتصالی بین یک ایستگاه مدیریت و عامل‌های آن برقرار نمی‌شود، بلکه هر مبادله یک ارتباط مجزا بین یک ایستگاه مدیریت و عامل آن است.

پروکسی‌ها

در SNMPv1 تمام عامل‌ها و همچنین خود ایستگاه‌های مدیریت بایستی UDP و IP را پشتیبانی نمایند. این امر مدیریت مستقیم بر بعضی دستگاه‌ها را محدود نموده و دستگاه‌های دیگری همچون بعضی پل‌ها و مُودم‌ها که هیچکدام از بخش‌های TCP/IP را پشتیبانی نمی‌نمایند از دَور خارج می‌سازد. علاوه برآن ممکن است سیستم‌های کوچک بسیاری (کامپیوترهای شخصی، ایستگاه‌های کاری، کنترل‌کننده‌های قابل برنامه‌ریزی) وجود داشته باشند که برای کارهای خود، TCP/IP را پیاده‌سازی کرده ولی تمایلی ندارند که بار اضافی SNMP، منطق عامل و نگهداری MIB را بعهده داشته باشند.

برای پشتیبانی دستگاه‌هائی که پیاده‌سازی SNMP را در خود ندارند، مقولهٔ پروکسی(proxy) خلق گردید. در این روش یک عامل SNMP بصورت پروکسی (وکیل) برای یک یا چند دستگاه عمل می‌کند. یعنی عامل SNMP به وکالت از طرف دستگاه‌های پروکسی شده، رفتار می‌کند.



شکل ۱- ۸ نقش SNMP

شکل ۲-۸ نوع معماری پروتکل درگیر را نشان می‌دهد. ایستگاه مدیریت سؤالاتی که در مورد دستگاه خاصی دارد را به عامل پروکسی خود می‌فرستد. عامل پروکسی، هر سؤال را به پروتکل مدیریتی بکار رفته بتوسط آن دستگاه تبدیل می‌کند. وقتی عامل پروکسی پاسخ سؤال را دریافت کرد، آن را برای ایستگاه مدیریت ارسال می‌دارد. بطریق مشابه اگر یک یادآوری یا اخطار از هر نوع از سوی دستگاه به پروکسی انتقال یابد، پروکسی آن را بصورت یک پیام trap برای ایستگاه مدیریت خواهد فرستاد.

SNMPv2 نه تنها استفاده از مجموعهٔ پروتکلی TCP/IP بلکه استفاده از سایر انواع پروتکل‌ها را نیز اجازه می‌دهد. SNMPv2 علی‌الخصوص برای اجرا روی بستهٔ پروتکلی OSI طراحی شده است. بنابراین SNMPv2 می‌تواند تعداد متنوع‌تری از پیکربندی‌های شبکه را مدیریت نماید. در رابطه با پروکسی‌ها، هر دستگاهی که پیاده‌سازی SNMPv2 را ندارد تنها از طریق یک پروکسی می‌تواند مدیریت شود. این امر حتی شامل دستگاه‌های SNMPv1 هم می‌شود. یعنی اگر یک دستگاه، نرم‌افزار عامل SNMPv1 را در پیاده‌سازی خود داشته باشد، تنها از طریق یک دستگاه پروکسی که عامل SNMPv2 و نرم‌افزار مدیریت SNMPv1 را داشته باشد می‌تواند به مدیر SNMPv2 دسترسی یابد.

مواردی که در بالا به آنها اشاره شد را روابط پروکسی خارجی در SNMPv2 گویند. علاوه برآن SNMPv2 از یک ارتباط پروکسی بومی که در آن دستگاه‌های پروکسی شده، SNMPv2 را حمایت می‌کنند پشتیبانی می‌کند. در مورد اخیر، یک مدیر SNMPv2 با یک گره SNMPv2 که بعنوان یک عامل کار می‌کند ارتباط برقرار می‌کند. این گره آنگاه بصورت یک مدیر برای دسترسی به دستگاه پروکسی شده عمل می‌کند که بعنوان عامل SNMPv2 کار خواهدکرد. علت پشتیبانی از چنین ارتباط غیرمستقیمی این است که کاربران را قادر سازد تا سیستم مدیریت شبکه‌های سلسله مراتبی و غیرمتمرکز را، چنان که بعداً توضیح داده خواهد شد، پیکربندی نمایند.

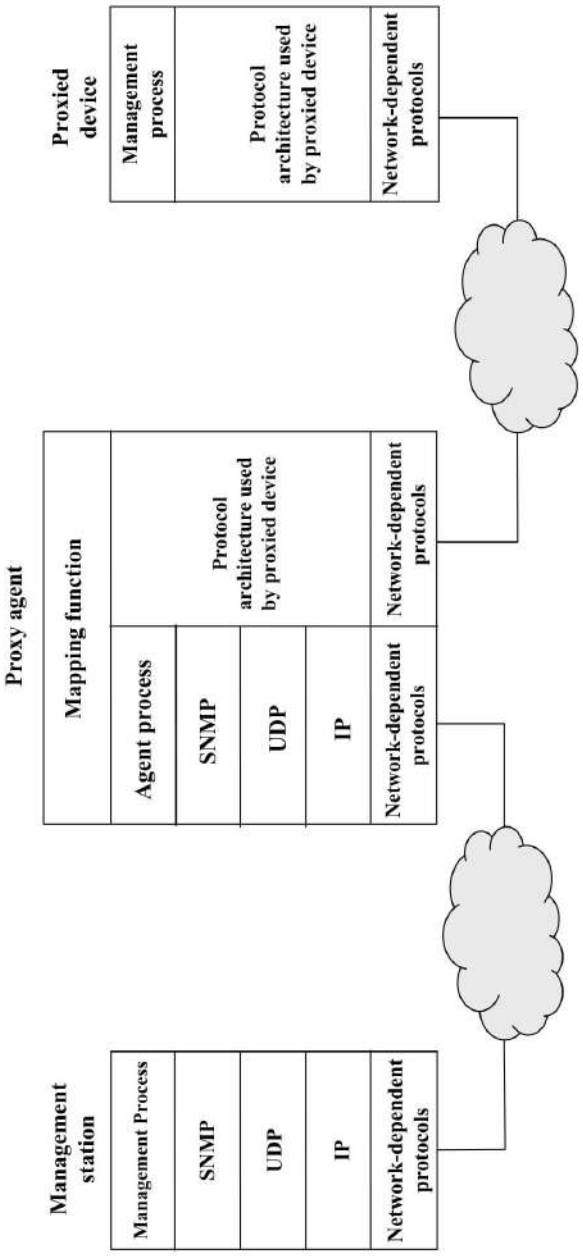
SNMPv2

قدرت SNMP در سادگی آن است. SNMP یک مجموعه پایه از ابزارهای مدیریت شبکه در یک بستهٔ نرم‌افزاری که بسهولت پیاده‌سازی و پیکربندی می‌شود را فراهم می‌آورد. از سوی دیگر چون کاربران برای مدیریت شبکه‌هائی که روز به روز توسعه یافته و بار کاری آنها افزایش می‌یابد هر روز بیشتر از دیروز به SNMP روآورده‌اند، کمبودهای آن کاملاً آشکار شده است. این کمبودها در سه گروه قرار دارند:

- عدم پشتیبانی از مدیریت توزیع‌شدهٔ شبکه
- نواقص عملیاتی
- نواقص امنیتی

کمبودهای اول و دوم در SNMPv2 مورد توجه قرار گرفتند که در سال ۱۹۹۳ منتشر شد و نسخهٔ تجدیدنظرشدهٔ آنهم در سال ۱۹۹۶ تهیه گردید (در حال حاضر RFCهای 1901, 1904 تا 1908، 2578 و 2579). SNMPv2 بسرعت مورد استقبال قرار گرفت و تعدادی از فروشنگان تنها چندماه پس از انتشار آن، محصولات منطبق با این استاندارد را به بازار فرستادند. نواقص امنیتی در SNMPv3 مورد توجه قرار گرفته است.

در بقیهٔ این بخش بطور مختصر مشخصه‌های جدید فراهم آمده بتوسط SNMPv2 را بررسی می‌کنیم. خصوصیات امنیتی SNMPv1 و SNMPv3 در بخش‌های بعدی توصیف خواهند شد.



شکل ۲-۸ پیکربندی پروکسی

مدیریت توزیع‌شدهٔ شبکه

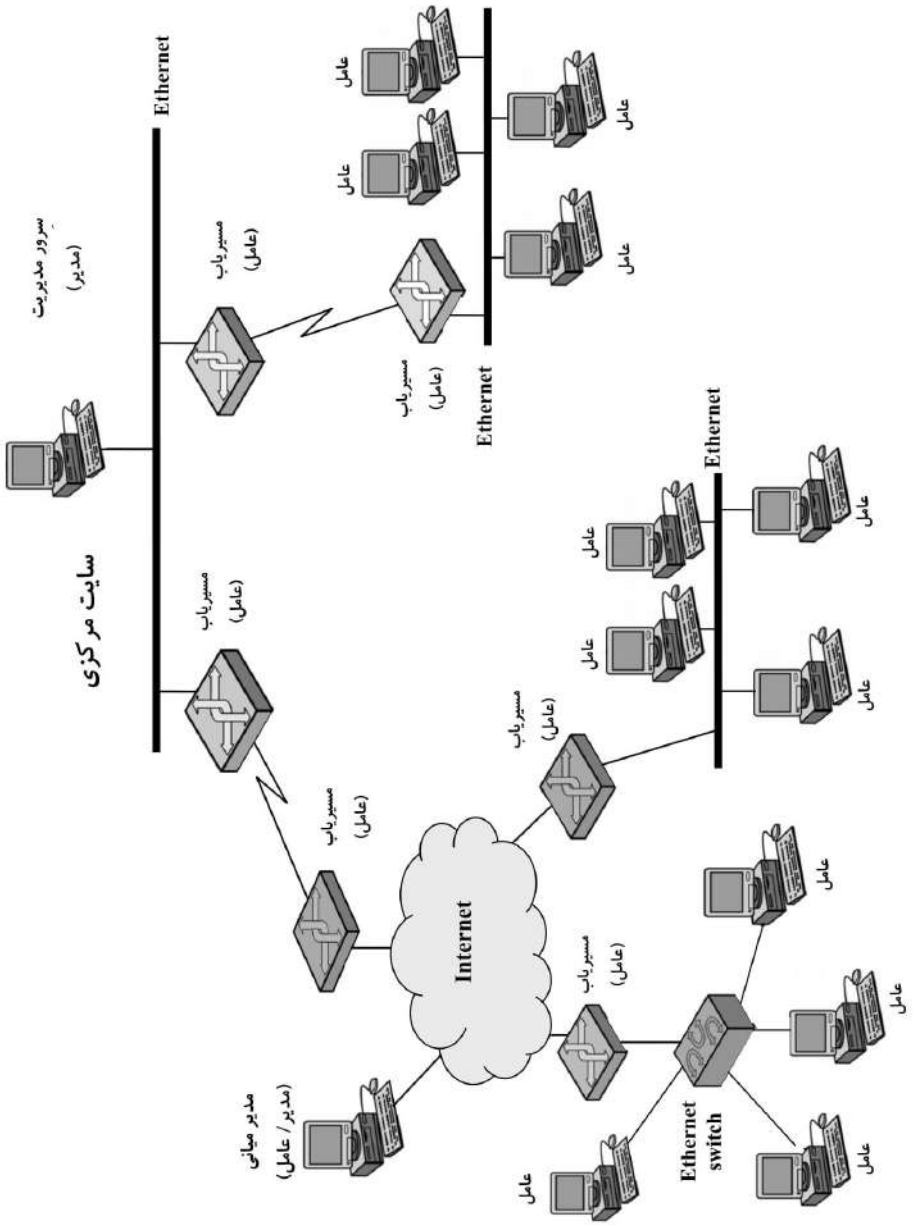
در روش سنّتی مدیریت متمرکز شبکه، یکی از میزبان‌ها در پیکربندی شبکه نقش ایستگاه مدیریت را دارد. ممکن است یک یا دو ایستگاه مدیریت دیگر نیز بعنوان پشتیبان ایستگاه اصلی عمل نمایند. مابقی تجهیزات شبکه دارای نرم‌افزارهای عامل و یک MIB هستند که پایش و کنترل از جانب ایستگاه مدیریت را امکان‌پذیر می‌نمایند. وقتی اندازهٔ شبکه‌ها بزرگ شده و بار ترافیکی آنها افزایش می‌یابد، چنین سیستم متمرکزی کارآئی نخواهد داشت. در این حالت فشار زیادی روی ایستگاه مدیریت وارد آمده و ترافیک پرحجمی ایجاد می‌شود که ناشی از گزارشات عوامل مدیریت بوده که تلاش می‌کنند تا سراسر شبکه را طی کرده و خود را به محل استقرار ایستگاه مدیریت برسانند. در چنین وضعیتی یک روش غیرمتمرکز توزیع شده، عملکرد بهتری دارد (شکل ۳-۸). در یک روش غیرمتمرکز مدیریت شبکه، ممکن است ایستگاه‌های مدیریتی سطح بالای متعددی وجود داشته باشند که به آنها سِرورهای مدیریت گویند. هر یک از چنین سِرورهائی ممکن است بطور مستقیم بخشی از عوامل را اداره نمایند ولی برای بسیاری از عامل‌ها، سِرور مدیریت مسئولیت خود را به یک مدیر میانی می‌سپارد. مدیر میانی، وظیفهٔ مدیریت برای پایش و کنترل عوامل تحت مسئولیت خود را دارد. مدیر میانی برای مدیر بالادست نقش یک عامل را بازی کرده، اطلاعات مورد نیاز او را فراهم نموده و دستورات او را می‌پذیرد. این نوع معماری مشکلات پردازش‌های مدیریتی را کم کرده و ترافیک کل شبکه را کاهش می‌دهد.

SNMPv2، هم از یک استراتژی کاملاً متمرکز و هم از یک استراتژی توزیع‌شده پشتیبانی می‌کند. در مورد اخیر بعضی سیستم‌ها هم نقش مدیر و هم نقش عامل را دارند. در نقش عامل، چنین سیستمی فرامین را از یک سیستم مدیریت مافوق می‌پذیرد. برخی از این فرامین مربوط به MIB محلی در عامل هستند. سایر فرامین به این امر نیاز دارند که عامل بعنوان یک پروکسی دستگاه‌های دور عمل کند. در این مورد، عامل پروکسی نقش مدیر برای دست‌یابی به اطلاعات یک عامل دور را داشته و آنگاه نقش یک عامل را از جهت ردکردن اطلاعات به مدیر مافوق خواهد داشت.

ارتقاء قابلیت‌های عملیاتی

جدول ۱-۸ قابلیت‌های ارتقاء یافتهٔ در SNMPv2 را نشان می‌دهد. هر دو پروتکل SNMP برحسب یک سری فرامین تعریف شده‌اند که بصورت واحدهای دیتای پروتکلی (PDU) منتقل می‌گردند. در مورد SNMPv1 پنج فرمان وجود دارد. یک مدیر، فرمان Get را برای یک عامل می‌فرستد تا اندازه موضوعات در MIB را برای او بفرستد. فرمان GetNext از این حقیقت استفاده می‌کند که موضوعات در یک MIB بصورت درختی سازمان داده شده‌اند. وقتی در یک فرمان GetNext از یک موضوع نام برده می‌شود، عامل، موضوع بعدی در درخت را پیدا کرده و اندازهٔ آن را برای مدیر برمی‌گرداند. GetNext از این جهت مفید است که به یک مدیر اجازه می‌دهد تا یک درخت، در محل عامل را، جهت یافتن موضوعات «به پیماید». فرمان Set به مدیر اجازه می‌دهد تا اندازه‌های موجود در عامل را به روزرسانی کرده و برای خلق و حذف ردیف‌های جداول از آن استفاده نماید. یک عامل از فرمان GetResponse برای پاسخ دادن به فرمان یک مدیر استفاده می‌کند. بالاخره فرمان Trap یک عامل را قادر می‌سازد تا بدون اینکه منتظر درخواست مدیریت شود، اطلاعات را برای مدیر بفرستد. بعنوان مثال یک عامل می‌تواند طوری پیکربندی شود تا در صورت پاره شدن یک لینک و یا افزایش ترافیک از آستانهٔ مشخصی، یک فرمان trap ارسال کند.

SNMPv2 تمام فرامین موجود در SNMPv1 را داشته و علاوه برآن دارای دو فرمان اضافی است. فرمان مهم‌تر، فرمان Inform است. این فرمان بتوسط یک ایستگاه مدیریت برای یک ایستگاه دیگر ارسال شده و همانند trap، اطلاعات مربوط به حالت‌ها یا پیشامدهای ایجادشده در یک فرستنده است. حسن فرمان Inform این است که از آن می‌توان در ساخت یک پیکربندی، برای تقسیم وظایف مدیریتی بین چند مدیر، در یک شبکهٔ بزرگ استفاده کرد.



شکل ۳-۸ مثالی از مدیریت توزیع شده شبکه

جدول ۱-۸ مقایسهٔ PDUهای SNMPv1 و SNMPv2

SNMPv1 PDU	SNMPv2 PDU	جهت ارسال	توصیف
GetRequest	GetRequest	مدیر به عامل	درخواست اندازه برای هر موضوع لیست شده
GetNextRequest	GetNextRequest	مدیر به عامل	درخواست اندازهٔ بعدی برای هر موضوع لیست شده
____	GetBulkRequest	مدیر به عامل	درخواست اندازه‌های متعدد
SetRequest	SetRequest	مدیر به عامل	تنظیم اندازه برای هر موضوع لیست شده
____	InformRequest	مدیر به مدیر	انتقال اطلاعات درخواست نشده
GetResponse	Response	عامل به مدیر یا مدیربه مدیر (SNMPv2)	پاسخ به درخواست مدیر
Trap	SNMPv2-Trap	عامل به مدیر	انتقال اطلاعات درخواست نشده

فرمان جدید دیگر، GetBulk است که به یک مدیر اجازه می‌دهد تا بلوک بزرگی از داده‌ها را درخواست نماید. فرمانGetBulk علی‌الخصوص برای انتقال تمام جداول، بتوسط یک فرمان منفرد، طراحی شده است.

یک اختلاف نهائی: فرمان Get در مورد SNMPv1 یکپارچه بوده در حالی که در مورد SNMPv2 یکپارچه نیست. اگر فرمان Get در SNMPv1 شامل لیستی از موضوعات باشد که اندازهٔ آنها درخواست شده است و حداقل یکی از این موضوعات در محل عامل موجود نباشد، تمام فرمان پذیرفته نخواهد شد. برای SNMPv2 چنین نبوده و بخشی از نتایج می‌تواند بازگشت داده شود. فرمان Get غیریکپارچه، استفادهٔ بهره‌ورتری از ظرفیت شبکه بتوسط مدیر را اجازه می‌دهد.

۸-۲ تسهیلات جامعه‌ای SNMPv1

SNMPv1 برابر آنچه که در RFC 1157 تعریف شده است، تنها شامل یک امکان امنیتی ابتدائی مبتنی برمفهوم جامعه (community) است. این امکان، سطح معینی از امنیت را ایجاد کرده ولی به حملات مختلف آسیب‌پذیر است [CERT02,JIAN02].

جوامع و نام‌های جوامع

همانند سایر کاربردهای توزیع شده، مدیریت شبکه از تعامل تعدادی موجودیت‌های کاربردی که از سوی یک پروتکل کاربردی حمایت می‌شوند، تشکیل می‌شود. در مورد SNMP، موجودیت‌های کاربردی مدیرها و عامل‌هائی هستندکه از SNMP استفاده می‌کنند.

مدیریت شبکه SNMP دارای چندین مشخصه است که شبیه کاربردهای توزیع شدهٔ دیگر نیستند. SNMP شامل یک رابطهٔ یک- به- چند بین یک مدیر و مجموعه‌ای از عوامل است: مدیر قادر است که موضوعات موجود در عوامل را بدست آورده و تنظیم کند. او همچنین قادر است که trapها را از عوامل دریافت دارد. بنابراین از نظر عملیاتی یا کنترلی، مدیر تعدادی از عوامل را «مدیریت می‌کند». همچنین ممکن است تعدادی مدیر وجود داشته باشدکه هریک از آنها تمام و یا زیرمجموعه‌ای از عوامل را پیکربندی و مدیریت نمایند. این زیرمجموعه‌ها ممکن است هم‌پوشانی داشته باشند.

بهمین ترتیب بایستی قادر باشیم تا شبکهٔ مدیریت SNMP را بصورت یک رابطهٔ یک- به- چند از طرف یک عامل با چند مدیر نگاه کنیم. هر عامل، MIB محلی خود را کنترل کرده و بایستی قادر باشد استفاده از آن MIB بتوسط تعدادی مدیر را کنترل کند. این کنترل دارای سه جنبه است:

- **سرویس اعتبارسنجی**: عامل ممکن است علاقه‌مند باشد تا دست‌یابی به MIB را تنها به مدیریت‌های معتبر اجازه دهد.
- **خط‌مشی دست‌یابی**: عامل ممکن است علاقه‌مند باشد تا امتیازات دست‌یابی متفاوتی به مدیران مختلف تخصیص دهد.
- **سرویس پروکسی**: یک عامل ممکن است بعنوان وکیل (پروکسی) سایر عوامل عمل نماید. این امر ممکن است شامل پیاده‌سازی سرویس اعتبارسنجی و/ یا خط‌مشی دست‌یابی برای سایر عوامل، در سیستم پروکسی باشد.

تمام این جنبه‌ها مرتبط با مسائل امنیتی هستند. در محیطی که مسئولیت مؤلفه‌های شبکه تقسیم‌بندی شده‌اند (مثلاً بین تعدادی واحدهای مدیریتی)، عامل‌ها نیازمند حفاظت خود و MIBهای خود از دست‌یابی‌های ناخواسته/ غیرمعتبر می‌باشند. SNMP برابر آنچه در RFC 1157 تعریف شده است، تنها یک امکان ابتدائی و محدود از چنین امنیتی با نام جامعه را فراهم می‌آورد.

یک **جامعهٔ SNMP (community)**، یک رابطهٔ بین یک عامل SNMP با جمعی از مدیران SNMP است که اعتبارسنجی، کنترل دست‌یابی و مشخصه‌های پروکسی را تعریف می‌کند. جامعه یک مفهوم محلی است که در محل یک عامل تعریف می‌شود. عامل برای هر ترکیب مطلوب از اعتبارسنجی، کنترل دست‌یابی و مشخصه‌های پروکسی، یک جامعه را تعریف می‌کند. به هر جامعه، یک نام یکتا (در درون این عامل) داده شده و مدیران متعلق به این جامعه بایستی از نام این جامعه در تمام فرامین get و set خود استفاده نمایند. یک عامل می‌تواند چندین جامعه را تشکیل دهد بطوری که عضویت مدیران در این جوامع هم‌پوشانی هم داشته باشند.

چون جوامع بطور محلی در یک عامل تعریف می‌شوند، عوامل مختلف ممکن است از نام واحدی استفاده کنند. یکسان بودن این نام‌ها مهم نبوده و نشان‌دهندهٔ هیچ شباهتی بین جوامع تعریف شده نیستند. بنابراین یک مدیر بایستی سابقهٔ نام و یا نام‌های مرتبط با هر عاملی که مایل به تماس با آن است را داشته باشد.

سرویس اعتبارسنجی

هدف سرویس اعتبارسنجی SNMPv1 این است که به گیرنده اطمینان دهد که یک پیام SNMPv1 از همان منبعی صادر شده است که ادعا می‌کند. SNMPv1 تنها یک روش ابتدائی برای احراز هویت را فراهم می‌آورد. هر پیام (تقاضای get یا put) از یک مدیر به یک عامل، شامل نام یک جامعه است. این نام بصورت یک کلمهٔ عبور عمل کرده و اگر فرستنده کلمهٔ عبور را بداند، معتبر تلقی خواهد شد.

با چنین فرم محدودی از اعتبارسنجی، بسیاری از مدیران شبکه حاضر نیستند که چیزی بجز پایش شبکه، یعنی عملیات get و trap را اجازه دهند. کنترل شبکه از طریق عمل set، طبیعتاً امر حساس‌تری است. نام جامعه می‌تواند آغازکنندهٔ یک رَویهٔ اعتبارسنجی باشد بشرط اینکه از نام فقط بصورت ابزار اولیه جستجوی کلمهٔ عبور استفاده شود. رَویهٔ اعتبارسنجی می‌تواند شامل مراحل پیچیده‌تری مثل رمزنگاری/رمزگشائی برای تأمین امنیت بیشتر باشد. این مسائل ورای قابلیت‌های RFC 1157 قرار دارد.

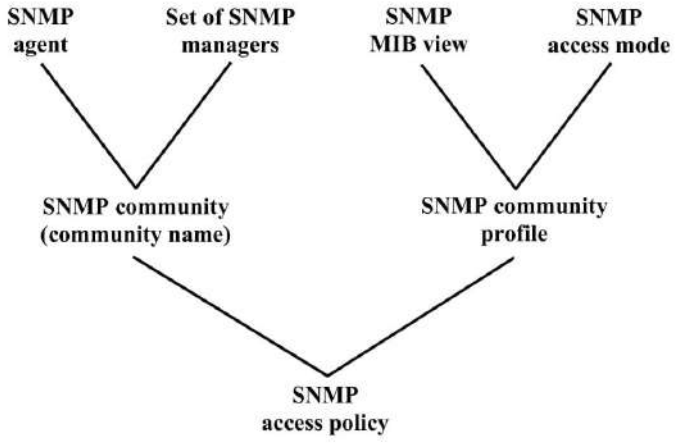
خط‌مشی دست‌یابی

با تعریف یک جامعه، یک عامل، دست‌یابی به MIB خود برای تعدادی از مدیران را محدود می‌کند. با استفاده از بیش از یک جامعه، عامل می‌تواند گروه‌های دست‌یابی مختلف برای مدیران مختلف تعریف کند. این کنترل دست‌یابی دو جنبه دارد:

- **منظر MIB در SNMP**: منظر(view) زیرمجموعه‌ای از موضوعات، در درون یک MIB است. منظرهای MIB مختلف ممکن است برای هر جامعه تعریف شود. مجموعهٔ موضوعات در یک منظر نیازی نیست که به یک زیرشاخهٔ منفرد MIB تعلق داشته باشد.
- **مُود دست‌یابی SNMP**: یک عنصر از مجموعه {READ-ONLY,READ-WRITE}است. یک مُود دست‌یابی برای هر جامعه تعریف می‌شود.

ترکیب یک منظرMIB و مُود دست‌یابی را یک **پروفایل جامعهٔ SNMP** گویند. بنابراین پروفایل یک جامعه شامل یک زیرمجموعهٔ تعریف‌شده از MIB در یک عامل، باضافهٔ یک مُود دست‌یابی به آن موضوعات است. مُود دست‌یابی SNMP بطور یکنواخت به تمام موضوعات در MIB view اعمال می‌گردد. بنابراین اگر مُود دست‌یابی READ-ONLY انتخاب شود، به تمام موضوعات منظر اعمال شده و مدیران را محدود می‌سازد: دسترسی به این منظر فقط READ-ONLY است.

با هر جامعهٔ تعریف شده بتوسط یک عامل، یک پروفایل جامعه مرتبط است. ترکیب یک جامعهٔ SNMP و یک پروفایل جامعهٔ SNMP را **خط‌مشی دست‌یابی SNMP** خوانند. شکل۴-۸ مفاهیمی را که مورد بحث قرار گرفت نشان می‌دهد.



شکل ۴-۸ مفاهیم مدیریتی SNMPv1

سرویس پروکسی

مفهوم جامعه در پشتیبانی سرویس پروکسی نیز مفید است. بخاطر آورید که یک پروکسی یک عامل SNMP است که به وکالت از طرف سایر دستگاه‌ها عمل می‌کند. معمولاً سایر دستگاه‌ها بیگانه هستند، یعنی TCP/IP و SNMP را پشتیبانی نمی‌کنند. در برخی موارد سیستم‌های پروکسی شده ممکن است SNMP را پشتیبانی کنند ولی پروکسی برای بحداقل رساندن تعامل بین دستگاه پروکسی شده و سیستم‌های مدیریت شبکه بکار می‌رود

برای هر دستگاهی که پروکسی وکیل آن است، یک خط‌مشی دست‌یابی SNMP تعریف می‌شود. بنابراین پروکسی می‌داند که کدام موضوعات MIB می‌توانند برای مدیریت سیستم‌های پروکسی شده (MIB view) و مُودهای دست‌یابی آن بکار روند.

۸-۳ SNMPv3

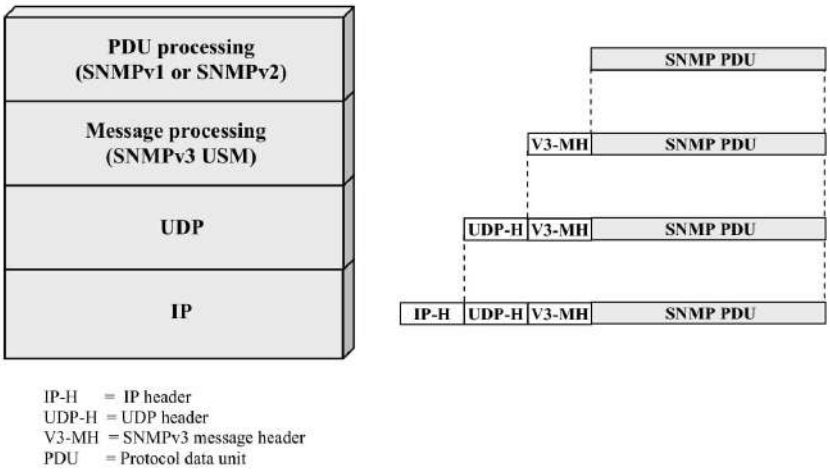
در سال ۱۹۹۸ میلادی، گروه کاری IETF SNMPv3 یک مجموعه از استانداردهای پیشنهادشده برای اینترنت که در حال حاضر RFC 2570 تا RFC 2576 را تشکیل می‌دهند فراهم نمود. این مجموعهٔ اسناد، یک چهارچوب برای بکارگیری قابلیت‌های امنیتی در مشخصه‌های کلی عملیات SNMPv1 یا SNMPv2 را شامل می‌شود. علاوه براین، اسناد یک مجموعه از قابلیت‌ها برای امنیت شبکه و کنترل دست‌یابی را فراهم می‌آورند.

توجه به این نکته مهم است که SNMPv3 یک جانشین برای SNMPv1 و/یا SNMPv2 نیست. SNMPv3 یک قابلیت امنیتی را تعریف می‌کند که می‌تواند بهمراه SNMPv2 (ترجیحاً) و یا SNMPv1 بکار رود. علاوه برآن، RFC 2571 یک معماری که در آن تمام نسخه‌های جاری و آتی SNMP می‌گنجند را توصیف می‌کند. RFC 2575 نیز یک امکان کنترل دست‌یابی را تعریف می‌کند که هدف آن این است که بطور مستقل از قابلیت هستهٔ SNMPv3 عمل کند. در این بخش یک نگاه کلی به RFC های 2570 تا 2576 نموده و توانائی‌های تعریف شده در آنها را بررسی می‌کنیم.

شکل ۵-۸ رابطهٔ میان نسخه‌های مختلف SNMP از نظر فرمت بکار رفته را نشان می‌دهد. اطلاعات بین یک ایستگاه مدیریت و یک عامل مدیریت به شکل یک پیام SNMP مبادله می‌شود. پردازش‌های مرتبط با امنیت در سطح پیام انجام می‌شود. بعنوان مثال SNMPv3 یک مدل امنیتی کاربر(User Security Model (USM که از میدان‌های موجود در سرآیند پیام استفاده می‌کند را مشخص می‌نماید. محمولهٔ یک پیام SNMP، یک PDU از SNMPv1 یا SNMPv2 است. یک PDU نمایش‌دهندهٔ یک عمل مدیریتی (مانند get یا set در مورد یک موضوع تحت مدیریت) است که با آن لیستی از نام متغیرهای مربوط به آن عمل همراه است.

RFCهای 2570 تا 2576 معماری کلی بعلاوهٔ ساختار پیام‌های خاص و خصوصیات امنیتی آنها را توصیف کرده ولی فرمت جدیدSNMP PDU را تعریف نمی‌کند. در نتیجه در داخل معماری جدید بایستی از فرمت‌های PDU نسخه‌های SNMPv1 یا SNMPv2 استفاده کرد. یک نوع پیاده‌سازی که از آن با عنوان SNMPv3 یاد می‌شود، شامل خصوصیات امنیتی و معماری تعریف شده در RFCهای 2570 تا 2576 بعلاوهٔ فرمت PDU و کارآئی‌های تعریف شده در اسناد SNMPv2 است. این موضوع در RFC 2570 چنین ذکر شده است: «SNMPv3 را می‌توان یک SNMPv2 باضافهٔ قابلیت‌های امنیتی و مدیریتی اضافی دانست.»

بقیهٔ این بخش چنین سازمان‌دهی شده است. ابتدا معماری SNMP که در RFC 2571 تعریف شده است را بطور مختصر معرفی می‌کنیم. سپس امکانات محرمانگی و اعتبارسنجی فراهم آمده بتوسط مدل امنیتی کاربر(USM) در SNMPv3 توصیف می‌گردد. بالاخره کنترل دست‌یابی و مدل کنترل دست‌یابی (VACM) را معرفی خواهیم کرد.



شکل ۵-۸ معماری پروتکل SNMP

معماری SNMP

معماری SNMP، برابر آنچه در RFC 2571 توصیف شده است، شامل مجموعه‌ای از موجودیت‌های توزیع‌شده و متعامل SNMP است. هر موجودیت، بخشی از قابلیت‌های SNMP را ایجاد کرده و ممکن است بصورت یک گرهٔ عامل، یک گرهٔ مدیریت و یا ترکیبی از این دو عمل نماید. هر موجودیت SNMP، شامل یک مجموعه از مدول‌هائی است که با هم تعامل کرده تا سرویس‌ها را فراهم آورند. این تعامل‌ها را می‌توان بصورت مجموعه‌ای از متغیرهای ابتدائی و پارامترهای انتزاعی مدل نمود.

RFC 2571 یک نیاز کلیدی طراحی برای SNMPv3 را مشخص می‌کند: یک معماری پودمانی طوری طراحی کنید که (۱) پیاده‌سازی روی محدودهٔ وسیعی از محیط‌های عملیاتی را اجازه دهد که برخی از آنها نیاز به عملکرد حداقل و ارزان قیمت داشته و برخی دیگر ممکن است دارای قابلیت‌های بیشتر برای مدیریت شبکه‌های بزرگ باشند، (۲) در مسیر استانداردسازی امکان انتقال بخش‌هائی از معماری به جلو، در صورت عدم تطبیق همهٔ بخش‌ها با هم، وجود داشته باشد و (۳) قابل سازگاری با مدل‌های امنیتی دیگر باشد.

موجودیت SNMP

هر موجودیت(entity) SNMP شامل یک موتور (engine) SNMP است. یک موتور SNMP عملیات ارسال و دریافت پیام‌ها، اعتبارسنجی و رمزنگاری/ رمزگشائی پیام‌ها و کنترل دست‌یابی به موضوعات مدیریت شده را پیاده‌سازی می‌نماید. این عملیات برای سرویس دادن به یک یا چند کاربرد، بتوسط موتور SNMP، پیکربندی شده و یک موجودیت SNMP را تشکیل می‌دهند.

هم موتور SNMP و هم کاربردهای پشتیبانی شده بتوسط این موتور، بصورت مجموعه‌ای از مدول‌های گسسته تعریف شده‌اند. این نوع معماری مزایای متعددی دارد. اول این که نقش موجودیت SNMP بتوسط مدول‌هائی که در آن پیاده‌سازی شده‌اند تعریف می‌شود. تعداد مشخصی از مدول‌ها، مورد نیاز یک عامل SNMP بوده در حالیکه تعداد مشخص دیگری (ممکن است هم‌پوشانی هم وجود داشته باشد) مورد نیاز یک مدیر SNMP هستند. ثانیاً ساختار پودمانی (مدولار) مشخصه‌ها باعث می‌شود که بتوان نسخه‌های مختلفی از هر مدول را تعریف کرد. این بنوبهٔ خود باعث می‌شود که (۱) قابلیت جایگزین کردن و یا ارتقاء توانائی‌ها برای جنبه‌های معینی از SNMP، بدون نیاز به عبور به نسخهٔ استاندارد شدهٔ بالاتر (مثلاً SNMPv4)، ایجاد گردد و (۲) بطور روشنی روش‌های هم‌زیستی، و استراتژی‌های عبور تعیین شود(RFC 2576).

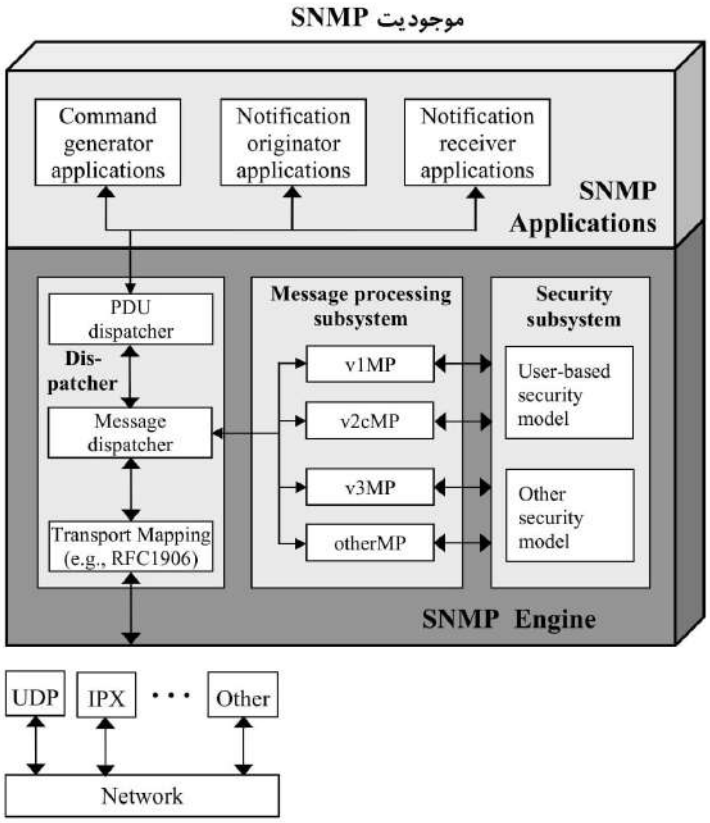
برای درک بهتر نقش هر مدول و رابطهٔ آن با مدول‌های دیگر، بهترین روش این است که به نحوهٔ استفاده از آنها در مدیرها و عامل‌ها در SNMP سنّتی توجه کنیم. اصطلاح *سنّتی(traditional)* که معادل *خالص(pure)* است از این جهت بکار رفته تا به این واقعیت اشاره کند که یک پیاده‌سازی لزومی ندارد حتماً کار یک مدیر خالص و یا یک عامل خالص را انجام دهد بلکه می‌توان مدول‌هائی را در کنار هم جمع کرد که هم وظایف مدیر و هم وظایف عامل را انجام دهند.

شکل ۶-۸ که برپایهٔ تصویر ارائه شده در RFC 2571 بناشده است بلوک دیاگرام یک **مدیر سنّتی یا خالص SNMP** را نشان می‌دهد. یک مدیر خالص SNMP با صدور فرمان‌هائی(get و set) با عوامل SNMP تعامل نموده و پیام‌های trap را دریافت می‌دارد. مدیر همچنین می‌تواند با سایر مدیران با صدور Inform Request PDU تعامل نموده، هشدارها (alerts) را فراهم کرده و یا فرامین Inform Response PDU را دریافت کند که در حقیقت تأئید دریافت فرامین Inform Request می‌باشند. در فرهنگ اصطلاحات SNMPv3، یک مدیرخالص SNMP شامل سه گروه از کاربردهاست.کاربرد مولد فرمان، داده‌های مدیریت در عامل‌های دور را پائیده و تغییرات لازم در آنها را بوجود می‌آورد. آنها از PDUهای SNMPv1 و/یا SNMPv2 که شامل Get، GetNext، GetBulk و Set است استفاده می‌کنند. یک کاربرد مولد اخطار، آغازگر پیام‌های آسنکرون است. در مورد یک مدیر خالص، Inform Request PDU برای این منظور بکار می‌رود. یک کاربرد گیرندهٔ اخطار، پیام‌های آسنکرون ورودی را پردازش می‌کند. اینها شامل PDUهای Inform Request، SNMPv2-Trap و SNMPv1-Trap هستند.

تمام کاربردهائی که در بالا توصیف شدند از سرویس‌های فراهم شده بتوسط موتور SNMP برای این موجودیت استفاده می‌کنند. موتور SNMP دو وظیفهٔ جمعی را انجام می‌دهد:

- PDUهای خارج‌شونده از کاربردهای SNMP را می‌پذیرد، پردازش‌های لازم را که شامل واردکردن کُدهای اعتبارسنجی و رمزنگاری است انجام داده و سپس PDUها را برای انتقال بصورت یک پیام، کپسولی می‌نماید.
- پیام‌های SNMP واردشونده را از لایهٔ حمل‌ونقل تحویل می‌گیرد، پردازش‌های لازم شامل اعتبارسنجی و رمزگشائی را انجام داده و سپس PDUها را از پیام استخراج کرده و آنها را به واحدهای کاربردی استفاده‌کننده از SNMP تحویل می‌دهد.

در یک مدیر خالص، موتور SNMP شامل یک حمل‌کننده (Dispatcher)، یک زیرسیستم پردازش پیام (Processing Subsystem) و یک زیرسیستم امنیت (Security Subsystem) است. حمل‌کننده بسادگی یک مدیر ترافیک است. برای PDUهای خارج‌شونده، حمل‌کننده PDUها را از کاربردها تحویل گرفته و کارهای زیر را انجام می‌دهد. برای هرPDU، حمل‌کننده نوع پردازش لازم برای پیام را تعیین کرده (یعنی SNMPv1، SNMPv2c و SNMPv3) و PDU را به مدول مناسب در زیرسیستم پردازش پیام عبور می‌دهد. به دنبال آن زیرسیستم پردازش پیام، یک پیام که شامل آن PDU و سرآیندهای مناسب هستند را باز می‌گرداند. حمل‌کننده آنگاه این پیام را برای انتقال به لایهٔ حمل‌ونقل می‌دهد.



شکل ۶-۸ مدیر سنّتی SNMP

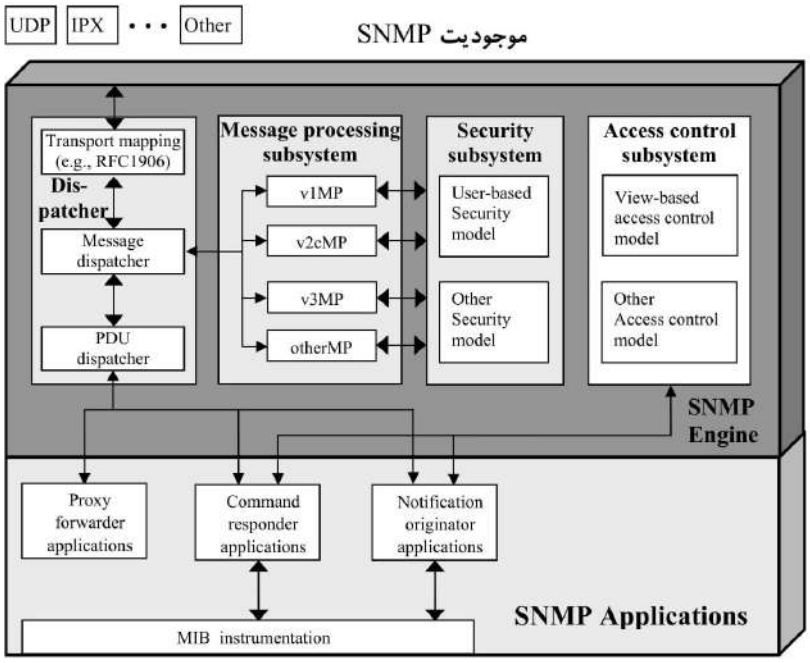
برای پیام‌های واردشونده، حمل‌کننده پیام‌ها را از لایهٔ حمل‌ونقل تحویل گرفته و عملیات زیر را انجام می‌دهد. حمل‌کننده، هر پیام را به مسیر مناسب که منتهی به مدول مناسب پردازش پیام است راهنمائی می‌کند. به دنبال آن زیرسیستم پردازش پیام، PDU موجود در پیام را بازپس می‌دهد. حمل‌کننده این PDU را به کاربرد مناسب عبور می‌دهد.

زیرسیستم پردازش پیام، PDUهای خارج شونده از حمل‌کننده را پذیرفته، آنها را با سرآیندهای مناسب تجهیز کرده و سپس به حمل‌کننده بازپس می‌دهد. زیرسیستم پردازش پیام همچنین پیام‌های ورودی را از حمل‌کننده قبول کرده، سرآیند هر پیام را پردازش نموده و PDU حمل‌شده در پیام را به حمل‌کننده برمی‌گرداند. یک پیاده‌سازی زیرسیستم پردازش پیام ممکن است تنها یک فرمت خاص پیام که مرتبط با یک نسخهٔ خاص SNMP (SNMPv1 ، SNMPv2c و یا SNMPv3) است را پشتیبانی کرده و یا ممکن است شامل تعدادی مدول باشد که هرکدام نسخهٔ خاصی از SNMP را پشتیبانی نمایند.

زیرسیستم امنیت، وظایف مربوط به اعتبارسنجی و رمزنگاری را انجام می‌دهد. هر پیام خارج‌شونده از زیرسیستم پردازش پیام، به زیرسیستم امنیت وارد می‌شود. بسته به سرویس مورد نیاز، زیرسیستم امنیت ممکن است PDU موجود در پیام و احتمالاً بخشی از میدان‌های سرآیند پیام را رمزنگاری کرده و ممکن است یک کُد اعتبارسنجی پیام تولید نموده و آن را در سرآیند پیام وارد کند. پیام پردازش‌شده آنگاه برای زیرسیستم پردازش پیام برگردانده می‌شود. به طریق مشابه، هر پیام واردشونده از زیرسیستم پردازش پیام به زیرسیستم امنیت عبور داده می‌شود. اگر لازم باشد، زیرسیستم امنیت کُد اعتبارسنجی را کنترل کرده، رمزگشائی را انجام داده و سپس پیام پردازش شده را به زیرسیستم پردازش پیام برمی‌گرداند. پیاده‌سازی زیرسیستم امنیت ممکن است یک یا چند مدل امنیتی خاص را پوشش دهد. تا کنون تنها مدل امنیتی تعریف شده User-Based Security Model (USM) برای SNMPv3 است که در RFC 2574 توصیف شده است.

شکل ۷-۸ که بر پایهٔ تصویر ارائه شده در RFC 2571 بنا شده است، بلوک دیاگرام یک **عامل سنّتی یا خالص SNMP** را نشان می‌دهد.

عامل سنّتی یا خالص ممکن است شامل سه نوع کاربرد باشد. کاربردهای پاسخ‌دهنده به فرامین، دست‌یابی به داده‌های مدیریت شده را فراهم می‌سازند. این کاربردها به درخواست‌های ورودی با گردآوری و/یا تنظیم موضوعات مدیریت شده و سپس صدور یک Response PDU پاسخ می‌دهند. یک کاربرد مولد اخطار، آغازگر پیام‌های آسنکرون می‌باشد. در مورد یک عامل خالص، PDUهای SNMPv2-Trap و SNMPv1-Trap برای این منظور مورد استفاده قرار می‌گیرند. یک کاربرد جلوبرندهٔ پروکسی، پیام‌ها را بین موجودیت‌ها، به جلو می‌راند.



شکل ۷-۸ عامل سنّتی SNMP

موتور SNMP برای یک عامل خالص، دارای تمام مؤلفه‌های موجود در موتور SNMP برای یک مدیر خالص باضافهٔ یک زیرسیستم کنترل دست‌یابی(Access Control Subsystem) است. این زیرسیستم وظیفهٔ شناسائی دست‌یابی‌های مجاز به MIB جهت خواندن و تنظیم موضوعات مدیریتی را داراست. این سرویس‌ها بر مبنای محتویاتPDUها انجام می‌شوند. یک پیاده‌سازی زیرسیستم امنیتی، ممکن است از یک یا چند مدول کنترل دست‌یابی پشتیبانی نماید. تا کنون تنها مدل امنیتی تعریف شده View-Based Access Control Model (VACM) است که در RFC 2575 توصیف شده است.

توجه کنید که عملیات مرتبط با امنیت در دو زیرسیستم مجزا سازمان‌دهی شده‌اند: امنیت و کنترل دست‌یابی. این یک مثال عالی از طراحی پودمانی است، زیرا دو زیرسیستم وظایف کاملاً مشخصی را انجام داده و بنابراین منطقی خواهد بود که استانداردسازی این دو مقوله بطور مستقل از هم انجام شود. زیرسیستم امنیت، وظیفهٔ کنترل سرّی بودن و معتبر بودن را بعهده داشته و روی پیام‌های SNMP عمل می‌کند. زیرسیستم کنترل دست‌یابی، وظیفهٔ کنترل دست‌یابی مجاز به اطلاعات مدیریتی را داشته و روی PDUهای SNMP عمل می‌کند.

فرهنگ واژه‌ها

جدول ۲-۸ بطور خلاصه بعضی از واژه‌هائی که در RFC 2571 معرفی شده‌اند را تعریف می‌کند. با هر موجودیت SNMP یک snmpEngineID یکتا مرتبط است. برای اهداف کنترل دست‌یابی، فرض می‌شود که هر موجودیت SNMP مدیریت تعـــدادی از مقـــوله‌های(context) اطلاعات مدیریت‌شده را بعهده دارد که هرکدام از آنها دارای یک contextName هستند که در آن موجودیت، یکتاست. برای تأکید بر اینکه در هر موجودیت یک مدیر واحد مقوله‌ها وجود دارد، هر موجودیت دارای یک contextEngineID یکتای مرتبط با آن است. چون یک ارتباط یک- به- یک بین موتور context و موتور SNMP در این موجودیت وجود دارد، contextEngineID از نظر اندازه برابر snmpEngineID است. کنترل دست‌یابی بتوسط مقوله‌های مشخصی که برای دست‌یابی به آنها تلاش شده است و هویت کاربر متقاضی، مدیریت می‌گردد. این کاربر که ممکن است یک فرد، یک کاربرد و یا گروهی از افراد یا کاربردها باشند، را رئیس(principal) خوانند.

سایر واژه‌های دارای اهمیت، مربوط به پردازش پیام‌ها می‌باشند. snmpMessageProcessingModel فرمت پیام و نسخهٔ SNMP پردازش پیام را مشخص می‌کند. snmpSecurityModel تعیین می‌کند که از کدام مدل امنیتی باید استفاده شود. snmpSecurityLevel تعیین می‌کند که کدام سرویس‌های امنیتی برای این کار مشخص درخواست شده‌اند. کاربر ممکن است فقط اعتبارسنجی، یا اعتبارسنجی بعلاوهٔ محرمانگی(رمزنگاری) و یا هیچکدام را درخواست کند.

کاربردهای SNMPv3

سرویس‌های بین مدول‌ها در یک موجودیت SNMP، در RFCها، برحسب فرامین ابتدائی(primitives) و پارامترها (parameters) تعریف شده‌اند. یک فرمان ابتدائی، عملی که باید انجام شود را مشخص کرده و از پارامترها برای عبور دادن دیتا و اطلاعات کنترلی استفاده می‌شود. این فرامین اولیه و پارامترها را می‌توان یک روش فرموله شده برای تعریف سرویس‌های SNMP دانست. فرم واقعی یک فرمان ابتدائی مستقل از پیاده‌سازی است و مثالی از آن احضار یک procedure است. در بحثی که به دنبال خواهد آمد، استفاده از شکل ۸-۸ که بر اساس تصویری در RFC 2571 بنا شده است ممکن است مفید واقع گردد تا ببینیم چگونه تمام این فرامین اولیه با هم جفت و جور می‌شوند. شکل ۸-۸الف دنباله‌ای از وقایعی را نشان می‌دهد که درآن یک کاربرد مولد فرمان یا مولد اخطار، درخواست ارسال یک PDU را می‌نماید و به دنبال آن یک پاسخ منطبق با این درخواست برای آن کاربرد پس فرستاده می‌شود. این پیشامدها در یک مدیر SNMP واقع

می‌شود. شکل ۸-۸ب پیشامدهای نظیر در یک عامل SNMP را نشان می‌دهد. در شکل نشان داده شده است که چگونه یک پیام ورودی منجر به تحویلPDU موجود در آن به یک کاربرد گشته و چگونه پاسخ کاربرد، منتج به یک پیام خروجی می‌گردد. توجه شود که بعضی پیکان‌ها در دیاگرام با نام یک primitive مشخص گردیده که نمایشگر یک درخواست است. پیکان‌های بدون نام، نمایشگر پاسخ به یک درخواست بوده و سایه‌ها نمایش انطباق بین درخواست و پاسخ مرتبط با آن است.

RFC 2573 بصورت کلی، رَویه‌هائی که هنگام تولید یک PDU برای ارسال و یا پردازش یک PDU ورودی، برای هر یک از کاربردها دنبال می‌شود را تعریف می‌کند. در همهٔ موارد، رَویه‌ها بر اساس تعامل با حمل‌کننده و برحسب Dispatcher Primitives تعریف می‌شوند.

جدول ۲-۸ فرهنگ واژه‌های SNMPv3

snmpEngineID
شناسهٔ یکتا و بدون ابهام یک موتور SNMP، و همچنین موجودیت SNMP که مرتبط با آن موتور است. این شناسه برحسب قرارداد بصورت Octet String تعریف می‌شود.

contextEngineID
بطور یکتا یک موجودیت SNMP که ممکن است نمونه‌ای از یک مقوله یا یک contextName خاص را شکل دهد تعریف می‌کند.

contextName
یک مقولهٔ بخصوص در یک موتور SNMP را معرفی می‌کند. این مقدار بعنوان یک پارامتر به Dispatcher و زیرسیستم کنترل دست‌یابی رد می‌شود.

scopedPDU
یک بلوک دیتا که شامل یک contextEngineID، یک contextName و یک SNMP PDU است. بصورت یک پارامتر به زیرسیستم امنیت عبور کرده و یا از آن گرفته می‌شود.

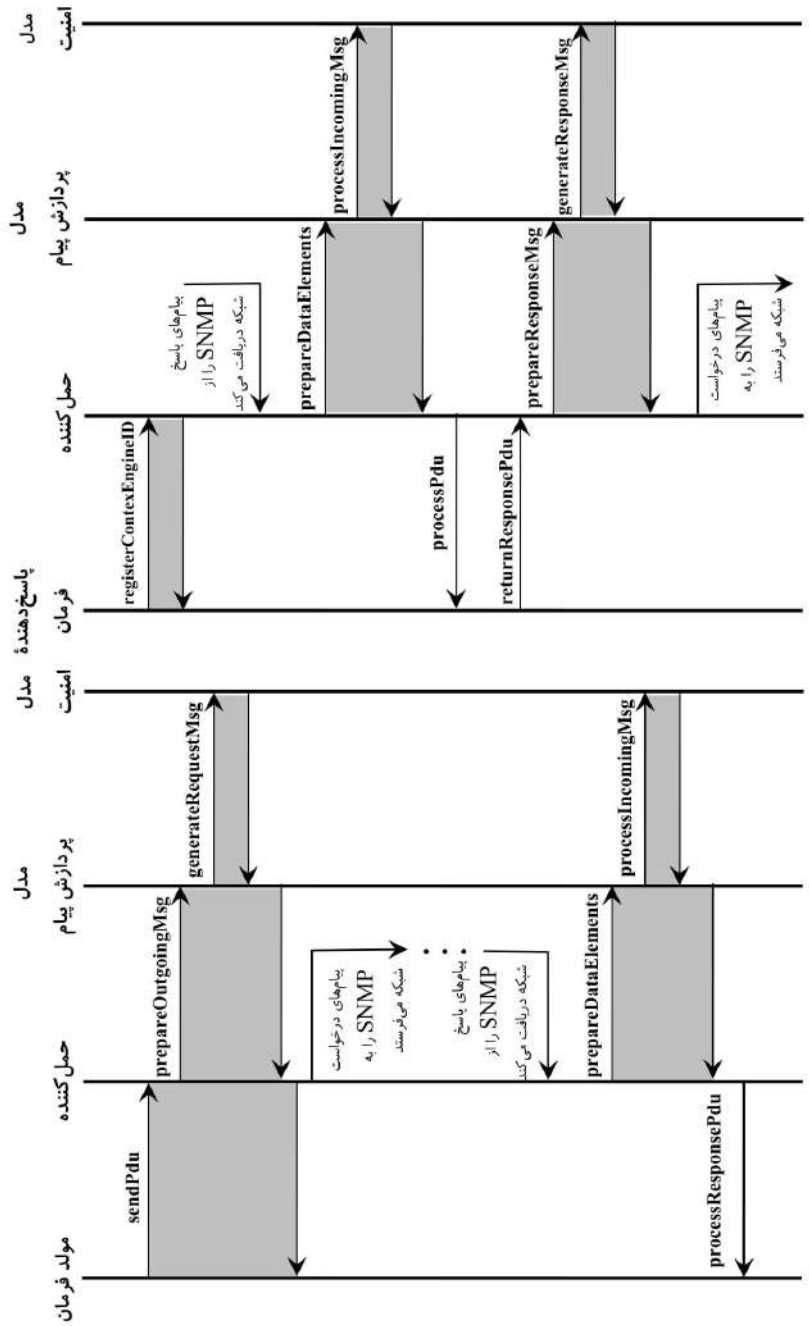
snmpMessageProcessingModel
شناسهٔ یکتای یک مدل پردازش پیام در زیرسیستم پردازش پیام است. مقادیر ممکن شامل SNMPv1، SNMPv2c و SNMPv3 است. بر حسب قرارداد بصورت Integer تعریف می‌شود.

snmpSecurityModel
شناسهٔ یکتای یک مدل امنیتی در زیرسیستم امنیت است. مقادیر ممکن شامل SNMPv1، SNMPv2c و USM است. بر حسب قرارداد بصورت Integer تعریف می‌شود.

snmpSecurityLevel
یک سطح امنیتی که در آن سطح، پیام‌های SNMP می‌توانند ارسال شده و یا عملیات پردازش صورت پذیرد و بدین صورت بیان می‌گردد که آیا اعتبارسنجی و/ یا محرمانگی بکار گرفته می‌شود یا نه. مقادیر ممکن noAuthnoPriv، authNoPriv و authPriv است و برحسب قرارداد بصورت Integer تعریف می‌شود.

principal
موجودیتی که از جانب او سرویس‌ها فراهم شده و یا پردازش‌ها انجام می‌شوند. یک principal می‌تواند فردی در یک نقش مشخص، مجموعه‌ای از افراد که هر کدام دارای نقش معین هستند، یک کاربرد و یا مجموعه‌ای از کاربردها و یا ترکیبی از همه این انواع باشد.

securityName
یک رشته قابل خواندن بتوسط انسان که نمایش‌دهندهٔ یک principal است. بصورت یک پارامتر در تمام فرامین اولیه SNMP ردوبدل می‌شود (Dispatcher ، Message Processing ، Security ، Access Control).



شکل ۸-۸ جریان عملیات SNMP

یک **کاربرد مولد فرمان** (**command generator application**) از فرامین ابتدائی حمل‌کنندهٔ sendPdu و processResponsePdu استفاده می‌کند. sendPdu، حمل‌کننده را با اطلاعاتی در مورد مقصد مورد نظر، پارامترهای امنیتی و PDU واقعی که باید ارسال شود تجهیز می‌کند. حمل‌کننده آنگاه مدل پردازش پیام (Message Processing Model) را بکار گرفته که آنهم بنوبهٔ خود مدل امنیتی (Security Model) را بخدمت طلبیده تا پیام را آماده نمایند. حمل‌کننده، پیام آماده شده را برای انتقال به لایهٔ حمل‌ونقل (مثلاً UDP) می‌فرستد. اگر آماده‌سازی پیام دچار مشکل شود، اندازهٔ برگشتی sendPdu primitive که بتوسط حمل‌کننده تنظیم می‌شود نمایشگر یک خطا خواهد بود. اگر آماده‌سازی پیام موفقیت‌آمیز باشد، حمل‌کننده یک شناسه sendPduHandle به این PDU اختصاص داده و اندازهٔ آن را به مولد فرمان برمی‌گرداند. مولد فرمان این sendPduHandle را ذخیره نموده تا بتواند PDU پاسخ متعاقب را، با این درخواست اولیه تطبیق دهد.

حمل‌کننده هر PDU، پاسخ ورودی را با استفاده از فرمان ابتدائی processResponsePdu به کاربرد مولد فرمان تحویل می‌دهد.

یک **کاربرد پاسخ‌دهندهٔ فرمان** (**command responder application**) از چهار فرمان ابتدائی حمل‌کننده (registerContextEngineID، unregisterContextEngineID، processPdu و returnResponsePdu) و یک فرمان ابتدائی Access Control Subsystem استفاده می‌کند.

registerContextEngineID primitive یک کاربرد پاسخ‌دهندهٔ فرمان را قادر می‌سازد تا خود را با یک موتور SNMP برای پردازش انواع PDU معین برای یک موتور context مرتبط نماید. وقتی یک پاسخ‌دهندهٔ فرمان، ثبت‌نام گردید، تمام پیام‌های آسنکرون دریافت‌شده که شامل ترکیب‌های ثبت‌نام‌شدهٔ contextEngineID و pduType مورد پشتیبانی هستند به پاسخ‌دهندهٔ فرمانی که برای پشتیبانی از این ترکیب ثبت‌نام شده است ارجاع می‌شوند. یک پاسخ‌دهندهٔ فرمان می‌تواند خود را از یک موتور SNMP که از unregisterContextEnginID primitive استفاده می‌کند جدا سازد.

حمل‌کننده هر PDU درخواست ورودی را با استفاده از processPdu primitive به کاربرد پاسخ‌دهندهٔ فرمان صحیح تحویل می‌دهد. به دنبال آن پاسخ‌دهندهٔ فرمان قدم‌های زیر را برمی‌دارد:

- پاسخ‌دهندهٔ فرمان، محتوای PDU درخواست را وارسی می‌کند. نوع عملیات بایستی با یکی از انواعی که قبلاً بتوسط این کاربرد ثبت نام شده است، تطبیق داشته باشد.
- پاسخ‌دهندهٔ فرمان تعیین می‌کند که آیا دست‌یابی برای عملیات مدیریتی تقاضا شده در این PDU مجاز است. برای این منظور AccessAllowed primitive احضار می‌گردد.
- پارامتر securityModel نشان می‌دهد که زیرسیستم کنترل دست‌یابی برای پاسخ به این درخواست از کدام مدل امنیتی باید استفاده کند. زیرسیستم کنترل دست‌یابی تعیین می‌کند که آیا این رئیس متقاضی (securityName) در این سطح امنیتی(securityLevel) حق درخواست این عمل مدیریتی (viewType) در این موضوع مدیریتی (variableName) در این مقوله (contextName) را داراست.
- اگر اجازهٔ دست‌یابی داده شد، پاسخ‌دهندهٔ فرمان عمل مدیریتی درخواست شده را انجام داده و یک PDU پاسخ را تهیه می‌کند. اگر دست‌یابی مجاز شناخته نشود، پاسخ‌دهنده فرمان، PDU پاسخ مناسب برای شکست عملیات را تهیه می‌کند.
- پاسخ‌دهندهٔ فرمان، حمل‌کننده را با یک returnResponsePdu فراخوانده تا PDU پاسخ را ارسال کند.

یک **کاربرد مولد اخطار (notification generator application)** همان رَویه‌های عمومی یک کاربرد مولد فرمان را دنبال می‌کند. اگر قرار است که یک Inform Request PDU ارسال شود، هم از sendPdu primitive و هم از processResponse primitive، بهمان روش کاربردهای مولد فرمان، استفاده می‌شود. اگر قرار است یک trap PDU ارسال شود تنها از sendPdu primitive استفاده می‌شود.

یک **کاربرد گیرندهٔ اخطار (notification receiver application)** از یک زیرمجموعه از رَویه‌های عمومی همانند کاربرد پاسخ‌دهندهٔ فرمان استفاده می‌کند. گیرندهٔ اخطار ابتدا بایستی ثبت‌نام شده تا PDU های Inform و/یا trap را دریافت کند. هر دو نوع PDU بتوسط یک processPdu primitive دریافت می‌شوند. برای پاسخگوئی به یک Inform PDU از یک returnResponsePdu primitive استفاده می‌شود.

یک **کاربرد جلوبرندهٔ پروکسی (proxy forwarder application)** از Dispatch primitives برای به جلو راندن پیام‌های SNMP استفاده می‌کند. جلوبرندهٔ پروکسی چهار نوع پیام اصلی را پشتیبانی می‌کند:

- پیام‌هائی که شامل انواع PDU از یک کاربرد مولد پیام هستند. جلوبرندهٔ پروکسی موتور SNMP مقصد، و یا نزدیک‌ترین موتور SNMP پائین دست را، تعیین کرده و PDU درخواست مناسب را ارسال می‌دارد.
- پیام‌هائی که شامل انواع PDU از یک کاربرد مولد اخطار هستند. جلوبرندهٔ پروکسی تعیین می‌کند که کدام موتور SNMP بایستی اخطار را دریافت کرده و PDU یا PDUهای مناسب را ارسال می‌کند.
- پیام‌هائی که شامل یک نوع Response PDU هستند. جلوبرندهٔ پروکسی تعیین می‌کند که کدام درخواست یا اخطار ارسال شدهٔ قبل با این پاسخ تطبیق داشته و PDU مناسب را ارسال می‌کند.
- پیام‌هائی که شامل یک گزارش‌اند. Report PDUها ارتباطات موتور- به- موتور SNMPv3 هستند. جلوبرندهٔ پروکسی تعیین می‌کند که کدام درخواست و یا اخطار بجلو رانده شدهٔ قبلی با این گزارش تطبیق داشته و گزارش را به آغازگر درخواست یا اخطار بازپس می‌فرستد.

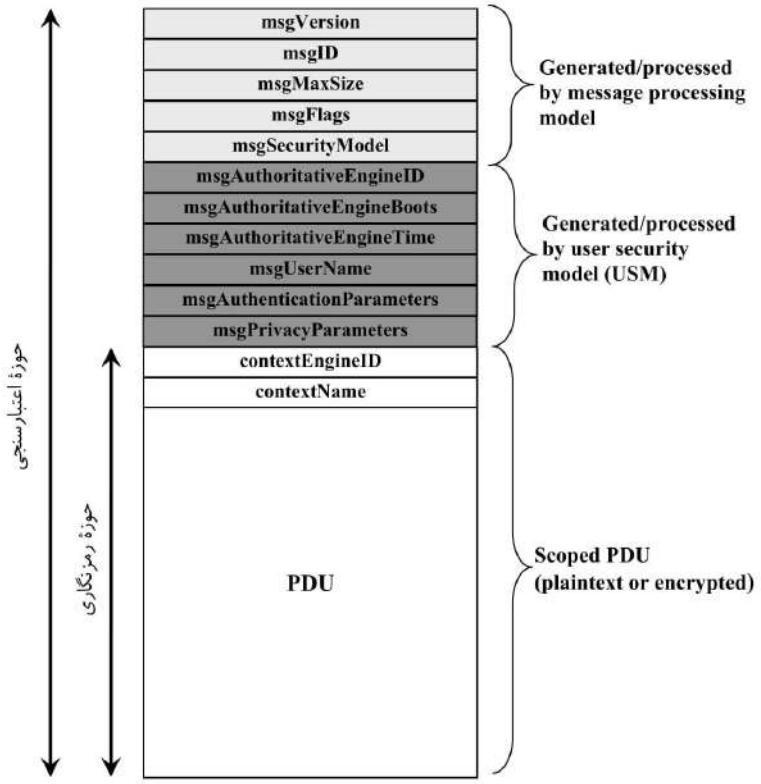
پردازش پیام و مدل امنیتی کاربر

پردازش پیام شامل یک مدل پردازش پیام همه- منظوره و یک مدل امنیتی خاص است. این ارتباط در شکل ۸-۸ نشان داده شده است.

مدل پردازش پیام

RFC 2572 یک مدل پردازش پیام همه- منظوره را تعریف کرده است. این مدل مسئول پذیرش PDUها از حمل‌کننده بوده، آنها را به فرم پیام کپسولی نموده و با استفاده از USM پارامترهای مرتبط با امنیت را در سرآیند پیام‌ها وارد می‌کند. مدل پردازش پیام همچنین پیام‌های ورودی را پذیرفته، USM را برای پردازش پارامترهای مرتبط با امنیت در سرآیند پیام بکار گرفته و PDUهای کپسولی شده را به حمل‌کننده تحویل می‌دهد.

شکل ۹-۸ ساختار پیام را نشان می‌دهد. پنج میدان اولیه بتوسط مدل پردازش‌گر پیام برای پیام‌های خارج‌شونده تولید، و بتوسط مدل پردازش‌گر پیام برای پیام‌های واردشونده پردازش می‌گردند. شش میدان بعدی نشان‌دهندهٔ پارامترهای امنیتی استفاده شده بتوسط USM هستند. بالاخره، PDU به همراه contextEngineID و contextName یک *scoped* PDU را تشکیل می‌دهند که برای پردازش PDU بکار می‌رود.



شکل ۹-۸ فرمت پیام SNMPv3 با USM

پنج میدان اول بشرح زیراند:

- **msgVersion**: برابر 3 (snmpv3) است.
- **msgID**: یک شناسهٔ یکتا که بین دو موجودیت SNMP برای هم‌آهنگ کردن پیام‌های درخواست و پاسخ بکار رفته و پردازش‌گر پیام نیز از آن برای هم‌آهنگ کردن پردازش پیام بتوسط مدل‌های مختلف زیرسیستم‌ها در معماری استفاده می‌کند. محدودهٔ این ID ، صفر تا $۲^{۳۱}-۱$ است.
- **msgMaxSize**: بیانگر اندازهٔ ماکزیمم یک پیام بر حسب اُکتت است که از سوی فرستنده پشتیبانی می‌شود و محدودهٔ آن ۴۸۴ تا $۲^{۳۱}-۱$ است. این اندازهٔ ماکزیمم سِگمنتی است که فرستنده می‌تواند از یک موتور SNMP دیگر بپذیرد (چه یک پاسخ و چه انواع دیگر پیام‌ها).

- **msgFlags**: یک اُکتت که کم اهمیت‌ترین سه بیت آن شامل سه پرچم است: reportableFlag، privFlag و authFlag. اگر reportableFlag = 1 باشدآنگاه یک Report PDU بایستی، در تحت شرایطی که می‌تواند باعث تولید یک Report PDU گردد، به فرستنده بازگردانده شود. وقتی این پرچم صفر است، امکان ارسال یک Report PDU وجود ندارد. پرچم reportableFlag در تمام پیام‌هائی که شامل یک درخواست (GET و SET) و یا یک Inform است برابر 1 و برای پیام‌هائی که شامل یک Response، یک Trap و یا یک Report PDU است برابر 0 قرار داده می‌شود. ReportableFlag به این امر کمک می‌کند که چه زمانی یک Report ارسال شود. از این امر تنها در مواردی استفاده می‌شود که در آن، بخش PDU پیام نتواند کُدگشائی شود (مثلاً وقتی که بعلت کلید ناصحیح، عمل رمزگشائی با شکست مواجه شود). privFlag و authFlag بتوسط فرستنده افراشته شده و برای نشان دادن سطح امنیتی تخصیص داده شده به پیام بکار می‌رود. privFlag = 1 به مفهوم اعمال رمزنگاری و privFlag = 0 به مفهوم اعمال اعتبارسنجی است. تمام ترکیب‌های ممکن بجز authFlag = 0 AND privFlag = 1 قابل اعمال است، یعنی فقط رمزنگاری بدون اعتبارسنجی ممکن نیست.
- **msgSecurityModel**: یک شناسه در محدودهٔ صفر تا $۲^{۳۱}-۱$ است که نشان می‌دهد کدام مدل امنیتی از طرف فرستنده برای آماده‌سازی پیام بکار گرفته شده است و بنابراین تعیین می‌کند که گیرنده بایستی از کدام مدل امنیتی برای پردازش پیام استفاده کند. اندازه‌های رزرو شده شامل 1 برای SNMPv1، 2 برای SNMPv2c (SNMPv2 با تسهیلات جامعه‌ای SNMPv1) و 3 برای SNMPv3 است.

مدل امنیتی کاربر

RFC 2574 مدل امنیتی کاربر User Security Model (USM) را تعریف می‌کند. USM سرویس‌های اعتبارسنجی و محرمانگی را برای SNMP فراهم می‌آورد. USM علی‌الخصوص برای حفظ امنیت در برابر تهدیدهای زیر طراحی شده است:

- **دستکاری اطلاعات**: یک موجودیت ممکن است یک پیام در حال ترانزیت، که بتوسط یک موجودیت معتبر تولید شده است، را طوری تغییر دهد که باعث انجام عملیات مدیریتی غیرمجاز گردد. نتیجهٔ این تهدید این است که یک موجودیت غیرمجاز بتواند هر پارامتر مدیریتی که از جمله شامل پیکربندی، عملیات و حسابرسی است را عوض کند.
- **نقاب‌گذاری**: عملیات مدیریتی غیرمجاز برای یک موجودیت ممکن است بتوسط آن موجودیت،که نقاب یک موجودیت مجاز را به چهره گذاشته است، انجام شود.
- **دستکاری جریان پیام**: SNMP برای کاربرد روی یک پروتکل حمل‌ونقل بدون اتصال طراحی شده است. این تهدید وجود دارد که پیام‌های SNMP جابجاشده، به تأخیرافتاده و یا بازخوانی شده و به عملیات غیرمجاز مدیریتی منجر گردند. برای مثال پیامی برای reboot کردن یک دستگاه ممکن است کپی شده و در آینده مجدداً ارسال گردد.
- **افشا**: یک موجودیت ممکن است مبادلات بین یک مدیر و یک عامل را مشاهده کرده و از این طریق اندازه‌های موضوعات مدیریت شده و همچنین پیشامدهای قابل اخطار را کشف کند. برای مثال، مشاهدهٔ یک مجموعه از فرامین که کلمات عبور را تغییر می‌دهد، یک حمله‌کننده را قادر خواهد ساخت تا کلمات عبور جدید را بدست آورد.

USM برای مقابله با دو تهدید زیر طراحی نشده است:

- **انکار سرویس**: یک حمله‌کننده ممکن است مانع مبادلات بین یک مدیر و یک عامل شود.
- **تحلیل ترافیک**: یک حمله‌کننده ممکن است الگوی عمومی ترافیک بین مدیران و عوامل را مشاهده نماید.

عدم وجود یک مکانیسم مقابله با تهدید انکار سرویس را ممکن است به دو دلیل زیر منطقی دانست: اول اینکه حملات انکار سرویس در بسیاری موارد از خرابی‌های شبکه، که خودکاربردهای مدیریت شبکه باید آن را مدیریت نمایند، قابل تفکیک نیستند. در ثانی یک حملهٔ انکار سرویس احتمالاً باعث از هم گسیختن تمام مبادلات شده و در نتیجه در مقولهٔ تسهیلات امنیتی کل شبکه، نه تنها آنچه در پروتکل مدیریت شبکه وجود دارد، قرار می‌گیرد. در مورد تحلیل ترافیک، بسیاری از پترن‌های ترافیکی مدیریت شبکه قابل پیش‌بینی بوده (مثلاً موجودیت‌های مختلف ممکن است از طریق فرامین SNMP، از سوی یک یا چند ایستگاه مدیریتی، با ضرب‌آهنگ منظم مدیریت شوند) و بنابراین محافظت از آنها در برابر شنود بیگانه امتیاز قابل توجه‌ای دربر ندارد.

توابع رمزنگاری

دو تابع رمزی برای USM تعریف شده است: اعتبارسنجی و رمزنگاری. برای حمایت از این دو تابع، یک موتور SNMP نیاز به دو مقدار دارد: یک کلید خصوصی‌سازی (privKey) و یک کلید اعتبارسنجی (authKey). اندازه‌های متفاوت این دو کلید برای کاربران زیر نگهداری می‌شوند:

- **کاربران محلی**: هر رئیسی (principal) در این موتور SNMP است که برای او عملیات مدیریت مجاز است.
- **کاربران دور**: هر رئیسی در یک موتور SNMP دور است که برای او ارتباطات مورد نیاز است.

این مقادیر از جمله صفات منتسب به کاربر برای هر کاربر مرتبط است. اندازه‌های privKey و authKey از طریق SNMP قابل دسترس نیستند.

USM استفاده از یک و یا هر دو پروتکل اعتبارسنجی HMAC-MD5-96 وHMAC-SHA-96 را مجاز می‌شمارد. HMAC که در فصل ۳ تشریح گردید، از یک تابع درهم‌ساز امن بعلاوهٔ یک کلید سرّی، برای تولید یک کُد اعتبارسنجی پیام، استفاده می‌کند. HMAC-MD5-96 از MD5 بعنوان تابع hash درونی استفاده می‌کند. یک authKey با طول ۱۶ اُکتت (۱۲۸ بیت) برای ورودی الگوریتم HMAC بکار می‌رود. الگوریتم یک خروجی ۱۲۸- بیتی را ایجاد کرده که تنها ۱۲ اُکتت (۹۶ بیت) آن مورد استفاده قرار می‌گیرد. برای HMAC-SHA-96 تابع hash درونی SHA-1 است. در اینجا authKey دارای طول ۲۰ اُکتت (۱۶۰ بیت) است. الگوریتم یک خروجی ۲۰- اُکتتی ایجاد می‌کند که در اینجا نیز فقط از ۱۲ اُکتت آن استفاده می‌شود.

USM از مُود زنجیره‌ای رمز قالبی(CBC) استاندارد رمزنگاری دیتا (DES) استفاده می‌کند. یک privKey با ۱۶ اُکتت طول برای ورودی پروتکل رمزنگاری بکار گرفته می‌شود. اولین ۸ اُکتت (۶۴ بیت) این privKey بعنوان کلید DES مورد استفاده قرار می‌گیرد. چون DES تنها به یک کلید ۵۶- بیتی نیاز دارد، کم اهمیت‌ترین بیت‌های هر اُکتت نادیده گرفته می‌شود. برای مُود CBC یک بردار اولیهٔ(IV) ۶۴- بیتی مورد نیاز است. آخرین ۸ اُکتت privKey شامل اندازه‌ای است که برای تولید IV از آن استفاده می‌گردد.

موتورهای مسئول و غیرمسئول

در هر انتقال پیام، یکی از دو واحد فرستنده یا گیرنده، بر طبق قواعد زیر بعنوان موتور SNMP مسئول انتخاب می‌شود:

- وقتی یک پیام SNMP شامل محموله‌ای است که انتظار یک پاسخ را دارد (مثل Get,GetNext,GetBulk,Set یا Inform PDU)، آنگاه گیرندهٔ چنین پیامی مسئول تلقی می‌گردد.
- وقتی یک پیام SNMP شامل محموله‌ای است که انتظار یک پاسخ را ندارد (مثل یک snmpv2-Trap، Response یا Report PDU)، آنگاه فرستندهٔ چنین پیامی، مسئول تلقی می‌گردد.

بنابراین، برای پیام‌هائی که از سوی یک مولد فرمان ارسال می‌شود و برای پیام‌های Inform که از جانب یک مولد اخطار ارسال می‌گردد، گیرنده مسئول است. برای پیام‌های یک پاسخ‌دهندهٔ فرمان و پیام‌های trap که از سوی یک مولد اخطار ارسال می‌شود، فرستنده مسئول است. این نوع تخصیص مسئولیت دو هدف را دنبال می‌کند:

- بهنگام بودن یک پیام با ساعت موتور مسئول ارزیابی می‌شود. وقتی یک موتور مسئول، یک پیام را ارسال می‌کند (Trap,Response,Report)، اندازهٔ جاری ساعت خود را در آن قرار می‌دهد تا گیرندهٔ غیرمسئول بتواند خود را با آن ساعت هماهنگ سازد. وقتی یک موتور غیرمسئول یک پیام را ارسال می‌کند (Get,GetNext,GetBulk,Set,Inform)، اندازهٔ تخمینی ساعت جاری مقصد را در پیام قرار می‌دهد تا مقصد بتواند بهنگام بودن پیام را ارزیابی نماید.
- عمل محلی‌کردن کلیدها که بعداً تشریح خواهد شد. یک رئیس منفرد را قادر خواهد ساخت تا کلیدهای ذخیره شده در موتورهای متعدد را صاحب شود. این کلیدها در موتور مسئول بنحوی محلی خواهند شد که رئیس منفرد، تنها مسئول یک کلید بوده و از ریسک امنیتی ذخیره‌سازی کپی‌های متعدد کلید در یک شبکهٔ توزیع‌شده جلوگیری شود.

منطقی است که گیرندهٔ PDUهای Command Generator و Inform را، بعنوان موتور مسئول، و در نتیجه کنترل‌کنندهٔ بهنگام بودن پیام دانست. اگر یک response یا trap بتأخیر افتاده و یا بازخوانی شود، خطرآن خیلی جدی نیست، در صورتی که Command Generator PDU، و تا حدودی Inform PDU، منجر به عملیات مدیریتی همچون خواندن و یا تغییر موضوعات MIB می‌گردند. بنابراین تضمین این امر که چنین PDUهائی به تأخیر نیفتاده و یا بازخوانی نشده است مهم بوده زیرا می‌توانند اثرات نامطلوبی داشته باشند.

پارامترهای پیام USM

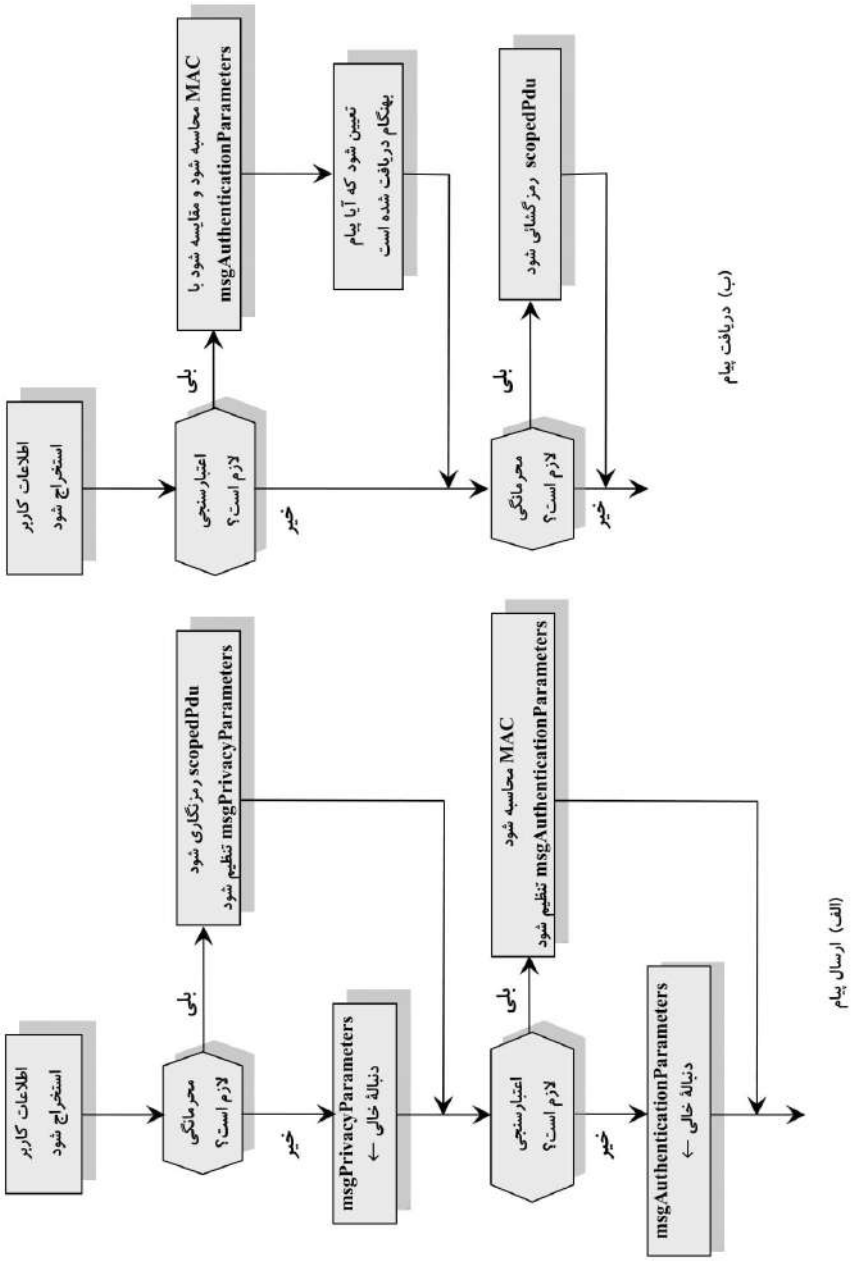
وقتی یک پیام خارج‌شونده، از سوی پردازش‌گر پیام به USM داده می‌شود، USM پارامترهای مرتبط با امنیت را در سرآیند پیام وارد می‌کند. وقتی یک پیام واردشونده از سوی پردازش‌گر پیام به USM داده می‌شود، USM پارامترهای امنیتی موجود در سرآیند پیام را بررسی و پردازش می‌نماید. پارامترهای مرتبط با امنیت به قرار زیراند:

- **msgAuthoritativeEngineID**: شناسهٔ snmpEngineID موتور SNMP مسئول در این مبادله است. بنابراین این مقدار در مورد یک Trap، یک Response و یک Report به مبدأ، و در مورد Get، GetNext، GetBulk، Set و Inform به مقصد اشاره می‌نماید.
- **msgAuthoritativeEngineBoots**: اندازهٔ snmpEngineBoots موتور SNMP مسئول در مبادلهٔ این پیام را نشان می‌دهد. موضوع snmpEngineBoots یک عدد صحیح در بازهٔ صفر تا $۲^{۳۱}-۱$ است که تعداد دفعاتی که موتور SNMP از پیکربندی اولیهٔ خود مجدداً آغازیدن گرفته است را نشان می‌دهد.
- **msgAutoritativeEngineTime**: اندازهٔ snmpEngineTime موتور SNMP مسئول در مبادلهٔ این پیام را نشان می‌دهد. موضوع snmpEngineTime یک عدد صحیح در بازهٔ صفر تا $۲^{۳۱}-۱$ است که تعداد ثانیه‌هائی را نشان می‌دهد که از آخرین باری که موتور SNMP مسئول، موضوع snmpEngineBoots را یک واحد اضافه کرده است، گذشته است. هر موتور SNMP مسئول، مسئولیت بالا بردن اندازهٔ snmpEngineTime خود به میزان یک واحد در هر ثانیه را داراست. یک موتور غیرمسئول، مسئولیت اضافه کردن snmpEngineTime خود، به اندازهٔ یک واحد برای هربار ارتباط با یک موتور مسئول، را بعهده دارد.
- **msgUserName**: کاربری (رئیس) که پیام از سوی او مبادله می‌گردد.
- **msgAuthenticationParameters**: اگر اعتبارسنجی برای این مبادله بکار نرود، خالی است. در غیر این صورت یک پارامتر اعتبارسنجی است. برای تعاریف فعلی USM، پارامتر اعتبارسنجی یک کُد اعتبارسنجی پیام HMAC است.
- **msgPrivacyParameters**: اگر محرمانگی برای این پیام بکار نرود، خالی است. در غیر این صورت یک پارامتر محرمانگی است. برای تعاریف فعلی USM، پارامتر محرمانگی مقداری است که بردار اولیه(IV) در الگوریتم CBC DES را نشان می‌دهد.

شکل ۱۰-۸ عملیات USM را خلاصه کرده است. برای انتقال پیام، اگر لازم باشد، ابتدا رمزنگاری انجام می‌شود. Scoped PDU رمزنگاری شده و در محمولهٔ پیام قرار می‌گیرد و اندازهٔ msgPrivacyParameters با مقداری پر می‌شود که برای تولید IV مورد نیاز است. آنگاه، اگر لازم باشد، اعتبارسنجی انجام می‌شود. تمام پیام که شامل scoped PDU است در ورودی HMAC قرار گرفته و کُد اعتبارسنجی بدست آمده در msgAuthenticationParameters قرار داده می‌شود. برای پیام‌های ورودی، اگر لازم باشد اعتبارسنجی انجام می‌شود. USM ابتدا MAC ورودی را با MAC محاسبه شدهٔ خود مقایسه کرده و اگر این دو با هم برابر باشند، پیام معتبر فرض می‌شود (از یک منبع مجاز صادر شده و در هنگام انتقال تغییر نکرده است). سپس USM کنترل می‌کند که آیا پیام، برابر آنچه بعداً تشریح خواهد شد، در درون یک بازهٔ زمانی مجاز دریافت شده باشد. اگر پیام بهنگام نباشد، غیر معتبر تلقی شده و معدوم خواهد شد. بالاخره اگر scoped PDU رمزنگاری شده باشد، USM رمزگشائی انجام داده و متن ساده را استخراج می‌کند.

مکانیسم USM برای کنترل بهنگام بودن

USM شامل مجموعه‌ای از مکانیسم‌های کنترل بهنگام بودن برای مقابله با تأخیر و یا بازخوانی پیام است. هر موتور SNMP که بتواند با ظرفیت یک موتور مسئول کارکند، بایستی دربرگیرندهٔ دو موضوع snmpEngineBoots و snmpEngineTime باشد که موضوع اخیر به زمان محلی در موتور اشاره دارد. وقتی یک موتور SNMP در ابتدا نصب می‌گردد اندازهٔ این دو موضوع برابر 0 تنظیم می‌گردد. پس از آن، snmpEngineTime در هر ثانیه یک واحد زیاد می‌شود.



شکل ۸-۱۰ پردازش پیام USM

اگر snmpEngineTime به مقدار ماکزیمم خود ۱-2^{31} برسد، snmpEngineBoots یک واحد اضافه می‌شود(مثل اینکه سیستم reboot شده باشد) و snmpEngineTime به صفر برگشته و مجدداً زیاد شدن را ادامه می‌دهد. با استفاده از یک مکانیسم سنکرونیزاسیون، یک موتور غیرمسئول می‌تواند تخمینی از اندازه‌های زمانی هر موتور مسئولی که با او ارتباط دارد را نگهداری کند. این اندازه‌های تخمینی در هر پیام خارج‌شونده قرار داده شده و به موتور مسئول دریافت‌کنندهٔ پیام اجازه می‌دهد تا تعیین کند که آیا پیام واردشونده بهنگام است یا خیر.

مکانیسم سنکرونیزاسیون بصورت زیر کار می‌کند. یک موتور غیرمسئول برای هر موتور SNMP مسئول که با او سروکار دارد، کپی سه مقدار را نگهداری می‌کند:

- **snmpEngineBoots**: آخرین اندازهٔ snmpEngineBoots برای هر موتور مسئول.
- **snmpEngineTime**: تخمین موتور غیرمسئول از snmpEngineTime موتور مسئول دور. این اندازه با موتور مسئول دور بصورتی که بعداً تشریح خواهد شد هم‌آهنگ می‌گردد. بین هم‌آهنگ‌سازی‌ها، این مقدار به میزان یک واحد در ثانیه افزایش یافته تا یک هم‌آهنگی نسبی با موتور مسئول دور بوجود آید.
- **latestReceivedEngineTime**: بزرگترین اندازهٔ msgAuthoritativeEngineTime که بتوسط این موتور از موتور مسئول دور دریافت گردیده است. این مقدار هربار که اندازهٔ بزرگتری از msgAuthoritativeEngineTime دریافت می‌گردد، بروزرسانی می‌شود. هدف این متغیر، حفاظت در برابر حملهٔ بازخوانی پیامی است که تخمین موتور SNMP غیرمسئول از snmpEngineTime را از جلورفتن بازدارد.

یک مجموعه از این سه متغیر، برای هر موتور مسئول دور که با این موتور ارتباط دارد، نگهداری می‌شود. از نظر منطقی، اندازه‌ها در نوعی حافظهٔ موقت نگهداری می‌شوندکه فهرستی از snmpEngineID هر موتور مسئول و اندازه‌های اخیر در آن ضبط شده است.

برای اینکه موتورهای غیرمسئول بتوانند هم‌آهنگی زمانی را حفظ کنند، هر موتور مسئول اندازهٔ boot و time را در کنار اندازهٔ snmpEngineID در هر پیام خروجی Response، Report و Trap خود در میدان‌های msgAuthoritativeEngineBoots، msgAuthoritativeEngineTime و msgAuthoritativeEngineID قرار می‌دهد. اگر پیام معتبر بوده و در پنجرهٔ زمانی معقول دریافت شود، آنگاه موتور غیرمسئول گیرنده، متغیرهای محلی خود (snmpEngineBoots، snmpEngineTime و latestReceivedEngineTime) برای آن موتور دور را بر طبق ضوابط زیر بروز می‌رساند:

۱- یک بروزرسانی وقتی انجام می‌شود که حداقل یکی از دو شرط زیر صحیح باشد:

- (msgAuthoritativeEngineBoots > snmpEngineBoots) **یا**
- (msgAuthoritativeEngineBoots = snmpEngineBoots)] **و**

[(msgAuthoritativeEngineTime > latestReceivedEngineTime)

اولین شرط می‌گویدکه بروزرسانی در صورتی باید واقع شود که اندازهٔ boot موتور مسئول از آخرین بروزرسانی افزایش یافته باشد. شرط دوم می‌گویدکه اگر اندازهٔ boot افزایش نیافته باشد، بروزرسانی فقط در صورتی باید انجام شود که زمان موتور ورودی بیشتر از زمان قبلی واردشدهٔ همین موتور باشد. زمان موتور ورودی در صورتی کمتر از آخرین زمان دریافت‌شدهٔ همین موتور خواهد بود که دو پیام ورودی خارج از نظم دریافت شده باشند. این امر هم بصورت عادی ممکن است واقع شود و هم ممکن است به دلیل یک حملهٔ بازخوانی صورت پذیرد و درهرحال موتور گیرنده بروزرسانی نخواهد شد.

۲- اگر بروزرسانی مجاز تشخیص داده شود، آنگاه تغییرات زیر انجام خواهد شد:

- اندازهٔ snmpEngineBoots به اندازهٔ msgAuthoritativeEngineBoots تغییر می‌یابد.
- اندازهٔ snmpEngineTime به اندازهٔ msgAuthoritativeEngineTime تغییر می‌یابد.
- اندازهٔ latestReceivedEngineTime به اندازهٔ msgAuthoritativeEngineTime تغییر می‌یابد.

اگر منطق را بچرخانیم می‌بینیم که اگر msgAuthoritativeEngineBoots < snmpEngineBoots باشد، آنگاه هیچ بروزرسانی انجام نخواهد شد. چنین پیامی نامعتبر تلقی شده و بایستی نادیده گرفته شود. اگر snmpEngineBoots = msgAuthoritativeEngineBoots بوده ولی msgAuthoritativeEngineTime < latestReceivedEngineTime باشد، بازهم بروزرسانی انجام نخواهد شد. در این مورد ممکن است پیام معتبر بوده ولی نظم آن بهم خورده باشد که در این صورت بروزرسانی snmpEngineTime منطقی نیست.

توجه شود که سنکرونیزاسیون تنها وقتی انجام خواهد شد که سرویس اعتبارسنجی برای این پیام بکار گرفته شده باشد و پیام بتوسط HMAC معتبر تلقی گردد. این محدودیت ضروری بوده زیرا فضای اعتبارسنجی شامل msgAuthoritativeEngineID، msgAuthoritativeEngineBoots و msgAuthoritativeEngineTime بوده و بنابراین از اعتبار این سه مقدار اطمینان حاصل خواهد شد.

SNMPv3 این قید را بوجود می‌آورد که برای اجتناب از حملات تأخیر و بازخوانی، پیام بایستی در یک پنجرهٔ زمانی معقول دریافت شده باشد. پنجرهٔ زمانی بایستی تا حد ممکن کوچک انتخاب گردد و درآن دقت ساعت‌های فرستنده و گیرنده، تأخیرهای رفت و برگشت پیام و دورهٔ تناوب هماهنگی ساعت‌ها ملحوظ شده باشد. اگر پنجرهٔ زمانی خیلی کوچک انتخاب شود، پیام‌های معتبر، بصورت پیام‌های نامعتبر جلوه خواهند کرد. از سوی دیگر بزرگ بودن پنجرهٔ زمانی، آسیب‌پذیری به تأخیرهای بداندیشانهٔ پیام را افزایش خواهد داد.

مورد مهم‌تر یک گیرندهٔ مسئول را بیشتر بررسی می‌کنیم. آزمایش بهنگام بودن در مورد یک گیرندهٔ غیرمسئول کمی متفاوت خواهد بود. با هر پیام ورودی که معتبر تشخیص داده شود و msgAuthoritativeEngineID آن با snmpEngineID این موتور برابر باشد، موتور اندازهٔ msgAuthoritativeEngineBoots و msgAuthoritativeEngineTime پیام ورودی را با اندازه‌های snmpEngineBoots و snmpEngineTime نگهداری شده در این موتور مطابقت خواهد داد. پیام ورودی در صورتی خارج از پنجرهٔ زمانی تشخیص داده می‌شود که یکی از شرایط زیر صحیح باشد:

- snmpEngineBoots = 2^{31}-1 **یا**
- msgAuthoritativeEngineBoots≠snmpEngineBoots **یا**
- اندازهٔ msgAuthoritativeEngineTime از اندازهٔ snmpEngineTime بمیزان ۱۵۰± ثانیه تفاوت داشته باشد.

شرط اول می‌گوید که اگر snmpEngineBoots دارای اندازهٔ ماکزیمم خود باشد، هیچ پیام ورودی معتبر تلقی نخواهد شد. شرط دوم می‌گوید که یک پیام بایستی دارای زمان boot مساوی با زمان boot موتور محلی باشد. برای مثال اگر موتور محلی reboot شده و موتور دور از زمان reboot با موتور محلی هم‌آهنگ نشده باشد، پیام‌های واردشده از آن موتور دور معتبر تلقی نخواهد شد. شرط آخر می‌گوید که زمان پیام‌های واردشده بایستی از زمان محلی منهای ۱۵۰ ثانیه بیشتر، و از زمان محلی باضافهٔ ۱۵۰ ثانیه کمتر باشد.

اگر یک پیام خارج از پنجرهٔ زمانی قرار داشته باشد، آنگاه آن پیام نامعتبر تلقی شده و یک نمایش خطا (notInTimeWindow) به مدول فرستندهٔ پیام برگردانده خواهد شد.

در اینجا نیز همانندکنترل سنکرونیزاسیون، کنترل بهنگام بودن تنها وقتی انجام می‌شود که سرویس اعتبارسنجی فعال بوده و پیام معتبر تشخیص داده شود تا از صحت سرآیندهای پیام اطمینان حاصل شده باشد.

محلی کردن کلید

یکی از لازمه‌های استفاده از سرویس‌های اعتبارسنجی و محرمانگی SNMPv3 این است که برای هر ارتباط بین یک رئیس مستقر در یک موتور غیرمسئول و یک موتور مسئول دور، یک کلید سرّی اعتبارسنجی و یک کلید سرّی رمزنگاری بین این دو به اشتراک گذاشته شود. این کلیدها، یک کاربر در یک موتور غیرمسئول (معمولاً یک سیستم مدیریت) را قادر می‌سازد تا اعتبارسنجی و محرمانگی را با سیستم‌های مسئول دور (معمولاً سیستم‌های عامل) که کاربر آنها را مدیریت می‌کند، عملیاتی نماید. RFC 2574 روش خلق، بروزرسانی و مدیریت این کلیدها را بیان می‌کند.

برای اینکه وظیفهٔ رؤسا از نظر مدیریت کلید سهل‌تر شود، هر رئیس لازم است که تنها یک کلید اعتبارسنجی و تنها یک کلید رمزنگاری را نگهداری نماید. این کلیدها در یک MIB ذخیره نشده‌اند و از طریق SNMP قابل دست‌یابی نیستند. در این قسمت ابتدا به نحوهٔ تولید این کلیدها از یک کلمهٔ عبور نگاه می‌کنیم. سپس به مفهوم محلی کردن کلید می‌پردازیم که یک رئیس را قادر می‌سازد تا در حالی که بطور محلی تنها صاحب یک کلید اعتبارسنجی و یک کلید رمزنگاری است، اما با هر موتور دور، یک کلید اعتبارسنجی یکتا و یک کلید رمزنگاری یکتا را به اشتراک بگذارد. این دو تکنیک ابتدا در [BLUM97a] پیشنهاد شدند.

یک کاربر نیاز به یک کلید محرمانگی ۱۶- اُکتتی و یک کلید اعتبارسنجی ۱۶ یا ۲۰- اُکتتی دارد. برای کلیدهائی که صاحبان آنها انسان هستند، مطلوب این است که کاربر بتواند از کلمهٔ عبوری که قابل خواندن بتوسط انسان باشد، و نه از کلیدی که از یک رشته بیت درست شده است، استفاده کند. بهمین جهت RFC 2574 یک الگوریتم برای نگاشت یک کلمهٔ عبور به یک کلید ۱۶ یا ۲۰- اُکتتی را تعریف کرده است. USM هیچ محدودیتی برای خود کلمهٔ عبور قائل نشده است اما سیاست‌های مدیریتی محلی بایستی کاربران را به استفاده از کلمات عبوری که بسهولت قابل حدس زدن نباشند مقید سازد. تولید کلید از کلمهٔ عبور بصورت زیر انجام می‌شود:

- کلمهٔ عبور کاربر را گرفته و با تکرار اندازهٔ کلمهٔ عبور به هر مقدار که لازم است و قطع کردن آخرین اندازهٔ تکرارشده در صورت لزوم، یک رشته بیت با طول 2^{20} اُکتت (۱,۰۴۸,۵۷۶ اُکتت) بنام digest0 ایجاد کنید. مثلاً یک کلمهٔ عبور ۸- کاراکتری (2^3 اُکتت)، بایستی 2^{17} بار با خودش جمع رشته‌ای شود تا digest0 را بوجود آورد.
- اگر یک کلید ۱۶- اُکتتی مورد نیاز است، تابع درهم‌ساز MD5 مرتبط با digest0، را محاسبه کرده تا digest1 حاصل شود. اگر یک کلید ۲۰- اُکتتی مورد نیاز است، تابع درهم‌ساز SHA-1 مرتبط با digest0، را محاسبه کرده تا digest1 بدست آید. digest1 همان کلید کاربر است.

یک حسن این روش این است که حملهٔ همه جانبهٔ لغت‌نامه‌ای که در آن یک دشمن تلاش می‌کند تا کلمات عبور مختلف را امتحان کرده، کلید مرتبط با هر کدام را ساخته و آنگاه این کلید را روی دیتای اعتبارسنجی یا رمزنگاری موجود امتحان کند را بشدت کند می‌سازد. برای مثال اگر یک دشمن یک پیام معتبر را استراق سمع کند، او می‌تواندبا کلیدهای

مختلف اندازهٔ HMAC را محاسبه نماید. اگر بین این مقدار و دیتای موجود تطبیقی یافت شود، دشمن می‌تواند تصور کند که کلمهٔ عبور کشف شده است. عملیاتی که در بالا تشریح شد، زمان لازم برای چنین حمله‌ای را بطور چشمگیری افزایش می‌دهد.

مزیت دیگر این تکنیک این است که کلیدهای کاربر را از نوع سیستم مدیریت شبکه (NMS) مجزا می‌کند. هیچ NMS ای مجبور نیست که اندازهٔ کلیدهای کاربر را نگهداری کند. در عوض هروقت لازم است، کلید کاربر از روی کلمهٔ عبور او ساخته می‌شود. [BLUM97b] ملاحظات زیرکه انگیزهٔ استفاده از کلمهٔ عبور مستقل از NSM است را لیست کرده است:

- اگر قرار باشد بجای تولید کلید از روی کلمهٔ عبور، خود کلید در جائی ذخیره گردد، یک روش این است که کلیهٔ کلیدهای سرّی در یک مخزن نگهداری شود. چنین روشی روی قابلیت اعتماد کل سیستم تأثیر سوء داشته زیرا ممکن است که خود مخزن در زمان مورد نیاز در دسترس نبوده و عیب‌یابی سیستم را غیرممکن سازد.
- از سوی دیگر اگر برای رفع مشکل بالا، مخازن متعددی ایجاد شوند، این امر با فراهم آوردن اهداف متعدد حمله برای دشمنان، امنیت را شکننده‌تر خواهد کرد.
- اگر یک مخزن متمرکز و یا چندین مخزن پراکنده مورد استفاده قرار گیرد، آنها را بایستی در محل‌های امن نگهداری کرد. این امر می‌تواند فرصت ایجاد یک "forward camp" در خلال عملیات اطفاء حریق (یعنی عیب‌یابی سیستم در هنگامی که بخش‌های غیرقابل پیش‌بینی از شبکه غیرعملیاتی شده و/ یا برای مدت زمان غیرقابل پیش‌بینی در دسترس نیستند) را کاهش دهد.

یک کلمهٔ عبور منفرد می‌تواند برای تولید یک کلید منفرد، هم برای اعتبارسنجی و هم رمزنگاری، بکار رود. روش امن‌تر این است که از دو کلمهٔ عبور استفاده شود که یکی برای تولید کلید اعتبارسنجی و دیگری برای تولید کلید رمزنگاری بکار رود.

یک کلید محلی شده، در RFC 2574، بصورت یک کلید سرّی مشترک بین یک کاربر و یک موتور SNMP مسئول تعریف شده است. هدف این است که کاربر لازم باشد تنها یک کلید (یا دو کلید، اگر اعتبارسنجی ومحرمانگی مورد نیاز است) را نگهداری کرده و بنابراین تنها لازم باشد فقط یک (یا دو) کلمهٔ عبور را بخاطر بسپارد. اما در واقع آسرار به‌اشتراک گذاشته شده بین یک کاربر بخصوص با هر موتور SNMP مسئول متفاوت است. عملی که بتوسط آن یک کلید منفرد کاربر به چندین کلید متفاوت یکتا که هرکدام مربوط به یک موتور SNMP دور هستند تبدیل می‌شود را محلی کردن کلید (key localization) گویند.[BLUM97a] انگیزه‌های این استراتژی را ذکر کرده که در اینجا آنها را بصورت خلاصه بیان می‌کنیم.

برای مدیریت کلید اهداف زیر را می‌توان تعریف نمود:

- هر سیستم عامل SNMP در یک شبکهٔ گسترده، دارای یک کلید یکتا برای هر کاربر معتبری است که می‌خواهد آن را مدیریت نماید. اگر کاربران متعددی بعنوان مدیران معتبر شناخته می‌شوند، عامل دارای یک کلید اعتبارسنجی یکتا و یک کلید رمزنگاری یکتا برای هر کاربر است. در نتیجه اگر کلید یک کاربر لو برود، کلیدهای کاربران دیگر لورفته نخواهند بود.
- کلیدهای یک کاربر برای عامل‌های مختلف متفاوت‌اند. بنابراین اگر یک عامل لو برود، تنها کلیدهای کاربر برای آن عامل لورفته و کلیدهای متعلق به عاملان دیگر لورفته نخواهند بود.
- مدیریت شبکه از هر نقطهٔ شبکه، صرف نظر از پیکربندی سیستم مدیریت شبکه (NMS)، امکان‌پذیر است. این امر به کاربر اجازه می‌دهد تا عملیات مدیریتی را از هر ایستگاه مدیریت انجام دهد. این قابلیت بتوسط الگوریتم تبدیل کلمهٔ عبور- به- کلید که قبلاً تشریح گردید امکان‌پذیر است.

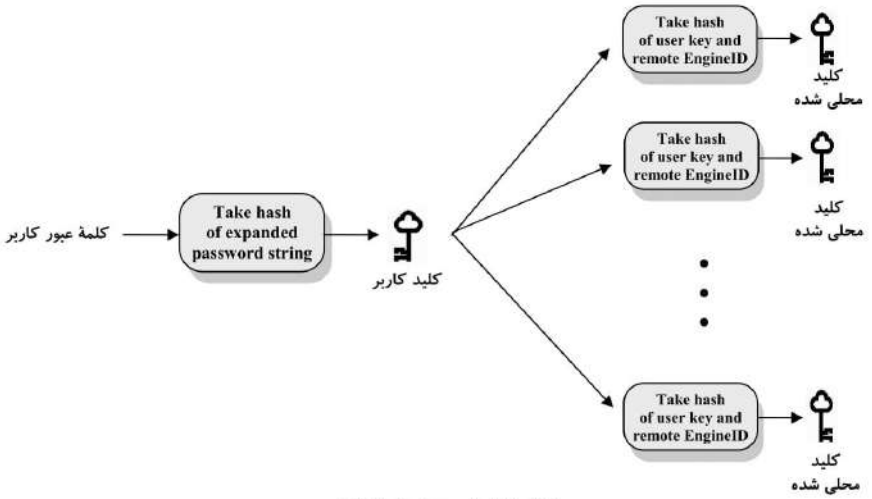
همچنین می‌توان به موارد زیر که بایستی از آنها اجتناب شود اشاره کرد:

- یک کاربر لازم است تعداد زیادی کلید را بخاطر سپارد (یا مدیریت کند) که این تعداد با اضافه شدن عوامل جدید مدیریت شده، بسرعت رشد می‌کند.
- یک دشمن که کلید مرتبط با یک عامل را کشف کند، قادر خواهد بود تا خود را جای هر عامل دیگر برای هر کاربر، و جای هر کاربر برای هر عامل دیگر جا بزند.

برای رسیدن به اهداف و ملاحظات ذکرشده، یک کلید منفرد کاربر بتوسط یک تابع یک- طرفهٔ برگشت‌ناپذیر (یعنی یک تابع hash) به فرم کلیدهای محلی شدهٔ متفاوت برای موتورهای مسئول متفاوت (عوامل متفاوت) درمی‌آید. روش کار چنین است:

- رشته بیت digest2 را از جمع رشته‌ای digest1 (قبلاً تشریح گردید)، اندازهٔ snmpEngineID موتور مسئول، و digest1 تولید کنید.
- اگر یک کلید ۱۶- اُکتتی مورد نیاز است، تابع درهم‌ساز MD5 رشتهٔ digest2 را بسازید. اگر یک کلید ۲۰- اُکتتی مورد نیاز است تابع درهم‌ساز SHA-1 آن را حساب کنید. خروجی بدست‌آمده، کلید محلی شدهٔ کاربر خواهد بود.

کلید محلی شده آنگاه می‌تواند در سیستم عامل به فرم امنی پیکربندی شود. نظر به یک- طرفه بودن MD5 و SHA-1 برای یک دشمن امکان نخواهد داشت که حتی در صورت کشف یک کلید محلی شده، کلید منفرد کاربر را پیدا کند. شکل ۱۱-۸ محلی کردن کلید را نشان می‌دهد.



شکل ۱۱-۸ محلی کردن کلید

کنترل دست‌یابی مبتنی بر منظر (View-Based)

کنترل دست‌یابی یک عمل امنیتی است که در سطح PDU انجام می‌شود. یک دستورالعمل کنترل دست‌یابی، مکانیسم‌هائی را برای تعیین اینکه آیا دست‌یابی به یک موضوع مدیریت شده در یک MIB محلی بتوسط یک رئیس دور مجاز است یا خیر تعریف می‌کند. مکانیسم‌های متعدد و متصوری را می‌توان برای کنترل دست‌یابی تعریف کرد. اسناد SNMPv3 مدل کنترل دست‌یابی view-based access control model (VACM) را تعریف می‌کند. VACM خود از یک MIB که خط‌مشی کنترل دست‌یابی برای این عامل را تعریف کرده است، استفاده کرده و استفاده از پیکربندی دور را ممکن می‌سازد.

VACM دارای دو مشخصهٔ مهم است:

- VACM تعیین می‌کند که آیا دست‌یابی به یک موضوع مدیریت شده در یک MIB محلی بتوسط یک رئیس دور مجاز است.
- VACM از یک MIB استفاده می‌کند که
  - خط‌مشی کنترل دست‌یابی برای این عامل را تعریف می‌کند.
  - استفاده از پیکربندی دور را ممکن می‌سازد.

عناصر یک مدل VACM

پنج عنصر که VACM را تشکیل می‌دهنددر RFC 2575 تعریف شده‌اند: گروه‌ها، سطح امنیتی، مقوله‌ها، منظرMIB و خط مشی دست‌یابی.

یک **گروه** (group) مجموعه‌ای از هیچ یا چند جفت<securityModel,securityName> را تعریف می‌کند که موضوعات مدیریتی SNMP می‌تواند از سوی آنها مورد دست‌یابی قرار گیرد. یک securityName به یک رئیس اشاره دارد و حقوق دست‌یابی تمام رؤسا در یک گروه یکسان است. هر گروه یک نام groupName یکتا دارد. مقولهٔ گروه یک ابزار سودمند برای طبقه‌بندی مدیران برحسب حقوق دست‌یابی آنهاست. بعنوان مثال، تمام مدیران ردهٔ بالا ممکن است دارای مجموعه‌ای از حقوق دست‌یابی بوده که با مجموعهٔ حقوق دست‌یابی مدیران میانی متفاوت باشد.

هر ترکیبی از securityModel و securityName حداکثر می‌تواند متعلق به یک گروه باشد. یعنی برای این عامل، رئیسی که ارتباطات او بتوسط یک securityModel خاص حفاظت می‌شود تنها می‌تواند در یک گروه جای گیرد.

حقوق دست‌یابی یک گروه می‌تواند برحسب **سطح امنیتی** (security level) پیام درخواست، متفاوت باشد. برای مثال، یک عامل ممکن است دست‌یابی read-only از یک پیام درخواست اعتبارسنجی نشده را بپذیرد ولی برای دست‌یابی به write، اعتبارسنجی را طلب نماید. علاوه برآن برای موضوعات حساس معینی، عامل ممکن است استفاده از سرویس محرمانگی برای درخواست و پاسخ آن را طلب نماید.

یک **مقوله MIB** (MIB context) یک زیرمجموعه با یک نام مشخص در یک MIB محلی است. ایجاد مقوله‌ها یک روش مفید برای گردآوردن موضوعاتی با سیاست‌های دست‌یابی یکسان تحت عنوان یک نام است.

مقوله، مفهومی مرتبط با کنترل دست‌یابی است. وقتی یک ایستگاه مدیریت با یک عامل ارتباط برقرارکرده تا به اطلاعات مدیریتی موجود در آن عامل دست یابد، آنگاه تعامل بین یک مسئول مدیریت با موتور SNMP عامل ایجاد می‌گردد و امتیازات کنترل دست‌یابی متعلق به این رئیس در این مقوله در یک منظر MIB گنجانده شده است. مقوله‌ها دارای مشخصه‌های کلیدی زیراند:

- یک موجودیت SNMP که بطور یکتا بتوسط یک contextEngineID مشخص می‌شود و ممکن است بیش از یک مقوله را نگهداری کند.
- یک موضوع و یا موردی از یک موضوع ممکن است در بیش از یک مقوله ظاهر شود.
- وقتی مقوله‌های متعددی وجود دارد، برای شناسائی موردی از یک موضوع، contextName و contextEngineID آن علاوه بر نوع موضوع و مورد آن بایستی شناسائی گردند.

اغلب علاقه‌مندیم تا دست‌یابی یک گروه خاص به زیرمجموعه‌ای از موضوعات مدیریت شده در یک عامل را محدود سازیم. برای رسیدن به این هدف، دست‌یابی به یک مقوله بتوسط **منظر MIB (view)** صورت می‌پذیرد که مجموعهٔ خاصی از موضوعات مدیریت شده (و بطور اختیاری موارد یک موضوع) را تعریف می‌کند. VACM از یک تکنیک قدرتمند و قابل انعطاف برای تعریف منظر MIB استفاده کرده که بر مفاهیم زیرشاخه‌های منظر و خانواده‌های منظر استوار است. منظر MIB بصورت یک مجموعه از زیرشاخه‌ها تعریف می‌شود که هر زیرشاخه یا در منظر وجود داشته و یا در آن موجود نیست.

موضوعات مدیریت شده در یک پایگاه داده محلی بصورت سلسله مراتبی و یا درختی و بر اساس شناسه‌های موضوعات سازمان می‌یابند. این پایگاه دادهٔ محلی شامل یک زیرمجموعه از انواع موضوعات تعریف شده بر اساس Internet-standard Structure of Management Information (SMI) بوده و شامل موارد موضوعاتی است که شناسه‌های آن منطبق با قراردادهای SMI است.

SNMPv3 شامل مفهومی به نام زیرشاخه است. یک زیرشاخه، بطور ساده یک گره در یک سلسله مراتب نام‌های MIB باضافهٔ تمام عناصر مرتبط با آن است. بطور رسمی‌تر، یک زیرشاخه ممکن است بصورت مجموعهٔ تمام موضوعات و موارد مختلف آنها بوده که پیشوند مشترک ASN.1 OBJECT IDENTIFIER در نام‌های آنان وجود دارد. طویل‌ترین پیشوند مشترک تمام موارد در یک زیرشاخه، شناسهٔ موضوع گرهٔ مادر در آن زیرشاخه است.

به هر مورد موجود در vacmAccessTable سه منظر MIB مرتبط است که یکی مربوط به دست‌یابی خواندن، یکی مربوط به دست‌یابی نوشتن و سومی مربوط به دست‌یابی اخطار است. هر منظر MIB شامل یک مجموعه از زیرشاخه‌های منظر است. هر زیرشاخهٔ منظر در منظر MIB یا حضور دارد و یا حضور ندارد. یعنی منظر MIB یا شامل تمام موارد موضوعی در زیرشاخه هست یا نیست. علاوه بر آن یک view mask برای این امر تعریف شده تا میزان اطلاعات پیکربندی لازم برای کنترل دست‌یابی به موارد کم‌ارزش (مثلاً کنترل دست‌یابی در سطح موردی یک موضوع) را کاهش دهد.

VACM یک موتور SNMP را قادر می‌سازد تا طوری پیکربندی شود که یک مجموعهٔ خاص از حقوق دست‌یابی که یک **خط‌مشی دست‌یابی** را تشکیل می‌دهد گردهم آورد. تعیین دست‌یابی به فاکتورهای زیر وابسته است:

- یک **رئیس** که درخواست دست‌یابی می‌کند. VACM یک عامل را قادر می‌سازد تا امتیازات دست‌یابی متفاوت برای کاربران مختلف قائل شود. برای مثال، یک سیستم مدیریت مسئول پیکربندی کل شبکه ممکن است امتیاز دست‌یابی وسیع تغییر اقلام در یک MIB محلی را داشته باشد درحالی‌که یک مدیر میانی با مسئولیت پایش، تنها حق read-only داشته و شاید تنها حق داشته باشد که به بخشی از MIB محلی دست یابد. همانطور که قبلاً بحث شد، رؤسا در گروه‌های مختلف دسته‌بندی شده و خط‌مشی دست‌یابی در رابطه با یک گروه تعریف می‌شود.
- **سطح امنیتی** که پیام درخواست SNMP در آن سطح قرار دارد. معمولاً یک عامل نیازمند به استفاده از اعتبارسنجی برای پیام‌هائی که شامل درخواست set (عمل نوشتن) هستند می‌باشد.

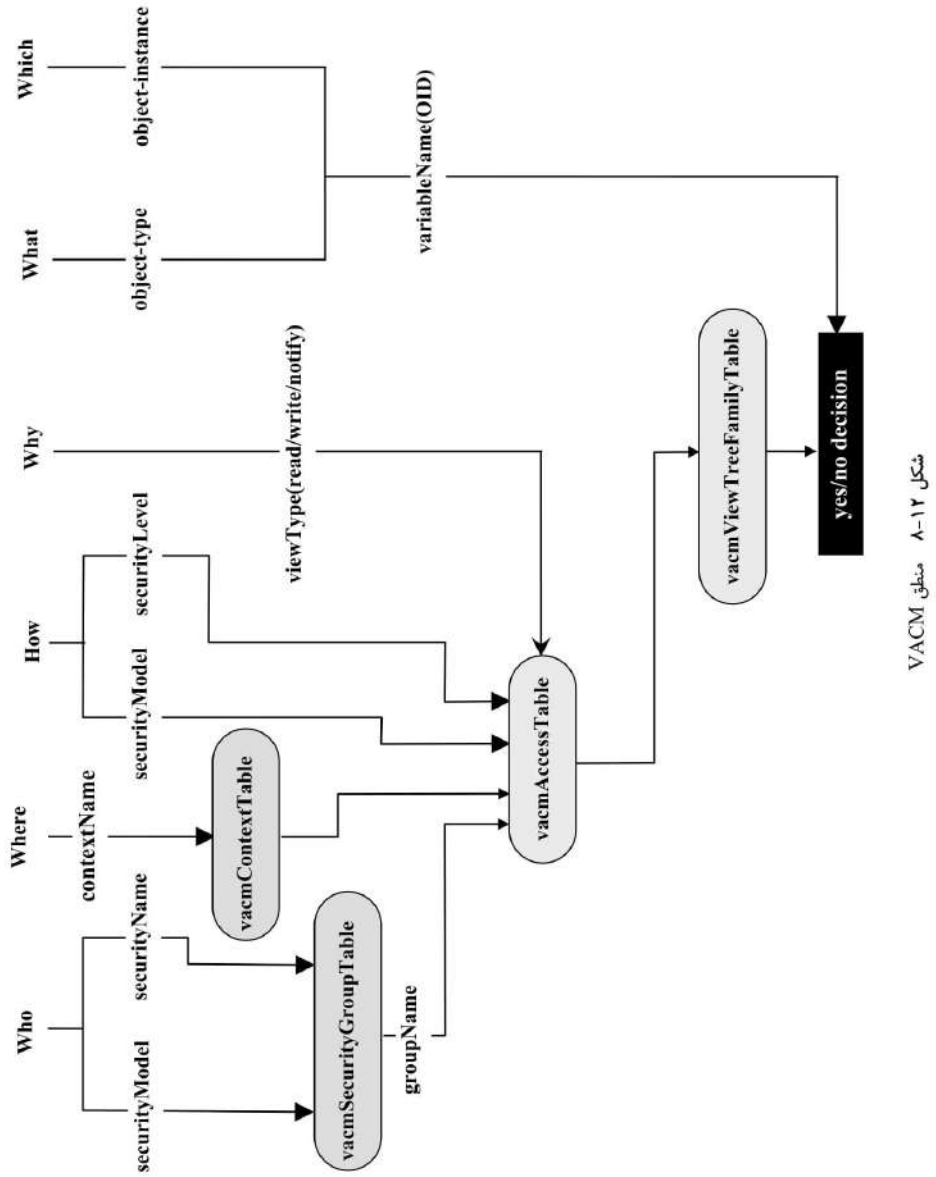
- **مدل امنیتی** استفاده شده برای پردازش پیام درخواست است. اگر مدل‌های امنیتی متعددی در یک عامل پیاده‌سازی شده باشد، عامل ممکن است طوری پیکربندی شود که به پیام‌هائی که با مدل‌های مختلف پردازش شده‌اند، سطوح دست‌یابی متفاوت تخصیص دهد. مثلاً اگر درخواست از طریق USM رسیده باشد اقلام معینی قابل دست‌یابی بوده در حالی که اگر مدل امنیتی SNMPv1 باشد این اقلام قابل دست‌یابی نباشند.
- **مقولهٔ MIB** برای این درخواست.
- **نوع موضوع** که دست‌یابی به آن درخواست شده است. برخی موضوعات، اطلاعات حیاتی‌تر و یا حساس‌تری را نسبت به بقیه نگاه می‌دارند و بنابراین خط‌مشی دست‌یابی بایستی وابسته به نوع موضوع مورد تقاضا باشد.
- **نوع دست‌یابی** تقاضا شده (خواندن، نوشتن، هشدار). خواندن، نوشتن و هشدار عملیات مدیریتی متمایز بوده و خط‌مشی‌های کنترل دست‌یابی مختلف ممکن است به هریک از آنها اعمال شود.

رَوَند کنترل دست‌یابی

یک کاربرد SNMP، VACM را از طریق isAccessAllowed primitive با پارامترهای ورودی securityModel، securityName، securityLevel، viewType، contextName و variableName بکار می‌گیرد. تمام این پارامترها برای تصمیم‌گیری نسبت به کنترل دست‌یابی لازم‌اند. بعبارت دیگر زیرسیستم کنترل دست‌یابی بنحوی تعریف شده است تا یک ابزار بسیار انعطاف‌پذیر برای پیکربندی کنترل دست‌یابی در عامل از طریق شکستن تصمیمات کنترل دست‌یابی به شش متغیر مجزا بوجود آورد.

شکل ۱۲-۸ که از شکلی در RFC 2575 اقتباس شده است، یک روش مفید برای مشاهدهٔ متغیرهای ورودی بوده و نشان می‌دهد که چگونه جداول مختلف در یک VACM MIB در اخذ تصمیمات کنترل دست‌یابی دخالت می‌کنند.

- **who**: ترکیب securityModel و securityName، معرفی‌کنندهٔ چه کسی در این عملیات است. این مقوله در واقع یک رئیس را تعریف کرده که ارتباطات او بتوسط یک securityModel حفاظت می‌شود. ترکیب بالا در این موتور SNMP حداکثر به یک گروه تعلق دارد. vacmSecurityToGroupTable با داشتن securityModel و securityName، groupName را استخراج می‌کند.
- **where**: contextName تعیین می‌کند که موضوع مدیریتی مطلوب در *کجا* بایستی یافت شود. vacmContextTable شامل یک لیست از contextNameهای شناخته شده است.
- **how**: ترکیب securityModel و securityLevel تعریف می‌کند که *چگونه* درخواست ورودی و Inform PDU حفاظت شده‌اند. ترکیب who، where و how هیچ و یا یک قلم از vacmAccessTable را مشخص می‌کند.
- **why**: viewType مشخص می‌سازد که *چرا* دست‌یابی درخواست شده است: برای خواندن، نوشتن و یا هشدار. مورد انتخاب شده در vacmAccessTable شامل یک MIB viewName برای هریک از سه نوع عمل بالاست و viewType برای انتخاب یک viewName مشخص است. این viewName، منظرMIB مناسب از vacmViewTreeFamilyTable را انتخاب می‌کند.
- **what**: variableName یک شناسهٔ موضوعات است که پیشوند آن یک نوع موضوع مشخص و پسوند آن یک مورد از موضوع مشخص را تعریف می‌کند. نوع موضوع نشان می‌دهد که چه نوع اطلاعات مدیریتی درخواست شده است.



شکل ۸-۱۲ منطق VACM

- **which**: مورد موضوع نشان می‌دهد که *کدام* قلم خاص اطلاعات درخواست شده است.

بالاخره variableName با منظر MIB استخراج شده مقایسه می‌گردد. اگر variableName با یک عنصر موجود در منظر MIB تطبیق داشته باشد، آنگاه دست‌یابی اعطا می‌گردد.

انگیزش

مفاهیمی که VACM را می‌سازند بنظر می‌رسد که منجر به تعریف پیچیده‌ای از کنترل دست‌یابی می‌گردند. انگیزهٔ معرفی چنین مفاهیمی، روشن ساختن ارتباطات موجود در کسب اطلاعات مدیریت و به حداقل رساندن نیازهای ذخیره‌سازی و پردازش در یک عامل است. برای فهم چنین انگیزه‌هائی به مورد زیر توجه کنید. در SNMPv1 مفهوم جامعه برای معرفی اطلاعات مرتبط با امنیت بکار می‌رود:

- هویت موجودیت درخواست‌کننده (ایستگاه مدیریت)
- هویت موجودیت انجام‌دهندهٔ درخواست (عاملی که برای خود و یا برای یک موجودیت پروکسی شده عمل می‌کند)
- هویت محلی که اطلاعات مدیریتی بایستی از آنجا دریافت شود (عامل یا واحد پروکسی شده)
- اطلاعات اعتبارسنجی
- اطلاعات کنترل دست‌یابی (مجاز بودن برای انجام عملیات درخواست شده)
- اطلاعات منظر MIB

با جمع کردن تمام این مفاهیم در یک متغیر منفرد، انعطاف‌پذیری و کارآئی از دست رفته‌اند. VACM همین مجموعه اطلاعات مرتبط با امنیت را، با استفاده از متغیرهای مشخص برای هر مورد فراهم آورده است. این یک مزیت قابل توجه نسبت به SNMPv1 است. این روش مفاهیم مختلف را از یکدیگر مجزا نموده تا اندازهٔ متغیرها بتوانند بطور جداگانه به هر مورد تخصیص یابند.

۸-۴ منابع مطالعاتی

[STAL99] یک بررسی کامل و تحلیلی از SNMP، SNMPv2 و SNMPv3 ارائه داده است. این کتاب همچنین مروری بر تکنولوژی مدیریت شبکه دارد.

STAL99 Stallings, W. *SNMP, SNMPv2, SNMPv3, and RMON 1 and 2*. Reading, MA: Addison-Wesley, 1999.

وِب سایت‌های مفید

- **SNMPv3 Web site**: این سایت بتوسط Technical University of Braunschweig نگهداری می‌شود. این سایت لینک‌هائی به RFCها و پیش‌نویس‌های اینترنت، کپی پیشنهادهای داده شده برای تغییرات به گروه‌های کاری، و همچنین لینک‌هائی به فروشندگان پیاده‌سازی‌های SNMPv3 فراهم آورده است.
- **The Simple Web site**: این سایت بتوسط University of Twente نگهداری می‌شود. منبع خوبی برای اطلاعات راجع به SNMP است که اشاره به پیاده‌سازی‌های عمومی این پروتکل داشته و همچنین شامل لیستی از کتاب‌ها و مقالات مرتبط است.

۸-۵ واژه‌های کلیدی، سؤالات مرورکنندهٔ بحث و مسائل

واژه‌های کلیدی

message processing model	مدل پردازش پیام	access policy	خط‌مشی دست‌یابی
network management	مدیریت شبکه	agent	عامل
proxy	پروکسی	community	جامعه
Simple Network Management Protocol (SNMP)	پروتکل سادهٔ مدیریت شبکه	community name	نام جامعه
User security model (USM)	مدل امنیتی کاربر	key localization	محلی کردن کلید
view-based access control model (VACM)	مدل کنترل دست‌یابی مبتنی بر منظر	management information base (MIB)	پایگاه اطلاعات مدیریت
		management station	ایستگاه مدیریت

سؤالات مرورکننده بحث

۸-۱ یک معماری مدیریت شبکه با چه مفهومی یکپارچه تلقی می‌شود؟

۸-۲ عناصر کلیدی مدل SNMP کدامند؟

۸-۳ یک MIB چیست؟

۸-۴ چه قابلیت‌ها یا فرامین اصلی در SNMPv1 فراهم آمده‌اند؟

۸-۵ وظیفهٔ یک پروکسی SNMP چیست؟

۸-۶ مفهوم جامعهٔ SNMPv1 را بطور مختصر بیان کنید.

۸-۷ رابطهٔ بین SNMPv1، SNMPv2 و SNMPv3 چیست؟

۸-۸ USM برای مقابله با چه تهدیدهائی طراحی شده است؟

۸-۹ تفاوت بین یک موتور مسئول با یک موتور غیرمسئول کدام است؟

۸-۱۰ محلی کردن کلید چیست؟

۸-۱۱ عناصری که VACM را می‌سازند لیست کرده و بطور مختصر تعریف نمائید.

مسائل

۸-۱ SNMPv1 یک نوع دیتا که Gauge نامیده می‌شود را تعریف کرده و آن را بصورت زیر تشریح کرده است:

> این نوع دیتا با کاربرد وسیع، نمایشگر یک عدد صحیح غیرمنفی است که می‌تواند زیاد یا کم شده ولی به یک مقدار ماکزیمم latch می‌گردد. استاندارد مقدار ماکزیمم $۲^{۳۱}-۱$ (۴,۲۹۴,۹۶۷,۲۹۵ دهدهی) برای Gauge را تعیین کرده است.

متأسفانه کلمهٔ *latch* تعریف نشده است و این امر دو تعبیر مختلف را بوجود آورده است. استاندارد SNMPv2 این ابهام را با تعریف زیر رفع کرده است:

> اندازهٔ یک Gauge وقتی دارای ماکزیمم مقدار خود است که اطلاعاتی که مدل می‌شود بزرگتر یا مساوی آن مقدار ماکزیمم باشد. اگر اطلاعات مدل شده پس از آن از مقدار ماکزیمم کمتر شود، Gauge هم کاهش می‌یابد.

الف- تعبیر دیگر چیست؟

ب- در نقاط قوت و ضعف این دو تعبیر بحث کنید.

۸-۲ در SNMPv1، برای هر موضوع در یک MIB یک MIB access Category تعریف شده است که هریک از اندازه‌های read-only، write-only و not-accessible می‌تواند به آن اختصاص یابد. یک read با یک عمل get یا trap انجام شده و یک write با عمل set صورت می‌پذیرد. برای write-only، موضوع ممکن است برای عملیات get و trap در دسترس باشد ولی این امر وابسته به پیاده‌سازی است. MIB Access Category ماکزیمم دست‌یابی مجاز به یک موضوع را تعیین می‌کند ولی در تسهیلات جامعه‌ای SNMPv1، Access Mode ممکن است برای یک پروفایل جامعهٔ بخصوص این دست‌یابی را بازهم محدودتر کند. در جدول زیر نوع دست‌یابی مجاز برای هر مورد را تعیین کنید.

MIB Access Category	SNMP Access Mode	
	READ-ONLY	READ-WRITE
read-only		
Read-write		
Write-only		
Not-accessible		

۸-۳ **الف**- RFC 2574 چنین بیان می‌کند که برای یک موتور غیرمسئول، اندازه‌های msgAuthoritativeEngineBoots و msgAuthoritativeEngineTime در یک سرآیند پیام خروجی تنها در صورتی تنظیم می‌شوند که پیام بخواهد بتوسط گیرندهٔ مسئول اعتبارسنجی گردد. چرا این محدودیت دارای معنی است؟

ب-از طرف دیگر، برای یک پیام Response از سوی یک موتور مسئول، اندازه‌های msgAuthoritativeEngineBoots و msgAuthoritativeEngineTime در سرآیند پیام‌های خارج‌شونده همیشه تنظیم میشوند. چرا بایستی چنین باشد؟

۸-۴ RFC 2574 چنین تعیین میکندکه سنکرونیزاسیون ساعت (بروزرسانی ساعت محلی بر حسب مقادیر ورودی) قبل از تأئید پنجرهٔ زمانی (کنترل اینکه پیام ورودی به هنگام است) صورت پذیرد. این امر بدین معنی است که یک موتور غیر مسئول می‌تواند تخمین خود از ساعت موتور مسئول را حتی در صورتی که پیام در خارج از پنجرهٔ زمانی مجاز قرار داشته باشد، بروزرسانی نماید. از زمان انتشار RFC، نظرات مخالف مستمری در این مورد در لیست پستی SNMPv3 وجود داشته است ولی تا زمان کتابت این کتاب بنظر نمی‌رسد که تغییری در بیان استاندارد رخ دهد. آموزنده است که به این مورد نگاهی بیندازیم. با تعاریف زیر:

MAEB = msgAuthoritativeEngineBoots
MAET = msgAuthoritativeEngineTime
SEB = تخمین محلی از snmpEngineBoots موتور مسئول دور
SET = تخمین محلی از snmpEngineTime موتور مسئول دور
LRET = snmpEngineTime اخیرترین

آنگاه فرض کنید که یک موتور غیرمسئول پیامی را دریافت می‌کند که برحسب آن

(MAEB = SEB) AND [LRET < MAET < (SET – 150)]

آنگاه شرایط برای بروزرسانی ساعت مهیا بوده و بنابراین این کار انجام می‌شود:

SET := MAET; LRET := MAET

حال وقتی به کنترل پنجرهٔ زمانی می‌رسیم، داریم

(MAEB = SEB) AND (MAET = SET)

و بنابراین پیام را بهنگام اعلام می‌کنیم. حال فرض کنید که ابتدا پنجرهٔ زمانی کنترل شود، آیا پیام را بهنگام و یا نابهنگام اعلام می‌کردیم؟

۸-۵ در نسخهٔ اولیهٔ انتشاریافتهٔ مشخصه‌های USM (RFC 2274)، در بحث پیرامون روش‌های سنکرونیزاسیون ساعت و تأئید پنجرهٔ زمانی، چنین آمده است: «توجه شود که این رَویه، اجازهٔ سنکرونیزاسیون اتوماتیک ساعت در صورتی که موتور SNMP غیرمسئول خارج از هماهنگی بوده و موتور SNMP مسئول بیش از ۱۵۰ ثانیه عقب‌تر از موتور SNMP غیرمسئول باشد، را نمی‌دهد.» این جمله پس از آنکه نویسندهٔ این کتاب به گروه کاری مربوطه خاطرنشان کرد که این امر همیشه صحیح نمی‌باشد از نسخهٔ بازنگری شده (RFC 2574) حذف گردید. از مثال مسألهٔ ۴-۸ استفاده کرده و نشان دهید که واقعاً بیان قبل همیشه صحیح نیست.

۸-۶ SNMPv3 فرض می‌کند که روش امنی برای تحویل کلیدهای محلی شده به سیستم‌های (عامل) اعتبارسنجی شده وجود دارد. این تحویل امن فراتر از حوزهٔ SNMPv3 است. این کار ممکن است بصورت دستی و یا بتوسط پروتکل امن دیگری انجام شود. وقتی یک کلید اولیه (یا یک جفت کلیدبرای اعتبارسنجی و محرمانگی) به یک عامل تحویل شد، SNMPv3 مکانیسمی را برای بروزرسانی کلیدها بصورت امن فراهم می‌آورد. مطلوب این است که برای ارتقاء امنیت، کلیدها هرچندوقت یکبار بروزرسانی شوند. یک کاربر می‌تواند از جانب خودش داوطلب تغییر کلید شده و برای این منظور کلمهٔ عبور جدیدی را ارائه دهد. بصورت مشابه، سیستم مدیریت شبکه (NMS) می‌تواند این کار را آغاز کرده و درخواست کلمهٔ

عبور جدید نماید. در هر صورت، کلید کاربر در NMS بروزرسانی می‌شود. سپس NMS می‌تواند یک کلید محلی برای هریک از عوامل ارتباطی را محاسبه نماید. پس از آن، NMS بایستی با هر عامل بطور امن ارتباط یابد تا عامل را وادار کند که کلید محلی شده خود را بروز برساند. دو حالت اختیاری زیر امکان‌پذیر است:

- کلید جدید را با استفاده از کلید قدیم بعنوان کلید رمز، رمزنگاری نماید.
- نوعی تابع یک- طرفه را بکار گرفته تا یک اندازه از کلید قدیمی درست شود. این اندازه را با کلید جدید XOR کرده و نتیجه را برای عامل بفرستد. عامل آنگاه می‌تواند نتیجهٔ ورودی را با همان تابع که به کلید قدیم اعمال شده است XOR کرده تا کلید جدید را بدست آورد.

SNMPv3 از فرمی به روش دوم استفاده می‌کند. مزیت این روش نسبت به روش اول چیست؟

۸-۷ روش برخورد SNMPv3، شامل استفاده از یک موضوع KeyChange در MIB سیستم هدف است. یک رئیس دور یا NMS این موضوع را تنظیم کرده که پس از آن بتوسط عامل بصورت خودکار برای بروزرسانی کلید مرتبط بکار می‌رود. الگوریتم شامل دو فاز است که یک فاز آن در موتور متقاضی و فاز دیگر آن در موتور عامل دور صورت می‌پذیرد.

عمل وقتی شروع می‌شود که یک متقاضی علاقه‌مند است تا یک کلید موجود keyOld را با یک کلید جدید keyNew تعویض کند. متقاضی قدم‌های زیر را برمی‌دارد:

۱- یک اندازهٔ random از یک تولیدکنندهٔ اعداد شبه تصادفی و یا تولیدکنندهٔ اعداد واقعی تصادفی تولید می‌کند.

۲- مقدار زیر را محاسبه می‌کند

$$digest = Hash(keyOld \parallel random)$$

که در آن Hash یا MD5 و یا SHA-1 است که بستگی به این دارد که آیا یک کلید ۱۶- اُکتتی و یا ۲۰- اُکتتی مورد نیاز بوده و $\parallel$ علامت جمع رشته‌ای است.

۳- محاسبه می‌کند

$$delta = digest \oplus keyNew$$

$$protocolKeyChange = (random \parallel delta)$$

که در آن $\oplus$ عمل XOR است.

۴- اندازهٔ protocolKeyChange آنگاه در یک فرمان Set به یک عامل فرستاده شده تا موضوع KeyChange در MIB عامل را بروز برساند.

عامل برای بروز رساندن کلید، با اندازهٔ ورودی چه باید بکند؟

۸-۸ یک روش ساده‌تر از آنچه در مسألهٔ قبل عنوان شد این است که keyOld را با keyNew بصورت XOR درآورده و نتیجه را ارسال کند. گیرنده آنگاه اندازهٔ XOR مقدار دریافت شده با keyOld را محاسبه کرده تا keyNew را تولید کند. چون یک حمله‌کننده keyOld را نمی‌داند، او نمی‌تواند که keyNew را بدست آورد. مزایای استفاده از عدد تصادفی و تابع امن یک- طرفه در مسألهٔ ۷-۸ در مقایسه با این روش چیست؟

قسمت سوم

امنیت سیستم

قسمت سوم کتاب به مقوله‌های امنیتی در سطح سیستم می‌پردازد که شامل تهدیدها و روش‌های مقابله با آنها از سوی مهاجمین و ویروس‌ها می‌باشد. این قسمت همچنین استفاده از دیوارهای آتش و سیستم‌های معتمد را مورد بحث قرار می‌دهد.

فصل ۹ مهاجمین

فصل ۹ مجموعهٔ متنوعی از تهدیدهائی که به دلیل وجود نقاط آسیب‌پذیر در سیستم‌های کامپیوتری مبتنی بر شبکه، از سوی نفوذگران متوجه سرویس‌ها و دست‌یابی به آنها می‌باشد، را مورد بررسی قرار می‌دهد. این فصل با بحثی در مورد انواع حملاتی که می‌تواند بتوسط کاربران غیر مجاز، یا مهاجمین، روی سیستم انجام شود شروع شده و روش‌های جلوگیری و تشخیص آنها را تحلیل می‌نماید. این فصل همچنین مقولهٔ مرتبط مدیریت کلمهٔ عبور را پوشش می‌دهد.

فصل ۱۰ نرم‌افزارهای بداندیش

فصل ۱۰ تهدیدهای نرم‌افزاری به سیستم‌ها، با تأکید خاصی بر ویروس‌ها و کِرم‌ها، را مورد بررسی قرار می‌دهد. این فصل با بررسی انواع مختلف نرم‌افزارهای موذی شروع شده و نگاه مفصل‌تری به ماهیت ویروس‌ها و کِرم‌ها می‌اندازد. بقیهٔ فصل نگاهی به روش‌های مقابله با این تهدیدها دارد. در انتها حملات توزیع شدهٔ انکار سرویس مورد بحث قرار می‌گیرد.

فصل ۱۱ دیوارهای آتش

یک روش استاندارد برای محافظت منابع کامپیوتری محلی از تهدیدهای خارجی، استفاده از دیوار آتش است. فصل ۱۱ اصول طراحی دیوار آتش را مورد بحث قرار داده و به تکنیک‌های معینی اشاره می‌کند. این فصل همچنین مقولهٔ مرتبط سیستم‌های معتمد را پوشش می‌دهد.

فصل ۹

مهاجمین

یکی از مسائل امنیتی مهم در سیستم‌های شبکه‌ای، تعرض خصمانه و یا حداقل ناخواستهٔ کاربران و نرم‌افزارهاست. تعرض کاربر می‌تواند بصورت اتصال غیرمجاز به ماشین، و یا در مورد یک کاربر معتبرکسب امتیازات یا انجام عملیاتی فراتر از آنچه برای او مجاز است، باشد. تعرض نرم‌افزاری می‌تواند بشکل یک ویروس، کِرم و یا اسب تروا ظاهر شود.

تمام این حملات مرتبط با امنیت شبکه‌اند زیرا ورود به سیستم می‌تواند از طریق شبکه صورت پذیرد. ولی در عین حال این حملات منحصراً مبتنی بر شبکه نمی‌باشند. یک کاربر با امکان دسترسی به یک ترمینال محلی ممکن است بدون استفاده از شبکهٔ میانی مبادرت به تعرض نماید. یک ویروس یا یک اسب تروا ممکن است نه از طریق شبکه بلکه از طریق یک دیسکت وارد سیستم شود. تنها کِرم یک پدیدهٔ کاملاً شبکه‌ای است. بنابراین تعرض به سیستم بحثی است که هم مورد توجه امنیت شبکه و هم مورد توجه امنیت کامپیوتر است.

نظر به اینکه این کتاب بر امنیت شبکه متمرکز شده است، ما نسبت به تجزیه و تحلیل مفصّل تک‌ها و پاتک‌های مرتبط با ورود غیرمجاز به سیستم تلاش نخواهیم کرد. در عوض در این بخش مروری کلی بر این مشکلات خواهیم داشت.

این فصل به موضوع تعرض‌کنندگان و یا مهاجمین می‌پردازد. در ابتدا ماهیت حمله را بررسی نموده و سپس نگاهی به استراتژی‌های مربوط به جلوگیری از حمله و در صورت شکست در جلوگیری، استراتژی‌های تشخیص حمله می‌پردازیم. در قسمت بعد مقوله‌های مربوط با مدیریت کلمهٔ عبور را بررسی می‌کنیم.

۹-۱ مهاجمین (INTRUDERS)

یکی از دو تهدید عمدهٔ امنیت سیستم، مهاجمین هستند (دیگری ویروس‌ها می‌باشند)، که معمولاً از آنها با نام هَکر (hacker) و یا کِرَکِر (cracker) یاد می‌شود. در یکی از مطالعات مهم مربوط به مهاجمین، آندرسن [ANDE80] سه دسته از مهاجمین را شناسائی نموده است:

- **نقاب‌دار(Masquerader)** : فردی است که مجاز به استفاده از کامپیوتر نیست ولی از کنترل‌های دست‌یابی به سیستم عبور کرده و اشتراک یک کاربر قانونی را مورد سوء استفاده قرار می‌دهد.
- **سوءاستفاده‌کننده (Misfeasor)**: یک کاربر قانونی است که به دیتا، برنامه‌ها و یا منابعی دست می‌یازد که قانوناً اجازهٔ دست‌یابی به آنها را ندارد. این فرد ممکن است مجاز به دست‌یابی به آنها باشد ولی از آنها بطور غیرقانونی در جهت خواسته‌های خود استفاده می‌نماید.
- **کاربر خُفیه (Clandestine User)**: فردی است که کنترل سوپروایزری سیستم را بدست گرفته و از این کنترل برای فرار از شناخته شدن و یا خنثی کردن عملیات شناسائی استفاده نماید.

مهاجم نقاب‌دار به احتمال زیاد یک فرد غیرخودی است. مهاجم سوءاستفاده‌کننده معمولاً یک فـــرد خودی است و مهاجم خفیه می‌تواند فردی خودی و یا غیرخودی باشد.

حملات مهاجمین می‌تواند از مرز بی‌خطر تا مرز خطرناک متفاوت باشد. در مرز بی‌خطر، افراد بسیاری وجود دارند که بدون قصد سوء به کاوش در اینترنت کنجکاو بوده و می‌خواهند بدانند چه چیز تازه‌ای در اینترنت وجود دارد. در مرز خطرناک افراد معدودی هستند که تلاش می‌کنند تا به داده‌های مهم دست یابند، داده‌ها را بصورت غیرمجاز تغییر داده و یا سیستم را از کار بیندازند.

خطر مهاجمین بنحو قابل توجهی برای عموم روشن شده و شاید واقعهٔ "Wily Hacker" در سالهای ۱۹۸۷-۱۹۸۶ میلادی که بتوسط Cliff Stoll [STOL88,89] ثبت شده است علت عمدهٔ آن باشد. در سال ۱۹۹۰ میلادی، در آمریکا یک نمایش قدرت بر علیه هَکِرهای غیرمجاز صورت پذیرفت که طی آن عده‌ای دستگیر و عده‌ای متهم شدند. یک محاکمهٔ جنجالی برپا گردید، تعدادی محکوم شده و مقادیر زیادی از تجهیزات کامپیوتری و دیتا ضبط گردید [STER92]. بسیاری از مردم تصور کردند که مساله حل شده و مشکل مهار گردیده است.

اما در حقیقت، مساله تحت کنترل قرار نگرفته است. بعنوان مثال گروهی در لابراتوارهای Bell گزارش داده‌اند که حملات دائم و مکرری از طریق اینترنت و بتوسط منابع مختلف در طول زمان روی سایت‌های کامپیوتری آنها صورت پذیرفته است [BELL92, BELL93]. در زمان ارائهٔ این گزارش، گروه فوق شاهد موارد زیر بودند:

- تلاش برای کپی کردن فایل کلمات عبور (بعداً دربارهٔ آن بحث خواهد شد)، بیشتر از هر دو روز یکبار.
- تقاضاهای مشکوک برای فراخوانی از دور (RPC = Remote Procedure Call) به میزان بیش از هر هفته یکبار.
- تلاش برای اتصال به ماشین‌های «طعمه» که وجود خارجی ندارند حداقل هر دو هفته یکبار.

مهاجمین بی‌خطر را ممکن است تحمل نمود ولی باید توجه داشت که چون آنها منابع را بکار می‌گیرند می‌توانند عملیات کاربران قانونی را با کندی مواجه سازند. البته راهی وجود ندارد که بتوان پیش‌بینی کرد آیا یک مهاجم بی‌خطر و یا خطرناک است. در نتیجه حتی برای سیستم‌هائی که مدارک حساس بخصوصی را نگاه نمی‌دارند، انگیزه‌ای برای کنترل این مساله وجود دارد.

مثالی که بطور آشکار خطر چنین تهاجمی را نشان می‌دهد در دانشگاه A&M تکزاس رخ داد [SAFF93]. در ماه اوت ۱۹۹۲ میلادی، مرکز کامپیوتر آن دانشگاه متوجه شد که یکی از دستگاه‌های آن برای حمله به کامپیوترها در نقطه‌ای دیگر از طریق اینترنت بکار گرفته شده است. با کنترل عملیات، پرسنل مرکز کامپیوتر دریافتند که پای چندین مهاجم غیرخودی در کار است که برنامه‌هائی با هدف کشف کلمات عبور را روی تعدادی از کامپیوترها اجرا می‌کنند (سایت شامل ۱۲,۰۰۰ دستگاه کامپیوتری متصل بهم بود). مرکز کامپیوتر ماشین‌های آلوده را جدا کرده، حفره‌های امنیتی را بسته و عملیات نرمال را از سرگرفت. چند روز بعد، یکی از مدیران سیستم محلی دریافت که تهاجم از سر گرفته شده است. بعداً روشن شد که حمله بسیار پیچیده‌تر از آنست که در ابتدا تصور می‌شد. فایل‌هائی بدست آمدکه حاوی صدها کلمهٔ عبورکشف شده بودند که بعضی از آنها متعلق به سرورهای اصلی و مثلاً امن بود. علاوه بر آن، یکی از ماشین‌های محلی بعنوان تابلوی اعلانات یک هَکِر مورد استفاده قرار گرفته بود که از آن هَکِرها برای ارتباط با هم و بحث در مورد تکنیک‌ها و پیشرفت کار استفاده می‌کردند.

تحلیل این حمله نشان داد که در واقع دو سطح از هَکِرها در این امر دخالت داشتند. هَکِرهای سطح بالا، کاربران پیچیده‌ای بودند که اطلاعات کافی از تکنولوژی داشتند و هَکِرهای سطح پائین «سربازان پیاده‌ای» بودند که صرفاً از برنامه‌های

عبور از سیستم استفاده کرده و نسبت به نحوهٔ عملکرد این برنامه‌ها اطلاعاتی نداشتند. این کار گروهی، دو سلاح عمده در زرادخانهٔ مهاجمین را با هم ترکیب کرده بود: اطلاعات پیچیده‌ای از اینکه چطور می‌توان مبادرت به هجوم کرد، و تمایل به صرف ساعات فراوان برای پیداکردن نقاط ضعف ورود به سیستم.

یکی از نتایج مرتبط با آشنائی فزاینده با مشکل تهاجم، ایجاد تعدادی از تیم های اورژانس کامپیوتری CERT (Computer Emergency Response Team) بوده است. این تیم‌ها، اطلاعات مربوط به نقاط آسیب‌پذیر سیستم‌ها را جمع کرده و آن را برای مدیران سیستم‌ها ارسال می‌کنند. متاسفانه هَکرها هم می‌توانند به این گزارشات CERT دسترسی یابند. در واقعهٔ A&M تکزاس، تحلیل‌های بعدی نشان داد که هَکرها برنامه‌هائی را ساخته بودند که با کمک آنها هر نقطهٔ ضعفی که بتوسط CERT گزارش شده بود مورد تجاوز قرار می‌گرفت. اگر حتی یک ماشین به توصیه‌های فوری CERT عمل ننموده بود، در مقابل این حملات آسیب‌پذیر باقی مانده بود.

علاوه بر برنامه‌های مربوط به دسترسی به کلمات عبور، مهاجمین سعی کردند تا نرم‌افزار ورود به سیستم را طوری دستکاری نمایند که بتوانند کلمه‌های عبور کاربرانی را که به آن سیستم متصل‌اند کشف کنند. این امر آنان را قادر ساخت تا مجموعهٔ عظیمی از کلمات عبور هَک شده را تهیه نموده و آن را در تابلوی اعلاناتی که خود در یکی از ماشین‌های قربانی ایجاد کرده بودند نصب کنند.

در این بخش به تکنیک‌های استفاده شده برای تهاجم نظر خواهیم کرد. آنگاه روش‌های تشخیص تهاجم را عنوان می‌کنیم. بالاخره به روش‌های مرتبط با کلمات عبور در جلوگیری از تهاجم توجه خواهیم نمود.

تکنیک‌های تهاجم

هدف مهاجم، کسب دست‌یابی به سیستم و یا افزایش محدودهٔ اختیارات وی در دست‌یابی به سیستم است. این امر نیاز به کسب اطلاعاتی از طرف مهاجم دارد که معمولاً حفاظت شده‌اند. در بیشتر موارد، این اطلاعات در کلمهٔ عبور کاربر خلاصه می‌شوند. با اطلاع از کلمهٔ عبور یک کاربر قانونی، یک مهاجم می‌تواند وارد سیستم شده و از تمام امتیازات مربوط به کاربر قانونی استفاده نماید.

معمولاً، یک سیستم بایستی فایلی را که شامل کلمات عبور تمام کاربران مجاز است در داخل خود نگهداری کند. اگر چنین فایلی بدون حفاظت در سیستم ذخیره شده باشد، آنگاه دسترسی به آن و پیدا کردن کلمات عبور برای هَکرها آسان خواهد بود. فایل کلمات عبور می‌تواند به دو طریق محافظت شود:

- **تابع یک- طرفه**: سیستم تنها یک فرم رمزشده از کلمهٔ عبور کاربر را نگهداری می‌کند. وقتی کاربر کلمهٔ عبوری را به سیستم عرضه می‌کند، سیستم این کلمهٔ عبور را به رمز درآورده و آن را با فرم ذخیره شدهٔ آن مقایسه می‌کند. در عمل، سیستم معمولاً یک تبدیل یک طرفه (برگشت‌ناپذیر) را انجام داده که درآن از کلمهٔ عبور برای تولید یک کلید رمزنگاری استفاده شده و یک خروجی با طول ثابت تولید می‌شود.
- **کنترل دست‌یابی**: دست‌یابی به فایل کلمات عبور، محدود به یک یا چند مشترک است.

اگر یکی از این دو روش مقابله با تهاجم و یا هردوی آنها در سیستم مستقر باشند، تلاش قابل توجهی لازم است تا یک مهاجم، هرچند قوی، بتواند کلمات عبور را پیدا کند. بر اساس یک بررسی انجام شده که شامل گفتگو با تعدادی هَکر کلمات عبور نیز بوده است [ALVA90]، تکنیک‌های پیداکردن کلمات عبور چنین‌اند:

۱- کلمات عبور پیش فرض مربوط به مشترکین که معمولاً به همراه سیستم عرضه می‌شوند، امتحان شود. بسیاری از مدیران سیستم، زحمت تغییر این کلمات عبور را بخود نمی‌دهند.
۲- تمام کلمات عبور کوتاه ممکن (بین ۱ تا ۳ حرف) امتحان شود.
۳- کلمات موجود در کتاب لغت حالت فعال سیستم و یا لیستی از کلمات عبور محتمل امتحان شود. نمونه‌هائی از کلمات عبور محتمل را می‌توان در تابلوی اعلانات هَکرها پیدا کرد.
۴- اطلاعات مربوط به کاربران جمع‌آوری شود. این اطلاعات شامل نام کامل آنها، نام همسر و فرزندان آنها، نام عکس‌های موجود در دفتر آنها و نام کتاب‌های مورد علاقه و یا موجود در محل کار آنها که سرگرمی فرد محسوب می‌شوند، می‌باشد.
۵- شمارهٔ تلفن کاربر، شمارهٔ شناسنامه، کُد ملی، شمارهٔ اطاق وی و اطاق‌هائی که بیشتر با آنها مرتبط است امتحان شود.
۶- تمام شماره‌های قانونی اتومبیل‌ها در شهر محل سکونت کاربر امتحان شود.
۷- از یک اسب تروا برای دور زدن محدودیت‌های دست‌یابی به سیستم استفاده شود.
۸- روی خط ارتباطی سیستم مورد نظر با کاربران دور، شنود گذاشته شود.

شش روش اول، راه‌های مختلف حدس‌زدن کلمهٔ عبور است. اگر یک مهاجم قرار باشد تا با تلاش‌های مداوم و حدس‌زدن‌های مکرر به یک سیستم راه یابد، تلاش او تلاش خسته‌کننده‌ای خواهد بود که بسهولت می‌تواند تشخیص داده شود. بعنوان مثال، یک سیستم می‌تواند بسادگی پس از هر سه بار تلاش ناموفق در وصل شدن به سیستم، تلاش بعدی را انکار نماید و بنابراین مهاجم مجبور خواهد بود که پس از این مدت دوباره به سیستم متصل گردد تا تلاش خود را از سر بگیرد. تحت چنین شرایطی، امتحان کردن بیش از یک مشت کلمهٔ عبور عملی نخواهد بود. ولی احتمال اینکه یک مهاجم زیرک چنین روش خامی را انتخاب کند کم است. بعنوان مثال اگر مهاجم بتواند با سطح پائینی از امتیازات به یک فایل رمز شدهٔ کلمات عبور دست یابد، استراتژی او این خواهد بود که فایل را گرفته و در فرصت کافی به کشف رمز کلمات عبور بپردازد تا کلمهٔ عبوری با امتیازات بیشتر دست‌یابی را کشف کند.

حملات حدسی در جائی ممکن و حتی خیلی مؤثر خواهد بودکه بتوان حدس‌های زیادی را بصورت اتوماتیک به مرحلهٔ اجرا گذاشت و هر حدس را امتحان کرد بدون اینکه عملیات حدس زدن بتوسط مسئولین شبکه قابل تشخیص باشد. در بخش‌های بعدی این فصل در مورد خنثی‌سازی حملات حدسی صحبت خواهیم کرد.

روش هفتم در لیست بالا، یعنی استفاده از اسب تروا، کاری است که مقابله با آن بسیار مشکل است. مثالی از یک برنامه که مرحلهٔ کنترل دست‌یابی را دور می‌زند در [ALVA90] ذکر شده است. در این مثال، یک کاربر دارای سطح پائین دست‌یابی یک بازی کامپیوتری تهیه کرده و اپراتور سیستم را به استفاده از آن در اوقات فراغت وسوسه نمود. برنامه ظاهراً یک بازی کامپیوتری بود ولی در خفا شامل کُدی بود که فایل کلمات عبور که رمزشده نبوده ولی از نظر دست‌یابی حفاظت شده بود را به داخل فایل کاربر کپی می‌نمود. چون بازی در سطح بالای دست‌یابی مربوط به اپراتور اجرا می‌گردید، قادر بود تا به فایل کلمات عبور دست یابد.

حملهٔ هشتم ذکرشده در لیست بالا، یعنی شنود خط، در مقولهٔ حفاظت فیزیکی است. با این حمله می‌توان از طریق تکنیک‌های رمزنگاری پیوند، که در بخش رمزنگاری به آن اشاره شد، مقابله کرد.

سایر تکنیک‌های تهاجم نیازی به فراگیری کلمهٔ عبور ندارند. مهاجمین می‌توانند با سوء استفاده از حملاتی نظیر سرریزکردن حافظهٔ موقت روی برنامه‌ای که با امتیازات خاصی در حال اجراست، به سیستم دست یابند. ارتقاء امتیازات نیز می‌تواند بهمین روش انجام شود.

حال به بررسی دو روش مهم مقابله با این تهاجمات، یعنی تشخیص و جلوگیری، می‌پردازیم. تشخیص، مرتبط با آگاه شدن از حمله چه قبل و چه بعد از آن است. جلوگیری، یک چالش امنیتی و یک جنگ دائمی در همهٔ زمان‌هاست. مشکل از این حقیقت سرچشمه می‌گیرد که مدافع بایستی همهٔ حملات ممکن را دفع کند، در حالی که مهاجم آزاد است تا ضعیف‌ترین حلقه در زنجیرهٔ دفاعی را پیدا کرده و از آنجا حمله نماید.

۹-۲ تشخیص تهاجم (INTRUSION DETECTION)

بهترین سیستم‌های جلوگیری از تهاجم، ناچاراً روزی شکست خواهند خورد. خط دفاعی دوم سیستم در مقابل تهاجم، تشخیص تهاجم است و این مساله در سال‌های اخیر محور بسیاری از تحقیقات بوده است. ملاحظات متعددی محرک اینگونه بررسی‌ها بوده‌اند که از آن جمله‌اند:

۱- اگر تهاجم باندازهٔ کافی سریع تشخیص داده شود، مهاجم را می‌توان شناسائی و قبل از اینکه صدمه‌ای به سیستم وارد شده و یا داده‌هائی مورد تعرض قرار گیرند او را از سیستم بیرون راند. تشخیص تهاجم حتی اگر باندازه کافی سروقت انجام نشود تا بتوان مهاجم را قبل از هرگونه عملی از سیستم خارج کرد، بازهم هر چقدر زودتر صورت پذیرد میزان صدمات وارد به سیستم کمتر بوده و بازیابی سیستم سریعتر انجام خواهد پذیرفت.

۲- یک سیستم تشخیص تهاجم کارآمد، می‌تواند به عنوان یک سیستم بازدارنده عمل کرده و از هجمه‌ها جلوگیری کند.

۳- تشخیص تهاجم باعث می‌شود که بتوان مجموعه‌ای از اطلاعات مربوط به تکنیک‌های تهاجم را جمع‌آوری کرد که خود می‌تواند تسهیلات جلوگیری از تهاجم را توسعه بخشد.

تشخیص تهاجم بر این فرض قرار گرفته است که رفتار یک مهاجم با رفتار یک کاربر قانونی بنحوی متفاوت خواهد بود که می‌توان آن را اندازه‌گیری کرد. البته نمی‌توان انتظار داشت که یک وجه تمایز کاملاً مشخص و مرزبندی شده بین حملهٔ یک مهاجم و استفادهٔ معمول یک کاربر قانونی از منابع مجاز، وجود داشته باشد. بلکه باید انتظار داشته باشیم که بخشی از عملیات این دو یکسان و غیرقابل تشخیص باشند.

شکل ۹-۱ بطور خیلی کلی ماهیت وظیفه‌ای که بعهدهٔ یک طراح سیستم تشخیص تهاجم است را نشان می‌دهد. اگرچه رفتار نوعی یک مهاجم با رفتار معمول یک کاربر معتبر متفاوت است ولی این رفتارها نقاط مشترکی هم دارند. به همین دلیل یک تعبیر نسبتاً وسیع از رفتار تهاجمی که باعث بدام انداختن مهاجمین زیادتری گردد، خواه ناخواه منجر به متهم نمودن عده‌ای از کاربران مجاز، بعنوان مهاجم نیز خواهد گردید. از سوی دیگر کوشش برای مبرا نمودن همهٔ کاربران مجاز از اینکه بعنوان مهاجم شناخته شوند، نیاز به تعاریف سفت و سختی از رفتار تهاجمی داشته که در نتیجه آن برخی مهاجمین نیز از شناخته شدن مصون خواهند ماند. بنابراین عمل تشخیص تهاجم یک مصالحهٔ هنری بین دو مقولهٔ فوق‌الذکر است.

در بررسی آندرسن [ANDE80]، چنین فرض شده است که می‌توان با اطمینان معقول، بین یک مهاجم نقاب‌دار و یک کاربر قانونی تفاوت قایل شد. رفتار یک کاربر قانونی را می‌توان با مشاهدهٔ تاریخچهٔ رفتاری او در گذشته جمع‌آوری کرد و هر تغییر قابل ملاحظه در نحوهٔ رفتار او را می‌توان تشخیص داد. آندرسن بیان می‌کند که عمل تشخیص یک رفتار مشکوک (کاربر قانونی که رفتاری غیر قانونی دارد)، سخت‌تر از تشخیص تهاجم است زیرا فاصلهٔ بین رفتار نرمال و رفتار غیرنرمال یک کاربر قانونی ممکن است بسیار کم باشد. آندرسن نتیجه گرفت که این نوع تخطّی صرفاً از طریق جستجوی رفتارهای نامعقول،

غیر قابل تشخیص خواهد بود. با وجود این، رفتارهای مشکوک را می‌توان با تعریف هوشمندانهٔ دسته‌ای از عملیات که استفادهٔ غیرمعتبر از سیستم را مشخص می‌کنند تشخیص داد. بالاخره، تشخیص کاربر خفیه کاری فراتر از حوزهٔ روش‌های صرفاً اتوماتیک است. این مشاهدات که در سال ۱۹۸۰ میلادی مورد توجه بودند، امروز نیز همچنان معتبرند.

[PORR92] روش‌های زیر برای تشخیص تهاجم را تعریف می‌کند:

۱- **تشخیص آماری ناهنجاری**: شامل جمع‌آوری اطلاعات مربوط به کاربران قانونی در طول زمان است.آنگاه تست‌های آماری به رفتار یک کاربر مشکوک اعمال شده تا با اطمینان نسبتاً بالا تعیین شود که این رفتار، رفتار یک کاربر قانونی نیست.

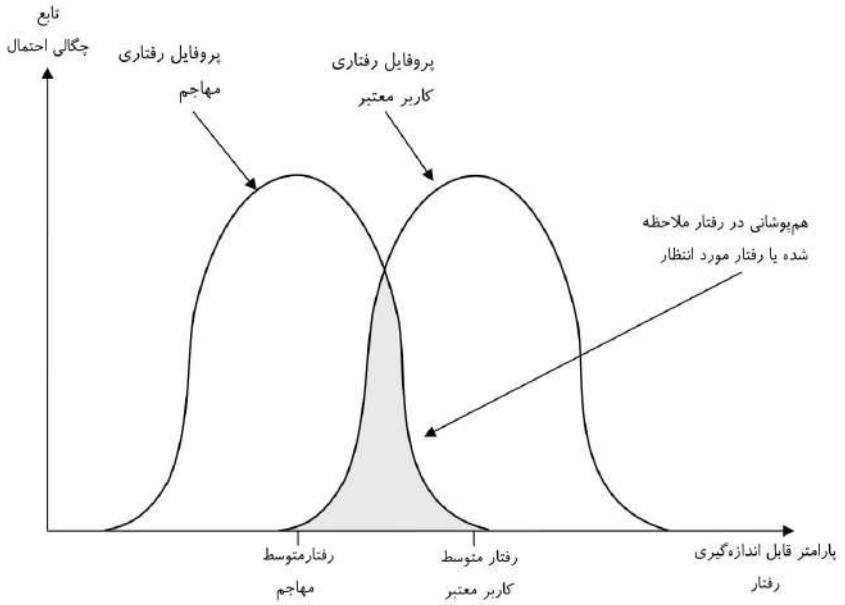
الف- تشخیص آستانه‌ای: این روش شامل تعریف آستانه‌هائی، مستقل از کاربر، برای تواتر وقوع اتفاقات مختلف است.

ب- تشخیص مبتنی بر پروفایل: یک پروفایل از فعالیت هرکاربر تهیه شده و برای تشخیص تغییرات رفتاری هر مشترک مجاز مورد استفاده قرار می‌گیرد.

۲- **تشخیص مبتنی بر قاعده**: شامل تلاش برای تعریف یک سری قواعد است که با استفاده از آن قواعد بتوان تصمیم گرفت که رفتار مشاهده شده، مربوط به یک مهاجم است.

الف- تشخیص موارد خلاف قاعده: قواعدی تهیه می‌شود که انحراف از رفتارهای نرمال را تشخیص دهد.

ب- تشخیص نفوذ: یک سیستم خبره است که به دنبال رفتارهای مشکوک می‌گردد.



شکل ۱-۹ پروفایل رفتاری مهاجمین و کاربران معتبر

در یک جمله، روش‌های آماری به دنبال تعریف رفتارهای هنجار و یا قابل انتظار هستند در حالی که روش‌های مبتنی بر قاعده، کوشش می‌کنند تا رفتارهای منظم را تعریف کنند.

بر اساس تعاریفی که قبلاً برای انواع مهاجمین داده شد، تشخیص آماری ناهنجاری در مورد مهاجمین نقاب‌دار، که به احتمال کم الگوی رفتاری کاربران مجاز را تقلید می‌کنند، مؤثر است. از سوی دیگر، این تکنیک‌ها ممکن است در مقابله با رفتارهای غیرمجاز مؤثر نباشند. برای چنین حملاتی، روش‌های مبتنی بر قاعده ممکن است قادر به شناسائی وقایع و تواتر وقوع آنها بوده و نفوذ بیگانه را مشخص سازند. در عمل یک سیستم ممکن است ترکیبی از هر دو روش را استفاده کرده تا بتواند در مقابل محدودهٔ وسیعی از حملات ایستادگی نماید.

سوابق ممیزی (Audit Records)

یکی از ابزارهای اصلی برای تشخیص تهاجم، سوابق ممیزی است. بعضی از سوابق فعالیت‌های جاری کاربران بایستی بعنوان ورودی یک سیستم تشخیص تهاجم نگهداری شوند. در این مورد از دو طرح استفاده می‌شود:

- **سوابق ممیزی بومی**: تقریباً تمام سیستم عامل‌های چندکاربره، دارای نرم‌افزار محاسبی می‌باشند که اطلاعات مربوط به فعالیت کاربران را جمع‌آوری می‌نماید. مزیت استفاده از این اطلاعات این است که هیچ نرم‌افزار جمع‌آوری دیگری مورد نیاز نیست. عیب آن نیز این است که سوابق بومی ممکن است اطلاعات مورد نیاز را نداشته و یا آن را به فرم مطلوب نداشته باشند.
- **سوابق ممیزی مخصوص تشخیص**: می‌توان یک تسهیلات جمع‌آوری اطلاعات را طوری برپا نمود که تنها سوابق مورد نیاز یک سیستم تشخیص تهاجم را جمع‌آوری نماید. یکی از مزایای چنین روشی این است که می‌توان آن را مستقل از سیستم تعریف کرد و به سیستم‌های مختلفی اعمال نمود. عیب آن میزان سربارهٔ اضافی، ناشی از داشتن دو بستهٔ نرم‌افزاری محاسباتی روی یک ماشین، است.

مثال خوبی از سوابق ممیزی مخصوص تشخیص، آن است که بتوسط Dorothy Denning [DENN87] خلق گردید. هر سابقهٔ ممیزی شامل میدان‌های زیر است:

- **عامل**: شروع‌کنندگان عملیات. یک عامل نوعاً کاربری در یک ترمینال است، ولی ممکن است پردازشی باشد که بجای یک کاربر و یا گروهی از کاربران عمل کند. تمام فعالیت‌ها از طریق فرمان‌هائی انجام می‌شود که از سوی عامل صادر شده است. عامل‌ها می‌توانند در گروه‌های مختلف با امتیازات دست‌یابی متفاوت طبقه‌بندی شده و این طبقات ممکن است هم‌پوشانی هم داشته باشند.
- **عمل**: عملیاتی است که بتوسط عامل روی موضوع انجام می‌شود. مثال‌هائی از این دست، ورود به سیستم، خواندن، عملیات I/O و غیره است.
- **موضوع**: چیزهائی که عمل روی آنها صورت می‌پذیرد. مثال‌های این مورد مانند فایل‌ها، برنامه‌ها، پیام‌ها، رکوردها، ترمینال‌ها، چاپگرها و یا ساختارهای ایجاد شده بتوسط کاربر یا برنامه است. وقتی یک عامل در معرض دریافت عملی، مانند دریافت e-mail، قرار می‌گیرد آنگاه همان عامل مبدل به یک موضوع می‌گردد. موضوع‌ها می‌توانند بر حسب نوعشان دسته‌بندی شوند. دسته‌بندی موضوعات ممکن است وابسته به شکل و یا محیط باشد. مثلاً عملیات روی یک پایگاه داده ممکن است براساس تأثیر روی کل پایگاه و یا از دید یک فایل آن پایگاه ممیزی گردد.

- **شرایط استثنائی**: در صورت وجود استثناء، آن را مشخص می‌سازد.
- **استفاده از منابع**: یک لیست کمّی است که مشخص می‌کند هر عنصر به چه میزان از منابع استفاده کرده است (مثلاً تعداد خطوطی که چاپ شده و یا نمایش داده شده است، تعداد رکوردهائی که خوانده و یا نوشته شده است، زمان پردازش، واحدهای I /O استفاده شده، زمان یک اجلاس).
- **زمان‌سنج**: یک برچسب زمانی یکتا که تاریخ و زمان وقوع عمل را نشان می‌دهد.

اکثر عملیات یک کاربر از تعدادی عمل ابتدائی تشکیل شده است. برای مثال، کپی کردن یک فایل شامل انجام فرمان کپی کاربر است که خود شامل اعمالی مانند کنترل اعتبار دست‌یابی به فایل، تنظیم کپی، خواندن از یک فایل و نوشتن فایل در جای دیگر است. فرمان زیر را در نظر بگیرید

COPY GAME.EXE TO <Library>GAME.EXE

که بتوسط آقای Smith برای کپی کردن یک فایل اجرائی GAME از فهرست جاری به فهرست دیگری بنام <Library> صادر شده است. سوابق ممیزی زیر ممکن است تولید شوند:

Smith	execute	<Library>COPY.EXE	0	CPU = 00002	11058721678
Smith	read	<Smith>GAME.EXE	0	RECORDS = 0	11058721679
Smith	execute	<Library>COPY.EXE	write-viol	RECORDS = 0	11058721680

در این مورد، عمل کپی ممکن است نادیده گرفته شود (abort)، زیرا Smith اجازه نوشتن فایلی در فهرست <Library> را نداشته باشد.

تجزیه عمل کاربر به عملیات ابتدائی‌تر دارای سه مزیت است:

۱- چون موضوعات، واحدهای قابل حفاظت در یک سیستم‌اند، استفاده از عملیات ابتدائی باعث می‌گردد تا ممیز بتواند تمام رفتارهائی که روی موضوع تأثیر می‌گذارند را ثبت کند. بنابراین سیستم می‌تواند تمام تلاش‌های ناموفق برای دست‌یابی را تشخیص داده (با ملاحظهٔ وضع غیرعادی موارد برگشتی) و همچنین تلاش‌های موفق را، با ملاحظهٔ وضع غیرعادی در مجموعهٔ موضوعاتی که عامل به آنها دسترسی دارد، پیدا کند.

۲- سوابق ممیزی مربوط به یک موضوع منفرد و یک عمل منفرد، مدل را ساده نموده و ساخت آن را آسان می‌سازند.

۳- بعلت ساختار ساده و یکنواخت سوابق ممیزی مخصوص تشخیص، کسب اطلاعات مربوط به آن و یا حداقل بخشی از این اطلاعات می‌تواند به سهولت، با انتقال سوابق ممیزی بومی به آن انجام شود.

تشخیص آماری ناهنجاری (Statistical Anomaly Detection)

همانطور که قبلاً بیان گردید، تکنیک‌های تشخیص آماری ناهنجاری در دو طبقهٔ وسیع جای می‌گیرند: سیستم‌های تشخیص آستانه‌ای و سیستم‌های مبتنی بر پروفایل. تشخیص آستانه‌ای شامل شمارش تعداد وقوع یک پیشامد بخصوص در محدودهٔ مشخصی از زمان است. اگر شمارش از تعداد معقولی که مورد انتظار است فراتر رود، آنگاه می‌توان فرض کرد که تهاجمی رخ داده است.

تحلیل آستانه‌ای به تنهائی یک تشخیص‌دهندهٔ مبتدی و غیرمؤثر، حتی در مورد حملات با پیچیدگی کم، است. هم اندازهٔ آستانه و هم فاصلهٔ زمانی بایستی تعیین شوند. بعلت تنوع رفتار کاربران، احتمال دارد که این آستانه‌ها نتایج مثبت اشتباه و یا نتایج منفی اشتباه قابل ملاحظه‌ای را ایجاد کنند. با وجود این، تشخیص‌دهنده‌های سادهٔ آستانه‌ای ممکن است در معیت تکنیک‌های پیچیدهٔ دیگر مفید واقع شوند.

تشخیص ناهنجاری مبتنی بر پروفایل، بر تعیین رفتارهای سابق تک‌تک کاربران و یا گروه کاربران تکیه کرده و سعی می‌کند انحراف از این نوع رفتار را نشان دهد. یک پروفایل ممکن است شامل مجموعه‌ای از پارامترها باشد تا تنها انحراف یک پارامتر از حالت نرمال آن، منجر به اعلام یک تهاجم نگردد.

تشخیص ناهنجاری مبتنی بر پروفایل بر اساس تحلیل سوابق ممیزی قرار دارد. سوابق ممیزی به دو طریق، ورودی تابع تشخیص تهاجم را تشکیل می‌دهند. اولاً، طراح بایستی نسبت به تعیین یک سری از معیارهای کمّی که می‌تواند رفتار کاربر را ارزیابی کند تصمیم بگیرد. تحلیل سوابق ممیزی در یک دوره از زمان می‌تواند برای تعیین پروفایل فعالیت یک کاربر معمولی مورد استفاده واقع شود. بنابراین سوابق ممیزی برای تعریف رفتار نوعی یک کاربر بکار می‌روند. ثانیاً، سابقهٔ ممیزی عملیات جاری، ورودی سیستم برای تشخیص تهاجم را تشکیل می‌دهند. یعنی مدل تشخیص تهاجم، سوابق ممیزی عملیات جاری را تحلیل کرده تا انحراف آن از رفتار متوسط کاربر را تشخیص دهد.

مثال‌هائی از معیارهای تشخیص، که در تشخیص تهاجم مبتنی بر پروفایل مفید هستند، بقرار زیراند:

- **شمارنده**: یک عدد صحیح غیرمنفی که می‌تواند زیاد شده ولی نمی‌تواند کم شود مگر اینکه با یک فرمان مدیریتی صفر گردد. معمولاً شمارش یک پیشامد در طول دورهٔ زمانی مشخصی مورد توجه است. نمونه‌هائی از این دست شامل تعداد تلاش‌ها برای ورود به سیستم از طرف یک کاربر مشخص در طول یک ساعت، تعداد دفعات اجرای یک فرمان در زمان یک اجلاس، و یا تعداد تلاش‌های ناموفق برای ورود به سیستم از طریق کلمات عبور غلط ظرف یک دقیقه است.

- **پیمانه**: یک عدد صحیح غیرمنفی که می‌تواند هم زیاد و هم کم شود. یک پیمانه معمولاً برای اندازه گیری مقدار یک چیز بکار می‌رود. مثال‌های این مورد نظیر تعداد اتصالات منطقی به یک برنامهٔ کاربردی و یا تعداد پیام‌های خروجی است که برای پردازش در صف انتظار قرار گرفته‌اند.
- **زمان سنج**: زمان بین وقوع دو پیشامد یکسان را نشان می‌دهد. مثال امر، فاصله زمانی بین دو ورود به سیستم از سوی یک کاربر مشخص است.
- **بکارگیری منابع**: اندازهٔ کمّی منابعی که در طول زمان مشخصی بکار گرفته شده‌اند را نشان می‌دهد. نمونهٔ این موارد عبارت از تعداد صفحات چاپ شده در خلال یکبار اتصال به سیستم و یا زمان صرف شده برای اجرای یک برنامه است.

با در دست داشتن این معیارها، تست‌های مختلفی را می‌توان برای تعیین اینکه فعالیت‌های جاری در محدودهٔ قابل قبول هستند و یا خیر، انجام داد. [DENN87] به عوامل زیر که می‌توانند مؤثر واقع شوند اشاره می‌کند:

- میانگین و انحراف معیار
- چندمتغیری
- فرآیند مارکوف
- سری‌های زمانی
- عملیاتی

ساده‌ترین تست آماری، اندازه‌گیری **میانگین و انحراف معیار** یک پارامتر در طول یک دورهٔ زمانی است. این اندازه‌ها منعکس‌کنندهٔ رفتار متوسط و تغییرات حول و حوش آن است. استفاده از میانگین و انحراف معیار قابل اعمال به شمارنده‌ها، زمان‌سنج‌ها و منابع دیگر می‌باشند. اما این معیارها معمولاً به تنهائی برای تشخیص تهاجم کافی نیستند.

یک مدل **چندمتغیری** برمبنای همبستگی بین دو یا چند متغیر قرار دارد. رفتار مهاجم را با ملاحظهٔ همبستگی بین وقایع (مثل همبستگی زمان پردازش با استفاده از یک منبع، و یا همبستگی تعداد دفعات ورود به سیستم با زمان استفاده از سیستم در هر بار)، با اطمینان بیشتری می‌توان تحلیل نمود .

یک مدل **فرآیند مارکوف** برای تعیین احتمال عبور بین حالات مختلف بکار می‌رود. بعنوان مثال این مدل ممکن است عبور از یک فرمان به فرمان مشخص دیگر را جستجو نماید.

یک مدل **سری زمانی** روی فواصل زمانی تکیه کرده و به دنبال زنجیره‌ای از وقایع می‌گردد که فاصلهٔ بین آنها یا خیلی زیادتر و یا خیلی کمتر از حد معمول می‌باشد. تست‌های آماری مختلفی می‌توانند برای تعیین زمان‌بندی غیرنرمال مورد استفاده قرار گیرند.

بالاخره یک **مدل عملیاتی**، بجای تحلیل اتوماتیک سوابق ممیزی، بر مبنای قضاوت در مورد آنچه غیرعادی تلقی می‌شود قرار دارد. در این روش نوعاً محدوده‌های ثابتی تعریف شده و اگر عملیاتی خارج از این محدوده‌ها انجام شود، شک نسبت به تهاجم قوت می‌پذیرد. چنین روشی وقتی خوب کار می‌کند که رفتار تهاجمی را بتوان از روی یک سری رفتارهای مشاهده شده نتیجه‌گیری کرد. بعنوان مثال، اگر در طول زمان کوتاهی تلاش زیادی برای ورود به سیستم انجام شود، می‌توان آن را نوعی تهاجم تلقی کرد.

بعنوان مثالی از بکارگیری این معیارها و مدل‌های مختلف، جدول ۱-۹ معیارهای متنوعی که در سیستم تشخیص تهاجم (IDES) انستیتوی تحقیقاتی استانفورد (SRI) بکار گرفته و آزمایش شده است را نشان می‌دهد [DENN87,JAVI91,LUNT88].

مزیت عمدهٔ استفاده از پروفایل‌های آماری این است که نیازی به داشتن معلومات قبلی از خطاهای امنیتی نیست. برنامهٔ تشخیص دهندهٔ تهاجم می‌آموزد که چه رفتاری نرمال است و آنگاه بدنبال انحراف از این رفتار نرمال می‌گردد. این روش متکی بر مشخصه‌های سیستم و نقاط آسیب‌پذیر آن نیست. در نتیجه این تکنیک می‌تواند بسهولت در سیستم‌های دیگر نیز بکار گرفته شود.

جدول ۱-۹ معیارهائی که می‌توانند برای تشخیص تهاجم بکار گرفته شوند

معیار	مدل	نوع تهاجم تشخیص داده شده
فعالیت مربوط به ورود به سیستم و اجلاس‌ها		
دفعات ورود به سیستم در روز و زمان	میانگین و انحراف معیار	مهاجمین احتمال دارد که در ساعات غیر اداری وارد سیستم شوند.
دفعات ورود به سیستم از مکان های مختلف	میانگین و انحراف معیار	مهاجمین ممکن است از مکانی وارد سیستم شوند که یک کاربر خاص بندرت و یا هیچگاه از آنجا وارد نمیشود.
زمان گذشته از آخرین ورود به سیستم	عملیاتی	وارد شدن از یک حساب مسدود.
زمان هر اجلاس	میانگین و انحراف معیار	انحراف قابل ملاحظه از زمان معمول ممکن است نشان یک مهاجم نقاب‌دار باشد.
میزان دیتای انتقال‌یافته به مکانی دوردست	میانگین و انحراف معیار	انتقال مقدار زیادی از دیتا به نقطه‌ای دور می‌تواند به نشان نشت اطلاعات حیاتی باشد.
بکارگیری منابع در هر اجلاس	میانگین و انحراف معیار	بکارگیری زیاد پردازشگر و I/O میتواند بعلت حضور مهاجم باشد.
دادن کلمهٔ عبور اشتباه در هنگام ورود به سیستم	عملیاتی	تلاش برای ورود به سیستم با حدس زدن کلمهٔ کلمهٔ عبور.
شکست در ورود به سیستم از ترمینال های بخصوص	عملیاتی	تلاش برای ورود غیرمجاز به سیستم.
فعالیت مربوط به اجرای برنامه‌ها و فرمان‌ها		
دفعات اجرای یک برنامه یا فرمان	میانگین و انحراف معیار	می‌تواند از طرف یک مهاجم باشد که از فرمان‌های مختلف استفاده می‌کند و یا ناشی از نفوذ موفق یک کاربر قانونی است که به فرمان‌های سطح بالا دسترسی یافته است.
بکارگیری منابع برنامه	میانگین و انحراف معیار	یک اندازهٔ غیر معقول میتواند نماینده ورود یک ویروس یا اسب تروا باشد که آثار جنبی آن بکارگیری پردازشگر یا بخش I/O است.
اجرا نشدن برنامه	عملیاتی	می‌تواند به تشخیص تهاجمی از سوی یک فرد برای دسترسی به امتیازات بالاتر منجر گردد.
فعالیت مربوط به دست یابی به فایل‌ها		
دفعات خواندن، نوشتن، ایجاد و حذف	میانگین و انحراف معیار	مقادیر غیرمعقول تلاش برای خواندن و نوشتن از طرف کاربران می‌تواند بیانگر حملهٔ بالماسکه‌ای یا مرورگری باشد.
رکوردهای خوانده شده و یا نوشته شده	میانگین و انحراف معیار	اندازه‌های غیرمعقول می‌تواند نمایش تلاش برای بدست آوردن اطلاعات حساس باشد.
شمارش دفعات تلاش ناموفق برای خواندن، نوشتن، ایجاد و یا حذف	عملیاتی	می‌تواند منجر به کشف کاربرانی شود که مصرانه قصد دسترسی به فایل‌های غیرمجاز را دارند.

تشخیص مبتنی بر قاعده تهاجم (Rule-Based Intrusion Detection)

روش‌های مبتنی بر قاعده، تهاجم را با مشاهدهٔ پیشامدها در یک سیستم و اعمال یک سری قواعد به آنها ردیابی می‌کنند تا به این تصمیم دست یابند که این مجموعهٔ فعالیت‌ها مشکوک و یا غیرمشکوک است. در بیان خیلی کلی، تمام روش‌ها را می‌توان به تشخیص ناهنجاری و یا شناسائی نفوذ دسته‌بندی کرد، اگرچه این دو گروه دارای نقاط مشترکی نیز هستند.

تشخیص مبتنی بر قاعدهٔ ناهنجاری (Rule-based anomaly detection)، از نظر روش برخورد و قدرت شبیه تشخیص آماری ناهنجاری است. در روش مبتنی بر قاعده، سوابق تاریخی فایل‌ها تحلیل شده تا رفتارهای کاربر شناسائی گشته و بصورت خودکار قواعدی که نمایش‌دهندهٔ این رفتارهاست تولیدگردند. قواعد ممکن است نشان‌دهندهٔ رفتارهای گذشته کاربران، برنامه‌ها، اولویت‌ها، شیارهای زمانی، ترمینال‌ها و غیره باشند. آنگاه رفتارهای زمان حال مشاهده شده و هر عمل با یک سری قواعد تهیه شده مقایسه گشته تا معلوم شود آیا این عمل جاری با تاریخچهٔ مربوط به اعمال گذشته همخوانی دارد یا خیر.

همانند تشخیص آماری ناهنجاری، تشخیص مبتنی بر قاعدهٔ ناهنجاری نیاز به شناخت نقاط آسیب‌پذیر امنیتی یک سیستم ندارد. بلکه روش، مبتنی بر مشاهدهٔ رفتار گذشته بوده و در واقع فرض می‌کند که آینده هم مثل گذشته خواهد بود. برای اینکه این روش اثربخش باشد، به یک پایگاه داده نسبتاً حجیم از قواعد نیازمندیم. برای مثال، روشی که در [VACC89] توصیف شده است بین 10^4 تا 10^6 قاعده دارد.

شناسائی مبتنی بر قاعده نفوذ (Rule-based penetration identification)، روش بسیار متفاوتی را برای تشخیص تهاجم بکار می‌برد که بر مبنای تکنولوژی سیستم‌های خبره قرار دارد. مشخصهٔ کلیدی چنین سیستم‌هائی استفاده از قواعد برای شناسائی نفوذهای شناخته شده و یا نفوذهائی که از نقاط ضعف شناخته شده استفاده می‌کنند می‌باشد. همچنین می‌توان قواعدی را تعریف کرد که رفتارهای مشکوک را، حتی وقتی که در محدودهٔ قانونی رفتارهای تعیین شده کاربردها هستند، شناسائی نماید. قواعدی که در این سیستم‌ها بکار می‌روند، نوعاً منحصر به ماشین و سیستم عامل بخصوصی می‌باشند. همچنین، چنین قواعدی بجای این‌که بر اساس تحلیل خودکار سوابق تولید شوند بتوسط «خبرگان» تدوین می‌شوند. روش معمول کار این است که با مدیران سیستم و تحلیل‌گران امنیتی مذاکره شده تا بر اساس تجربیات آنان مجموعه‌ای از سناریوهای نفوذ و وقایع کلیدی که امنیت سیستم هدف را به مخاطره می‌اندازند جمع‌آوری گردد. روشن است که کارآئی این روش، بستگی به مهارت افراد مرتبط در جمع‌آوری قواعد خواهد داشت.

مثال ساده‌ای از نوع قواعدی که می‌تواند مورد استفاده قرار گیرد را می‌توان در NIDX، یکی از اولین سیستم‌هائی که از شواهد تاریخی برای نسبت دادن درجاتی از شک و شبهه به فعالیت‌ها استفاده کرده است، پیدا کرد [BAUE88]. مثال‌هائی از این شواهد تاریخی چنین‌اند:

۱- کاربران نبایستی فایل‌هائی که در پوشه‌های شخصی کاربران دیگر است را بخوانند.
۲- کاربران نبایستی در فایل‌های کاربران دیگر بنویسند.
۳- کاربرانی که پس از گذشت چندساعت مجدداً وارد کامپیوتر می‌شوند معمولاً به همان فایل‌هائی رجوع می‌کنندکه قبلاً به آنها مراجعه کرده‌اند.
۴- کاربران معمولاً تجهیزات مرتبط با دیسک‌ها را مستقیماً باز نمی‌کنند بلکه متکی به قابلیت‌های سطوح بالاتر سیستم عامل هستند.
۵- کاربران نبایستی بیش از یکبار به یک سیستم وارد شوند.
۶- کاربران از برنامه‌های سیستمی کپی تهیه نمی‌کنند.

روش شناسائی نفوذ در IDES بیانگر استراتژی دنبال شده است. سوابق ممیزی در زمان جمع‌آوری بررسی شده و با قواعد مبنا مقایسه می‌شوند. اگر بین این دو شباهتی وجود داشته باشد *نرخ شک* کاربر افزایش می‌یابد. اگر تعداد تطبیق مشاهدات با قواعد مبنا از یک آستانه فراتر رود، آنگاه شناسائی یک بیگانه گزارش می‌شود.

روش IDES مبتنی بر بررسی سوابق ممیزی است. یک نقطهٔ ضعف این طرح، نداشتن قابلیت انعطاف است. برای یک سناریوی نفوذ، ممکن است بتوان سوابق متنوعی را که هر یک از آنها با دیگری اختلاف جزئی داشته باشد معیار قضاوت قرار داد. مشکل این است که شاید نتوان این مقدار تنوع زیاد را در قواعد صریحی جمع‌آوری نمود. روش دیگر تهیهٔ یک مدل سطح بالاتری است که مستقل از سوابق ممیزی عمل کند. مثالی از این دست، مدل گذر از حالات USTAT است [ILGU93]. USTAT بجای استفاده از رفتارهای مشخص با جزئیات کامل که در مکانیسم ثبت سوابق UNIX دیده شده است از رفتارهای کلی استفاده می‌کند. USTAT روی یک سیستم عامل SunOS ساخته شده که سوابق ممیزی ۲۳۹ پیشامد را فراهم می‌سازد. از این تعداد تنها ۲۸ تا برای پردازش اولیه بکار می‌روند که منجر به ۱۰ عمل مختلف می‌شوند (جدول ۲-۹). تنها با استفاده از این اعمال و پارامترهائی که با هر عمل بکار گرفته می‌شود، یک دیاگرام حالت که فعالیت مشکوک را تعیین می‌کند ساخته می‌شود. چون تعدادی از پیشامدهای قابل ممیزی مختلف به تعداد کمتری عمل نگاشت می‌شوند، روش خلق قواعد ساده‌تر است. علاوه براین، مدل دیاگرام حالت پس از مشاهدهٔ رفتارهای تهاجمی جدید، به فرم ساده‌تری قابل جرح و تعدیل خواهد بود.

جدول ۲-۹ عملیات USTAT در برابر انواع پیشامدهای SunOS

USTAT Action	SunOS Event Type
Read	open_r, open_rc, open_rtc, open_rwc open_rwtc, open_rt, open_rw, open_rwt
Write	truncate, ftruncate, creat, open_rtc, open_rwc, open_rwtc, open_rt, open_rw, open_rwt, open_w, open_wt open_wc, open_wct
Create	mkdir, creat, open_rc, open_rtc, open_rwc, open_rwtc, open_wc, open_wtc, mknod
Delete	rmdir, unlink
Execute	exec, execve
Exit	exit
Modify_Owner	chown, fchown
Modify_Perm	chmod, fchmod
Rename	rename
Hardlink	link

خطای نرخ پایهٔ (Base-Rate Fallacy)

یک سیستم تشخیص تهاجم برای اینکه کاربرد عملی داشته باشد، بایستی بتواند درصد قابل توجهی از تهاجم‌ها را تشخیص داده و در عین حال نرخ آلارم‌های کاذب را در حد قابل قبولی پائین نگاه دارد. اگر تنها درصد متوسطی از تهاجم‌های واقعی شناسائی شوند، سیستم حس امنیتی کاذبی را ایجاد خواهد کرد. از سوی دیگر اگر سیستم در مواقعی که تهاجمی وجود ندارد هشدار دهد (آلارم کاذب)، آنگاه یا مدیران سیستم آلارم‌ها را جدی نخواهند گرفت و یا زمان زیادی در راه تحلیل آلارم‌های کاذب تلف خواهد شد.

متاسفانه، بعلت احتمالی بودن وقایع مرتبط با این مساله، بسیار مشکل است که بتوان نرخ بالای کشف تهاجمات را با نرخ کم آلارم‌های کاذب همراه کرد. بطور کلی اگر تعداد واقعی تهاجمات در مقایسه با تعداد دفعات استفادهٔ قانونی از سیستم کم باشد، آنگاه نرخ آلارم‌های کاذب بالا خواهد بود مگر اینکه تست‌ها بتوانند بصورت فوق‌العاده‌ای بین صحت و سقم وقایع تفاوت قائل شوند. یک بررسی از سیستم‌های تشخیص تهاجم که در [AXEL00] گزارش شده است، نشان می‌دهد که سیستم‌های جاری بر مشکل خطای نرخ پایهٔ فایق نشده‌اند. برای آگاهی مختصری از ریاضی این بحث به ضمیمهٔ ۹- الف مراجعه شود.

تشخیص توزیع شدهٔ تهاجم

تا زمان‌های اخیر، کار بر روی سیستم‌های تشخیص تهاجم منحصر به تجهیزات یک سیستم منفرد و متکی به خود بود. سازمان‌های کنونی نوعاً نیاز به یک دفاع کارآمد از مجموعهٔ توزیع شدهٔ میزبان‌هائی را دارند که بتوسط شبکه‌های LAN و یا اینترنت بهم متصل‌اند. اگرچه می‌توان برای هر میزبان، یک عملیات دفاعی تشخیص تهاجم را بطور جداگانه بوجود آورد ولی با هماهنگی و همکاری بین سیستم‌های تشخیص تهاجم منصوبه بر روی شبکه، می‌توان دفاع مؤثرتری را ایجاد کرد.

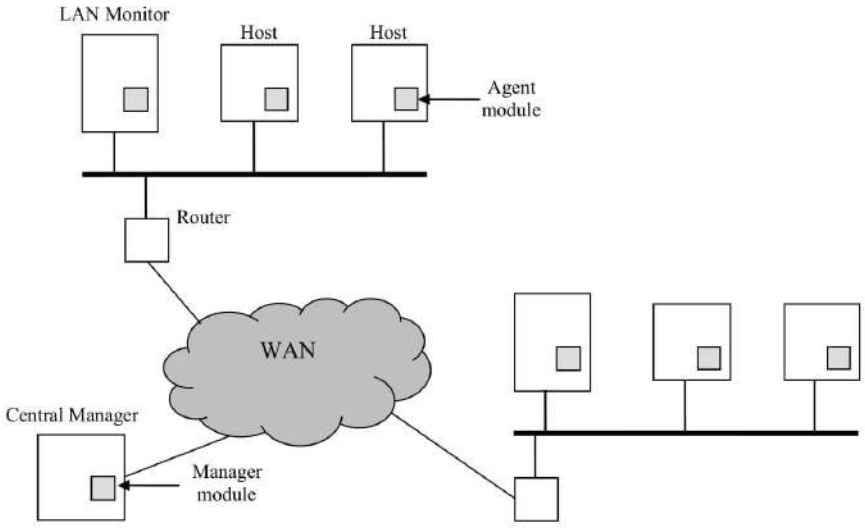
Porras مقوله‌های عمده زیر را در طراحی یک سیستم توزیع‌شدهٔ تشخیص تهاجم یادآوری می‌کند[PORR92]:

- یک سیستم توزیع‌شدهٔ تشخیص تهاجم ممکن است به استفاده از فرمت‌های مختلف سوابق ممیزی نیاز داشته باشد. در یک محیط نامتجانس، سیستم‌های مختلف از روش‌های جمع‌آوری سوابق بومی مختلفی استفاده کرده و اگراز سیستم تشخیص تهاجم استفاده نمایند، ممکن است فرمت‌های مختلفی از سوابق ممیزی امنیتی را بکار گیرند.
- یک و یا چند گره از شبکه بعنوان نقاط جمع‌آوری و تحلیل داده‌های مرتبط با سیستم‌های شبکه بکار خواهند رفت. بنابراین دیتای مربوط به سوابق ممیزی یا بصورت خام و یا بصورت خلاصه، بایستی در عرض شبکه منتقل گردد و لازم است که اصالت و محرمانگی این داده‌ها تضمین شود. اصالت داده‌ها از اینجهت مورد نیاز است که یک مهاجم نتواند فعالیت‌های خود را با تغییر دادن سوابق ممیزی انتقال یافته، در پشت نقابی پنهان نماید. محرمانگی داده‌ها نیز از اینجهت مهم است که اطلاعات ممیزی می‌توانند اطلاعات گرانقیمتی باشند.
- هم یک معماری متمرکز و هم یک معماری غیرمتمرکز می‌تواند برای این منظور بکار رود. در یک معماری متمرکز، تنها یک نقطهٔ مرکزی جمع‌آوری و تحلیل داده‌های ممیزی وجود دارد. این امر وظیفهٔ ارتباط دادن گزارشات ورودی به یکدیگر را تسهیل نموده ولی باعث ایجاد یک گلوگاه و یا یک نقطهٔ خرابی منفرد می‌شود. در یک معماری غیرمتمرکز بیش از یک مرکز جمع‌آوری و تحلیل داده‌ها وجود داشته که البته بایستی با هم هماهنگی نموده و اطلاعات را مبادله کنند.

مثال خوبی از یک سیستم توزیع‌شدهٔ تشخیص تهاجم، در دانشگاه کالیفرنیا در Davis طراحی شده است [HEBE92,SNAP91]. شکل ۲-۹ معماری کلی این طرح را نشان می‌دهدکه از سه مؤلفهٔ اصلی تشکیل می‌گردد:

- **مدول عامل میزبان**: یک مدول جمع‌آوری سوابق ممیزی که بصورت یک پردازش پشت پرده در یک سیستم پایش شده عمل می‌کند. هدف آن جمع‌آوری داده‌های مربوط به پیشامدهای مرتبط با امنیت در سیستم میزبان و انتقال آن به یک مدیر مرکزی است.
- **مدول عامل پایشگر LAN**: بهمان روش مدول عامل میزبان کار کرده بجز اینکه ترافیک LAN را تجزیه و تحلیل کرده و نتایج را به مدیر مرکزی گزارش می‌کند.
- **مدول مدیریت مرکزی**: گزارشات را از میزبان‌ها و پایشگر LAN گرفته، آنها را پردازش نموده، همبستگی آنها را جستجو کرده و تهاجم را تشخیص می‌دهد.

روش طوری طراحی شده است که مستقل از نوع سیستم عامل و یا روش جمع‌آوری سوابق ممیزی سیستم است. شکل ۳-۹ [SNAP91] روش کلی برخورد با مسأله را نشان می‌دهد. واحد عمل‌کننده هر سابقه‌ای را که بتوسط سیستم‌های ثبت سوابق بومی جمع‌آوری می‌شود می‌گیرد. فیلتر نشان داده شده طوری طراحی شده است که تنها سوابقی را که از نظر امنیتی دارای اهمیت هستند نگاه دارد. سوابق نگاهداری شده آنگاه طوری فرمتشان عوض می‌شود تا به فرم یک فرمت استاندارد که سوابق ممیزی میزبان (HAR) خوانده می‌شود درآیند. آنگاه یک مدول منطقی مبتنی بر الگو، سوابق را بمنظور کشف فعالیت‌های مشکوک تجزیه و تحلیل می‌کند. در پائین‌ترین سطح، واحد عمل‌کننده به دنبال پیشامدهائی می‌گردد که مستقل از هر پیشامد قبلی مورد توجه‌اند. شکست در دسترسی به یک فایل، تلاش برای دست‌یابی به فایل‌های سیستم، و تغییر



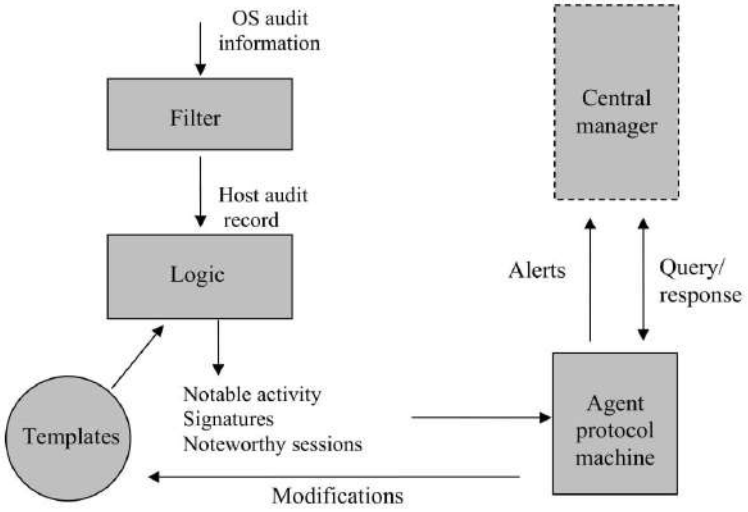
شکل ۲-۹ معماری برای تشخیص توزیع شدهٔ تهاجم

در کنترل دست‌یابی به یک فایل، مثال‌هائی از این نوع‌اند. در یک سطح بالاتر، واحد عمل‌کننده به دنبال زنجیره‌ای از پیشامدها مانند پترن حمله‌های شناخته شده (امضاءها) می‌گردد. بالاخره بر اساس پروفایل تاریخچهٔ رفتاری یک کاربر، رفتار مشکوک آن کاربر را مورد توجه قرار می‌دهد. در این مقوله، تعداد برنامه‌های اجراشده، تعداد فایل‌های مورد استفاده قرار گرفته و امثال آن مورد توجه خواهند بود.

وقتی فعالیت مشکوکی تشخیص داده می‌شود، یک هشدار برای مدیریت مرکزی ارسال می‌شود. مدیریت مرکزی شامل یک سیستم خبره بوده که می‌تواند از داده‌های دریافت شده، گمانی را استخراج نماید. مدیریت ممکن است از هر میزبان، نسخهٔ HAR او را درخواست نماید تا آنها را با یکدیگر مقایسه کند.

پایشگر LAN نیز اطلاعاتی را برای مدیریت مرکزی فراهم می‌سازد. این عامل، اتصالات میزبان‌ها به یکدیگر و سرویس‌های استفاده شده و حجم ترافیک را ثبت می‌کند. همچنین پیشامدهای قابل توجه از قبیل تغییر ناگهانی بار شبکه، استفاده از سرویس‌های مرتبط با امنیت، و فعالیت‌های شبکه مانند *rlogin* بتوسط همین عامل جستجو می‌شود.

معماری نشان داده شده در شکل‌های ۲-۹ و ۳-۹ کاملاً عام و قابل انعطاف است. این معماری زیربنای لازم برای برخوردی مستقل از سیستم، که می‌تواند از تشخیص تهاجم مربوط به یک سیستم منفرد تا ارتباط دادن فعالیت‌های سایت‌های مختلف شبکه برای تشخیص یک فعالیت مشکوک را بپوشاند، تأمین می‌کند.



شکل ۳-۹ معماری عامل

طعمه‌ها (Honeypots)

یک نوآوری نسبتاً جدید در تکنولوژی تشخیص تهاجم، honeypot است. honeypotها طعمه‌هائی هستند که با اغفال مهاجم او را از سیستم‌های حیاتی دور نگاه می‌دارند. honeypotها برای مقاصد زیر طراحی می‌شوند:

- منحرف کردن مهاجم از دست‌یابی به سیستم‌های حیاتی
- جمع‌آوری اطلاعات در مورد فعالیت‌های مهاجم
- تشویق مهاجم به باقی ماندن در سیستم تا زمانی که مدیران سیستم بتوانند عکس‌العمل نشان دهند

این سیستم‌ها با اطلاعات ساختگی که بظاهر ارزشمند جلوه می‌کنند طوری پر می‌شوند که کاربران قانونی معمولاً به آنها دسترسی پیدا نمی‌کنند. بنابراین هرگونه تلاش برای دست‌یابی به honeypot مشکوک تلقی خواهد شد. سیستم با پایشگرهای حساس و ثبت‌کننده‌های وقایع طوری تجهیز شده است که این نوع دست‌یابی‌های مشکوک را تشخیص داده و اطلاعات مربوط به فعالیت‌های تهاجمی را جمع‌آوری نماید. چون هر حمله‌ای بر علیه honeypot موفقیت‌آمیز بنظر خواهد رسید، مدیران شبکه فرصت کافی برای تجهیز شدن و دنبال کردن مهاجم، قبل از اینکه سیستم‌های اصلی در معرض خطر قرار گیرند، را خواهند داشت.

تلاش‌های اولیه، در استفاده از یک کامپیوتر honeypot و آدرس‌های IP مخصوصی که برای جلب مهاجمین انتخاب شده بود، خلاصه می‌شد. تحقیقات جدیدتر بر ساخت شبکه‌های honeynet که تمام سازمان را تقلید کرده و احتمالاً از دیتای واقعی یا شبیه‌سازی شده استفاده کنند متمرکز شده است. بمحض اینکه هَکرها در محدودهٔ شبکه قرار گیرند، مسئولین می‌توانند رفتار آنها را با جزئیات کامل مشاهده کرده و سیاست‌های دفاعی مناسب را بیابند.

فرمت مبادلهٔ تشخیص تهاجم (Intrusion Detection Exchange Format)

برای تسهیل ساخت سیستم‌های گستردهٔ تشخیص تهاجم که بتوانند در عرض محدودهٔ وسیعی از رایانه‌ها و محیط‌ها عمل نمایند، نیاز به استانداردهائی است که بتوانند عملیات مختلف بین‌شبکه‌ای را حمایت کنند. چنین استانداردهائی، کانون فعالیت گروه کاری تشخیص تهاجم IETF را تشکیل می‌دهد. هدف این گروه کاری تعریف فرمت‌های دیتا و مبادلهٔ روش‌هائی برای به اشتراک گذاشتن اطلاعات جالب در زمینهٔ تشخیص تهاجم و سیستم‌های پاسخگو به مساله و همچنین سیستم‌های مدیریتی است که ممکن است نیاز به تعامل با آنها داشته باشند. خروجی‌های این سیستم کاری، شامل موارد زیراند:

۱- یک سند نیازمندی‌ها که توصیف‌کنندهٔ نیازهای عملیاتی سطح بالا برای ارتباط بین سیستم‌های تشخیص تهاجم، و همچنین نیازهای ارتباطی بین سیستم‌های تشخیص تهاجم با سیستم‌های مدیریت، بهمراه دلایل مستدل و منطقی برای این نیازهاست. برای توجیه این نیازها بایستی از سناریوهای مناسب استفاده شود.

۲- تعیین مشخصه‌های یک زبان محاورهٔ مشترک برای تشخیص،که فرمت داده‌هائی که این نیازها را ارضاء می‌کنند را هم تعیین نماید.

۳- یک سند ساختاری که بهترین پروتکل‌های موجود برای ارتباط بین سیستم‌های تشخیص تهاجم را شناسائی نموده و مشخص می‌سازد که فرمت‌های انتخاب شده دیتا چگونه با آنها مرتبط‌اند.

در زمان نگارش این مطلب، تمام این اسناد در مرحلهٔ پیش‌نویس اسناد اینترنتی هستند.

۹-۳ مدیریت کلمهٔ عبور

حفاظت کلمهٔ عبور

خط مقدم دفاع در مقابل مهاجمین سیستم، کلمهٔ عبور است. تقریباً تمام سیستم‌های چندکاربره از کاربران می‌خواهندکه نه تنها یک نام یا شناسه (ID) را وارد سیستم کنند بلکه یک کلمهٔ عبور (password) را نیز عرضه نمایند. کلمهٔ عبور برای سنجش اعتبار ID فردی که می‌خواهد به سیستم وارد شود مورد استفاده قرار می‌گیرد. ID نیز بنوبهٔ خود، امنیت را به راه‌های زیر فراهم می‌آورد:

- ID تعیین می‌کند که آیا کاربر برای دست‌یابی به سیستم دارای اعتبار است. در بعضی سیستم‌ها، تنها کسانی که قبلاً ID آنها در سیستم ذخیره شده است می‌توانند به سیستم دست یابند.
- ID امتیازاتی را که به کاربر اختصاص داده شده است، تعیین می‌کند. کاربران محدودی ممکن است دارای وظایف سوپروایزری سیستم باشند که طبیعتاً بایستی به آنها این اجازه داده شود که فایل‌ها را خوانده و یا عملیاتی را انجام دهند که معمولاً از نظر پنهان است. بعضی سیستم‌ها دارای امکانات پذیرش میهمان و یا افراد ناشناس بوده و طبیعتاً این افراد بایستی محدودیت‌های زیادتر و یا امتیازات کمتری نسبت به سایرین داشته باشند.
- ID بصورتی که معمولاً کنترل دست‌یابی منصفانه نامیده می‌شود مورد استفاده قرار می‌گیرد. بعنوان مثال، یک کاربر ممکن است با لیست نمودن IDهای کاربران دیگر به آنها اجازه دهد تا فایل‌هائی که متعلق به اوست را بخوانند.

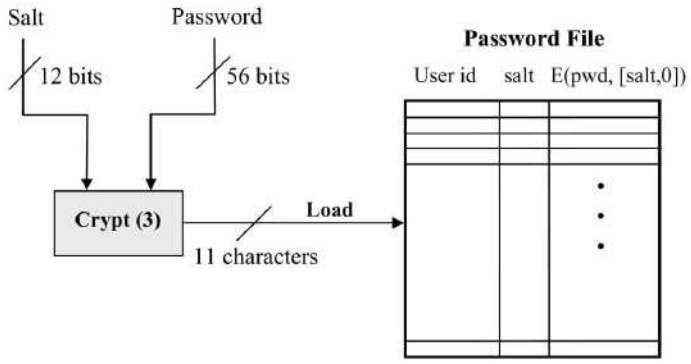
آسیب‌پذیری کلمات عبور

برای فهم ماهیت خطراتی که متوجه سیستم‌های مبتنی بر کلمهٔ عبور است، اجازه دهید روشی که بصورت گسترده در UNIX استفاده می‌شود و در آن هیچگاه کلمات عبور به فرم باز ذخیره نمی‌شوند را مطالعه کنیم. روش چنین است (شکل ۴-۹الف). هر کاربر یک کلمهٔ عبور که حداکثر دارای هشت کاراکتر قابل چاپ است را امتحان می‌کند. این کلمهٔ عبور به یک مقدار ۵۶- بیتی (با استفاده از کُد ۷- بیتی ASCII) تبدیل می‌شود که بعنوان کلید ورودی یک روش رمزنگاری بکار می‌رود. روش رمزنگاری که crypt (3) نام دارد مبتنی بر DES است. الگوریتم DES بنحوی دستکاری شده است که از یک "salt" ۱۲- بیتی استفاده می‌کند. نوعاً این مقدار وابسته به زمانی است که کلمهٔ عبور به کاربر اختصاص داده می‌شود. الگوریتم DES دستکاری شده، روی یک دیتای ورودی که یک بلوک ۶۴- بیتی از 0 های باینری است عمل می‌کند. خروجی الگوریتم سپس بعنوان ورودی یک رمزنگار دوم مورد استفاده قرار می‌گیرد. این پردازش مجموعاً برای ۲۵ بار رمزنگاری تکرار می‌شود. خروجی ۶۴- بیتی این پردازش آنگاه به یک دنبالهٔ ۱۱-کارکتری تبدیل می‌گردد. آنگاه hash کلمهٔ عبور بهمراه کپی متن ساده "salt" در فایل کلمهٔ عبور متناظر با ID کاربر ذخیره می‌گردد. نشان داده شده است که این متد در برابر انواع مختلفی از حملات شکستن رمز امن است [WAGN00].

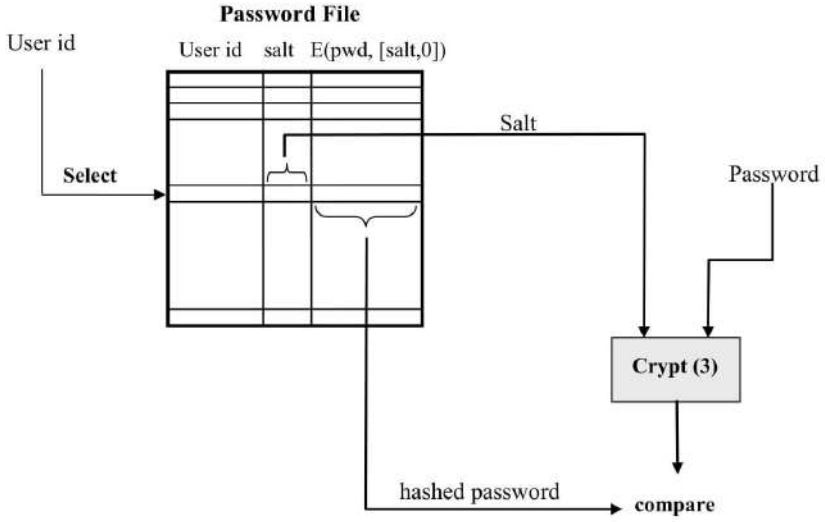
"salt" سه منظور را برآورده می‌سازد:

- نمی‌گذارد کلمات عبور مشابه در فایل کلمات عبور مشاهده شوند. حتی اگر دو کاربر کلمهٔ عبور یکسانی را انتخاب کنند، چون این کلمات عبور در زمان‌های متفاوتی به کاربران اختصاص یافته‌اند، بنابراین کلمات عبور پردازش شدهٔ انتهائی دو کاربر متفاوت خواهند بود.

- بطور مؤثری طول کلمات عبور را زیاد می‌کند بنحوی که کاربر لازم نیست دو کاراکتر اضافی را بخاطر بسپارد. در نتیجهٔ این عمل، تعداد کلمات عبور بمیزان ۴,۰۹۶ برابر زیاد شده و حدس زدن یک کلمهٔ عبور سخت‌تر می‌گردد.
- استفاده از سخت‌افزار آماده DES، که می‌تواند حملهٔ همه جانبه به کلمهٔ عبور را تسهیل کند، را غیرممکن می‌سازد.



(الف) بارگذاری یک کلمهٔ عبور جدید



(ب) تأئید یک کلمهٔ عبور

شکل ۴-۹ روش برخورد با کلمهٔ عبور در UNIX

وقتی یک کاربر می‌خواهد به یک سیستم UNIX وارد شود، او یک ID و یک کلمهٔ عبور را به سیستم عرضه می‌دارد. سیستم عامل با استفاده از ID وارد فایل لیست کلمات عبور شده و "salt" و کلمهٔ عبور رمزشده را بازخوانی می‌کند. "salt" و کلمهٔ عبور ارائه شده بتوسط کاربر، بعنوان ورودی‌های الگوریتم رمزنگاری مورد استفاده قرار می‌گیرند. اگر نتیجه امر برابر اندازهٔ ذخیره شدهٔ قبلی بود، کلمهٔ عبور پذیرفته می‌شود.

عمل رمزنگاری برای خنثی کردن حملاتی که استراتژی آنها حدس زدن کلمهٔ عبور است بکار می‌رود. اجرای نرم‌افزاری DES در مقایسه با نمونهٔ سخت‌افزاری آن کند بوده و تکرار ۲۵ مرتبهٔ آن، زمان لازم را ۲۵ برابر بیشتر می‌کند. از زمان طراحی اولیهٔ این روش تا کنون، دو تغییر در الگوریتم آن داده شده است. اولاً نسخه‌های جدیدتر الگوریتم، سرعت آن را افزایش داده‌اند. بعنوان مثال کِرم اینترنتی توصیف شده در فصل ۱۰ قادر شد تا با استفاده از یک الگوریتم رمزنگاری بهره‌ورتر از آنچه که در ابتدا بصورت استاندارد در سیستم‌های UNIX نصب شده بود، صدها کلمهٔ عبور را در طول مدت نسبتاً کوتاهی، در سیستم مورد تهاجم، حدس بزند. ثانیاً، عملکرد سخت‌افزارها روز به روز بهتر شده بطوری که اجازه می‌دهد الگوریتم‌های نرم‌افزاری نیز سریع‌تر عمل کنند.

بنابراین کلمهٔ عبور در سیستم UNIX با دو تهدید مواجه است. اول اینکه یک کاربر ممکن است با استفاده از تسهیلات پذیرش میهمان و یا روش دیگری به ماشین دسترسی پیدا کرده و آنگاه یک برنامهٔ حدس‌زنندهٔ کلمهٔ عبور، که شکنندهٔ کلمهٔ عبور خوانده می‌شود، را روی ماشین اجرا کند. مهاجم بایستی قادر باشد تا صدها و شاید هزارها کلمات عبور ممکن را با بکارگیری حداقل منابع امتحان کند. علاوه برآن، اگر یک دشمن قادر باشد تا یک نسخه از فایل کلمات عبور را بدست آورد، آنگاه یک برنامهٔ cracker (شکننده) خواهد توانست تا سر فرصت و روی ماشین دیگری کلمات عبور را کشف کند.

بعنوان مثال در ماه اوت سال ۱۹۹۳ میلادی، حضور یک شکنندهٔ کلمات عبور روی اینترنت گزارش گردید[MADS93]. این عمل با استفاده از یک کامپیوتر موازی Thinking Machines Corporation و به میزان ۱,۵۶۰ رمزنگاری در ثانیه برای هر واحد حاصل گردید. با استفاده از چهار واحد پردازش در هر گره پردازشگر (یک پیکربندی استاندارد)، این عمل به ۸۰۰,۰۰۰ رمزنگاری در ثانیه روی یک ماشین با ۱۲۸ گره (که اندازهٔ متوسطی است)، و ۶/۴ میلیون رمزنگاری در ثانیه روی یک ماشین با ۱,۰۲۴ گره افزایش یافت.

حتی این نرخ حدسیات، یک مهاجم صاحب عقل را قانع نمی‌سازد تا با استفاده از یک حملهٔ همه‌جانبه بخواهد تمام ترکیبات ممکن کاراکترها برای کشف یک کلمهٔ عبور را امتحان کند. بجای آن شکننده‌های کلمات عبور اغلب از این واقعیت استفاده می‌کنند که بعضی کاربران از کلمه‌های عبوری استفاده می‌کنند که به سهولت قابل حدس زدن هستند.

برخی کاربران وقتی اجازه دارند که کلمهٔ عبور را خود انتخاب کنند، کلمه‌ای را برمی‌گزینند که بطور ساده‌لوحانه‌ای کوتاه است. نتایج یک بررسی در دانشگاه Purdue در جدول ۳-۹ نشان داده شده است. در این بررسی، تغییر کلمات عبور روی ۵۴ ماشین مورد مطالعه قرار گرفت که تقریباً ۷,۰۰۰ شمارهٔ کاربری را در بر می‌گرفت. تقریباً ۳٪ کلمات عبور از ۳ کاراکتر و یا کمتر تشکیل شده بودند. یک مهاجم می‌توانست حملهٔ خود را با آزمایش کردن تمام کلمات عبور ممکن با طول ۳ و کمتر آغاز کند. یک علاج ساده این است که سیستم هر کلمهٔ عبور کمتر از مثلاً ۶ کاراکتر را نپذیرد و یا مثلاً کاربران را مجبور کند تا کلمات عبوری که حتماً ۸ کاراکتر داشته باشند را انتخاب نمایند. اکثر کاربران در مورد اعمال چنین محدودیتی شکایت نخواهند کرد.

طول کلمهٔ عبور تنها بخشی از مشکل است. بسیاری افراد وقتی اجازه دارند تا کلمهٔ عبور را خود انتخاب نمایند، کلمه‌ای را برمی‌گزینند که قابل حدس است. آنها مثلاً نام خود و یا نام خیابان منزل خود و یا یک کلمهٔ موجود در کتاب لغت و غیره را انتخاب می‌کنند. این امر کار شکستن کلمهٔ عبور را تسهیل می‌کند و شکنندهٔ کلمهٔ عبور فقط کافی است فایل کلمات عبور را

جدول ۳-۹ طول‌های مشاهده شده برای کلمات عبور [SPAF92a]

Length	Number	Fraction of Total
1	55	.004
2	87	.006
3	212	.02
4	449	.03
5	1260	.09
6	3035	.22
7	2917	.21
8	5772	.42
Total	13787	1.0

با کلمات عبور محتمل مقایسه نماید. چون بسیاری از مردم از کلمات عبور قابل حدس استفاده می‌کنند، این نوع استراتژی تقریباً در تمام سیستم‌ها موفق خواهد بود.

یک گزارش از موفقیت‌آمیزبودن حدس زدن کلمهٔ عبور در [KLEI90]، نمایش داده شده است. نویسندهٔ گزارش، فایل‌های کلمات عبور UNIX را از منابع متنوعی جمع‌آوری نموده که شامل تقریباً ۱۴,۰۰۰ کلمهٔ عبور رمزشده است. نتیجهٔ، که نویسنده بحق آن را رعب‌آور توصیف کرده است، در جدول ۴-۹ نشان داده شده است. رویهم‌رفته تقریباً یک چهارم کلمات عبور، لو رفته بودند. استراتژی مورد استفاده چنین بود:

۱- نام کاربر، حرف اول نام او، نام فامیل او، شمارهٔ حساب و سایر اطلاعات شخصی او را امتحان کنید. در مجموع ۱۳۰ تبدیل مختلف برای هر کاربر مورد آزمایش قرار گرفته بود.

۲- کلمات موجود در کتاب لغت‌های مختلف را امتحان کنید. نویسنده، یک کتاب لغت با بیش از ۶۰,۰۰۰ لغت را جمع‌آوری کرده است که شامل کتاب لغت برخط روی سیستم، و لیست‌های مختلفی برابر آنچه در جدول آمده است می‌باشد.

۳- تبدیل‌های متفاوتی روی لغات جمع‌آوری شده در بند ۲ انجام دهید. این شامل تبدیل حرف اول لغت به حرف بزرگ و یا یک کاراکتر کنترلی، تبدیل تمام لغت به حروف بزرگ، معکوس کردن کلمه و تبدیل حرف "O" به رقم "0" و غیره است. این تبدیل‌ها تقریباً یک میلیون کلمه به لیست اضافه می‌کنند.

۴- تبدیل‌های بزرگ کردن یک یا چند حرف که در بند ۳ انجام نشده است را به لغات بند ۲ اعمال کنید. این امر تقریباً دو میلیون کلمهٔ اضافی را به لیست اضافه می‌نماید.

بنابراین، تست یادشده تقریباً ۳ میلیون کلمه را در بر می‌گرفت. با استفاده از سریع‌ترین Thinking Machine که قبلاً از آن یاد شد، رمزنگاری تمام این کلمات با استفاده از تمام مقادیر "salt" به زمانی کمتر از یک ساعت نیاز داشت. بخاطر داشته باشید که چنین جستجوی جامعی تقریباً ۲۵٪ احتمال موفقیت داشته است در حالی که حتی یک مورد موفقیت هم ممکن است برای دست‌یابی به امتیازات وسیعی در سیستم کافی باشد.

جدول ۴-۹ کلمات عبور لو رفته از یک مجموعهٔ نمونهٔ با ۱۳,۷۹۷ حساب [KLEI90]

Type of Password	Search Size	Number of Matches	Percentage of Password Matched	Cost/Benefit Ratio *
User/account name	130	368	2.7%	2.830
Character sequences	866	22	0.2%	0.025
Numbers	427	9	0.1%	0.021
Chinese	392	56	0.4%	0.143
Place names	628	82	0.6%	0.131
Common names	2239	548	4.0%	0.245
Female names	4280	161	1.2%	0.038
Male names	2866	140	1.0%	0.049
Uncommon names	4955	130	0.9%	0.026
Myths&legends	1246	66	0.5%	0.053
Shakespearean	473	11	0.1%	0.023
Sport terms	238	32	0.2%	0.134
Science fiction	691	59	0.4%	0.085
Movies and actors	99	12	0.1%	0.121
Cartoons	92	9	0.1%	0.098
Famous people	290	55	0.4%	0.190
Phrases and patterns	933	253	1.8%	0.271
Surnames	33	9	0.1%	0.273
Biology	58	1	0.0%	0.017
System dictionary	9683	1027	7.4%	0.052
Machine names	9018	132	1.0%	0.015
Mnemonics	14	2	0.0%	0.143
King James Bible	7525	83	0.6%	0.011
Miscellaneous words	3212	54	0.4%	0.017
Yiddish words	56	0	0.0%	0.000
Asteroids	2407	19	0.1%	0.007
Total	62727	3340	24.2%	0.053

* Cost /Benefit Ratio = Number of Matches / Search Size

کنترل دست‌یابی

یکی از راه‌های مقابله با تهاجم به کلمات عبور، جلوگیری از دست‌یابی دشمن به فایل کلمات عبور است. اگر قسمتی از فایل که کلمات عبور رمزشده در آن نگهداری می‌شود تنها بتوسط یک کاربر دارای امتیازات ویژه قابل دست‌یابی باشد، آنگاه دشمن بدون دانستن کلمهٔ عبور آن کاربر ویژه، امکان دسترسی به فایل را نخواهد داشت. [SPAF92a] چند اشکال در این استراتژی را یادآوری می‌کند:

- بسیاری از سیستم‌ها، شامل بیشتر سیستم‌های UNIX، در معرض تعرض‌های پیش‌بینی نشده قرار دارند. همینکه مهاجمی توانست به نحوی به سیستم راه یابد، ممکن است بخواهد مجموعه‌ای از کلمات عبور را بدست آورد تا از طریق حساب‌های مختلف برای ورود به سیستم اقدام نموده و بدین نحو خطر تشخیص خود را کم کند. همچنین یک کاربر دارای اشتراک ممکن است بخواهد از حساب مشترک دیگری استفاده کرده تا داده‌های گران‌قیمت را خوانده و یا در سیستم خرابکاری نماید.

- یک رویداد حفاظتی ممکن است فایل کلمات عبور را ناخوانا ساخته و در نتیجه تمام اشتراک‌ها را به مخاطره اندازد.
- برخی کاربران، دارای اشتراک روی ماشین‌های دیگر در نواحی حفاظتی دیگری هستند ولی از همین کلمهٔ عبور در آنجا هم استفاده می‌کنند. بنابراین اگر فردی بتواند کلمهٔ عبور آنها روی یک ماشین را کشف کند، ماشین دیگری در محل دیگری نیز می‌تواند در معرض خطر واقع شود.

بهمین دلیل، یک استراتژی مؤثرتر این است که کاربران را مجبور کرد تا کلمهٔ عبوری را انتخاب کنند که حدس زدن آن مشکل باشد.

استراتژی‌های انتخاب کلمهٔ عبور

درسی که از دو آزمایش بالا (جداول ۳-۹ و ۴-۹) آموخته می‌شود این است که، در صورتی که امر به خود آنها واگذار شود، بسیاری از کاربران کلمهٔ عبوری را انتخاب می‌کنند که یا خیلی کوتاه بوده و یا حدس زدن آن خیلی آسان است. از سوی دیگر اگر به کاربران کلمات عبوری اختصاص یابد که از هشت کاراکتر قابل چاپ تصادفی تشکیل شده باشد، شکستن کلمهٔ عبور تقریباً غیرممکن خواهد بود. اما در این صورت بخاطر سپردن کلمهٔ عبور برای بیشتر کاربران امری غیرممکن است. خوشبختانه حتی اگر فضای کلمات عبور را به دنباله‌ای از کاراکترها که بطور معقول قابل بخاطر سپردن هستند محدود کنیم، باز هم اندازهٔ فضا بحدی بزرگ خواهد ماند که شکستن کلمهٔ عبور عملی نخواهد بود. بنابراین هدف ما این است که کلمات عبور قابل حدس را بنحوی حذف کنیم که باز هم کاربر قادر باشد کلمهٔ عبور خود را بخاطر بسپارد. در این مقوله از چهار تکنیک عمده استفاده می‌شود:

- تعلیم کاربر
- استفاده از کلمات عبوری که بتوسط کامپیوتر تولید می‌شود
- کنترل غیرفعال کلمهٔ عبور
- کنترل فعال کلمهٔ عبور

کاربران را بایستی از اهمیت بکاربردن کلمات عبوری که بسختی قابل حدس زدن هستند آگاه کرد و برای آنها خط‌مشی مناسبی برای انتخاب کلمات عبور قوی ارائه داد. این استراتژی **تعلیم کاربر** در اکثر سازمان‌ها، علی‌الخصوص در جاهائی که جمعیت زیاد بوده و تغییر و تحول زیادی در جریان است، احتمال موفقیت کمی دارد. بسیاری از کاربران خط‌مشی‌ها را بسهولت زیرپا می‌گذارند. برخی دیگر ممکن است قضاوت صحیحی نسبت به یک کلمهٔ عبور محکم نداشته باشند. مثلاً کاربران زیادی (اشتباهاً) معتقدند که معکوس کردن حروف یک کلمه، و یا بزرگ نوشتن حرف آخر یک کلمه، آن را غیرقابل حدس خواهد ساخت.

کلمات عبور تولید شده بتوسط کامپیوتر نیز دارای مشکل‌اند. اگر کلمهٔ عبور طبیعت کاملاً تصادفی داشته باشد، کاربر قادر به بخاطر سپردن آن نخواهد بود. حتی اگر کلمهٔ عبور قابل تلفظ هم باشد، کاربر ممکن است در بخاطر سپردن آن مشکل داشته و وسوسه شود که آن را یادداشت نماید. بطور کلی، روش ساخت کلمهٔ عبور بتوسط کامپیوتر دارای تاریخچه‌ای است که عدم تمایل به پذیرش آن از سوی کاربران را نشان می‌دهد. FIPS PUB 181 یکی از بهترین طراحی‌های مربوط به تولید اتوماتیک کلمهٔ عبور را ارائه داده است. این استاندارد نه تنها روش عمل را توصیف می‌کند بلکه کُد برنامه نوشته شده به

زبان C را نیز ارائه می‌دهد. الگوریتم با ایجاد بخش‌های قابل تلفظ، آنها را به‌هم چسبانده تا یک کلمهٔ عبور را ایجاد نماید. یک مولد اعداد تصادفی برای تولید دنبالهٔ کاراکترها جهت ساخت بخش‌های کلمهٔ عبور بکار می‌رود.

یک استراتژی **کنترل غیرفعال کلمهٔ عبور** این است که خود سیستم هرچندوقت یکبار برنامهٔ شکستن کلمهٔ عبور داخلی خود را اجرا کرده تا کلمات عبور قابل حدس را پیدا کند. سیستم اگر بتواند یک کلمهٔ عبور را با حدس بدست آورد آن را حذف کرده و کاربر را از این موضوع مطلع می‌کند. این تاکتیک چندین نقطهٔ ضعف دارد. اول اینکه اگر قرار باشد تا عمل درست انجام شود، منابع سیستم را بشدت بکار خواهد گرفت. دوم اینکه یک دشمن مصمم که قادر به ربودن یک فایل کلمات عبور باشد می‌تواند ساعتها و یا حتی روزها تمام زمان CPU را برای دسترسی به هدف خود بکار گیرد. علاوه بر این هر کلمهٔ عبور موجود تا زمانی که کنترل غیرفعال کلمهٔ عبور، آن را کشف کند قابل تعرض خواهد ماند.

امیدوارکننده‌ترین روش برای امنیت کلمهٔ عبور، یک **کنترل‌کنندهٔ فعال کلمهٔ عبور** است. در این روش به یک کاربر اجازه داده می‌شود تا کلمهٔ عبور خود را انتخاب کند. ولی در هنگام انتخاب، سیستم به دنبال این می‌گردد که آیا این انتخاب مجاز است و در صورتی که این شرط برقرار نباشد آن را نمی‌پذیرد. چنین کنترل‌کننده‌هائی بر اساس این فلسفه قرار دارند که با هدایت کافی از طرف سیستم، کاربران می‌توانند کلمات عبور قابل بخاطرسپردن را از بین مجموعهٔ بزرگی از کلمات عبور که احتمال حدس زدن آنها در یک حملهٔ لغت‌نامه‌ای محتمل است، انتخاب کنند.

حیلهٔ کنترل‌کنندهٔ فعال کلمهٔ عبور این است که میخواهد توازنی بین خواست کاربر با استحکام کلمهٔ عبور ایجاد کند. اگر سیستم کلمات عبور زیادی را دفع کرده و قبول نکند، کاربران از این شکایت خواهند کرد که انتخاب کلمهٔ عبور کاری مشکل است. اگر سیستم از یک الگوریتم ساده برای تعریف آنچه قابل قبول است استفاده کند، این خود هدایت‌کنندهٔ شکنندگان رمز عبور برای بهبود بخشیدن به روش‌های گمانه‌زنی خود خواهد بود. در بقیهٔ این قسمت، نگاهی به برخوردهای مختلف در زمینهٔ کنترل فعال کلمهٔ عبور خواهیم انداخت.

اولین روش، یک سیستم ساده اعمال قانون است. بعنوان مثال قوانین زیر را می‌توان اعمال کرد:

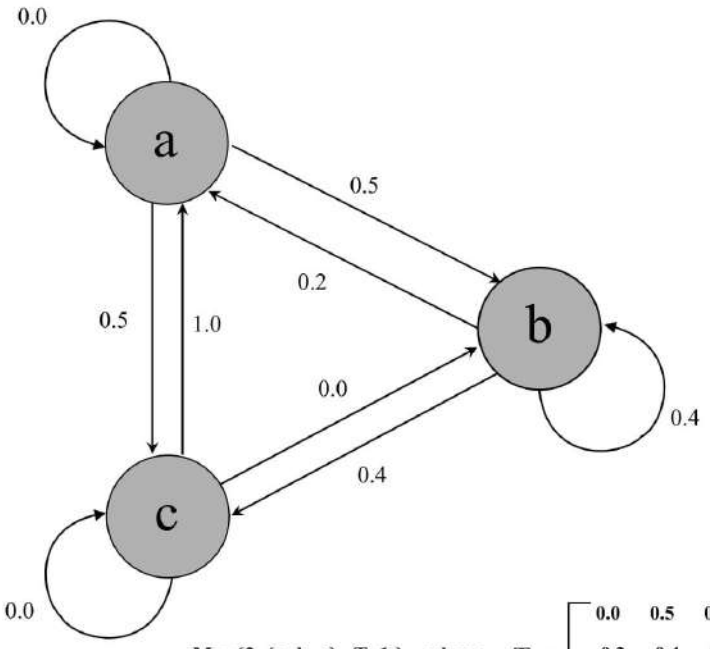
- تمام کلمات عبور بایستی حداقل ۸ کاراکتر طول داشته باشند.
- در ۸ کاراکتر اول کلمهٔ عبور بایستی حداقل یک حرف بزرگ، یک حرف کوچک، یک عدد و یک علامت وجود داشته باشد.

این قوانین را می‌توان بهمراه نصایح دیگر برای کاربر بیان نمود. اگرچه این نوع برخورد نسبت به تعلیم سادهٔ کاربران ارجحیت دارد، ولی برای ناامید کردن شکنندگان کلمهٔ عبور ممکن است کافی نباشد. این روش به قفل‌شکنان هشدار می‌دهد که کدام کلمات عبور را امتحان نکنند ولی بازهم ممکن است منجر به شکستن کلمهٔ عبور شود.

روش ممکن دیگر، جمع‌آوری یک لیست از کلمات عبور «بد» است. وقتی کاربری کلمهٔ عبوری را انتخاب می‌کند، سیستم آن را آزمایش کرده تا اطمینان یابد که در لیست غیرقابل قبول قرار نداشته باشد. این روش دارای دو مشکل است:

- **فضا**: برای مؤثر بودن روش، لیست بایستی خیلی بزرگ باشد. بعنوان مثال، لیستی که در پروژهٔ Purdue [SPAF92a] مورد استفاده قرار گرفت بیش از ۳۰ مگابایت حافظه را اشغال می‌کرد.
- **زمان**: زمان لازم برای جستجو در چنین لیست بزرگی، خود ممکن است خیلی زیاد باشد. علاوه بر آن برای کنترل کردن تمام تبدیلات ممکن روی لغات لیست، یا آن کلمات بایستی در لیست وجود داشته باشند، که حجم لیست را عظیم خواهد کرد، و یا برای هر کلمه پردازش دیگری نیز مورد نیاز خواهد بود.

دو تکنیک امیدوارکننده برای ساخت یک کنترل‌کنندهٔ فعال مؤثر و بهره‌ور، که بر مبنای نپذیرفتن لغات یک لیست قرار دارد، وجود دارد. یکی از این دو روش از یک مدل مارکوف (Markov) برای تولید کلمات عبور قابل حدس استفاده می‌کند [DAVI93]. شکل ۵-۹ یک نسخهٔ ساده شده از چنین مدلی را نشان می‌دهد. این مدل زبانی را نشان می‌دهد که الفبای آن دارای سه حرف است. حالت سیستم در هر لحظه برابر آخرین حرفی است که در سیستم پردازش شده است. مقدار نشان داده شده روی هر فلش این احتمال را نشان می‌دهد که حرفی به دنبال حرف دیگر قرار گیرد. بنابراین اگر مثلاً حرف فعلی a باشد، احتمال این که حرف بعدی b باشد 0.5 است.



مثالی از دنباله‌ای که مربوط به این زبان است : abbcacaba

مثالی از دنباله‌ای که مربوط به این زبان نیست : aacccbaaa

شکل ۵-۹ مثالی از مدل مارکوف

یک مدل مارکوف در حالت کلی دارای چهار عنصر [$m, A, \mathbf{T}, k$] است که درآن m تعداد حالات، A فضای حالات، $\mathbf{T}$ ماتریس احتمالات عبور از یک حالت به حالت دیگر و k مرتبهٔ مدل است. برای یک مدل مرتبهٔ k ام احتمال عبور به یک حرف بخصوص بستگی به نظم k حرف تولیدشدهٔ قبلی دارد. شکل ۵-۹ یک مدل سادهٔ مرتبهٔ اول را نشان می‌دهد.

دست اندرکاران از تولید و استفاده از یک مدل مرتبهٔ دوم نیز خبر می‌دهند. برای شروع، از تمام کلمات عبور قابل حدس یک لغت‌نامه ساخته می‌شود. آنگاه ماتریس عبور به طریق زیر ساخته می‌شود:

۱- ماتریس فرکانس $\mathbf{f}$ را بسازید، که در آن $\mathbf{f}(i,j,k)$ تعداد وقوع سه حرفی‌های شامل کاراکترهای i ام، j ام و k ام است. مثلاً کلمه عبور *parsnips* سه حرفی‌های ips , nip , sni , rsn , ars , par را تولید می‌کند.

۲- برای هر دوحرفی ij $\mathbf{f}(i,j,\infty)$ را بعنوان تعداد کل سه حرفی‌هائی که با ij شروع می‌شود حساب کنید. مثلاً $\mathbf{f}(a,b,\infty)$ شامل تعداد کل سه حرفی‌های بصورت aba، abb، abc و غیره خواهد بود.

۳- مؤلفه‌های ماتریس $\mathbf{T}$ را بصورت زیر محاسبه کنید:

$$T_{(i,j,k)} = \frac{f_{(i,j,k)}}{f_{(i,j,\infty)}}$$

نتیجهٔ امر مدلی است که ساختار کلمات لغت‌نامهٔ ساخته شده را نشان می‌دهد. با این مدل، سؤال «آیا این کلمهٔ عبور بد است؟» به سؤال «آیا این دنباله (کلمهٔ عبور) از این مدل مارکوف تولید می‌شود؟» تبدیل می‌گردد. برای یک کلمهٔ عبور داده شده، احتمالات تمام سه حرفی‌های آن را می‌توان جستجو کرد. آنگاه می‌توان نوعی تست آماری استاندارد برای تعیین اینکه آیا این کلمهٔ عبور بتوسط این مدل قابل تولید هست یا نیست را مورد استفاده قرار داد. کلمات عبوری که تولید آنها بتوسط این مدل محتمل است، پذیرفته نمی‌گردند. نویسندگان مقاله نتایج خوبی برای مدل مرتبهٔ دوم را گزارش داده‌اند. سیستم آنها تقریباً تمام کلمات موجود در لغت نامهٔ آنها را بدام انداخته و بسیاری از کلمات عبور مناسبی که برای کاربر راحت هستند را می‌پذیرد.

یک روش کاملاً متفاوت بتوسط Spafford [SPAF92a,SPAF92b] گزارش شده است. این روش بر اساس استفاده از یک فیلتر Bloom [BLOO70] قرار دارد. برای شروع، عمل فیلتر Bloom را توضیح می‌دهیم. یک فیلتر Bloom از مرتبهٔ k از یک مجموعهٔ k تائی از توابع hash مانند $H_1(x),H_2(x),...,H_k(x)$ تشکیل شده است که در آن هر تابع، یک کلمهٔ عبور را به اندازهٔ hash آن در محدودهٔ 0 تا N-1 تبدیل می‌کند. یعنی

$$H_i(X_j) = y \qquad 1 \le i \le k\,; \qquad 1 \le j \le D \qquad 0 \le y \le N\text{-}1$$

که در آن

X_j = کلمهٔ j ام در لغت نامهٔ کلمات عبور

D = تعداد کلمات در لغت نامهٔ کلمات عبور

آنگاه روش زیر به لغت‌نامه اعمال می‌شود:

۱- یک جدول hash از N بیت تعریف می‌شود که تمام بیت‌ها در ابتدا 0 اند.

۲- برای هر کلمهٔ عبور، اندازه‌های k مقدار hash آن محاسبه شده و بیت‌های نظیر آن در جدول hash به 1 تعویض می‌شوند. بنابراین اگر برای مقداری از i و j، $H_i(X_j) = 67$ باشد، آنگاه بیت شصت و هفتم جدول hash مساوی 1 می‌شود و اگر بیت قبلاً 1 بوده است 1 باقی می‌ماند.

وقتی کلمهٔ عبور جدیدی به کنترل‌کنندهٔ کلمات عبور عرضه می‌شود، اندازه‌های k مقدار hash آن محاسبه می‌گردد. اگر تمام بیت‌های نظیر جدول hash مساوی 1 باشند، آنگاه کلمهٔ عبور پذیرفته نخواهد شد. تمام کلمات عبوری که در لغت‌نامه وجود دارند رد می‌شوند. اما بازهم تعدادی جواب مثبت اشتباه خواهیم داشت (یعنی کلمات عبوری که در لغت‌نامه قرار ندارند ولی جدول hash آنها تطبیق دارد). برای روشن شدن مطلب، روشی با دو تابع hash را درنظر بگیرید. فرض کنید که کلمات عبور *undertaker* و *halkhogan* در لغت‌نامه هستند اما *xG%#jj98* در آن نیست. علاوه بر آن فرض کنید

$H_1(\text{undertaker}) = 25$ $H_1(\text{hulkhogan}) = 83$ $H_1(\text{xG\%\#jj98}) = 665$

$H_2(\text{undertaker}) = 998$ $H_2(\text{hulkhogan}) = 665$ $H_2(\text{xG\%\#jj98}) = 998$

اگر کلمهٔ عبور xG%#jj98 به سیستم عرضه شود، با وجود این که در لغت‌نامهٔ سیستم قرار ندارد رد خواهد شد. اگر تعداد زیادی از این جواب‌های مثبت غلط از سیستم بیرون داده شود، انتخاب کلمهٔ عبور برای کاربران سخت خواهد شد. بنابراین علاقه‌مندیم روشی را انتخاب کنیم که این جواب‌های مثبت غلط را به حداقل برساند. می‌توان نشان داد که احتمال یک جواب مثبت غلط برابر مقدار تقریبی زیر است

$$P \approx (1 - e^{kD/N})^k = (1 - e^{k/R})^k$$

و یا بصورت معادل آن

$$R \approx \frac{-k}{\ln(1 - P^{1/k})}$$

که در آن

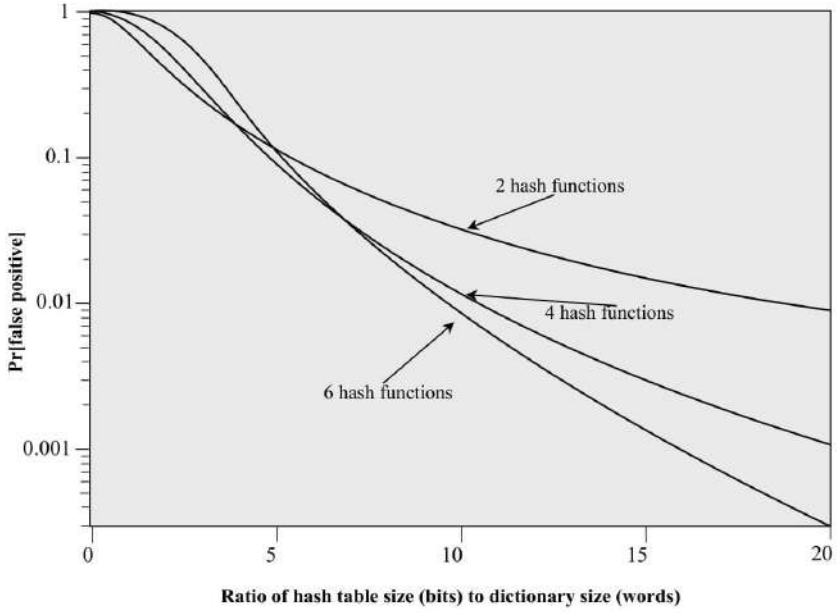
k = تعداد توابع hash

N = تعداد بیت‌های جدول hash

D = تعداد کلمات موجود در لغت‌نامه

$R = N/D$ نسبت اندازهٔ جدول hash (bits) به اندازهٔ لغت‌نامه (words)

شکل ۶-۹، P را بر حسب تابعی از R برای مقادیر مختلف k رسم کرده است. فرض کنید که یک لغت‌نامه یک میلیون لغت داشته و می‌خواهیم احتمال نپذیرفتن یک کلمهٔ عبور که در این لغت‌نامه قرار ندارد برابر ۰/۰۱ باشد. اگر شش تابع hash انتخاب کنیم، نسبت مورد نیاز $R = ۹/۶$ است. بنابراین یک جدول hash با $۹/۶ \times ۱۰^۶$ بیت و یا تقریباً ۱/۲ مگابایت حافظه مورد نیاز است. در مقایسه، ذخیره کردن تمام لغت‌نامه به حدود ۸ مگابایت حافظه نیازمند است. بنابراین فشردگی حاصل تقریباً با فاکتور ۷ خواهد بود. علاوه بر این، کنترل کردن کلمهٔ عبور شامل محاسبهٔ آسان شش تابع hash بوده و مستقل از حجم لغت‌نامه است در حالی که در صورت استفاده از کل لغت‌نامه، حجم جستجو بسیار وسیع‌تر خواهد شد.



شکل ۶-۹ عملکرد فیلتر Bloom

۹-۴ منابع مطالعاتی

دو منبع بررسی کامل تشخیص تهاجم [BACE00] و [PROC01] هستند. یک بررسی مختصرتر ولی کاملاً ارزشمند در [BACE01] ارائه شده است. دو مقالهٔ تحقیقی کوتاه ولی سودمند [KENT00] و [MCHU00] می‌باشند. [NING04] آخرین دستاوردهای تکنیک‌های تشخیص تهاجم را بررسی کرده است. [HONE01] توصیف محکمی از honeypots نموده و تحلیلی از ابزارها و روش‌های هکرها را ارائه می‌دهد.

BACE00 Bace, R. *Intrusion Detection*. Indianapolis, IN: Macmillan Technical Publishing, 2000.
BACE01 Bace, R., and Mell, P. *Intrusion Detection System*. NIST Special Publication SP 800-31, November 2000.
HONE01 The Honeynet Project. *Know Your Enemy: Revealing the Security Tools, Tactics, and Motives of the Blackhat Community*. Reading, MA: Addison-Wesely, 2001.
KENT00 Kent, S. "On the Trail of Intrusions into Information Systems." *IEEE Spectrum*, December 2000.
MCHU00 McHugh, J.; Christie, A.; and Allen, j."The Role of Intrusion Detection Systems." *IEEE Software*, September/October 2000.
NING04 Ning, P., et al. "Techniques and tools for analyzing Intrusion Alerts." *ACM Transactions on Information and system Security*, May 2004.
PROC01 Proctor, P. *The Practical Intrusion Detection Handbook*. Upper Saddle River, NJ: Prentice Hall, 2001.

وِب سایت‌های مفید

- **CERT Coordination Center**، سازمانی که از دل تیم پاسخگوئی به فوریت‌های کامپیوتری، بتوسط آژانس پروژه‌های پیشرفتهٔ تحقیقاتی وزارت دفاع آمریکا بیرون آمد. این سایت اطلاعات مفیدی در مورد تهدیدهای امنیتی اینترنت، آسیب‌پذیری‌ها و آمار حملات دارد.
- **Honeynet Project**، یک پروژهٔ تحقیقاتی که مطالعهٔ تکنیک‌های هَکرها، و ساخت محصولات honeypot را بعهده دارد.
- **Honeypots**، مجموعهٔ خوبی از مقالات تحقیقاتی و گزارشات تکنیکی.
- **Intrusion Detection Working Group**، شامل کلیهٔ اسنادی است که بتوسط گروه تولید شده است.

۹-۵ واژه‌های کلیدی، سؤالات مرورکنندهٔ بحث و مسائل

واژه‌های کلیدی

intrusion detection exchange format	فرمت مبادلهٔ تشخیص تهاجم
password	کلمهٔ عبور
rule-based intrusion detection	تشخیص مبتنی بر قاعدهٔ تهاجم
salt	
statistical anomaly detection	تشخیص آماری ناهنجاری

audit record	سوابق ممیزی- گزارشات ممیزی
Bayes' theorem	قضیهٔ Bayes
base-rate fallacy	خطای نرخ پایه
honeypot	طعمه
intruder	مهاجم
intrusion detection	تشخیص تهاجم

سؤالات مرورکننده بحث

۹-۱ سه دسته از مهاجمین را نام برده و عملکرد آنها را بطور مختصر تعریف کنید.

۹-۲ دو تکنیک معمول برای حفاظت از فایل کلمات عبور کدامند؟

۹-۳ سه نتیجهٔ مفید استفاده از یک سیستم تشخیص تهاجم کدامند؟

۹-۴ تفاوت بین تشخیص آماری ناهنجاری و تشخیص مبتنی بر قاعده تهاجم چیست؟

۹-۵ چه مقادیری در تشخیص مبتنی بر پروفایل تهاجم مفید می‌باشند ؟

۹-۶ فرق بین تشخیص مبتنی بر قاعدهٔ ناهنجاری و شناسائی مبتنی بر قاعدهٔ نفوذ چیست؟

۹-۷ یک honeypot چیست ؟

۹-۸ در مقولهٔ مدیریت کلمهٔ عبور در سیستم عامل UNIX، یک "salt" چیست؟

۹-۹ چهار تکنیک مورد استفاده جلوگیری از حدس زدن کلمهٔ عبور را نام برده و عملکرد آنها را بطور مختصر تعریف کنید.

مسائل

۹-۱ یک رانندهٔ تاکسی در یک تصادف منجر به فوت شبانه، فردی را مصدوم نموده و فرار می‌کند. دو شرکت تاکسیرانی سبز و آبی در شهر فعالیت دارند. به شما گفته می‌شود:

- ۸۵٪ تاکسی‌های شهر سبز و ۱۵٪ آنها آبی هستند.
- یک شاهد، رنگ تاکسی باعث تصادم را آبی گزارش کرده است.

دادگاه میزان اعتماد به شاهد در تحت شرایط شب حادثه را سنجیده و نتیجه گرفته است که شاهد احتمالاً ۸۰٪ در شناسائی رنگ تاکسی موفق بوده است. احتمال اینکه تاکسی مورد نظر آبی بوده و سبز نباشد، چقدر است؟

۹-۲ فرض کنید که کلمات عبور از یک ترکیب چهارتائی کاراکترها از یک الفبای دارای ۲۶ کاراکتر انتخاب می‌شوند. فرض کنید که یک مهاجم بتواند کلمات عبور با نرخ یک کلمه در ثانیه را امتحان کند.

الف- با فرض اینکه تا پایان هر تلاش مهاجم هیچ فیدبکی به او داده نشود، زمان مورد انتظار برای کشف کلمهٔ عبور صحیح چقدر است؟

ب- با فرض اینکه بمحض وارد نمودن یک کاراکتر اشتباه، مهاجم پیام خطا دریافت نماید، زمان مورد انتظار برای کشف کلمهٔ عبور صحیح چقدر است؟

۹-۳ فرض کنید که عناصر یک منبع k تائی بصورت یکنواخت روی عناصر یک هدف p تائی نگاشت می‌شوند. اگر هر رقم بتواند یکی از r مقدار را داشته باشد، آنگاه تعداد عناصر منبع r^k و تعداد عناصر هدف مقدار کمتر r^p خواهد بود. یک عنصر مشخص منبع مثل x_i به یک عنصر مشخص هدف مثل y_j نگاشت می‌شود.

الف- احتمال اینکه با یکبار تلاش، عنصر صحیح منبع بتوسط مهاجم انتخاب شود چقدر است؟

ب- احتمال اینکه یک عنصر متفاوت منبع x_k $(x_i \neq x_k)$ که به همان عنصر هدف y_j نگاشت می‌شود انتخاب شود چقدر است؟

ج- احتمال اینکه با یکبار تلاش، عنصر صحیح هدف بتوسط مهاجم انتخاب شود چقدر است؟

۹-۴ یک تولیدکنندهٔ کلمات عبور قابل تلفظ، برای هر کلمهٔ عبور شش حرفی بطور تصادفی دو بخش را انتخاب می‌کند. فرم هر بخش CVC (consonant,vowel,consonant) است که در آن $V = < a,e,i,o,u >$ و $C = \overline{V}$ است

الف- تعداد کلمات عبور ممکن چندتاست؟

ب- احتمال اینکه یک مهاجم یک کلمهٔ عبور را درست حدس بزند چقدر است؟

۹-۵ فرض کنید که کلمات عبور منحصر به استفاده از ۹۵ کاراکتر قابل چاپ کُد ASCII بوده و هر کلمهٔ عبور از ۱۰ کاراکتر تشکیل شده باشد. فرض کنید که یک شکنندهٔ کلمات عبور با نرخ رمزنگاری ۶/۴ میلیون رمزنگاری در هر ثانیه برای کشف کلمهٔ عبور مشغول بکار شود. روی یک سیستم UNIX چقدر طول خواهد کشید تا این رمزشکن بتواند همهٔ کلمات عبور ممکن را امتحان کند؟

۹-۶ نظر به ریسک‌های شناخته شده سیستم کلمات عبور UNIX، اسناد SunOS-4.0 پیشنهاد می‌کند که فایل کلمات عبور را برداشته و بجای آن یک فایل قابل خواندن بتوسط عموم که /etc/publickey نامیده می‌شود را جایگزین نمائیم. برای کاربر A، اطلاعات ورودی فایل شامل یک شناسهٔ کاربر ID_A کلید عمومی کاربر PU_a و کلید خصوصی نظیر آن PR_a است. کلید خصوصی با استفاده از DES و با کلیدی که از کلمهٔ عبور login کاربر (P_a) استخراج می‌شود رمز می‌شود. وقتی کاربر A به سیستم وصل می‌شود، سیستم $E [P_a , PR_a]$ را رمزگشائی کرده تا PR_a را بدست آورد.

الف- سیستم آنگاه تائید می‌کند که P_a بطور صحیح عرضه شده است. چگونه؟

ب- یک دشمن چگونه می‌تواند به این سیستم حمله کند؟

۹-۷ روش رمزنگاری کلمات عبور مورد استفاده در سیستم UNIX یک طرفه است و ممکن نیست که آن را بصورت معکوس بکار برد. در اینصورت آیا صحیح است که بگوئیم این روش در حقیقت یک کُد hash بوده و رمزنگاری کلمهٔ عبور نیست.

۹-۸ بیان کردیم که منظور کردن "salt" در روش کلمات عبور UNIX مشکل حدس زدن کلمهٔ عبور را بمیزان ۴,۰۹۶ برابر بالا می‌برد. اما "salt" بصورت متن ساده در همان جائی ذخیره می‌شود که کلمهٔ عبور رمزشده قرار می‌گیرد. بنابراین آن دو کاراکتر برای مهاجم معلوم بوده و نیازی به حدس زدن ندارد. در اینصورت چرا بیان می‌شود که "salt" امنیت را افزایش می‌دهد؟

۹-۹ با فرض اینکه شما مسالهٔ قبل را بطور صحیح پاسخ داده و اهمیت "salt" را درک کرده‌اید، به این سؤال پاسخ دهید. آیا این امکان وجود ندارد که با افزایش بسیار زیاد "salt" ، مثلاً به ۲۴ یا ۴۸ بیت بتوان همهٔ cracker ها را مأیوس کرد؟

۹-۱۰ فیلتر Bloom مورد بحث در بخش ۳-۹ را در نظر بگیرید. k را برابر تعداد توابع hash، N را برابر تعداد بیت‌های جدول hash و D را برابر تعداد کلمات لغت‌نامه فرض کنید.

الف- نشان دهید که تعداد بیت‌های 0 مورد انتظار در جدول hash از رابطهٔ زیر بدست می‌آید

$$\phi = (1 - k / N)^{D}$$

ب- نشان دهید که احتمال اینکه یک کلمهٔ ورودی که در لغت‌نامه وجود ندارد بطور اشتباه بعنوان یک کلمهٔ موجود در لغت‌نامه پذیرفته شود مساوی است با

$$P = (1 - \phi)^{k}$$

ج- نشان دهید که رابطهٔ قبل را می‌توان بصورت تقریبی $P \approx (1 - e^{-kD/N})^k$ نشان داد.

۹-۱۱ یک سیستم دست‌یابی به فایل طرح کنید که به کاربران معینی حق دست‌یابی خواندن و نوشتن به یک فایل را بر مبنای اجازه‌ای که بتوسط سیستم تعیین می‌گردد، بدهد. دستورات برنامه بایستی به فرمت زیر باشند:

تلاش کاربر A برای خواندن فایل F: READ (F, User A)
تلاش کاربر A برای ذخیره کردن فایل F که احتمالاً دستکاری هم شده است: WRITE (F, USER A)
هر فایل دارای یک header record است که شامل امتیازات اعتبارسنجی است. یعنی این سرآیند شامل لیستی از افرادی است که مجاز به خواندن و/ یا نوشتن هستند. این فایل قرار است با کلیدی رمزنگاری شود که در اشتراک کاربران نبوده و فقط برای سیستم شناخته شده است.

ضمیمهٔ ۹- الف خطای نرخ پایه (The Base-Rate Fallacy)

ابتدا نتایج مهمی از تئوری احتمالات را یادآوری نموده و سپس خطای نرخ پایهٔ را تشان می‌دهیم.

احتمال شرطی و پیشامدهای مستقل

اغلب لازم است تا احتمال پیشامدی را که مشروط به پیشامد دیگر است بدانیم. اثر این شرط این است که بعضی از وقایع از فضای نمونه حذف می‌شوند. مثلاً احتمال اینکه در انداختن دو طاس جمع خال‌ها ۸ باشد در صورتی که بدانیم یکی از خال‌ها حتماً زوج است چیست؟ می‌توان چنین استدلال کرد: چون یک طاس زوج است و جمع خال‌ها نیز زوج است بنابراین حتما طاس دوم هم بایستی زوج باشد. بنابراین، سه نتیجه موفق متساوی‌الاحتمال وجود دارد: (۶و۲) ، (۴و۴) و (۲و۶) که ازبین کل تعداد حالات ممکن ۲۷ = ۳×۳ - ۳۶ = (تعداد پیشامدهائی که هر دو تاس فرداند) - ۳۶ محاسبه می‌شوند. احتمال نتیجه شده برابر ۱:۹ - ۳:۲۷ است.

در حالت کلی، **احتمال شرطی** پیشامد A با فرض اینکه پیشامد B واقع شده باشد با Pr[A|B] نمایش داده شده و بصورت عبارت زیر تعریف می‌شود

$$\Pr[A \mid B] = \frac{\Pr[AB]}{\Pr[B]}$$

که در آن فرض می‌شود Pr[B] غیرصفر است.

در مثال ما، {جمع ۸ باشد} = A و {حداقل یکی از خالها زوج باشد} = B است. Pr[AB] تمام پیشامدهائی که در آنها جمع برابر ۸ و حداقل یکی از خالها زوج است را می‌پوشاند. همانطور که دیدیم سه پیشامد این‌چنینی وجود دارد. بنابراین Pr[AB]=3/36=1/12 . حال می‌توانیم Pr[A|B] را حساب کنیم:

Pr[A|B]=(1/12)/(3/4)=1/9

این نتیجه با استدلال قبلی همخوان است.

دو پیشامد A و B را **مستقل** ازهم گوئیم اگر Pr[AB]= Pr[A]Pr[B] باشد. بسولت می‌توان مشاهده نمود که اگر A و B مستقل باشند، Pr[A|B]=Pr[A] و Pr[B|A]=Pr[B] است.

قضیهٔ Bayes

یکی از مهم‌ترین نتایج تئوری احتمالات، قضیهٔ Bayes است. در ابتدا باید فرمول احتمال کلّ را بیان کنیم. هرگاه مجموعه‌ای از پیشامدهای دو به دو ناسازگار را داشته باشیم که اجتماع آنها همهٔ پیشامدهای ممکن را در بر بگیرد، و همچنین اگر یک پیشامد فرضی A را در نظر بگیریم، آنگاه میتوان نشان داد که

$$\Pr[\ A\ ] = \sum_{i=1}^{n} \Pr[\ A\ |\ E_i\ ]\Pr[\ E_i\ ] \qquad (۹-۱)$$

قضیهٔ Bayes را می‌توان بصورت زیر بیان کرد:

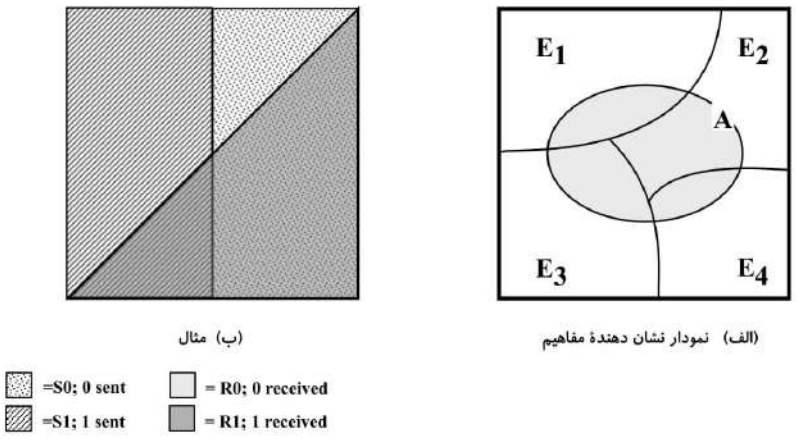
$$\Pr[E_i|A] = \frac{\Pr[A\,|\,E_i]\Pr[E_i]}{\Pr[A]} = \frac{\Pr[A\,|\,E_i]\Pr[E_i]}{\sum_{j=1}^{n}\Pr[A\,|\,E_j]\Pr[E_j]} \qquad (۹-۲)$$

شکل ۷-۹ الف مفهوم احتمال کلّ و قضیهٔ Bayes را نشان می‌دهد.

قضیهٔ Bayes برای محاسبهٔ «عواقب آینده»، یعنی احتمال وقوع پیشامدی با در دست داشتن شواهد مثبت در آن رابطه بکار می‌رود. برای مثال فرض کنید که دنباله‌ای از صفرها و یک‌ها از یک کانال نویزی منتقل می‌شوند. فرض کنید S0 و S1 به ترتیب پیشامدهای مرتبط با ارسال یک 0 و یک 1 در زمان معینی بوده و R0 و R1 نیز بترتیب پیشامدهای دریافت این 0 و 1 باشند. بازهم فرض کنید که ما احتمالات خروج این علائم از منبع را می‌دانیم و مثلاً $\Pr[S1] = P$ و $\Pr[S0] = 1-P$ است. حال خط را می‌پائیم تا ببینیم اگر 0 ارسال شود و یا 1 ارسال شود، هرچندوقت یکبار در دریافت آنها خطا خواهیم داشت و در این رابطه احتمالات $\Pr[R0\,|\,S1] = P_a$ و $\Pr[R1\,|\,S0] = P_b$ را حساب می‌کنیم. اگر یک 0 دریافت شود، ما می‌توانیم با استفاده از قضیهٔ Bayes احتمال شرطی یک خطا را محاسبه کنیم، یعنی احتمال شرطی این که یک 1 ارسال شده باشد در صورتی که یک 0 دریافت شده است:

$$\Pr[S1|R0] = \frac{\Pr[R0\,|\,S1]\Pr[S1]}{\Pr[R0|S1]\Pr[S1] + \Pr[R0|S0]\Pr[S0]} = \frac{PaP}{PaP + (1-Pa)(1-P)}$$

شکل ۷-۹ب معادلهٔ بالا را نشان می‌دهد. در شکل، فضای نمونه با یک مربع واحد نمایش داده شده است. نصف مربع متناظر با S0 و نصف دیگر آن متناظر با S1 بوده و بنابراین $\Pr[S0] = \Pr[S1] = 0.5$ است. بهمین ترتیب نصف مربع متناظر با R0 و نصف آن متناظر با R1 بوده و در نتیجه $\Pr[R0] = \Pr[R1] = 0.5$ است. در ناحیه‌ای که نمایشگر S0 است، 1/4 آن ناحیه متناظر با R1 بوده و بنابراین $\Pr[R1\,|\,S0] = 0.25$است. سایر احتمالات شرطی بهمین ترتیب روشن‌اند.



شکل ۷-۹ نمایش احتمال کلّ و قضیهٔ Bayes

نمایش خطای نرخ پایه

حالت زیر را در نظر بگیرید. از فرد بیماری برای تشخیص یک نوع مرض، آزمایشی گرفته شده است که نتیجهٔ آن مثبت است (یعنی این مرض را دارد). به شما گفته می‌شود:

- صحت آزمایش ۸۷٪ است (یعنی اگر بیمار این مرض را داشته باشد در ۸۷٪ موارد نتیجهٔ آزمایش مثبت است، و اگر بیمار این مرض را نداشته باشد بازهم در ۸۷٪ موارد نتیجهٔآزمایش صحیح است).
- احتمال بروز این مرض در یک جمعیت برابر ۱٪ است.

با فرض این که نتیجهٔ آزمایش مثبت است، احتمال این‌که بیمار این مرض را نداشته باشد چقدر است؟ بعبارت دیگر احتمال اینکه این یک اعلام خطر کاذب باشد چقدر است؟ برای یافتن جواب صحیح، نیاز به قضیهٔ Bayes داریم:

$$\Pr[\ well\ /\ positive\ ] = \frac{\Pr[\ positive\ /\ well\ ]\Pr[\ well\ ]}{\Pr[\ positive\ /\ disease\ ]\Pr[\ disease\ ] + [\Pr[\ positive\ /\ well\ ]\Pr[\ well\ ]}$$

$$= \frac{(0.13)(0.99)}{(0.87)(0.01) + (0.13)(0.99)} = 0.937$$

بنابراین در اکثریت وسیعی از موارد، وقتی شرایط مرض تشخیص داده می‌شود، این تشخیص یک تشخیص کاذب است.

این مسأله، در یک تحقیق [PIAT91]، به تعدادی افراد عرضه شد. بیشتر سوژه‌ها جوابشان ۱۳٪ بود. اکثریت بزرگی که شامل تعدادی از پزشکان نیز می‌شدند، عددی زیر ۵۰٪ را تخمین می‌زدند. خیلی از پزشکانی که حدسشان اشتباه بود با تأسف بیان می‌کردندکه «اگر شما صحیح می‌گوئید پس دلیلی برای آزمایشات کلینیکی وجود ندارد!» علت اینکه بیشتر افراد در پاسخ به این سؤال دچار اشتباه شده بودند این است که در محاسبهٔ ذهنی خود نرخ اصلی پیشامد(نرخ پایه) را منظور نکرده بودند. این خطا به نام *خطای نرخ پایه (base-rate fallacy)* مشهور است.

چگونه این معضل را میتوان حل کرد؟ فرض کنید بتوانیم هر دو نتیجه صحیح را به ۹۹/۹٪ برسانیم. یعنی فرض کنید بخواهیم که Pr[positive/disease] = 0.999 و Pr[negative/well] = 0.999 باشد. با قراردادن این دو عدد در فرمول (۲-۹)، Pr[well/positive] = 0.09 بدست می‌آید. بنابراین اگر بتوانیم تشخیص مریض بودن و یا مریض نبودن را با احتمال ۹۹/۹٪ درست پیش‌بینی نمائیم، آنگاه نرخ سیگنال کاذب تنها ۹٪ خواهد بود. این نتیجهٔ خیلی بهتری است ولی هنوز ایده‌آل نیست. بار دیگر دقت ۹۹/۹٪ را در نظر گرفته ولی فرض کنید که احتمال بروز مرض در یک جمعیت تنها ۰/۰۰۰۱ = ۱۰,۰۰۰ : ۱ باشد. در اینصورت نرخ آلارم کاذب ۹۱٪ خواهد شد. در حالات واقعی، [AXWL00] چنین نتیجه‌گیری کردکه احتمالات مرتبط با سیستم‌های تشخیص تهاجم آنچنان‌اند که نرخ آلارم کاذب رضایت‌بخش نمی‌باشد.

فصل ۱۰

نرم افزارهای بداندیش

ین فصل نرم‌افزارهای بداندیش(malware)، علی‌الخصوص ویروس‌ها و کِرم‌ها را مورد بررسی قرار می‌دهد.

۱۰-۱ ویروس‌ها و تهدیدهای مرتبط با آنها

شاید پیچیده‌ترین تهدیدها برای سیستم‌های کامپیوتری، بتوسط برنامه‌هائی صورت می‌پذیرد که از نقاط آسیب‌پذیر این سیستم‌ها سوءاستفاده می‌کنند. در این زمینه هم با برنامه‌های کاربردی و هم با برنامه‌های کمکی همانند ویرایش‌گرها (editors) و کامپایلرها (compilers) سروکار داریم.

این بخش را با مروری بر طیف این تهدیدهای نرم‌افزاری شروع می‌کنیم. بقیهٔ بخش به ویروس‌ها و کِرم‌ها اختصاص داشته و پس از نگاهی به ماهیت آنها، راه‌های مقابله با آنها را بررسی می‌کنیم.

برنامه‌های موذی

بعلت عدم وجود یک اجماع جهانی روی واژه‌های مربوط به این بخش و اختلاط بعضی از گروه‌ها با یکدیگر، تعریف اصطلاحات مربوط به این مقوله دشواری‌هائی را بوجود می‌آورد. جدول ۱۰-۱ که عمدتاً از [SZOR05] اقتباس شده است، راهنمای خوبی در این مورد است.

نرم‌افزارهای بداندیش را می‌توان به دو دسته تقسیم کرد: آنهائی که به یک برنامهٔ میزبان نیاز دارند و آنهائی که بطور مستقل عمل کرده و نیاز به محملی ندارند. دستهٔ اول ضرورتاً بخش‌هائی از یک برنامه بوده و نمی‌توانند مستقل از یک برنامهٔ کاربردی، برنامهٔ کمکی و یا برنامهٔ سیستمی وجود داشته باشند. ویروس‌ها، بمب‌های لاجیک و درب‌های مخفی مثال‌هائی از این دست‌اند. دستهٔ دوم برنامه‌های کاملی هستندکه می‌توانند مستقلاً بتوسط سیستم عامل کامپیوتر برنامه‌ریزی و اجرا شوند. کِرم‌ها و زامبی‌ها از این مقوله‌اند.

همچنین می‌توان بین آن دسته از تهدیدهای نرم‌افزاری که تکثیر نمی‌شوند در مقابل آنهائی که تکثیر می‌شوند، تفاوت قائل شد. اولی برنامه‌ها و یا تکه‌هائی از برنامه هستند که با یک چاشنی فعال می‌شوند. مثال‌های این مورد، بمب‌های لاجیک، درب‌های مخفی و زامبی‌ها هستند. دستهٔ بعد تکه‌ای از یک برنامه و یا برنامهٔ مستقلی است که وقتی اجرا شود ممکن است کپی‌های متعددی از خود را تولید نموده و در آینده بتوسط همان سیستم و یا سیستم‌های دیگر مرتبط با آن سیستم فعال شوند. ویروس‌ها و کِرم‌ها از این مقوله‌اند.

در بقیهٔ این بخش بطور مختصر به تشریح هریک از این نرم‌افزارهای بداندیش، بجز ویروس‌ها و کِرم‌ها که بطور مستقل تشریح خواهند شد، می‌پردازیم.

جدول ۱-۱۰ واژه‌های مربوط به نرم‌افزارهای بداندیش

نام	توصیف
Virus	خود را به یک برنامه متصل کرده و کپی‌هائی از خود را به برنامه‌های دیگر منتقل می‌کند
Worm	برنامه‌ای که کپی‌های خود را به کامپیوترهای دیگر منتقل می‌کند
Logic Bomb	وقتی فعال می‌شود که پیشامد خاصی روی دهد
Trojan Horse	برنامه‌ای که شامل قابلیت‌های اضافی غیرمنتظره است
Backdoor (trapdoor)	دستکاری یک برنامه بطوری که دست‌یابی غیرمجاز به عملیاتی را امکان‌پذیر نماید
Exploits	کُد مختص به یک آسیب‌پذیری منفرد و یا مجموعه‌ای از آسیب‌پذیری‌ها
Downloaders	برنامه‌ای که اقلام جدیدی را روی ماشین مورد تهاجم نصب می‌کند. یک downloader معمولاً با یک نامهٔ الکترونیک ارسال می‌شود
Auto-rooter	ابزارهای یک نفوذگر بداندیش که از آنها برای ورود به ماشین‌های جدید از راه دور استفاده می‌کند
Kit (virus generator)	مجموعه‌ای از ابزارها برای تولید ویروس‌های جدید بصورت خودکار
Spammer programs	برای ارسال حجم زیادی از هَرزنامه‌های الکترونیک بکار می‌رود
Flooders	برای حمله به شبکه‌های کامپیوتری از طریق ایجاد حجم بالائی از ترافیک بکار می‌رود تا یک حملهٔ انکار سرویس (DoS) را سازمان دهد
Keyloggers	حرکات صفحه کلید در یک کامپیوتر مورد حمله را می‌پاید
Rootkit	مجموعه‌ای از ابزارهای نفوذگری که پس از اینکه نفوذگر به سیستم راه یافت از آنها برای دسترسی به root-level استفاده می‌کند
Zombie	برنامه‌ای که روی یک ماشین آلوده شده فعال می‌شود تا حملات بر روی ماشین‌های دیگر را سامان دهد

درب مخفی (Backdoor)

یک درب مخفی، منفذی سرّی برای ورود به یک برنامه است که به فردی که از وجود این درب مطلع است اجازه می‌دهد که بدون عبور از موانع امنیتی طراحی شده به برنامه راه یابد. درب‌های مخفی برای سالیان متمادی بتوسط برنامه‌نویسان برای تدابیر تست و اشکال‌زدائی برنامه‌ها بطور قانونی مورد استفاده قرار گرفته‌اند. تعبیهٔ این درب‌ها معمولاً به این دلیل است که اجرای عادی برنامه نیاز به مراحل تأیید هویت و یا عبور از مراحل وقت‌گیر اجرائی و وارد نمودن مقادیر متعدد دارد. برای اشکال‌زدائی از برنامه، تولیدکنندهٔ برنامه ممکن است علاقه‌مند به کسب امتیازاتی برای ورود به برنامه و اجتناب از طی همهٔ مراحل اعتبارسنجی لازم باشد. همچنین برنامه نویس ممکن است بخواهد مطمئن شود که در صورت ایجاد اختلال در مراحل تأئید هویت، خود خواهد توانست از طریق دیگری وارد برنامه شده و آن را اصلاح نماید. درب مخفی در حقیقت کُدی برای شناخت دنبالهٔ مخصوصی از ورودی‌ها بوده و می‌تواند یک شمارهٔ کاربری خاص و یا دنبالهٔ غیرمحتملی از وقایع را شامل گردد.

درب‌های مخفی، وقتی‌که برای دسترسی غیرقانونی بتوسط برنامه‌نویس‌های غیرمسئول مورد استفاده قرار گیرند، تهدیدی جدی بحساب می‌آیند. درب مخفی، ایدهٔ اصلی نمایش آسیب‌پذیری در فیلم *War Games* بوده است. مثال دیگری از این تهدید آن بود که در جریان توسعهٔ Multics، تست‌های نفوذ در برنامه بتوسط «تیم ببر» نیروی هوائی امریکا انجام شد (نوعی شبیه‌سازی دشمن). یکی از تکنیک‌های بکار رفته این بود که یک برنامه جعلی بروزرسانی سیستم عامل، برای یکی از سایت‌هائی که از Multics استفاده می‌کرد ارسال شود. این برنامه حاوی یک اسب تروا (بعداً تشریح خواهد شد) بود که می‌توانست از طریق یکی از درب‌های مخفی وارد شده و امکان دسترسی به سیستم را در اختیار تیم عمل‌کننده قرار دهد. این تهدید آنقدر زیرکانه برنامه‌ریزی شده بود که طراحان Multics حتی بعد از اینکه از وجود آن مطلع شدند نتوانستند آن را پیدا نمایند[ENGE80].

ساخت کنترل‌هائی در سیستم عامل برای جلوگیری از سوءاستفاده از درب‌های مخفی کاری مشکل است. معیارهای امنیت بایستی بر توسعهٔ برنامه و فعالیت‌های مربوط بروزرسانی نرم‌افزار، نظارت دقیق داشته باشند.

بمب لاجیک (Logic Bomb)

یکی از قدیمی‌ترین انواع تهدیدهای نرم‌افزاری که قبل از ویروس‌ها و کرم‌ها وجود داشت و هم اکنون نیز بکار گرفته می‌شود، بمب لاجیک است. بمب لاجیک بصورت یک کُد در یک برنامه کاملاً قانونی تعبیه شده و طوری تنظیم می‌گردد که در صورت حصول شرایط خاصی «منفجر گردد». مثال‌هائی از شرایط خاص که می‌توانند بمب را منفجر سازند، حضور و یا عدم حضور فایل‌های مشخص، روز خاصی از هفته و یا تاریخ مشخصی از سال و یا حتی هنگام استفادهٔ فرد مشخصی از برنامه است. وقتی چاشنی بمب روشن شود، بمب ممکن است بخشی از دیتا و یا تمام فایل‌ها را تغییر داده و یا پاک کند، باعث توقف سیستم شده و یا صدمات دیگری را بجا گذارد. مثال خیره‌کننده‌ای از اینکه چگونه بمب‌های لاجیک می‌توانند عمل کنند مربوط به مورد Tim Lloyd است که با ارسال یک بمب لاجیک، کارفرمای خود یعنی شرکت Omega Engineering را با بیش از ده میلیون دلار خسارت مواجه کرد. عمل او استراتژی رشد سازمان را مختل نموده و نهایتاً به بیکار شدن ۸۰ کارگر انجامید[GAUD00]. Lloyd نهایتاً به ۴۱ ماه زندان و پرداخت دو میلیون دلار غرامت محکوم گردید.

اسب‌های تروا (Trojan Horses)

یک اسب تروا، یک برنامه یا مجموعه‌ای از فرمان‌های مفید و یا ظاهراً مفید است که شامل کُدهای پنهانی بوده و وقتی بکار گرفته شود عمل ناخواسته و یا مضری را انجام دهد.

برنامه‌های اسب تروا می‌توانند بمنظور انجام غیرمستقیم عملی بکار روند که کاربر غیرمجاز نمی‌تواند آن را بصورت مستقیم انجام دهد. بعنوان مثال، برای دست‌یابی به فایل‌های کاربر دیگری در یک سیستم اشتراکی، یک کاربر ممکن است یک برنامهٔ اسب تروا را طوری طراحی نماید که وقتی اجرا شود محدودیت‌های خواندن فایل‌های کاربر دیگر را از بین برده و فایل‌ها بتوسط همهٔ کاربران دیگر قابل دست‌یابی شوند. نویسندهٔ برنامه آنگاه می‌تواند با قراردادن برنامهٔ خود بعنوان یک برنامهٔ مفید در یک فهرست به اشتراک گذاشته شده و دادن نام اغواکننده‌ای به آن، کاربران دیگر را به اجرای برنامه ترغیب نماید. مثالی از این مورد برنامه‌ای است که بتواند بطور واضح لیستی از فایل‌های کاربر را با فرمت دلخواهی تهیه نماید. پس از اینکه کاربر دیگری این برنامه را اجرا نمود، آنگاه نویسندهٔ برنامه خواهد توانست به اطلاعات موجود در فایل او دست یابد. مثالی از یک برنامهٔ اسب تروا که تشخیص آن مشکل خواهد بود، برنامهٔ کامپایلری است که طوری دستکاری شده باشد که در هنگام کامپایل کردن برنامه‌های خاصی همانند برنامهٔ ورود به سیستم، یک کُد را وارد آن نماید[THOM84]. این کُد یک درب

مخفی در برنامهٔ ورود به سیستم(login) ایجاد کرده که به نویسندهٔ برنامه اجازه می‌دهد که با استفاده از کلمهٔ عبور بخصوصی وارد سیستم شود. با خواندن متن اصلی برنامهٔ ورود به سیستم، هرگز نمی‌توان به وجود این اسب تروا پی‌برد.

محرک بسیار معمول دیگر در استفاده از اسب تروا، تخریب داده‌هاست. ظاهراً بنظر خواهد رسید که برنامه کار مفیدی را انجام می‌دهد(مثلاً یک برنامهٔ ماشین حساب)، ولی برنامه در خفا فایل‌های کاربر را پاک می‌کند. بعنوان مثال، یکی از مدیران شبکهٔ CBS با اسب تروائی مورد حمله قرار گرفت که تمام حافظهٔ کامپیوترش را پاک نمود [TIME90]. اسب تروا در یک برنامهٔ گرافیکی که در یک BBS الکترونیک آگهی شده بود قرار داشت.

زامبی (Zombie)

یک زامبی برنامه‌ای است که بطور مخفیانه کامپیوتر دیگری را که به اینترنت وصل است در اختیار گرفته و از طریق آن کامپیوتر حملاتی را انجام دهد که حتی برای خود خلق‌کنندهٔ زامبی نیز دنبال کردن آن دشوار خواهد بود. از زامبی‌ها در حملات انکار سرویس و معمولاً علیه وِب سایت‌ها استفاده می‌شود. زامبی روی صدها کامپیوتر متعلق به افراد غیرمشکوک لانه کرده و آنگاه با ایجاد یک حملهٔ همه جانبه از ترافیک اینترنتی روی وِب سایت هدف، کار آن را مختل می‌سازد. بخش ۳-۱۰ زامبی‌ها را در حوزهٔ حملات انکار سرویس مورد بحث قرار می‌دهد.

ماهیت ویروس‌ها

یک ویروس، یک برنامه است که می‌تواند سایر برنامه‌ها را از طریق دستکاری آنها «آلوده» سازد. در این جرح و تعدیل کپی دیگری از ویروس ایجاد می‌شود که بعداً می‌تواند برنامه‌های دیگر را آلوده کند.

ویروس‌های بیولوژیک، تکه‌های کوچکی از کُدهای ژنتیک هستند- DNA یا RNA - که می‌توانند سازوکار یک سلول زنده را در اختیار گرفته و با دوز و کلک آن را وادار سازند تا هزاران نمونه معیوب از ویروس اولیه را تولید کند. همانند همزاد بیولوژیکی خود، یک ویروس کامپیوتری در کُد خود دستورالعمل ساخت نمونه‌های یکسانی از خود را حمل می‌کند. با استقرار در کامپیوتر میزبان، ویروس کنترل موقت سیستم عامل دیسک(DOS) را در اختیار می‌گیرد. سپس هرگاه این کامپیوتر آلوده با نرم‌افزار غیرآلوده‌ای تماس یابد، یک کپی جدید از ویروس به برنامهٔ جدید منتقل می‌گردد. در نتیجه آلودگی می‌تواند بتوسط کاربران غیرمشکوک که مشغول تبادل برنامه روی یک شبکه هستند از یک کامپیوتر به کامپیوتر دیگر گسترش یابد. در محیط یک شبکه، قابلیت دست‌یابی به کاربردها و سرویس‌های سیستم که روی کامپیوترهای مختلف قرار دارند، استعداد بالائی را برای گسترش یک ویروس به وجود می‌آورد.

یک ویروس می‌تواند هر کاری را که برنامه‌های دیگر انجام می‌دهند، انجام دهد. تنها فرق آن با برنامه‌های دیگر این است که خود را به برنامهٔ دیگری چسبانده و وقتی آن برنامه اجرا می‌شود ویروس نیز مخفیانه کار خود را انجام می‌دهد. وقتی ویروس فعال شود، می‌تواند هر عملی مثل پاک کردن فایل‌ها و برنامه‌ها را انجام دهد.

در طول زمان حیات خود، یک ویروس از چهار فاز مختلف عبور می‌کند:

- **فاز خفتن**: در این فاز ویروس خفته است ولی بالاخره بتوسط واقعه‌ای مانند رسیدن تاریخ مشخصی، حضور برنامه و یا فایل دیگری، و یا عبور ظرفیت دیسک سخت از حد معینی بیدار و فعال می‌شود. همه ویروس‌ها از این مرحله عبور نمی‌کنند.
- **فاز انتشار**: ویروس یک کپی کاملاً مشابه با خود را در برنامه‌های دیگر و یا بخش‌های معینی از دیسک بوجود می‌آورد. هر برنامه آلوده شده خود شامل ویروس مشابه‌ای بوده که می‌تواند وارد فاز انتشار شود.

- **فاز شروع به فعالیت:** ویروس برای انجام عملی که برای آن خلق شده است، فعال می‌شود. همانند فاز خفتن، این فاز نیز می‌تواند بتوسط وقایع متنوعی از قبیل شمارش تعداد دفعاتی که این ویروس شبیه خود را ایجاد کرده است، فعال شود.
- **فاز اجرا:** ویروس کار خود را انجام داده است. نتیجهٔ این عمل ممکن است بدون خطر، همانند ظاهرشدن یک پیام روی مونیتور، و یا خطرناک مانند آسیب زدن و یا تخریب برنامه‌ها و یا فایل‌ها باشد. مثال‌های دیگر از تخریب، اشغال فضای حافظه، ایجاد تناقضات نرم‌افزاری و سخت‌افزاری، و یا رفتار غیرنرمال سیستم است.

بیشتر ویروس‌ها کار خود را طوری انجام می‌دهندکه مختص سیستم عامل خاص و یا در بعضی موارد پایهٔ سخت‌افزاری مشخصی است. بنابراین آنها طوری طراحی می‌شوند که از جزئیات و نقاط ضعف سیستم‌های بخصوص استفاده کنند.

ساختار ویروس

یک ویروس می‌تواند به ابتدای یک برنامهٔ اجرائی و یا انتهای یک برنامهٔ اجرائی وصل گردد و یا ممکن است بطریق دیگری در درون برنامه جای داده شود. نکتهٔ اصلی در این مورد این است که وقتی برنامهٔ آلوده به ویروس بکار گرفته می‌شود، اول کُدهای مربوط به ویروس اجرا شده و سپس کُدهای برنامه اجرا می‌گردند.

یک فرم بسیار عمومی از ساختار ویروس در شکل ۱-۱۰ نشان داده شده است (بر مبنای [COHE94]). در این مورد کُد ویروس V در ابتدای برنامه اجرائی قرار گرفته است و فرض بر این است که نقطهٔ ورود به برنامه، خط اول آن است.

```
        program V :=

{goto main;
      1234567;

          subroutine infect-executable :=
              {loop:
              file := get-random-executable-file;
              if (first-line-of-file = 1234567)
                      then goto loop
                      else prepend V to file; }

          subroutine do-damage :=
              {whatever damage is to be done}

          subroutine trigger-pulled :=
              {return true if some condition holds}

main:       main-program :=
                  {infect-executable;
                  if trigger-pulled then do-damage;
                  goto next;}
next:

 }
```

شکل ۱-۱۰ یک ویروس ساده

یک برنامهٔ آلوده با کُد مربوط به ویروس شروع شده و چنین عمل می‌کند. اولین خط، کُد پرش به برنامهٔ اصلی ویروس است. خط دوم نشانگر خاصی است که بتوسط ویروس مورد استفاده قرار گرفته تا مشخص شود که آیا قربانی مورد توجه قبلاً با این ویروس آلوده شده است یا خیر. وقتی برنامه احضار می‌شود،کنترل بلافاصله به برنامهٔ ویروس منتقل می‌شود. برنامهٔ ویروس به دنبال فایل‌های اجرائی آلوده نشده می‌گردد و آنها را آلوده می‌سازد. سپس ویروس ممکن است عملی را انجام دهد که معمولاً برای سیستم زیان‌آور است. این عمل می‌تواند هربار که برنامه احضار می‌شود صورت پذیرفته و یا یک بمب لاجیک باشد که تنها تحت شرایط خاصی عمل کند. بالاخره ویروس کنترل را به برنامهٔ اولیه منتقل می‌سازد. اگر فاز آلوده‌سازی برنامه بطور معقولی سریع انجام پذیرد، احتمال اینکه کاربر متوجه شود که برنامه آلوده و یا غیرآلوده است کم خواهد بود.

ویروسی همانند آنچه تشریح گردید بسهولت تشخیص داده می‌شود زیرا یک نسخهٔ آلوده از یک برنامه، از یک نسخهٔ آلوده نشده طولانی‌تر است. یک روش برای اینکه از این سهولت تشخیص اجتناب شود این خواهد بود که فایل اجرائی را طوری فشرده نمود که نسخه‌های آلوده و غیرآلوده دارای طول مساوی باشند. شکل ۲-۱۰ [COHE94] منطق لازم را بطور کلی نشان می‌دهد. خطوط کلیدی برنامهٔ ویروس شماره‌گذاری شده و شکل ۳-۱۰ [COHE94] عملیات مربوط را نشان می‌دهد. فرض کنید که برنامهٔ P_1 با ویروس CV آلوده شده است. وقتی این برنامه بکار گرفته می‌شود، کنترل به ویروس منتقل شده و مراحل زیر طی می‌شود:

۱- برای هر فایل آلوده نشده P_2 که پیدا شود، ویروس ابتدا این فایل را فشرده کرده و P'_2 را تولید می‌کند که از برنامهٔ اولیه باندازهٔ برنامهٔ ویروس کوتاه‌تر است.
۲- یک کپی از ویروس اول به برنامهٔ فشرده شده اضافه می‌شود.
۳- نسخهٔ فشرده شدهٔ برنامهٔ آلوده شده اولیه P'_1 از فشردگی خارج می‌گردد.
۴- برنامهٔ اولیهٔ غیرفشرده اجرا می‌شود.

در این مثال، ویروس کاری بجز انتشار انجام نمی‌دهد. همانند مثال قبل، ویروس می‌تواند حاوی یک بمب لاجیک باشد.

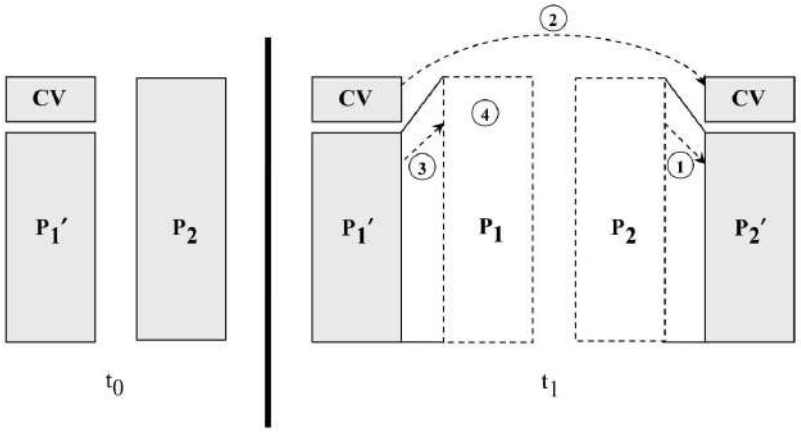
```
        program CV : =

{goto main;
      01234567;

      subroutine infect-executable :=
            {loop:
                  file := get-random-executable-file;
            if (first-line-of-file = 01234567) then goto loop;
      (1)   compress file;
      (2)   prepend CV to file;
            }

main: main-program :=
            {if ask-permission then infect-executable;
      (3)   uncompress rest-of-file;
      (4)   run uncompressed file;}
            }
```

شکل ۲-۱۰ منطق یک ویروس فشرده‌سازی شده



شکل ۳-۱۰ یک ویروس فشرده‌سازی شده

آلودگی اولیه

همینکه یک ویروس با آلوده کردن یک برنامه وارد سیستم شد، در موقعیتی است که می‌تواند بعضی و گاهی تمام فایل‌های دیگر آن سیستم را آلوده نماید. بنابراین برای جلوگیری از آلودگی‌های ویروسی باید از اول از ورود ویروس جلوگیری کرد. متاسفانه پیشگیری کاری بس دشوار است زیرا ویروس‌ها می‌توانند بخشی از هریک از برنامه‌های خارجی باشند. بنابراین مگر اینکه کسی از صفر شروع کرده و تمام برنامه‌های سیستمی و کاربردی را خود بنویسد والا همیشه خطر ویروس وجود دارد.

انواع ویروس ها

از زمانیکه ویروس‌ها برای اولین بار هویدا شدند، جنگ تسلیحاتی بین نویسندگان ویروس و نویسندگان نرم‌افزارهای ضد ویروس دائمی شد. تا برنامهٔ ضد ویروس مؤثری برای مقابله با ویروس‌های موجود آماده می‌گردید، ویروسی از نوع دیگر خلق می‌شد[STEP93]. موارد ذیل را بعنوان چشمگیرترین انواع ویروس معرفی می‌نماید:

- **ویروس انگلی(parasitic)**: این ویروس قدیمی‌ترین و هنوز معمول‌ترین نوع ویروس است. یک ویروس انگلی خود را به فایل‌های اجرائی چسبانده و وقتی آن فایل‌ها اجرا می‌گردند، فایل‌های اجرائی دیگر را پیدا کرده و آنها را آلوده می‌سازد.
- **ویروس ساکن درحافظه(memory-resident)**: بعنوان بخشی از یک برنامهٔ سیستم، در حافظهٔ اصلی کامپیوتر لانه می‌کند. ویروس از این نقطه هر برنامه‌ای را که اجرا می‌شود آلوده می‌سازد.

- **ویروس بخش راه‌اندازی(boot sector)**: یک رکورد و یا بخش اصلی راه‌اندازی را آلوده کرده و وقتی سیستم از روی دیسک مخصوص راه‌انداز که حاوی ویروس است بالا می‌آید، گسترش می‌یابد.
- **ویروس پنهان‌شونده(stealth)**: نوعی از ویروس است که مخصوصاً برای مخفی ماندن از دید نرم‌افزارهای ضد ویروس طراحی شده است.
- **ویروس چندچهره(polymorphic)**: ویروسی است که با هربار آلوده سازی فایل‌ها ظاهر خود را عوض کرده و بنابراین تشخیص آن با مقایسه با یک نمونهٔ قبل، غیرممکن است.
- **ویروس دگردیس(Metamorphic)**: این ویروس نیز همانند ویروس چندچهره در هر بار آلوده‌سازی ظاهر خود را عوض می‌کند. تفاوت در این است که این ویروس در هر بار آلوده‌سازی کاملاً خود را بازنویسی کرده و بنابراین کار تشخیص را دشوارتر می‌کند. ویروس‌های دگردیس ممکن است علاوه بر تغییر دادن چهرهٔ خود، رفتار خود را نیز عوض کنند.

قبلاً به نمونه‌ای از **ویروس پنهان‌شونده** اشاره گردید: ویروسی که از فشرده‌سازی استفاده کرده و حجم برنامه آلوده را دقیقاً به همان اندازهٔ حجم برنامهٔ اولیه نگاه می‌دارد. تکنیک‌های بسیار پیچیده‌تری نیز در مورد ساخت این ویروس ممکن است. بعنوان مثال، یک ویروس می‌تواند منطق حایل شدن در روال I/O دیسک را طوری تغییر دهد که وقتی کوششی برای خواندن بخش‌های مشکوک دیسک که شامل این روال‌هاست صورت می‌پذیرد، ویروس برنامهٔ اولیهٔ غیرآلوده را عرضه نماید.

یک **ویروس چندچهره** ویروسی است که در هنگام تکثیر کپی‌هائی را ایجاد می‌کند که از نظر عمل همانند نوع اولیهٔ آن بوده ولی از نظر ظاهر، الگوی بیت‌های آن تغییر کرده است. همانند یک ویروس پنهان شونده، هدف این ویروس علاوه بر تخریب، شکست دادن برنامه‌هائی است که به دنبال ویروس می‌گردند. در این مورد «امضاء» ویروس در هر کپی متفاوت خواهد بود. برای کسب چنین قابلیتی ویروس ممکن است بطور تصادفی دستوراتی را وارد برنامه کرده و یا نظم دستورالعملهای مستقل را بهم بزند. روش مؤثر دیگری که ممکن است پیش‌گرفته شود، استفاده از رمزنگاری است. قسمتی از ویروس که معمولاً *موتور تغییر (mutation engine)* نامیده می‌شود یک کلید رمز تصادفی را تولید کرده و بتوسط آن بقیهٔ ویروس به رمز در می‌آید. کلید بهمراه ویروس ذخیره شده، و خود موتور جستجو نیز تغییر می‌کند. وقتی یک برنامهٔ آلوده احضار می‌شود، ویروس از کلید پنهان شده استفاده کرده و ویروس را از رمز درمی‌آورد. وقتی ویروس تکثیر شد، یک کلید تصادفی دیگر انتخاب می‌شود.

سلاح دیگری که در زرادخانهٔ نویسندگان ویروس‌ها موجود است، جعبه ابزار تولید ویروس است. چنین جعبه ابزاری یک فرد نسبتاً تازه کار را قادر می‌سازد تا در مدت زمان کوتاهی، تعدادی ویروس خلق نماید. اگرچه ویروس‌هائی که به کمک این جعبه ابزارها خلق می‌شوند پیچیدگی کمتری نسبت به ویروس‌های دست‌ساز دارند، ولی با وجود این تعداد ویروس‌هائی که سریعاً بتوسط آن بوجود می‌آیند خود مشکلی برای روش‌های مبارزه با ویروس خواهد بود.

ویروس‌های ماکرو (Macro Viruses)

در اواسط دههٔ ۱۹۹۰ میلادی، ویروس‌های ماکرو فزاینده‌ترین نوع ویروس‌ها بودند. ویروس‌های ماکرو به دلایل متعددی تهدیدکننده‌های عمده بشمار می‌آیند:

۱- یک ویروس ماکرو مستقل از سیستم عامل کامپیوتر است. تقریباً تمام ویروس‌های ماکرو اسناد Microsoft Word را آلوده می‌سازند. هر پایه سخت‌افزاری و سیستم عاملی که Word را حمایت نماید می‌تواند آلوده شود.

۲- ویروس‌های ماکرو اسناد، و نه بخش‌های قابل اجرای کُد برنامه، را آلوده می‌سازند. بیشتر اطلاعات ورودی یک کامپیوتر را اسناد، و نه برنامه‌ها، تشکیل می‌دهند.

۳- ویروس‌های ماکرو به سهولت گسترش می‌یابند. یکی از روش‌های بسیار معمول از طریق پست الکترونیک است.

ویروس‌های ماکرو از خصیصه‌ای که در برنامهٔ Word و برنامه‌های دیگر Microsoft Office مانندExcel وجود داشته و ماکرو نامیده می‌شود استفاده می‌کنند. یک ماکرو اصالتاً یک برنامهٔ اجرائی است که در یک سند مربوط به Word یا فایل نوع دیگری جاسازی شده است. معمولاً کاربران ماکروها را برای خودکار کردن عملیات تکراری و بنابراین استفادهٔ کمتر از صفحه کلید بکار می‌گیرند. زبان ماکرو معمولاً نوعی از زبان برنامه‌نویسی Basic است. یک کاربر ممکن است دنباله‌ای از حرکات کلیدها را در یک ماکرو تعریف نموده و آن را طوری تنظیم نماید که وقتی یک کلید عملیاتی و یا ترکیب کوتاهی از چند کلید بکار گرفته می‌شوند، برنامهٔ ماکرو اجرا گردد..

نسخه‌های پی‌درپی Word که بعداً به بازار آمده‌اند، حفاظت بیشتری را در برابر ویروس‌های ماکرو فراهم نموده‌اند. بعنوان مثال مایکروسافت یک ابزار اختیاری Macro Virus Protection عرضه نموده است که فایل‌های مشکوک Word را تشخیص داده و به مشتری خطر جدی بازکردن یک فایل شامل ماکرو را گوشزد می‌کند. سازندگان محصولات متنوع ضد ویروس نیز ابزارهائی را برای تشخیص و تصحیح ویروس‌های ماکرو تولید کرده‌اند. همانند سایر انواع ویروس‌ها، مسابقهٔ تسلیحاتی در زمینهٔ ویروس‌های ماکرو نیز جریان دارد ولی آنها دیگر جزو تهدیدکننده‌های اصلی به شمار نمی‌آیند.

ویروس‌های پست الکترونیک (E-mail Viruses)

نوآوری جدید دیگر در مقولهٔ نرم‌افزارهای بداندیش، بتوسط ویروس پست الکترونیک حاصل گردید. اولین ویروس‌های پست الکترونیک که بسرعت تکثیر می‌شدند، همانند Melissa، از یک ماکروی Microsoft Word پنهانی در یک فایل پیوست به یک نامهٔ الکترونیک بهره می‌گرفتند. اگر دریافت‌کنندهٔ نامه، پیوست نامه را باز نماید ماکروی Word فعال می‌شود. آنگاه:

۱- ویروس پست الکترونیک، خود را برای کلیهٔ کسانی که نامشان در لیست آدرس کاربر بازکنندهٔ نامه موجود باشد می‌فرستد.

۲- ویروس تخریب محلی انجام می‌دهد.

در پایان سال ۱۹۹۹میلادی، نسخهٔ قدرتمندتری از ویروس پست الکترونیک ظاهرگردید. این نسخهٔ جدیدترمی‌توانست صرفاً با بازکردن یک نامهٔ الکترونیک حاوی ویروس فعال گردد و نیازی به بازکردن پیوست نامه نمی‌بود (مثل ویروس I Love You). این ویروس از دستورالعمل‌های زبان برنامه‌نویسی Visual Basic که تسهیلات پست الکترونیک را فراهم می‌سازند استفاده می‌کند.

بنابراین نسل جدیدی از بَدافزارها را مشاهده می‌کنیم که از طریق نامهٔ الکترونیک وارد شده و از خصوصیات نرم‌افزاری پست الکترونیک استفاده کرده تا خود را در سراسر اینترنت گسترش دهند. ویروس به محض فعال شدن به انتشار خود می‌پردازد (چه با بازکردن e-mail و چه با بازکردن پیوست e-mail) و خود را به تمام آدرس‌هائی که در صندوق پستی میزبان آلوده قرار دارد ارسال می‌کند. در نتیجه در حالی که قبلاً ماه‌ها و یا سال‌ها طول می‌کشید تا یک ویروس منتشر گردد، حال در ظرف مدت کوتاهی ویروس به همه جا راه می‌یابد. این موضوع کار نرم‌افزارهای ضدویروس را بسیار مشکل می‌کند تا بتوانند قبل از اینکه ویروس لطمات زیادی به محدودهٔ بزرگی وارد نماید، آن را کشف و نابود کنند. بالاخره، درجهٔ بالاتری از امنیت بایستی در نرم‌افزارهای کمکی و کاربردی اینترنتی روی کامپیوترهای شخصی تعبیه شود تا با این تهدید فزاینده مبارزه نمایند.

کِرم‌ها (Worms)

یک کِرم یک برنامه است که می‌تواند خود را تکثیر کرده و کپی‌های تکثیرشده را در عرض یک شبکهٔ کامپیوتری از رایانه‌ای به رایانهٔ دیگر منتقل نماید. پس از ورود به یک سیستم، کِرم ممکن است فعال شده، شروع به تکثیر نموده و مجدداً انتشار یابد. علاوه بر انتشار، کِرم معمولاً کارهای ناخواسته‌ای را نیز انجام می‌دهد. یک ویروس پست الکترونیک بعضی از مشخصات یک کِرم را داراست زیرا خود را از یک سیستم به سیستم دیگر منتقل می‌سازد. ولی با وجود این ما آن را یک ویروس می‌گوئیم زیرا برای انتقال آن واسطهٔ انسانی لازم است. یک کِرم بطور فعال به دنبال آلوده کردن سیستم‌های دیگر است و هر ماشینی که آلوده می‌شود خود بعنوان یک پایگاه خودکار برای حمله به ماشین‌های دیگر بکار می‌رود.

برنامه‌های مربوط به کِرم‌ها، از ارتباطات شبکه‌ای برای انتقال از یک سیستم به سیستم دیگر استفاده می‌کنند. همینکه یک کِرم شبکه در یک سیستم فعال گردید، می‌تواند بصورت یک ویروس یا باکتری عمل نموده، یا یک اسب تروا را در سیستم وارد کرده و یا به هر میزان عملیات تخریبی و یا قطع سرویس انجام دهد.

برای تکثیر خود، یک کِرم شبکه از برخی قابلیت‌های شبکه استفاده می‌کند. مثال‌هائی از این مورد چنین‌اند:

- **تسهیلات پست الکترونیک:** یک کِرم کپی خود را به سایر سیستم‌ها پست می‌کند.
- **قابلیت اجرا در دوردست:** یک کِرم یک کپی خود را در سیستم دیگر اجرا می‌کند.
- **قابلیت ورود به سیستم در دوردست:** یک کِرم بعنوان یک کاربر در یک سیستم دوردست وارد شده و سپس فرامینی را بکار می‌گیرد که خود را از یک سیستم به سیستم دیگر کپی نماید.

کپی جدید برنامهٔ کِرم آنگاه روی سیستم دوردست اجرا شده که علاوه بر عملیاتی که درآن سیستم انجام می‌دهد، بهمان ترتیب قبل خود را گسترش هم می‌دهد.

یک کِرم شبکه همان خصوصیات یک ویروس کامپیوتری را از خود نشان می‌دهد: یک فاز خفته، یک فاز انتشار، یک فاز شروع به فعالیت و یک فاز اجرا. فاز انتشار معمولاً عملیات زیر را انجام می‌دهد:

۱- با بررسی جداول کامپیوتر میزبان و یا سایر آدرس‌های موجود در فایل‌های سیستم دوردست، به دنبال سیستم‌هائی می‌گردد که آنها را آلوده سازد.

۲- یک اتصال با سیستم دوردست برقرار می‌کند.

۳- خود را در سیستم دوردست کپی کرده و ترتیبی اتخاذ می‌کند که کپی اجرا شود.

کِرم شبکه همچنین ممکن است قبل از آلوده‌کردن سیستم تلاش کند تا بفهمد که آیا سیستم قبلاً آلوده شده است یا خیر؟ در یک سیستم با برنامه‌های متعدد، کِرم ممکن است با پوشیدن لباس مبدّل خود را بجای یک پردازش و یا نام دیگری که برای اپراتور سیستم آشنا باشد جا بزند.

همانند ویروس‌ها، مبارزه با کِرم‌های شبکه کاری دشوار است.

کِرم موریس (Morris)

تا تولید نسل جدید کِرم‌ها، مشهورترین کِرم شناخته شده کِرمی بود که در سال ۱۹۸۸ میلادی بتوسط Robert Morris در اینترنت رها گردید. کِرم Morris برای گسترش در سیستم‌های UNIX طراحی شده بود و تکنیک‌های متفاوتی برای انتشار را بکار می‌بست. وقتی یک نسخهٔ آن اجرا می‌گردید، اولین وظیفهٔ آن شناسائی میزبان‌های مرتبط با میزبان

جاری بود که ورود به خود از طرف میزبان جاری را اجازه می‌دادند. کِرم این عمل را با آزمایش لیست‌ها و جداول متعددی که شامل جداول سیستم‌ها که مشخص می‌نماید کدام ماشین‌ها طرف اعتماد سیستم جاری بوده، فایل‌های پستی کاربر برای انتقال نامه‌های الکترونیک، جداولی که کاربران برای دسترسی به حساب‌های دوردست بکار می‌برند، و همچنین برنامه‌ای که اتصالات شبکه را نشان می‌دهد، شروع می‌کرد. برای هر میزبان جدید کشف شده، کِرم روش‌های متعددی برای کسب دست‌یابی را امتحان می‌نمود:

۱- سعی میکرد تا بعنوان یک کاربر قانونی به میزبان دور وارد شود. در این روش، کِرم اول تلاش می‌نمود تا فایل کلمه عبور محلی را شکسته (cracking) و سپس از ID و کلمهٔ عبور کشف شده استفاده کند. فرض براین بود که کاربران متعددی همین کلمهٔ عبور را در سیستم‌های مختلف بکار می‌برند. برای بدست آوردن کلمات عبور، کِرم یک برنامهٔ شکستن کلمهٔ عبور را اجرا می‌نمود که شامل مراحل زیر بود:
(الف) نام حساب شخص و همهٔ جایگشت‌های آن را امتحان می‌کرد.
(ب) یک لیست داخلی ۴۳۲- تائی از کلمات عبور که Morris آنها را محتمل می‌دانست آزمایش می‌شد.
(ج) تمام کلمات موجود در لغت‌نامهٔ محلی سیستم امتحان می‌شد.

۲- از یک اِشکال در پروتکل finger سوءاستفاده کرده تا محل یک کاربر دور را حدس بزند.

۳- از یک درب مخفی در گزینهٔ اشکال‌زدائی پردازش دور، که نامه‌ها را ارسال و دریافت می‌نماید سوءاستفاده می‌کرد.

اگر هریک از عملیات فوق به موفقیت می‌انجامید، کِرم ارتباط با مترجم فرامین سیستم عامل را حاصل می‌نمود. سپس کِرم به این مترجم یک برنامهٔ bootstap ارسال کرده، فرمانی برای اجرای این برنامه صادر کرده و از سیستم خارج می‌شد. برنامهٔ bootstrap در حین اجرا، برنامهٔ مادر را صدا زده و بقیه کِرم را پیاده می‌کرد. کِرم جدید سپس اجرا می‌گردید.

حملات جدید کِرم‌ها

دورهٔ مدرن تهدید کِرم‌ها با رها شدن کِرم Code Red در ماه ژوئیهٔ سال ۲۰۰۱ میلادی آغاز شد. Code Red از یک حفرهٔ امنیتی در IIS (Microsoft Internet Information Server) برای نفوذ و گسترش سوءاستفاده می‌کند. این کِرم همچنین کنترل‌کنندهٔ فایل‌های سیستمی در Windows را غیرفعال می‌کند. کِرم بصورت تصادفی از آدرس‌های IP استفاده کرده تا میزبان‌های دیگر را آلوده سازد. در خلال دورهٔ زمانی مشخصی، کِرم فقط انتشار می‌یابد. سپس با بمباران کردن یک وِب‌سایت با بسته‌های دیتا از میزبان‌های مختلف، یک حملهٔ انکار سرویس (Denial of Service) را آغاز می‌نماید. کِرم آنگاه فعالیت خود را به حال تعلیق درآورده ولی بطور تناوبی فعال می‌شود. در موج دوم حملات، Code Red تقریباً ۳۶۰,۰۰۰ سِرور را در ظرف ۲۴ ساعت آلوده نمود. علاوه بر فاجعه‌ای که برای سِرور هدف ایجاد می‌کند، Code Red می‌تواند مقادیر حجیمی از ظرفیت اینترنت را در اختیار گرفته و سرویس را مختل سازد.

Code Red II نوع دیگری از این کِرم است که Microsoft IIS هدف آن است. علاوه براین، کِرم جدید یک درب مخفی در کامپیوتر قربانی ایجاد کرده که به یک هَکر اجازه می‌دهد تا فعالیت‌های این کامپیوتر را هدایت نماید.

در اواخر سال ۲۰۰۱، کِرم دیگری با قابلیت‌های متنوع بنام Nimda هویدا گردید. Nimda برای گسترش از سازوکارهای متعددی استفاده می‌کند:

- از یک کلاینت به کلاینت دیگر، از طریق پست الکترونیک.
- از یک کلاینت به کلاینت دیگر، از طریق قابلیت‌های اشتراکی یک شبکهٔ باز.
- از یک سرور وب به کلاینت، از طریق مرورکردن سایت‌های مورد حمله قرارگرفته.
- از یک کلاینت به سرور وب، از طریق اسکن کردن فعال و سوءاستفاده از نقاط آسیب‌پذیر فهرست‌های Microsoft IIS 4.0/5.0.
- از یک کلاینت به سرور وب، ازطریق جستجو برای درب‌های مخفی بجامانده بتوسط کرم‌های "Code Red II".

این کرم، اسناد وب (مثل فایل‌های با پسوند .htm، .html و .asp) و فایل‌های اجرائی دیگر در سیستم آلوده شده را تغییر داده و نسخه‌های متعددی از خود تحت نام‌های متفاوت را خلق می‌کند.

در اوائل سال ۲۰۰۳، کرم SQL Slammer ظاهر گردید. این کرم از نقطهٔ ضعف سرریزشدن حافظهٔ موقت سرور Microsoft SQL سوءاستفاده می‌کرد. Slammer بطور فوق‌العاده‌ای متراکم بود و بسرعت گسترش می‌یافت بطوری که در ظرف ده دقیقه ۹۰٪ میزبان‌های آسیب‌پذیر را آلوده می‌کرد. پایان سال ۲۰۰۳ شاهد ظهور کرم Sobig.f بود که از سرورهای باز پروکسی سوءاستفاده کرده و آنها را به پایگاهی برای ارسال هَرزنامه تبدیل می‌کرد. در اوج فعالیت خود، Sobig.f برابر گزارش‌های اعلام شده، عامل ارسال یک پیام در هر ۱۷ پیام بود و در اولین ۲۴ ساعت حضور، یک میلیون کپی از خود برجای گذاشت.

Mydoom یک کرم پرحجم پست الکترونیک بود که در سال ۲۰۰۴ ظاهر گردید. این کرم از شگرد فزایندهٔ ایجاد یک درب مخفی در کامپیوترهای آلوده شده استفاده می‌کرد که به هَکرها اجازه می‌داد تا از این طریق به داده‌های مهمی همچون کلمات عبور و شمارهٔ کارت‌های اعتباری دست یابند. Mydoom تا هزاربار در هر دقیقه تکثیر می‌شد و برابر گزارشات، اینترنت را با ارسال ۱۰۰ میلیون پیام در عرض ۳۶ ساعت آلوده کرد.

وضعیت تکنولوژی کرم‌ها

وضعیت فعلی تکنولوژی کرم‌ها شامل موارد زیر است:

- **حمله به سیستم عامل‌های مختلف**: کرم‌های جدیدتر منحصر به رایانه‌هائی با سیستم عامل Windows نبوده بلکه می‌توانند به سیستم عامل‌های متعددی حمله نمایند که از آن جمله انواع UNIX می‌باشد.
- **استثمار چندگانه**: کرم‌های جدید به طرق مختلف به سیستم‌ها نفوذ کرده و از سرورهای وب، مرورگرها، پست الکترونیک، به اشتراک گذاشتن فایل‌ها و سایر کاربردهای مبتنی بر شبکه سوءاستفاده می‌کنند.
- **گسترش فوق سریع**: یکی از روش‌هائی که برای سرعت بخشیدن به گسترش کرم از آن استفاده می‌شود یک پیش اسکن اینترنت برای جمع‌آوری آدرس ماشین‌های آسیب‌پذیر است.
- **چندچهرگی**: برای فرار از کشف شدن، عبور از فیلترها و خنثی کردن تحلیل‌های برخط، کرم‌ها از تکنیک ویروس‌های چندچهره استفاده می‌کنند. هر کپی کرم یک کُد جدید ایجاد کرده که از نظر عملکرد دارای فرامین مشابه و تکنیک‌های رمزنگاری یکسان هستند.
- **دگردیسی**: علاوه بر عوض کردن ظاهر خود، کرم‌های دگردیس دارای یک فهرست از الگوهای رفتاری متفاوت‌اند که در مراحل مختلف انتشار از آنها استفاده می‌کنند.

- **وسیلهٔ حمل و نقل**: چون کِرم‌ها می‌توانند به سرعت سیستم‌های زیادی را آلوده نمایند، اغلب محمل خطرناکی برای حمل سایر ابزارهای حملات گسترده از قبیل زامبی‌ها در حملات گستردهٔ انکار سرویس هستند.
- **استثمار غافلگیرانه**: برای گسترش ماکزیمم و ایجاد حداکثر غافل‌گیری، یک کِرم باید از یک آسیب‌پذیری ناشناخته که تنها امکان کشف آن در یک محیط عمومی شبکه امکان‌پذیر است استفاده کند.

۱۰-۲ روش‌های مقابله با ویروس‌ها

بکارگیری آنتی‌ویروس‌ها

راه حل ایده‌آل در برابر تهدید ویروس‌ها، جلوگیری از ورود آنهاست: در وهلهٔ اول به ویروس اجازه ندهید که وارد سیستم شود. دست‌یابی به این هدف معمولاً غیرممکن است ولی پیش‌گیری، حملات موفق ویروس‌ها را کاهش خواهد داد. بهترین عمل بعدی انجام کارهای زیر است:

- **تشخیص**: وقتی ویروس سیستمی را آلوده کرد، از این اتفاق آگاه شوید و محل ویروس را کشف کنید.
- **شناسائی**: وقتی مشخص شد که ویروسی وجود دارد، نوع ویروس را تعیین کنید.
- **پاک‌سازی**: وقتی ویروس شناسائی گردید، همهٔ آثار ویروس را از برنامهٔ آلوده شده پاک کرده و برنامه را بحالت اولیه برگردانید. ویروس را از تمام سیستم‌های آلوده شده طوری برانید که آلودگی به جاهای دیگر سرایت نکند.

اگر حضور ویروس تشخیص داده شد ولی شناسائی و یا پاک‌سازی آن میسر نگردید، آنگاه چارهٔ امر آن است که برنامهٔ آلوده را نابود کرده و یک نسخهٔ جدید و پاک را جانشین آن نمائیم.

تکنولوژی ساخت ویروس‌ها و آنتی‌ویروس‌ها دست در دست هم بجلو می‌روند. ویروس‌های اولیه، کُدهای نسبتاً ساده‌ای بودند و می‌توانستند با بسته‌های نرم‌افزاری ضدویروس نسبتاً ساده شناسائی و دفع شوند. همینطور که جنگ تسلیحاتی ویروس‌ها تکامل یافت، هم ویروس‌ها و هم الزاماً آنتی ویروس‌ها تکامل‌یافته‌تر و پیچیده‌تر شدند.

[STEP93] چهار نسل از نرم‌افزارهای ضدویروس را شناسائی نموده است:

- نسل اول: اِسکنرهای ساده
- نسل دوم: اِسکنرهای کشف‌کننده
- نسل سوم: دام‌های شکارکنندهٔ فعالیت
- نسل چهارم: محافظت تمام عیار

یک اِسکنر **نسل اول** برای شناسائی یک ویروس، نیاز به امضاء آن ویروس دارد. ویروس ممکن است شامل "wildcard" بوده ولی الزاماً دارای همان ساختار و همان پترن بیت‌ها در همهٔ نسخ خود است. چنین اِسکنرهای مخصوص امضاء، فقط قابلیت تشخیص ویروس‌های شناخته شده را داشتند. نوع دیگری از اِسکنرهای نسل اول سابقه‌ای از اندازهٔ برنامه را نگاه داشته و به دنبال تغییر اندازه برنامه می‌گردند.

یک اِسکنر **نسل دوم** متکی بر امضاء خاصی نیست. در عوض اِسکنر از قوانین ذهنی کشف‌کننده استفاده کرده و بدنبال آلودگی‌های محتمل ویروس می‌گردد. یک نوع از این اِسکنرها به جستجوی کُدهائی می‌پردازد که معمولاً با ویروس‌ها مرتبط می‌باشند. بعنوان مثال، یک اِسکنر ممکن است به جستجوی ابتدای یک حلقهٔ رمزنگاری که در ویروس چندچهره مورد استفاده

قرار می‌گیرد پرداخته و کلید رمز را پیدا کند. همینکه کلید کشف شد، اسکنر برای شناسائی ویروس آن را رمزگشائی نموده و سپس آلودگی را برطرف کرده و برنامه را اصلاح می‌کند.

روش دیگری در نسل دوم آنتی‌ویروس‌ها، کنترل صحت برنامه است. یک جمع کنترلی می‌تواند به هر برنامه الصاق شود. اگر ویروسی برنامه را بدون تغییردادن جمع کنترلی آلوده نماید، آنگاه یک آزمایش کنترل صحت، تغییر را آشکار می‌سازد. برای مقابله با ویروسی که آنقدر پیچیده باشد که بتواند در هنگام آلوده‌سازی برنامه، جمع کنترلی آن را تغییر دهد می‌توان از یک تابع درهم‌ساز رمزشده استفاده کرد. کلید رمزنگاری در جائی جدا از برنامه ذخیره شده تا ویروس نتواند یک کُد جدید hash تولید کرده و آن را به رمز درآورد. با استفاده از یک تابع درهم‌ساز بجای یک جمع کنترلی ساده، ویروس از جرح و تعدیل برنامه بنحوی که بتواند همان کُد hash سابق را تولید کند، عاجز می‌ماند.

نسل سوم آنتی‌ویروس‌ها، برنامه‌های ساکن در حافظه هستند که ویروس را با فعالیت آن، و نه با ساختار آن، در یک برنامهٔ آلوده شده شناسائی می‌کنند. حسن چنین برنامه‌هائی این است که لازم نیست تا امضاءها و قواعد ذهنی برای ردیف وسیعی از ویروس‌ها را تولید کرد، بلکه تنها کافی است که مجموعهٔ کوچکی از فعالیت‌ها که نمایشگر تلاش برای آلوده‌سازی سیستم است را شناسائی نموده و سپس برای پاک‌سازی مداخله کرد.

محصولات **نسل چهارم**، بسته‌های نرم‌افزاری شامل مجموعه‌ای از آنتی‌ویروس‌ها بوده که به همراه هم مورد استفاده قرار می‌گیرند. اینها شامل اسکنرها و دام‌های شکارکننده فعالیت می‌باشند. بعلاوه چنین بسته‌ای شامل قابلیت‌های کنترل دست‌یابی هستند که توان ویروس‌ها در ورود به یک سیستم را کم کرده و آنگاه قابلیت یک ویروس برای بروزرسانی فایل‌ها برای گسترش آلودگی را محدود می‌سازند.

جنگ تسلیحاتی ادامه دارد. با بسته‌های نرم‌افزاری آنتی‌ویروس‌های نسل چهارم، استراتژی دفاعی سازمان یافته‌تری بکار گرفته می‌شود و حوزهٔ دفاع به معیارهای عام‌تری از امنیت کامپیوتر اعمال می‌گردد.

تکنیک‌های پیشرفتهٔ آنتی‌ویروس‌ها

در زمینهٔ آنتی‌ویروس‌ها، مرتباً روش‌های پیچیده‌تر و محصولات پیشرفته‌تری پدیدار می‌گردند. در اینجا به دو نمونه از مهم‌ترین آنها اشاره می‌کنیم.

رمزگشائی ژنریک

تکنولوژی رمزگشائی ژنریک GD (Generic Decryption)، برنامه‌های آنتی‌ویروس را قادر می‌سازد که حتی پیچیده‌ترین ویروس‌های چندچهره را تشخیص داده و در عین حال سرعت اسکن نمودن برای یافتن ویروس‌ها را بالا نگاه دارند[NOCH97]. بیاد آوریدکه وقتی فایلی که حاوی یک ویروس چندچهره است اجرا می‌شود، ویروس بایستی خود را رمزگشائی کرده و فعال شود. برای تشخیص چنین ساختاری، فایل‌های اجرائی از یک اسکنر GD که شامل عناصر زیر است استفاده می‌کنند:

- **مقلّد پردازشگر مرکزی (CPU Emulator)**: یک کامپیوتر مجازی مبتنی بر نرم‌افزار است. فرامین مربوط به یک فایل اجرائی بجای اجرا روی پردازشگر اصلی بتوسط این مقلّد ترجمه می‌گردند. مقلّد شامل نسخه‌های نرم‌افزاری همهٔ رجیسترها و سایر بخش‌های سخت‌افزاری پردازشگر می‌باشد بطوری که پردازشگر اصلی تحت تاثیر برنامه‌هائی که بتوسط مقلّد مورد تعبیر قرار می‌گیرند واقع نمی‌شود.

- **اسکنر امضاء ویروس (Virus signature scanner)**: مدولی که کُد برنامهٔ هدف، بمنظور یافتن امضاء ویروس‌های شناخته شده، را اسکن می‌کند.
- **مدول کنترل مقلّد (Emulation control module)**: اجرای کُد برنامهٔ هدف را کنترل می‌کند.

در ابتدای هر شبیه‌سازی، مقلّد شروع به ترجمهٔ خط به خط فرامین برنامهٔ هدف می‌کند. در نتیجه اگر کُد برنامه شامل یک بخش رمزگشائی که ویروس را رمزگشائی و آشکار می‌کند باشد، این کُد ترجمه می‌گردد. در واقع ویروس کار برنامهٔ آنتی‌ویروس را با ظاهر کردن ویروس انجام می‌دهد. هرچند وقت یکبار نیز، مدول کنترل عمل ترجمه را متوقف نموده و برنام را بمنظور یافتن امضاء ویروس‌ها اسکن می‌نماید.

در هنگام ترجمه، کُد برنامهٔ هدف نمی‌تواند هیچ آسیبی به محیط حقیقی کامپیوتر شخصی وارد کند زیرا برنامه در یک محیط کاملاً کنترل شده ترجمه می‌شود.

مشکل‌ترین مقولهٔ طراحی در استفاده از یک اسکنر GD، تعیین زمان لازم برای ترجمه است. معمولاً عناصر یک ویروس در فاصلهٔ کوتاهی پس از شروع اجرای یک برنامه فعال می‌شوند، ولی الزاماً اینطور نیست. هرچقدر زمان تقلید یک برنامهٔ بخصوص بتوسط اسکنر زیادتر باشد، احتمال شکار یک ویروس مخفی زیادتر است. ولی برنامهٔ آنتی‌ویروس نمی‌تواند زمان و منابع زیادی را برای مدت طولانی در اختیار گیرد زیرا در اینصورت کاربر از تأخیر طولانی شاکی خواهد شد.

سیستم مصون دیجیتال (Digital Immune System)

سیستم مصون دیجیتال، یک سیستم حفاظتی مفصّل ضد ویروس است که بتوسط IBM طراحی شده است [KEPH97a, KEPH97b]. محرک تولید این سیستم، تهدید فزایندهٔ انتشار ویروس‌های اینترنتی بوده است. ابتدا مختصری راجع به این نوع تهدید صحبت کرده و آنگاه روش IBM را تشریح می‌کنیم.

در قدیم، تهدید ویروس‌ها دارای این مشخصهٔ بود که گسترش ویروس‌های جدید و تغییر شکل آنها کند بود. نرم‌افزارهای آنتی‌ویروس معمولاً بصورت ماهیانه به روز درآمده و این امر برای کنترل مساله کافی بود. همچنین در فرم سنّتی، اینترنت نقش نسبتاً کوچکی در گسترش ویروس‌ها را ایفاد می‌نمود. اما همانطور که [CHES97] خاطرنشان می‌سازد، دو روند عمده در تکنولوژی اینترنت اثر فزاینده‌ای در سرعت گسترش ویروس در سال‌های اخیر داشته است:

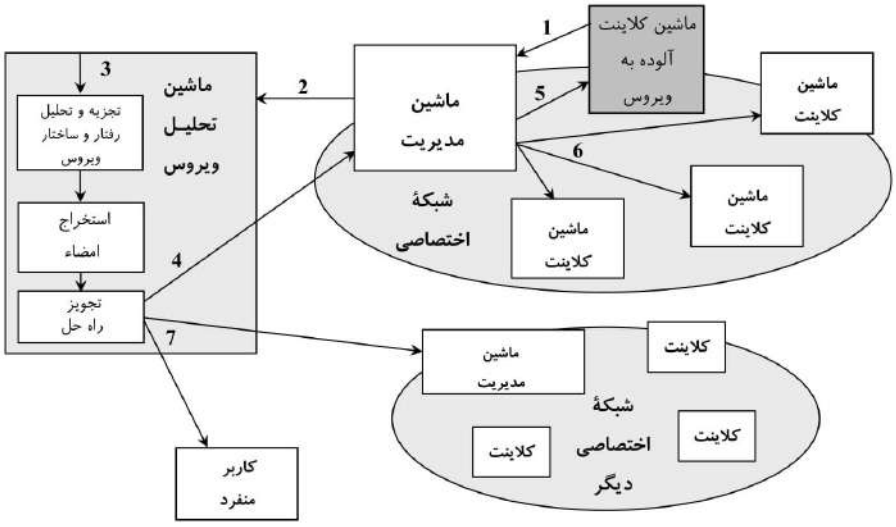
- **سیستم‌های یکپارچهٔ پستی**: سیستم‌هائی همانند Lotus Notes و Microsoft Outlook امر ارسال هر چیزی به هرکسی، و کار با اقلام دریافت شده را بسیار آسان کرده است.
- **سیستم‌های برنامه‌های سیار**: قابلیت‌هائی همچون Java و Active X به برنامه‌ها اجازه می‌دهد تا بخودی‌خود از یک سیستم به سیستم دیگر عبور نمایند.

در پاسخ به تهدیدهائی که بتوسط قابلیت‌های جدید اینترنت حاصل شده‌اند، IBM یک سیستم مصون دیجیتال مربوط به خود را تولید کرده است. این سیستم، استفاده از مقلّد برنامه که در قسمت قبل از آن یاد شد را توسعه داده و یک سیستم مقلّد و تشخیص ویروس را ساخته است. هدف این سیستم، عکس‌العمل سریع و ایزوله کردن ویروس‌ها بمحض ورود به سیستم است. همینکه یک ویروس جدید وارد سازمانی شود، سیستم مصون بطور خودکار آن را محاصره کرده، تجزیه و تحلیل نموده، تشخیص داده، محاصره کرده، آن را از بین برده و اطلاعات مربوط به ویروس را به سیستم‌هائی که از آنتی‌ویروس IBM استفاده می‌کنند ارسال کرده تا ویروس قبل از اینکه بتواند در جای دیگری اجرا شود، کشف و خنثی گردد.

شکل ۴-۱۰ مراحل اصلی عملیات در سیستم مصون دیجیتال را نشان می‌دهد:

۱- یک برنامهٔ پایشگر روی هر PC از یک سری ذهنیات در رابطه با رفتار سیستم، تغییرات مشکوک برنامه‌ها، و یا امضاء محلی در مورد اینکه ممکن است ویروسی وجود داشته باشد استفاده می‌کند. برنامهٔ پایشگر یک نسخه از هر برنامه‌ای را که فکر می‌کند آلوده است به یک ماشین مدیریت در سازمان ارسال می‌کند.
۲- ماشین مدیریت، نسخهٔ برنامه را بصورت رمز درآورده و آن را برای یک ماشین مرکزی تجزیه و تحلیل ویروس ارسال می‌دارد.
۳- این ماشین محیطی را ایجاد می‌کند که در آن برنامهٔ آلوده می‌تواند با امنیت کامل اجرا شود. روش‌های اجرائی در این مورد شامل تقلید و یا خلق محیط حفاظت شده‌ای است که در آن محیط، برنامهٔ مشکوک بتواند اجرا و پایش گردد. ماشین تجزیه و تحلیل ویروس آنگاه داروئی برای شناسائی و حذف ویروس تجویز می‌کند.
۴- داروی تجویز شده به ماشین مدیریت برگشت داده می‌شود.
۵- ماشین مدیریت، داروی تجویز شده را به کلاینت می‌دهد.
۶- داروی تجویز شده همچنین برای سایر کلاینت‌های سازمان ارسال می‌گردد.
۷- مشترکین سراسر دنیا بطور منظم آنتی‌ویروس‌های جدید را دریافت می‌کنند تا آنها را از ویروس‌های جدید حفظ کنند.

موفقیت سیستم مصون دیجیتال وابسته به توانائی ماشین تجزیه و تحلیل ویروس در تشخیص ویروس‌های جدید و ابتکاری می‌باشد. با تحلیل مداوم و پائیدن ویروس‌هائی که در محیط پیدا می‌شوند بایستی بتوان نرم‌افزار مصون دیجیتال را بصورت دائم برای مبارزه با تهدیدها به روز نگاه داشت.



شکل ۴-۱۰ سیستم مصون دیجیتال

نرم افزار سدکنندهٔ رفتار (Behavior-Blocking Software)

بر خلاف اسکنرهائی که به تاریخچهٔ ویروس‌ها و یا بعبارتی به جستجوی اثر انگشت ویروس‌ها می‌پردازند، نرم‌افزار سدکنندهٔ رفتار با سیستم عامل یک کامپیوتر میزبان جفت شده و رفتار برنامه را بمنظور کشف رفتارهای بداندیشانه بصورت بلادرنگ می‌پاید. نرم‌افزار سدکنندهٔ رفتار آنگاه جلوی رفتارهای موذیانه را قبل از اینکه بتوانند روی سیستم تأثیر منفی گذارند، سد می‌کند. رفتارهای پایش شده می‌تواند شامل موارد زیر باشد:

- تلاش برای بازکردن، مشاهده، حذف و یا تعویض فایل‌ها.
- تلاش برای فرمت کردن درایوها و سایر عملیات غیرقابل برگشت.
- ایجاد تغییرات در منطق فایل‌های اجرائی برنامه‌های ماکرو.
- تغییرات در تنظیمات حیاتی سیستم مانند تغییر در تنظیمات بالاآمدن کامپیوتر.
- تغییرات در نامهٔ الکترونیک و پیام‌های فوری کلاینت.
- تحریک به شروع ارتباطات شبکه‌ای.

اگر سدکنندهٔ رفتار تشخیص دهد که اجرای یک برنامه باعث بروز رفتارهای موذیانه خواهد شد، می‌تواند این رفتارها را در هنگام اجرا مسدود کرده و یا نرم‌افزار بداندیش را خاتمه دهد. این عمل رجحان عمده‌ای نسبت به روش‌های تشخیص آنتی‌ویروس‌های مبتنی بر اثر انگشت و یا سوابق ویروس‌ها دارد. در حالی که عملاً میلیاردها روش مختلف برای سردرگم‌کردن و تغییر دستورات در یک ویروس یا کِرم وجود دارد، که بسیاری از آنها نمی‌توانند از طرف یک اسکنر عمل‌کننده بر اساس سوابق مورد تشخیص واقع شوند، ولی بالاخره یک برنامهٔ بداندیش باید موضوع کاملاً تعریف‌شده‌ای را از سیستم عامل درخواست کند. با فرض اینکه سدکنندهٔ رفتار بتواند چنین درخواستی را تشخیص دهد، خواهد توانست عملیات بداندیشانه را صرف نظر از اینکه منطق برنامه چقدر پیچیده و سردرگم‌کننده باشد، تشخیص و جلوی آن را سد کند.

روشن است که توانائی پایش نرم‌افزار در حال اجرا در زمان حال، مزیت بزرگی را برای سدکنندهٔ رفتار به ارمغان می‌آورد ولی این امر دارای نقاط ضعفی نیز هست. چون برنامهٔ بداندیش قبل از اینکه همهٔ رفتارهای آن مشاهده شود بایستی روی سیستم هدف اجرا گردد، ممکن است قبل از اینکه به توسط سیستم سدکنندهٔ رفتار شناسائی و مسدود گردد، صدمات زیادی را به سیستم وارد نماید. بعنوان مثال، یک ویروس جدید ممکن است تعدادی از فایل‌های ظاهراً غیرمهم را در دیسک سخت جابجا نموده و سپس به یک فایل تنها حمله کرده و آن را آلوده سازد که همین آلودگی اخیر باعث تشخیص و سدشدن راه آن گردد. اگرچه آلودگی اصلی مسدود شده است ولی کاربر ممکن است در پیداکردن فایل‌های خود دچار مشکل شود که خود باعث از دست رفتن کارآئی و یا بدتر از آن خواهد بود.

۳-۱۰ حملات توزیع شدهٔ انکار سرویس

حملات توزیع‌شدهٔ انکار سرویس DDoS (Distributed Denial of Service) یک تهدید مهم امنیتی برای سازمان‌ها محسوب گردیده و بنظر می‌رسد که اینگونه حملات در حال افزایش‌اند[VIJA02]. در یک بررسی سه هفته‌ای در سال ۲۰۰۱، محققین بیش از ۱۲,۰۰۰ حمله به بیش از ۵,۰۰۰ هدف مشخص را شناسائی کردند. هدف‌ها انواع متفاوتی داشته و کمپانی‌های شناخته شدهٔ بزرگی مانند Amazon و Hotmail تا ISPهای کوچک خارجی و اتصالات تلفنی را می‌پوشاندند

[MOOR01]. حملات DDoS با بمباران کردن سرورها، شبکه‌ها و یا حتی کاربران انتهائی با ترافیک بی‌خاصیت باعث می‌شوند که کاربران قانونی نتوانند به آن منابع دسترسی یابند. در یک حملهٔ DDos معمول، تعداد زیادی از میزبان‌های مورد تهاجم قرارگرفته گردهم می‌آیند تا بسته‌های دیتای بی‌حاصل را ارسال نمایند. در سال‌های اخیر، روش‌های حمله و ابزارهای بکارگرفته شده برای این منظور پیچیده‌تر، مؤثرتر و دنبال کردن آن تا هدف نهائی سخت‌تر شده‌اند بطوری که تکنولوژی‌های دفاع، در برابر حملات بسیار گسترده قادر به مقاومت نیستند[CHAN02].

یک حملهٔ انکار سرویس (DoS) تلاشی برای ناکام گذاشتن کاربران قانونی در استفاده از آن سرویس است. وقتی این حمله از یک میزبان منفرد و یا یک گرهٔ شبکه شروع می‌شود آن را به سادگی یک حملهٔ DoS می‌خوانند. یک تهدید جدی‌تر حملهٔ DDoS است. در یک حملهٔ DDoS، یک مهاجم قادر است تا تعدادی از میزبان‌های روی اینترنت را به نحوی سازمان‌دهی نماید که همه با هم و یا بصورت متوالی به یک هدف حمله کنند. این بخش به حملات DDoS می‌پردازد. در ابتدا به ماهیت و انواع حمله اشاره می‌کنیم. سپس به نحوهٔ آماده‌سازی شبکه‌ای از میزبان‌ها برای انجام حمله می‌پردازیم. در انتها، روش‌های مقابله با این حملات را بررسی خواهیم کرد.

توصیف حملهٔ DDoS

یک حملهٔ DDoS تلاش می‌کند تا منابع هدف حمله را طوری بکار گیرد تا از دادن سرویس باز مانند. یک روش برای طبقه‌بندی حملات DDoS دسته‌بندی آنها بر اساس منبع بکارگرفته شده است. در حالت کلی، منبع بکارگرفته شده منابع داخلی سیستم هدف و یا ظرفیت انتقال دیتای شبکهٔ محلی سیستم هدف است.

یک مثال ساده از **حمله به منابع داخلی**، حملهٔ SYN flood است. شکل ۵-۱۰الف مراحل این حمله را نشان می‌دهد:

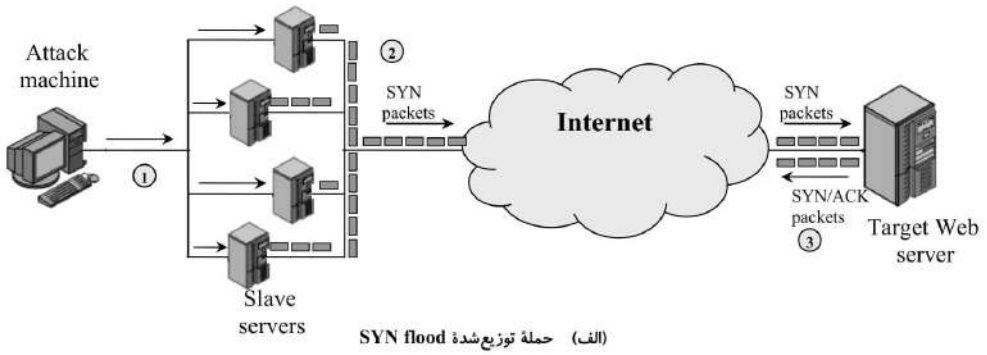
۱- حمله‌کننده کنترل تعدادی از میزبان‌های روی اینترنت را به دست گرفته و آنها را برای حمله به سرور وب تعلیم می‌دهد.

۲- میزبان‌های بخدمت گرفته شده شروع به ارسال بسته‌های TCP/IP SYN (synchronize/initialization) برای هدف، با اطلاعات غلط در موردآدرس‌های برگشتی IP، می‌نمایند.

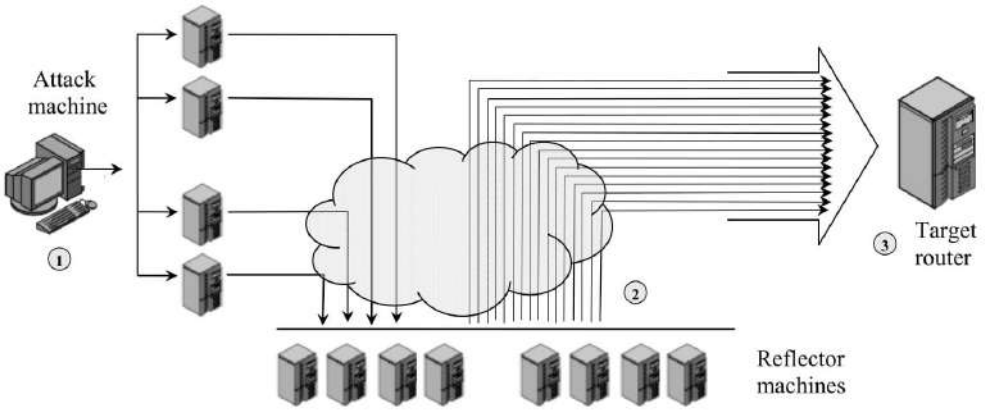
۳- هر بستهٔ SYN، یک درخواست برای گشودن یک اتصال TCP است. برای هریک از چنین بسته‌هائی، سرور وب با یک بستهٔ SYN/ACK (synchronize/acknowledge) جواب داده و تلاش می‌کند تا یک اتصال TCP را با واحد TCP در آدرس IP ساختگی ایجاد کند. سرور وب برای هر درخواست SYN که منتظر پاسخ است یک ساختار داده را نگاه داشته و همینطور که درخواست‌های بیشتری وارد می‌شود کم‌کم در باطلاق فرو می‌رود. نتیجهٔ امر این است که در حالی که ماشین قربانی شده، منتظر کامل شدن اتصالات «نیمه باز» قلابی است از پاسخ دادن به اتصالات قانونی باز می‌ماند.

حالت ساختار داده TCP، یک هدف معمول از نوع منابع داخلی بوده ولی به هیچوجه تنها منبع داخلی نیست. [CERT01] مثال‌های دیگری را ارائه می‌دهد:

۱- در بسیاری سیستم‌ها، تعداد معدودی ساختار داده وجود دارند که اطلاعات پردازشی را نگاه می‌دارند (شناسه‌های پردازش، جداول پردازش و غیره). یک مهاجم ممکن است بتواند با نوشتن یک برنامهٔ ساده و یا چند فرمان که کاری جز خلق کپی‌های مکرر از خود کاری انجام نمی‌دهند، این ساختارها را اشغال کند.



(الف) حملهٔ توزیع‌شدهٔ SYN flood



(ب) حملهٔ توزیع‌شدهٔ ICMP

شکل ۵-۱۰ مثال‌هائی از حملات سادهٔ DDoS

۲- یک مهاجم همچنین ممکن است تلاش کند تا ظرفیت دیسک را به طرق دیگر پرکند که از آن جمله‌اند:

- خلق تعداد بیشماری از پیام‌های پستی
- خلق خطاهای عمدی که بایستی ثبت شوند
- قراردادن فایل‌ها در نواحی ناشناس ftp یا نواحی ناشناس محیط‌های اشتراکی شبکه

شکل ۵-۱۰ب مثالی از یک **حمله به منابع انتقال دیتا** را نشان می‌دهد. قدم‌های زیر برداشته می‌شوند:

۱- حمله‌کننده، کنترل چندین میزبان روی اینترنت را به دست می‌گیرد و آنها را تعلیم می‌دهد تا بسته‌های ICMP ECHO با آدرس جعل شدهٔ IP هدف را به دسته‌ای از میزبان‌ها که بعنوان منعکس‌کننده عمل می‌کنند ارسال نمایند.
۲- گره‌های واقع در سایت منعکس‌کننده، درخواست‌های جعلی را دریافت کرده و در جواب بسته‌های پاسخ echo را برای سایت هدف ارسال می‌کنند.
۳- مسیریاب هدف تهاجم، با بسته‌های ارسالی از سایت منعکس‌کننده بمباران شده و دیگر ظرفیتی برای انتقال ترافیک قانونی باقی نمی‌ماند.

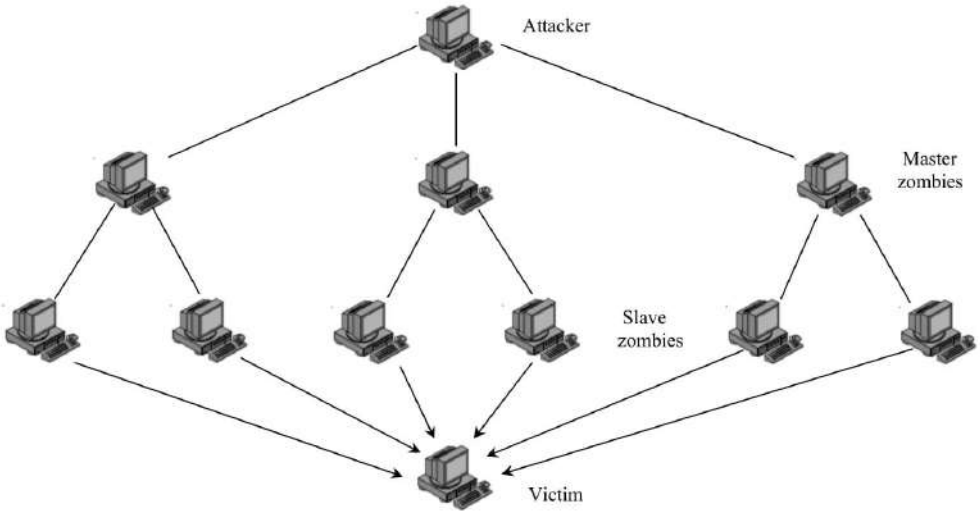
روش دیگری برای طبقه‌بندی حملات DDoS، تقسیم آنها به حملات مستقیم و حملات انعکاسی است. در یک **حملهٔ مستقیم (direct DDoS)** که در شکل ۶-۱۰الف نشان داده شده است، حمله‌کننده قادر است تا نرم‌افزارهای زامبی را در تعدادی از سایت‌هائی که در اینترنت پراکنده‌اند بنشاند. اغلب حملهٔ DDoS شامل دو سطح از ماشین‌های زامبی است: زامبی های ارباب و زامبی‌های بَرده. میزبان‌های هردو نوع زامبی با برنامهٔ بداندیش آلوده شده‌اند. حمله‌کننده، زامبی‌های ارباب را هماهنگ و به حمله وامی‌دارد که آنها هم به نوبهٔ خود زامبی‌های بَرده را هماهنگ و به حمله وامی‌دارند. استفاده از دو سطح زامبی، دنبال کردن حمله و یافتن منشاء آن را دشوارتر نموده و شبکهٔ حمله‌کنندهٔ مقاوم‌تری را بوجود می‌آورد.

یک **حملهٔ انعکاسی (reflector DDoS)**، لایهٔ دیگری از ماشین‌ها را در حمله وارد می‌کند (شکل ۶-۱۰ب). در این نوع حمله، زامبی‌های بَرده بسته‌های دیتائی را می‌سازند که نیاز به پاسخی دارد که شامل آدرس IP هدف بعنوان آدرس منبع در سرآیند بستهٔ IP است. این بسته‌ها به ماشین‌های غیرآلوده‌ای ارسال می‌شوند که منعکس‌کننده خوانده می‌شوند. ماشین‌های غیرآلوده با بسته‌هائی به مقصد ماشین هدف به این درخواست‌ها پاسخ می‌دهند. یک حملهٔ DDoS انعکاسی بآسانی می‌تواند ماشین‌های بیشتر و ترافیک بیشتری را نسبت به حملهٔ DDoS مستقیم درگیر کرده و بنابراین آسیب آن بیشتر است. علاوه برآن، یافتن منشاء حمله و فیلتر کردن بسته‌های حمله‌کننده سخت‌تر بوده زیرا حمله از تعدادی ماشین غیرآلوده که در سطح وسیع گسترده‌اند سرچشمه می‌گیرد.

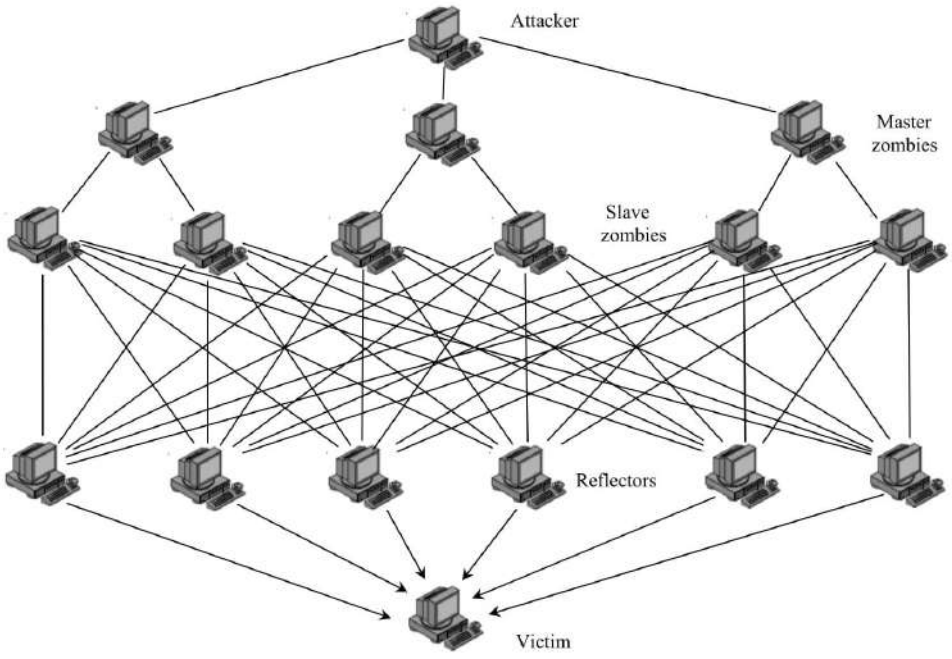
ایجاد شبکهٔ حمله‌کننده

اولین قدم در یک حملهٔ DDoS این است که حمله‌کننده تعدادی نرم‌افزار زامبی را در تعدادی ماشین بکارد تا آنها بعداً بتوانند حمله را آغاز کنند. اجزاء ضروری این مرحله از حمله عبارتند از:

۱- نرم‌افزاری که بتواند حملهٔ DDoS را انجام دهد. نرم‌افزار باید بتواند روی تعداد زیادی ماشین اجرا شده، بایستی بتواند حضور خود را پنهان کرده، باید بتواند ارتباط خود را با حمله‌کننده حفظ کرده و یا از یک مکانیسم خود انفجاری بهره‌مند بوده، و بالاخره بایستی قادر باشد تا حملهٔ برنامه‌ریزی شده را روی هدف اجرا نماید.
۲- وجود یک آسیب‌پذیری در تعداد زیادی از سیستم‌ها. حمله‌کننده بایستی از وجود یک نقطهٔ آسیب‌پذیر یا حفرهٔ امنیتی که تعداد زیادی از مدیران سیستم‌ها و کاربران منفرد از آن غافل‌اند باخبر بوده تا بتواند نرم‌افزار زامبی را در آن نقاط نصب نماید.
۳- یک استراتژی برای یافتن ماشین‌های آسیب‌پذیر، فرایندی که اِسکن کردن نامیده می‌شود.



(الف) حملهٔ DDoS مستقیم



(ب) حملهٔ DDoS انعکاسی

شکل ۶-۱۰ انواع حملات بمبارانی DDoS

در عمل اِسکن کردن، حمله‌کننده ابتدا به جستجوی تعدادی ماشین آسیب‌پذیر پرداخته و آنها را آلوده می‌سازد. آنگاه بطور معمول، نرم‌افزار زامبی که در ماشین آلوده نصب شده است همان عمل اِسکن کردن را تکرار کرده تا یک شبکهٔ بزرگ گسترده از ماشین‌های آلوده ایجاد شود. [MIRK04] از استراتژی‌های زیر برای عمل اِسکن نام می‌برد:

- **Random**: هر میزبان به دام افتاده، به آدرس‌های تصادفی در فضای آدرس‌های IP نفوذ کرده و برای هرکدام از یک seed مختلف استفاده می‌کند. این تکنیک حجم بالائی از ترافیک اینترنت را بوجود می‌آورد که ممکن است حتی قبل از آغاز حملهٔ اصلی سرویس را مختل سازد.
- **Hit-list**: حمله‌کننده ابتدا یک لیست طولانی از ماشین‌هائی که پتانسیل آسیب‌پذیری دارند را تهیه می‌کند. این امر ممکن است به کندی و در طول زمان انجام شود تا از تشخیص این که حمله‌ای در شرف وقوع است اجتناب شود. وقتی که لیست تهیه و جمع‌آوری گردید، حمله‌کننده شروع به آلوده کردن ماشین‌های موجود در لیست می‌نماید. به هر ماشین آلوده شده، بخشی از لیست برای اِسکن کردن واگذار می‌شود. این استراتژی به اِسکن سریع تعداد زیادی ماشین در مدت کوتاهی منجر شده که می‌تواند تشخیص وقوع آلودگی را با دشواری مواجه سازد.
- **Topological**: این روش از اطلاعات موجود در ماشین قربانی استفاده کرده تا میزبان‌های جدیدی را برای اِسکن کردن بیابد.
- **Local subnet**: اگر میزبانی که پشت یک دیوار آتش قرار دارد بتواند آلوده شود، آنگاه این میزبان در شبکهٔ محلی خود به دنبال آلوده کردن اهداف دیگر خواهد رفت. این میزبان از ساختار آدرسی زیرشبکه استفاده کرده تا میزبان‌های دیگری را که در غیر اینصورت تحت حفاظت دیوار آتش می‌بودند پیدا کند.

روش‌های مقابله با DDoS

در حالت کلی سه خط دفاعی در برابر حملات DDoS وجود دارد [CHAN02]:

- **جلوگیری از حمله و بازدارندگی (قبل از حمله)**: این مکانیسم‌ها قربانی را قادر می‌سازد تا بدون اینکه برای کلاینت‌های قانونی از سرویس دادن باز ماند، در برابر تلاش برای حمله مقاومت نماید. تکنیک‌های این مورد شامل اعمال سیاست‌های مناسب برای بکارگیری منابع و تدارک دیدن منابع رزرو در صورت تقاضاست. علاوه براین، مکانیسم‌های بازدارنده، سیستم‌ها و پروتکل‌های اینترنتی را طوری جرح و تعدیل می‌نماید که احتمال حملهٔ DDoS کم شود.
- **تشخیص حمله و فیلتر کردن (در طول حمله)**: این مکانیسم‌ها تلاش می‌کنند تا حمله را هرچه سریع‌تر تشخیص داده و عکس‌العمل بلادرنگ نشان دهند. این امر اثرات حمله بر هدف را به حداقل می‌رساند. تشخیص شامل جستجوی رفتارهای مشکوک است. پاسخ شامل فیلترکردن بسته‌های دیتائی است که احتمالاً بخشی از حمله هستند.
- **جستجوی منشاء حمله و شناسائی (در طول حمله و بعد از آن)**: این تلاشی برای پیدا کردن منبع حمله بعنوان قدمی در جهت جلوگیری از حملات آتی است. این روش، حتی اگر موفقیت‌آمیز باشد، معمولاً نتایج سریعی را در جهت مقابله با حملهٔ درحال انجام بدست نمی‌دهد.

چالش عمده در مبارزه با حملات DDoS روش‌های متنوع عملیاتی آنهاست. پاتک‌های DDoS بایستی به همراه تهدیدها تکامل یابند.

۱۰-۴ منابع مطالعاتی

برای یک فهم کامل از ویروس‌ها، باید کتاب [SZOR05] را خواند. منبع بسیار خوب دیگر [HARL01] است. مقالات خوب در مورد ویروس‌ها و کِرم‌ها [CASS01]، [FORR97]، [KEPH97] و [NACH97] می‌باشند. [MEIN01] اطلاعات مفیدی در مورد کِرم Code Red را در اختیار می‌گذارد.

[PATR04] یک بررسی ارزنده از حملات DDoS است. [MIRK04] یک توصیف کامل از حملات DDoS و پاتک‌های آنها را ارائه می‌کند. [CHAN02] یک بررسی خوب از استراتژی‌های دفاعی DDoS است.

CASS01 Cass, S. "Anatomy of Malice." *IEEE Spectrum*, November 2001.

CHAN02 Chang, R. "Defending Against Flooding-Based Distributed Denial-of-Servise Attacks: A Tutorial." *IEEE Communications Magazine*, October 2002.

FORR97 Forrest, S.; Hofmeyr, S.; and Sommayaji, A. "Computer Immunology." *Communications of the ACM*, October 1997.

HARL01 Harley, D.; Slade, R.; and Gattiker, U. *Viruses Revealed*. New York: Osborne/McGraw-Hill, 2001.

KEPH97 Kephart, J.; Sorkin, G.; Chess,D.; and White, S."Fighting Computer Viruses." *Scientific American*, November 1997.

MEIN01 Meinel, C. "Code Red for the Web." *Scientific American*, October 2001.

MIRK04 Mirkovic, J., and Relher,P."A Taxonomy of DDoS Attack and DDoS Defence Mechnisms." *ACM SIGCOMM Computer Communications Review*, April 2004.

NACH97 Nachenberg, C. "Computer Virus-Antivirus Coevolution." *Communications of the ACM*, January 1997.

PATR04 Patrikakis, C.; Masikos, M.; and Zouraraki, O."Distributed Denial of Service Attacks." *The Internet Protocol Journal*, December 2004.

SZOR05 Szor, p., *The Art of Computer Virus Research and Defence*.Reading,MA:Addison-Wesley,2005.

وِب سایت‌های مفید

- **AntiVirus Online**، سایت IBM دربارهٔ اطلاعات مربوط به ویروس‌ها.
- **Vmyths**، مختص برملاکردن خطرات ویروس‌ها و رفع شبهات مربوط به ویروس‌های واقعی.
- **DDoS Attacks/Tools**، لیست گسترده‌ای از لینک‌ها و اسناد مربوط به این بحث.

۱۰-۵ واژه‌های کلیدی، سؤالات مرورکنندهٔ بحث و مسائل

واژه‌های کلیدی

auto-rooter	نوعی ابزار نفوذگری	macro virus	ویروس ماکرو
backdoor	درب مخفی	malicious software (malware)	نرم‌افزار بداندیش
digital immune system	سیستم مصون دیجیتال	polymorphic virus	ویروس چندچهره
direct DDoS attack	حملهٔ DDoS مستقیم	reflector DDoS attack	حملهٔ DDoS انعکاسی
distributed denial of service	انکار سرویس توزیع شده	rootkit	ابزار دسترسی به ریشهٔ برنامه
downloaders	بارگذاری کننده	spammer program	برنامهٔ ارسال هَرزنامه‌ها
e-mail virus	ویروس پست الکترونیک	stealth virus	ویروس پنهان شونده
exploits	استثمارگرها	trapdoor	درب مخفی
flooder	حملهٔ سیلابی	trojan horse	اسب تروا
keylogger	ثبت کنندهٔ حرکات صفحه کلید	virus	ویروس
kit	جعبه ابزار	worm	کِرم
logic bomb	بمب لاجیک	zombie	زامبی

سؤالات مرورکننده بحث

۱۰-۱ نقش فشرده‌سازی در عملکرد یک ویروس چیست؟
۱۰-۲ نقش رمزنگاری در عملکرد یک ویروس چیست؟
۱۰-۳ فازهای مختلف عملکرد یک ویروس یا کِرم کدام‌اند؟
۱۰-۴ یک کِرم در حالت کلی چگونه انتشار می‌یابد؟
۱۰-۵ یک سیستم مصون دیجیتال چیست؟
۱۰-۶ نرم‌افزار سدکنندهٔ رفتار چگونه کار می‌کند؟
۱۰-۷ یک DDoS چیست؟

مسائل

۱۰-۱ در برنامهٔ ویروس شکل ۱۰-۱ یک اشتباه وجود دارد. آن چیست؟

۱۰-۲ این سؤال مطرح است که آیا می‌توان برنامه‌ای نوشت که بتواند یک نرم‌افزار را تجزیه و تحلیل کرده و مشخص نماید که آیا این نرم‌افزار ویروس است یا نه؟ فرض کنید که برنامه‌ای مانند D داریم که قادر به این کار است. یعنی، برای هر برنامهٔ P اگر برنامهٔ D(P) را اجرا کنیم نتیجه یا مثبت (P ویروس است) و یا منفی (P ویروس نیست) خواهد بود. حال برنامهٔ زیر را در نظر بگیرید:

```
program CV :=
  { ...
  main-program :=
          {if D(CV) then goto next :
                  else infect-executable ;
          }
next :
  }
```

در برنامهٔ بالا infect-executable یک مدول است که حافظه را بمنظور یافتن برنامه‌های اجرائی اسکن کرده و خود را در آن برنامه‌ها کپی می‌کند. تعیین کنید که آیا D می‌تواند در مورد اینکه CV یک ویروس است تصمیم صحیح بگیرد؟

فصل ۱۱

دیوارهای آتش

دیوارهای آتش می‌توانند یک وسیلهٔ مؤثر برای حفاظت یک سیستم محلی، و یا شبکه‌ای از سیستم‌ها، در مقابل تهدیدهای امنیتی مبتنی بر شبکه بوده و در عین حال دسترسی به دنیای خارج، از طریق شبکه‌های WAN و اینترنت را حفظ کنند.

این فصل را با مروری بر اصول عملکرد و نحوهٔ طراحی دیوارهای آتش آغاز می‌کنیم. سپس به مقولهٔ امنیت خود دیوار آتش پرداخته و علی‌الخصوص فرضیهٔ یک سیستم معتمد یا سیستم عامل امن را بررسی می‌نمائیم.

۱-۱۱ اصول طراحی دیوارهای آتش

سیستم‌های اطلاعات در شرکت‌ها، دوایر دولتی و سایر سازمان‌ها بطور پیوسته در حال تکامل بوده‌اند:

- سیستم متمرکز پردازش داده‌ها، با یک رایانهٔ بزرگ مرکزی، که تعدادی پایانه که مستقیماً به آن وصل‌اند را حمایت می‌نماید.
- شبکه‌های محلی (LAN) که PCها و پایانه‌ها را به هم و به رایانهٔ مرکزی متصل می‌کند.
- شبکه‌های اختصاصی که شامل تعدادی از LANها، PCهای متصل بهم، سِرورها و شاید یک یا دو رایانهٔ بزرگ مرکزی است.
- شبکه‌های وسیع تجاری، شامل تعدادی از شبکه‌های اختصاصی در مناطق جغرافیائی مختلف، که بتوسط یک WAN خصوصی با هم ارتباط دارند.
- اتصال اینترنتی که در آن شبکه‌های اختصاصی مختلف همگی به اینترنت وصل بوده و ممکن است بتوسط یک WAN خصوصی به هم متصل باشند.

برای اکثر سازمان‌ها، اتصال به اینترنت دیگر امروز یک امر تشریفاتی نیست. استفاده از اطلاعات و سرویس‌های موجود در اینترنت از ضروریات سازمانی محسوب می‌شود. علاوه برآن تک‌تک کاربران درون یک سازمان نیز تمایل و نیاز به دسترسی به اینترنت دارند و اگر این امر بتوسط شبکهٔ LAN سازمان آنها فراهم نگردد، از قابلیت‌های خط تلفن استفاده کرده و PC خود را از طریق یک فراهم‌آورندهٔ سرویس اینترنتی (ISP) به اینترنت متصل می‌سازند. از سوی دیگر در حالی‌که دسترسی به اینترنت منافعی را برای سازمان به ارمغان می‌آورد ولی دنیای خارج را نیز قادر می‌سازد تا به تجهیزات شبکه‌های محلی دسترسی یافته و با آنها تبادل اطلاعات داشته باشد. این امر تهدیدی را برای سازمان بوجود می‌آورد. اگرچه ممکن است که هر ایستگاه کاری و سِرور یک شبکه را با تجهیزات امنیتی قوی تجهیز کرد ولی چنین حفاظتی، یک روش عملی مناسب نیست. شبکه‌ای با صدها و شاید هزارها سیستم را در نظر بگیرید که از معجونی از نسخه‌های متفاوت UNIX و Windows استفاده می‌کند. وقتی یک شکاف امنیتی کشف شود، هر سیستمی که تحت تأثیر این امر قرار گرفته بایستی برای رفع مشکل

ارتقاء یابد. راه حل دیگر که بطور فزاینده‌ای مورد پذیرش قرار گرفته است، استفاده از دیوار آتش است. دیوار آتش بین شبکهٔ اختصاصی و اینترنت قرار می‌گیرد تا یک پیوند کنترل شده را ایجاد کرده و یک دیوار امنیتی خارجی را در پیرامون شبکه ایجاد نماید. هدف این دیوار پیرامونی، این است که شبکه را از حملات امنیتی اینترنتی حفاظت کرده و با فراهم آوردن تنها یک منفذ، مسئولین شبکه را قادر سازد تا از آن منفذ موارد امنیتی و ممیزی شبکه را کنترل کنند. دیوار آتش ممکن است یک کامپیوتر تنها و یا مجموعه‌ای از چند سیستم کامپیوتری باشد که با تعامل با یکدیگر وظایف دیوار آتش را انجام دهند.

در این بخش ابتدا مشخصات کلی دیوارهای آتش را بررسی می‌کنیم. سپس به انواع دیوارهای آتش که امروز مورد استفاده‌اند نظری می‌اندازیم. بالاخره تعدادی از متداول‌ترین پیکربندی‌های دیوارهای آتش را مورد بررسی قرار می‌دهیم.

مشخصه‌های دیوار آتش (Firewall)

[BELL94b] اهداف طراحی یک دیوار آتش را چنین بیان می‌کند:

۱- تمام ترافیک داخل به خارج و بالعکس بایستی از میان دیوار آتش عبور نماید. این امر با مسدود کردن فیزیکی تمام دسترسی‌ها به شبکهٔ محلی، بجز از طریق دیوار آتش حاصل می‌گردد. از پیکربندی‌های متنوعی در این مورد می‌توان استفاده کرد که بعداً مورد بحث قرار خواهد گرفت.

۲- تنها به ترافیک معتبر، برابر آنچه خط‌مشی امنیتی محلی آن را تعریف کرده است، اجازهٔ عبور داده می‌شود. انواع متنوعی از دیوارهای آتش مورد استفاده قرار می‌گیرندکه هریک انواع مختلفی از خط‌مشی‌ها را ایجاد می‌کنند. بعداً در این باره بحث خواهیم کرد.

۳- خود دیوار آتش در مقابل نفوذ بیگانه دارای امنیت است. این امر استفاده از یک سیستم مورد اعتماد یا یک سیستم عامل امن را ایجاب می‌نماید. این موضوع را نیز بعداً مورد بحث قرار خواهیم داد.

[SMIT97] چهار تکنیک عام، که دیوارهای آتش برای کنترل دست‌یابی و عملیاتی کردن خط‌مشی امنیتی سایت مورد استفاده قرار می‌دهند، را ذکر کرده است. در ابتدا دیوارهای آتش عمدتاً روی کنترل سرویس نظارت داشتند ولی تکامل آنها باعث شده است که هرچهار منظور را مورد توجه قرار دهند:

- **کنترل سرویس**: نوع سرویس‌های اینترنتی، چه در محدودهٔ شبکه و چه در خارج از محدودهٔ شبکه، که می‌توانند قابل دست‌یابی باشند را تعیین می‌کند. دیوار آتش ممکن است ترافیک را بر اساس آدرس IP و شمارهٔ پورت TCP فیلتر کند، ممکن است نرم‌افزار پروکسی(proxy)، که درخواست هرنوع سرویس قبل از عبور آن به مقصد را دریافت و تحلیل می‌نماید، فراهم سازد و یا ممکن است خود بستر نرم‌افزار سِرور همانند سرویس وِب و یا پست الکترونیک باشد.
- **کنترل جهت**: جهتی که فقط در آن جهت درخواست سرویس بخصوصی پذیرفته شده و اجازهٔ عبور از دیوار آتش دارد را تعیین می‌کند.
- **کنترل کاربر**: دست‌یابی به یک سرویس، بر اساس اینکه کدام کاربر می‌خواهد از آن استفاده کند، را کنترل می‌کند. این کنترل معمولاً به کاربران داخل محدودهٔ دیوار آتش (کاربران محلی) اعمال می‌شود. این سرویس همچنین ممکن است به ترافیک ورودی کاربران خارج از محدوده نیز اعمال شود که در این حالت نیاز به نوعی روش اعتبارسنجی همانند IPSec است.

- **کنترل رفتار**: کنترل چگونگی استفاده از سرویس را بعهده دارد. بعنوان مثال دیوار آتش ممکن است نامه‌های الکترونیک را برای جلوگیری از عبور هَرزنامه (spam) فیلتر کند و یا ممکن است دسترسی خارجی را تنها به بخشی از اطلاعات سِرور وِب ممکن سازد.

قبل از پرداختن به جزئیات انواع دیوار آتش و پیکربندی‌های مختلف آن، بهتر است آنچه را که می‌توان از یک دیوار آتش انتظار داشت خلاصه کنیم. قابلیت‌های زیر معمولاً در حوزهٔ عملکرد یک دیوار آتش قرار دارد:

۱- یک دیوار آتش یک گلوگاه منفرد را تعریف می‌کند تا کاربران غیرمعتبر را از شبکهٔ محافظت شده دور نگاه داشته، سرویس‌های بالقوه خطرآفرین را از ورود به شبکه و خروج از شبکه مانع شده، و حفاظت از انواع متنوع حملات تقلیدIP و مسیریابی را ایجاد کند. استفاده از یک گلوگاه منفرد، مدیریت امنیت را تسهیل می‌نماید زیرا قابلیت‌های امنیتی در یک سیستم تنها، و یا مجموعه‌ای از سیستم‌ها متمرکز می‌شود.

۲- یک دیوار آتش، محلی برای پائیدن پیشامدهای مرتبط با امنیت را ایجاد می‌کند. ممیزی‌ها و آلارم‌ها می‌توانند روی یک سیستم دیوار آتش بنا نهاده شوند.

۳- یک دیوار آتش یک بستر مناسب برای چندین عمل اینترنتی است که ربطی به امنیت ندارند. اینها شامل یک مترجم آدرس شبکه است که آدرس‌های محلی را به آدرس‌های اینترنتی نگاشت نموده و یا وظیفه‌ای مدیریتی است که اتصال به اینترنت را اجازه داده و یا ممیزی می‌نماید.

۴- یک دیوار آتش می‌تواند بعنوان بستر IPSec عمل نماید. با استفاده از قابلیت مُود تونل (tunnel mode) توصیف شده در فصل ۶، دیوار آتش می‌تواند برای ایجاد شبکه‌های خصوصی مجازی (VPN) بکار رود.

دیوارهای آتش محدودیت‌های مخصوص به خود را نیز داشته که شامل موارد ذیل‌اند:

۱- دیوار آتش نمی‌تواند در مقابل حملاتی که دیوار آتش را دور می‌زنند مقاومت کند. سیستم‌های داخلی شبکه ممکن است از قابلیت شماره‌گیری تلفنی برای اتصال به یک ISP استفاده کنند. یک LAN داخلی ممکن است از یک مخزن مُودم استفاده کند که قابلیت اتصال به اینترنت از طریق خط تلفن برای کارمندانی که در ماموریت خارج از سازمان هستند را فراهم نماید.

۲- دیوار آتش در برابر تهدیدهای داخلی همانند یک کارمند ناراضی و یا کارمندی که سهواً به یک نفوذگر خارجی کمک می‌کند حفاظتی ایجاد نمی‌کند.

۳- دیوار آتش نمی‌تواند در برابر انتقال برنامه‌ها یا فایل‌های ویروسی ایجاد حفاظت نماید. با توجه به تنوع سیستم‌های عامل و تنوع برنامه‌های کاربردی در درون یک محدوده، برای دیوار آتش غیرعملی و شاید غیرممکن است که تمام فایل‌های ورودی، e-mailها و پیام‌ها را بمنظور یافتن ویروس‌ها اِسکن نماید.

انواع دیوارهای آتش

شکل ۱-۱۱ سه نوع معمول دیوار آتش را نشان می‌دهد: فیلترهای بسته‌های دیتا (packet filters)، دروازه‌های سطح کاربرد (application-level gateways) و دروازه‌های سطح مدار (circuit-level gateways). هریک از این سه نوع را به نوبت بررسی می‌کنیم.

مسیریاب فیلترکنندهٔ بسته‌های دیتا (Packet- Filtering Router)

یک مسیریاب فیلترکنندهٔ بسته‌ها، یک سری قواعد را به بسته‌های IP ورودی اعمال کرده و آنگاه یا آنها را بجلو رانده و یا معدوم می‌کند. مسیریاب نوعاً طوری پیکربندی می‌شود که بسته‌های دیتا را در هر دو جهت (بسمت داخل و بسمت خارج شبکه) فیلتر نماید. قواعد فیلترینگ براساس اطلاعاتی است که در یک بستهٔ اطلاعاتی دیتا وجود دارد:

- **آدرس IP منبع**: آدرس IP سیستمی که بستهٔ IP را ارسال کرده است (مثلاً 192.168.1.1).
- **آدرس IP مقصد**: آدرس IP مقصدی که بستهٔ دیتا قصد رسیدن به آن را دارد (مثلاً 192.168.1.2).
- **آدرس سطح حمل‌ونقل منبع و مقصد**: شمارهٔ پورت سطح حمل‌ونقل (مثلTCP یا UDP)که کاربردهائی مثل SNMP یا TELNET را تعریف می‌کند.
- **میدان پروتکل IP**: نوع پروتکل حمل‌ونقل را تعریف می‌کند.
- **مدار واسط**: برای یک مسیریاب با ۳ پورت و بیشتر، اینکه از کدام مدار واسط مسیریاب، بسته خارج شده و یا به کدام مدار واسط مسیریاب، بسته وارد می‌شود.

فیلتر بسته‌ها معمولاً بصورت لیستی از قواعد، که مبتنی بر تطبیق با میدان‌های سرآیند IP یا TCP است، تنظیم می‌گردد. اگر بین قواعد وضع شده با میدان‌های بسته تطابقی یافت شود، آن قاعده برای تعیین اینکه بسته عبور کرده و یا نابود شود بکار گرفته می‌شود. اگر تطبیقی یا قاعده‌ای یافت نشود آنگاه براساس پیش‌فرض عمل خواهد شد. برای پیش‌فرض دو حالت ممکن وجود دارد:

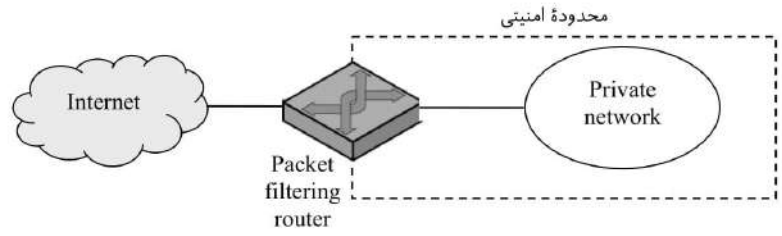
- **پیش‌فرض = *نابودی***: آنچه که مجاز تعریف نشده است ممنوع است.
- **پیش‌فرض = *عبور***: آنچه که ممنوع تعریف نشده است مجاز است.

پیش‌فرضی که نابودی را پیشنهاد می‌دهد، محافظه‌کارانه‌تر است. در ابتدا جلوی همه چیز سد می‌شود و سرویس‌ها را بایستی مورد به مورد تعریف و اضافه نمود. این روش برای کاربران ملموس‌تر بوده و احتمال اینکه آنها دیوار آتش را بعنوان یک مانع جدّی تلقی کنند بیشتر است. پیش‌فرض عبور، عملیات کاربران انتهائی را تسهیل کرده ولی امنیت کمتری را فراهم می‌سازد. در این حالت مسئول امنیت شبکه بایستی بواقع هوشیار بوده و نسبت به هر تهدید امنیتی جدید، بمحض اینکه کشف شود، واکنش نشان دهد.

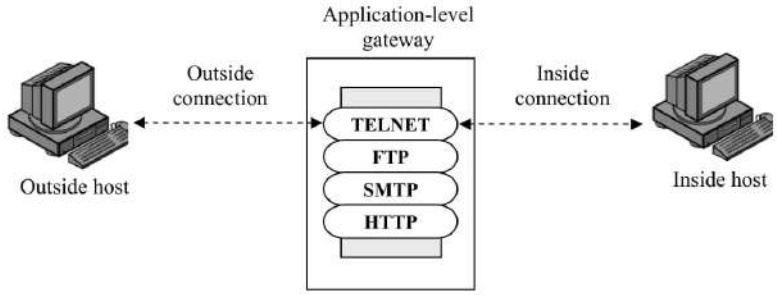
جدول ۱-۱۱ از [BELL94]، مثال‌هائی از مجموعهٔ قواعد فیلترینگ بسته‌های دیتا را نشان می‌دهد. در هر مجموعه، قوانین از بالا به پائین اعمال می‌شوند. علامت " * " در یک میدان، یک نمایشگر عام بوده که بجای آن هرچیزی می‌تواند قرار داشته باشد. فرض بر این است که پیش‌فرض = نابودی به بسته‌های دیتا اعمال می‌شود.

الف: نامه‌های وارد به محدوده (پورت ۲۵ برای SMTP ورودی است) فقط به مقصد یک دروازه میزبان مجاز است. ولی نامه‌های یک میزبان خارجی بخصوص، SPIGOT، مسدود شده زیرا این میزبان تاریخچه‌ای حاکی از ارسال فایل‌های حجیم در پیام‌های پستی خود دارد.

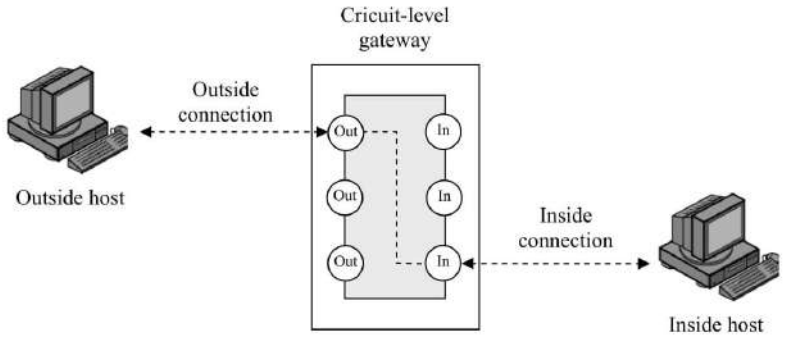
ب: این حالت بیان صریح سیاست مربوط به پیش فرض است. تمام مجموعه‌های قواعد در انتهای کار بطور ضمنی از این قانون پیروی می‌کنند.



(الف) مسیریاب فیلترکنندهٔ بسته‌های دیتا



(ب) دروازهٔ سطح کاربرد



(ج) دروازهٔ سطح مدار

شکل ۱-۱۱ انواع دیوار آتش

جدول ۱-۱۱ مثال‌هائی از فیلترینگ بسته‌های دیتا

action	ourhost	port	theirhost	port	comment
block	*	*	SPIGOT	*	we don't trust these people
allow	OUR-GW	25	*	*	connection to our SMTP port

(الف)

action	ourhost	port	theirhost	port	comment
block	*	*	*	*	default

(ب)

action	ourhost	port	theirhost	port	comment
allow	*	*	*	25	connection to their SMTP port

(ج)

action	src	port	dest	port	flags	comment
allow	{our hosts}	*	*	25		our packets to their SMTP port
allow	*	25	*	*	ACK	their replies

(د)

action	src	port	dest	port	flags	comment
allow	{our hosts}	*	*	*		our outgoing calls
allow	*	*	*	*	ACK	replies to our calls
allow	*	*	*	>1024		traffic to nonservers

(هـ)

ج: این مجموعهٔ قواعد بیانگر این است که هر میزبان داخلی می‌تواند به خارج e-mail بزند. یک بستهٔ TCP با پورت مقصد شمارهٔ ۲۵ به سرور SMTP ماشین مقصد ارسال می‌گردد. اشکال این قاعده این است که استفاده از پورت ۲۵ برای دریافت SMTP، تنها یک پیش‌فرض است. یک دستگاه خارجی می‌تواند طوری پیکربندی شود که کاربرد دیگری برای پورت ۲۵ داشته باشد. با عمل به این قاعده یک مهاجم می‌تواند با ارسال بسته‌هائی با شمارهٔ ۲۵ در TCP، به کامپیوترهای داخل محدوده دسترسی یابد.

د: این مجموعهٔ قواعد به نتایجی منجر می‌گردد که در بند (ج) به آن دست نمی‌یافتیم. قواعد از مشخصهٔ اتصالات TCP بهره می‌گیرند. هروقت اتصالی برقرار شود، پرچم ACK یک سگمنت TCP به اهتزاز درآمده تا سگمنت‌های ارسال شده از سمت دیگر را تائید کند. بنابراین مجموعهٔ این قواعد بیان می‌دارند که بسته‌های IP که آدرس‌های IP مبداء آنها یکی از میزبان‌های داخلی مشخص بوده و شمارهٔ پورت TCP مقصد آنان ۲۵ است پذیرفته می‌گردند. همچنین بسته‌های ورودی که شمارهٔ پورت مبداء آنها ۲۵ بوده و پرچم ACK را در سگمنت TCP در اهتزاز دارند، مجاز به عبور می‌باشند. توجه شود که ما سیستم‌های منبع و مقصدکه بایستی این قواعد را بطور صریح تعریف کنند، با روشنی کامل مشخص نموده‌ایم.

هـ: این مجموعهٔ قواعد، یکی از روش‌های مدیریت اتصالات FTP است. درFTP از دو اتصال TCP استفاده می‌شود: یک اتصال کنترلی برای برقراری مقدمات انتقال فایل و یک اتصال دیتا برای انتقال واقعی خود فایل. اتصال دیتا از

یک شمارهٔ پورت متفاوت که بطور پویا برای این امر اختصاص می‌یابد، استفاده می‌کند. بیشتر سرورها روی شمارهٔ پورت‌های پائین کارکرده که در نتیجه بیشتر نیز هدف حملات قرار می‌گیرند. بیشتر مکالمات خارجی تمایل به استفاده از پورت‌های با شماره های بالاتر، نوعاً بالای ۱,۰۲۳، دارند. بنابراین این قاعده به موارد زیر اجازه می‌دهد:

- بسته‌هائی که از داخل شبکه سرچشمه گرفته‌اند
- بسته‌های پاسخ برای یک اتصال که از یک ماشین داخلی سرچشمه گرفته‌اند
- بسته‌هائی که برای یک پورت شمارهٔ بالا در یک ماشین داخلی ارسال می‌شوند

این روش نیازمند این است که سیستم‌ها طوری پیکربندی شوند که تنها پورت‌های مناسب بکار گرفته شوند.

مجموعهٔ قواعد (هـ) مشکلاتی که در راه برخورد با کاربرها در سطح فیلترکردن بسته‌ها وجود دارد را خاطرنشان می‌سازد. روش دیگری برای برخورد با FTP و کاربردهای مشابه، استفاده از یک دروازه سطح کاربرد است که بعداً در این بخش به آن خواهیم پرداخت.

یکی از محاسن مسیریاب فیلترکنندهٔ بسته‌ها، سادگی آن است. همچنین فیلترهای بسته‌های دیتا نوعاً از نظر کاربران شفاف بوده و خیلی سریع هستند.[WACK02] نقاط ضعف زیر را برای دیوارهای آتش از این نوع برمی‌شمارد:

- چون دیوارهای آتش فیلترکنندهٔ بسته‌ها، داده‌های لایهٔ بالاتر را بررسی نمی‌کنند، آنها نمی‌توانند از حملاتی که عملیات و یا نقاط آسیب‌پذیر مختص به کاربرد را هدف قرار می‌دهند، جلوگیری کنند. بعنوان مثال یک دیوار آتش فیلترکنندهٔ بسته‌ها نمی‌تواند فرامین کاربردی مشخصی را بلوکه کند و بنابراین تمام عملیات موجود در آن کاربرد مجاز شناخته می‌شود.
- بعلت اطلاعات محدود دیوار آتش فیلترکنندهٔ بسته‌ها، عملیات اتصال به شبکه در این فیلتر محدود است. عملیات اتصال به سیستم معمولاً شامل همان اطلاعاتی است که از آنها برای تصمیم‌گیری در مورد کنترل دست‌یابی استفاده می‌شود (آدرس منبع، آدرس مقصد و نوع ترافیک).
- بیشتر دیوارهای آتش فیلترکنندهٔ بسته‌ها روش‌های پیشرفتهٔ تصدیق هویت کاربران را به خدمت نمی‌گیرند. بازهم این محدودیت عمدتاً بعلت فقدان کارآئی سیستم دیوار آتش در لایهٔ بالاتر است.
- آنها معمولاً در مقابل حملات و استثماری که از مشکلات درونی TCP/IP و پشتهٔ پروتکلی آن ناشی می‌شود، همانند *network layer address spoofing* آسیب‌پذیرند. بسیاری از دیوارهای آتش فیلترکنندهٔ بسته‌ها قادر به تشخیص یک بستهٔ لایهٔ شبکه، که اطلاعات آدرسی لایهٔ سوم OSI آن تغییر داده شده است، نیستند. مهاجمین معمولاً از تقلید آدرس برای عبور از کنترل‌های امنیتی یک دیوار آتش استفاده می‌کنند.
- بالاخره بعلت تعداد کم پارامترهای دخیل در تصمیم‌گیری نسبت به کنترل دست‌یابی، دیوارهای آتش فیلترکنندهٔ بسته‌ها در معرض رخنه‌های امنیتی ناشی از پیکربندی نامناسب قرار دارند. بعبارت دیگر بصورت خیلی ساده و تصادفی، یک دیوار آتش فیلترکنندهٔ بسته‌ها ممکن است طوری پیکربندی شود که به ترافیک، نوع منابع و نوع مقاصدی که قاعدتاً بایستی بر اساس خط‌مشی امنیت اطلاعات سازمان از عبور آنها جلوگیری شود، اجازهٔ عبور دهد.

برخی از انواع حملات و روش‌های معقول دفاع در برابر آنها، که ممکن است بر علیه دیوارهای آتش فیلترکنندهٔ بسته‌ها انجام شود، بقرار زیراند:

- **تقلیدآدرسIP (IP spoofing)**: مهاجم، بسته‌هائی را از خارج ارسال می‌دارد که در محل آدرس IP منبع آنها، آدرس یک میزبان داخلی جا داده شده است. حمله‌کننده امیدوار است که استفاده از یک آدرس تقلیدی باعث شود که او بتواند در سیستم‌هائی که بسادگی فقط آدرس منبع را کنترل نموده و در آن بسته‌های مشخص میزبان‌های داخلی مورد اعتماد پذیرش می‌گردند، نفوذ یابد. ضدحملهٔ این روش این است که بسته‌هائی که از یک مدار واسط خارجی وارد شده ولی دارای آدرس یک منبع داخلی هستند را نابود کرد.
- **حملات مسیریابی منبع**: ایستگاه منبع، مسیر عبور یک بسته در اینترنت را تعیین کرده و امیدوار است که این روش، موانع امنیتی که اطلاعات مسیریابی را کنترل نمی‌کنند دور بزند. ضدحملهٔ این روش این است که تمام بسته‌هائی که از این ابزار استفاده می‌کنند را نابود کرد.
- **حملات فِرگمنت‌های کوچک**: مهاجم از ابزار IP fragmentation استفاده کرده تا فِرگمنت‌های فوق‌العاده کوچکی را خلق نموده و اطلاعات سرآیند TCP را مجبور سازد تا در یک فِرگمنت مجزا از دیتا قرار گیرد. این حمله برای فریب دادن قواعد فیلترینگ که مبتنی بر اطلاعات سرآیند TCP هستند طراحی شده است. حمله کننده امیدوار است که تنها اولین فِرگمنت بتوسط مسیریاب فیلترکننده بررسی شده و بقیه فِرگمنت‌ها عبور نمایند. این حمله را می‌توان با معدوم کردن تمام بسته‌هائی که نوع پروتکل آنها TCP بوده و IP fragment offset آنها برابر ۱ است، خنثی کرد.

دیوارهای آتش تفتیش‌کننده حالت (Stateful Inspection Firewall)

یک فیلتر معمولی بسته‌های دیتا، تصمیمات فیلترینگ را بر مبنای تک‌تک بسته‌ها انجام داده و مضامین هیچ لایه بالاتر را بکار نمی‌گیرد. برای فهم این مطلب که مضامین لایهٔ بالاتر چه بوده و چرا یک فیلتر سنّتی بسته‌های دیتا نسبت به این امر دارای محدودیت است، لازم است که این موضوع کمی شکافته شود. اکثر کاربردهای استاندارد که روی لایهٔ TCP قرار دارند از یک مدل کِلاینت/ سِرور پیروی می‌کنند. بعنوان مثال در پروتکل سادهٔ انتقال نامه‌های الکترونیک(SMTP)، نامه از یک سیستم کِلاینت به یک سیستم سِرور منتقل می‌شود. سیستم کِلاینت پیام‌های e-mail را معمولاً بتوسط کاربر تولید می‌کند. سیستم سِرور پیام‌های پست الکترونیک ورودی را پذیرفته و آنها را در صندوق پستی کاربر قرار می‌دهد. SMTP یک اتصال TCP بین کِلاینت و سِرور برقرار می‌کند که در آن شمارهٔ پورت TCP سِرور که تعیین‌کنندهٔ کاربرد SMTP سِرور است، ۲۵ می‌باشد. شمارهٔ پورت TCP برای SMTP کِلاینت، عددی بین ۱,۰۲۴ و ۱۶,۳۸۳ است که بتوسط پروتکل SMPT کِلاینت تعیین می‌گردد.

در حالت کلی وقتی کاربردی که از TCP استفاده می‌کند تماسی با میزبان دوردست می‌گیرد، یک اتصال TCP ایجاد نموده که در آن شمارهٔ پورت TCP کاربرد دوردست (سِرور)، عددی کمتر از ۱,۰۲۴ بوده و شمارهٔ پورت TCP کاربرد محلی (کِلاینت)، عددی بین ۱,۰۲۴ و ۱۶,۳۸۳ است. اعداد کمتر از ۱,۰۲۴ شمارهٔ پورت‌های «شناخته شده» بوده و بطور دائمی به کاربردهای مشخصی تخصیص می‌یابند (مثلاً ۲۵ برای SMTP). اعداد بین ۱,۰۲۴ و ۱۶,۳۸۳ بطور پویا تولید شده و تنها برای زمان حیات یک اتصال TCP دارای اهمیت‌اند.

یک دیوار آتش فیلترکنندهٔ بسته‌های دیتا بایستی به ترافیک داخل محدوده که دارای شمارهٔ پورت‌های بالا بوده و بر مبنای TCP می‌باشند اجازهٔ عبور دهد. این امر یک نقطهٔ آسیب‌پذیر را ایجاد می‌نماید که ممکن است بتوسط کاربران غیرمجاز مورد سوءاستفاده قرار گیرد.

جدول ۲-۱۱ مثالی از جدول وضعیت اتصالات دیوار آتش از نوع تفتیش‌کنندهٔ حالت [WACK02]

Source Address	Source Port	Destination Address	Destination Port	Connection State
192.168.1.100	1030	210.9.88.29	80	Established
192.168.1.102	1031	216.32.42.123	80	Established
192.168.1.101	1033	173.66.32.122	25	Established
192.168.1.106	1035	177.231.32.12	79	Established
223.43.21.231	1990	192.168.1.6	80	Established
219.22.123.32	2112	192.168.1.6	80	Established
210.99.212.18	3321	192.168.1.6	80	Established
24.102.32.23	1025	192.168.1.6	80	Established
223.212.212	1046	192.168.1.6	80	Established

یک فیلتر بسته‌های دینا از نوع تفتیش‌کننده حالت، ترافیک TCP را با ایجاد یک فهرست از اتصالات TCP خارج از محدودهٔ شبکه همانند جدول ۲-۱۱، با محدودیت‌های بیشتری مواجه می‌سازد. برای هر اتصال برقرار شده، یک قلم به جدول اضافه می‌گردد. فیلتر بسته‌های دینا اکنون به ترافیک ورودی که بمقصد پورت‌های شمارهٔ بالا ارسال شده‌اند در صورتی اجازهٔ عبور می‌دهد که بسته‌ها دارای پروفایل یکی از اقلام موجود در جدول باشند.

دروازهٔ سطح کاربرد (Application - Level Gateway)

یک دروازهٔ سطح کاربرد که یک سِرور پروکسی (proxy) نیز خوانده می‌شود برای ترافیک سطح کاربرد بصورت یک رله عمل می‌کند (شکل ۱-۱۱ب). کاربر با یک برنامهٔ کاربردی TCP/IP همانند Telnet یا FTP با دروازه تماس می‌گیرد و دروازه از کاربر نام میزبان دوری که کاربر تمایل به اتصال با او دارد را سؤال می‌کند. وقتی کاربر پاسخ داده و یک IP معتبر و اطلاعات اعتبارسنجی لازم را ارائه نمود، دروازه با برنامهٔ کاربردی میزبان دور تماس گرفته و سِگمنت‌های TCP شامل داده‌های کاربر را بین دو نقطهٔ انتهائی رله می‌کند. اگر دروازه کُد پروکسی خاصی را فراهم ننماید، سرویس عملیاتی نشده و داده‌ها نمی‌توانند از عرض دیوار آتش عبور کنند. علاوه براین دروازه را می‌توان طوری پیکربندی کرد که تنها از حالت خاصی از کاربردها که مسئول شبکه آنها را قابل پذیرش می‌داند حمایت کرده و برای حالات دیگر از دادن سرویس خودداری نماید.

دروازه‌های سطح کاربرد نسبت به فیلترهای بسته‌های دینا امن‌ترند. بجای تلاش برای مقابله با ترکیبات فوق‌العاده زیاد ممکن که بایستی در سطح TCP و IP مجاز یا ممنوع شناخته شوند، دروازهٔ سطح کاربرد تنها کافی است نسبت به تعداد محدودی از کاربردها دقت نماید. علاوه بر این ثبت و ممیزی تمام ترافیک ورودی در سطح کاربرد، کاری آسان است.

عیب اصلی این نوع دروازه، سربارهٔ پردازش بیشتر در هربار اتصال است. در واقع اتصال بین دو کاربر انتهائی در بین راه شکسته شده و دروازه با قرار گرفتن در این نقطهٔ مَفصلی، بایستی همهٔ ترافیک دو جهت را کنترل نماید.

دروازهٔ سطح مدار (Circuit - Level Gateway)

نوع سوم دیوار آتش، دروازهٔ سطح مدار است (شکل ۱-۱۱ج). این دیوار آتش می‌تواند یک سیستم منفرد بوده و یا می‌تواند عمل خاصی باشد که بتوسط دروازهٔ سطح کاربرد برای کاربردهای معینی انجام شود. یک دروازهٔ سطح کاربرد اجازهٔ یک اتصالTCP سر- به- سر را نمی‌دهد بلکه دروازه‌ای بین دو اتصال TCP ایجاد می‌کندکه یک اتصال بین خودش و استفاده‌کننده از TCP در میزبان داخلی، و اتصال دیگر بین خودش و استفاده‌کننده از TCP در یک میزبان خارجی واقع شده است. زمانی که دو اتصال برقرار گردید دروازه معمولاً سگمنت‌های TCP را، بدون اینکه محتویات آنها را بررسی کند، از یک اتصال به اتصال دیگر رله می‌کند. تدبیر امنیتی بکارگرفته شده در این مورد فقط تعیین آن است که برقراری چه اتصالاتی مجاز هستند.

دروازهٔ سطح مدار معمولاً وقتی مورد استفاده قرار می‌گیرد که مسئول سیستم به کاربران داخل سیستم اعتماد داشته باشد. دروازه را می‌توان طوری پیکربندی کرد که سرویس سطح کاربرد یا سرویس پروکسی در اتصالات داخل محدوده، و عملیات سطح مدار در اتصالات خارج محدوده، را حمایت نماید. در این نوع پیکربندی، دروازه می‌تواند سربارهٔ بررسی داده‌های کاربردی ورودی برای عملیات ممنوع را تحمل کرده ولی درعوض نیازی به بررسی سربارهٔ داده‌های خروجی برای این منظور نداشته باشد.

مثالی از اجرای یک دروازهٔ سطح مدار، بستهٔ نرم‌افزاری SOCKS [KOBL92] است. نسخهٔ پنجم SOCKS در RFC 1928 تعریف شده است. این تعریف چنین است:

> پروتکل توصیف شده در اینجا بدین منظور طراحی شده است تا بستری برای کاربردهای کلاینت/ سرور در هردو بُعد TCP و UDP فراهم کرده تا بطور سهل و امن از سرویس‌های یک دیوار آتش شبکه استفاده نمایند. پروتکل اصولاً یک "shim-layer" بین لایه‌های کاربرد و حمل‌ونقل است و در چنین حالتی سرویس‌های دروازه‌ای سطح شبکه مثل جلوراندن پیام‌های ICMP را فراهم نمی‌سازد.

SOCKS شامل مؤلفه‌های زیر است:

- سرور SOCKS که روی یک دیوار آتش بر پایهٔ UNIX کار می‌کند.
- کتابخانهٔ کلاینت SOCKS که روی میزبان‌های داخلی حفاظت شده بتوسط دیوار آتش کار می‌کند.
- نسخه‌های SOCKS تهیه شده چندین برنامهٔ استاندارد کلاینت همانند FTP و TELNET. پیاده‌سازی پروتکل SOCKS معمولاً شامل دوباره کامپایل کردن و یا تغییر مسیردادن کاربردهای مبتنی بر TCP کلاینت برای استفاده از کپسولی کردن مناسب روتین‌های کتابخانهٔ SOCKS می‌باشد.

وقتی یک کلاینت مبتنی بر TCP بخواهد با شیئی اتصال برقرار کند که تنها از طریق دیوار آتش قابل دست‌یابی است (چنین امری به پیاده‌سازی مربوط می‌شود)، بایستی یک اتصال TCP به پورت مناسب SOCKS بر روی سرور SOCKS برقرار نماید. سرویس SOCKS روی پورت ۱,۰۸۰ TCP قرار دارد. اگر درخواست اتصال پذیرفته شود، کلاینت وارد یک مذاکره برای تعیین روش اعتبارسنجی مورد استفاده شده، از متد انتخاب شده استفاده کرده و آنگاه تقاضای رلهٔ داده‌ها را می‌نماید. سرور SOCKS تقاضا را ارزیابی نموده و اتصال مناسب را یا برقرار و یا به آن پاسخ رد می‌دهد. مبادلات UDP به روش مشابه‌ای بررسی و اجرا می‌گردند. فی‌الواقع یک اتصال TCP باز می‌شود تا اعتبار یک کاربر را چه برای دریافت و چه برای ارسال سگمنت‌های UDP بسنجد و این سگمنت‌ها تا زمانی که اتصال TCP باز است به جلو رانده می‌شوند.

میزبان دژدار (Bastion Host)

یک میزبان دژدار سیستمی است که بتوسط مدیریت دیوار آتش بعنوان یک استحکامات اساسی در امنیت شبکه شناخته شده است. نوعاً میزبان دژدار بعنوان بستری برای دروازه‌های سطح کاربرد و یا دروازه‌های سطح مدار عمل می‌کند. مشخصات عمومی یک میزبان دژدار چنین است:

- بستر سخت‌افزاری میزبان دژدار یک نسخهٔ امن از سیستم عامل خود را اجرا کرده که در نتیجه آن را به یک سیستم معتمد تبدیل می‌کند.
- تنها سرویس‌هائی روی میزبان دژدار نصب می‌شوند که مدیریت شبکه آنها را ضروری می‌داند. اینها شامل کاربردهای پروکسی مانند SMTP, FTP, DNS , Telnet و اعتبارسنجی کاربرند.
- میزبان دژدار ممکن است قبل از اینکه یک کاربر اجازهٔ دست‌یابی به سرویس‌های پروکسی را داشته باشد، نیازمند اعتبارسنجی‌های بیشتری باشد. علاوه برآن هر سرویس پروکسی ممکن است قبل از اینکه به کاربر اجازهٔ استفاده از خود را بدهد، نوعی تصدیق هویت خود را طلب نماید.
- هر پروکسی طوری پیکربندی می‌شود که تنها زیرمجموعه‌ای از مجموعهٔ فرامین کاربردی استاندارد را حمایت نماید.
- هر پروکسی طوری پیکربندی می‌شود که اجازهٔ دست‌یابی به سیستم‌های میزبان خاصی را بدهد. این بدین معنی است که مجموعهٔ فرامین/ قابلیت‌ها تنها به یک زیرمجموعه از سیستم‌های شبکهٔ حفاظت شده محدود گردد.
- هر پروکسی با ثبت کردن همهٔ رخدادهای ترافیکی، اطلاعات کاملی از هر اتصال و زمان هر اتصال را کسب و نگهداری می‌کند. ممیزی اتصالات یک ابزار ضروری برای کشف و خنثی کردن حملات مهاجمین است.
- هر مدول پروکسی یک بستهٔ نرم‌افزاری کوچک است که فقط برای مقاصد امنیت شبکه طراحی شده است. بعلت سادگی نسبی آن، کنترل کردن چنین مدول‌هائی برای کشف موارد نقض امنیت ساده‌تر است. بعنوان مثال، یک کاربرد معمول پستی در UNIX ممکن است شامل بیش از ۲۰,۰۰۰ خط برنامه باشد در حالی‌که یک پروکسی پستی ممکن است کمتر از ۱,۰۰۰ خط برنامه داشته باشد.
- هر پروکسی مستقل از سایر پروکسی‌های میزبان دژدار است. اگر مشکلی در عملکرد هر پروکسی پیش آید، و یا نقطهٔ آسیب‌پذیری در آن کشف گردد، می‌توان بدون اینکه عملیات کاربردی سایر پروکسی‌ها مختل شوند آن را از روی سیستم برداشت. همچنین اگر جمع کاربران برای سرویس جدیدی نیاز به حمایت داشته باشند، مدیر شبکه می‌تواند بسهولت پروکسی مورد نیاز را روی میزبان دژدار نصب نماید.
- یک پروکسی بجز در ابتدا، و برای خواندن فایل پیکربندی خود، معمولاً نیازی به دسترسی به دیسک ندارد. این امر کار را برای یک مهاجم که بخواهد sniffer اسب تروا و یا سایر فایل‌های خطرناک را روی میزبان دژدار نصب کند دشوار می‌سازد.
- هر پروکسی بصورت یک کاربر فاقد امتیاز، در یک شاخهٔ خصوصی و امن روی میزبان دژدار اجرا می‌شود.

پیکربندی‌های دیوار آتش

علاوه بر استفاده از یک پیکربندی ساده شامل یک سیستم منفرد، همانند مسیریاب فیلترکنندهٔ بسته‌ها و یا یک دروازهٔ منفرد (شکل ۱-۱۱)، پیکربندی‌های پیچیده‌تری نیز ممکن بوده و در واقع معمول‌ترند. شکل ۲-۱۱ سه نوع پیکربندی متداول دیوار آتش را نشان می‌دهد. هریک از آنها را بنوبت بررسی می‌کنیم.

در پیکربندی **دیوارآتش حفاظت‌کنندهٔ میزبان با یک دژ** (screened host firewall, single-homed bastion) (شکل ۲-۱۱الف)، دیوار آتش شامل دو سیستم است که یک مسیریاب فیلترکنندهٔ بسته‌ها و یک میزبان دژدار می‌باشند. معمولاً مسیریاب طوری پیکربندی می‌شودکه:

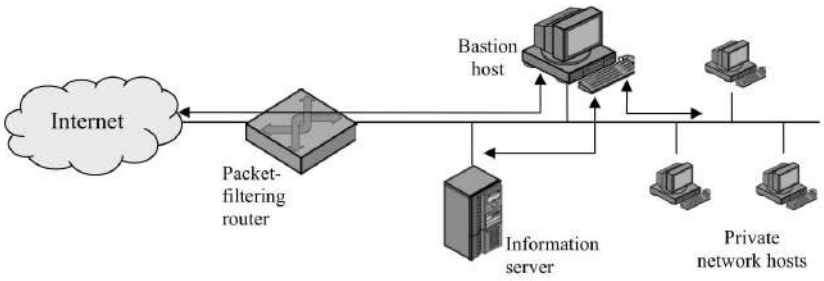
۱- برای ترافیکی که از سمت اینترنت وارد می‌شود، تنها بسته‌های IP که مقصد آنها میزبان دژدار است اجازهٔ ورود دارند.
۲- برای ترافیک خروجی از شبکهٔ داخلی، تنها بسته‌های IP که از میزبان دژدار صادر می‌شوند اجازهٔ خروج دارند.

میزبان دژدار عملیات احراز هویت و پروکسی را انجام می‌دهد. این پیکربندی دارای امنیت بیشتری از یک مسیریاب فیلترکنندهٔ بسته‌ها و یا یک دروازهٔ سطح کاربرد بوده و علت این امر دو چیز است. اول اینکه این پیکربندی هم فیلترینگ سطح بسته و هم فیلترینگ سطح کاربرد را انجام داده و قابلیت انعطاف قابل ملاحظه‌ای در تعریف خط‌مشی‌های امنیتی بوجود می‌آورد. دوم اینکه یک مهاجم قبل از اینکه بتواند امنیت شبکهٔ داخلی را به مخاطره اندازد، قاعدتاً بایستی از دو سیستم مجزا عبور کند.

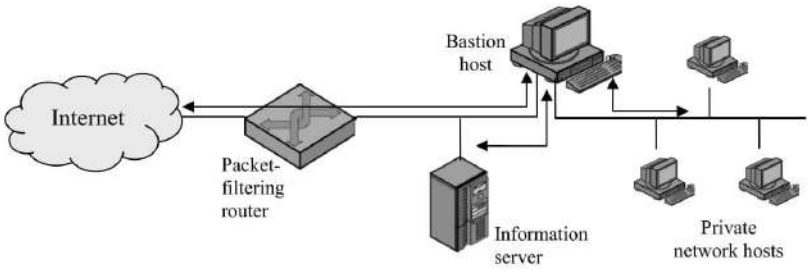
این پیکربندی همچنین انعطافی را در فراهم آوردن دسترسی مستقیم به اینترنت بوجود می‌آورد. بعنوان مثال، شبکهٔ داخلی ممکن است شامل یک سِرور اطلاعاتی عمومی همانند یک سِرور وِب بوده باشد که برای آن نیازی به اِعمال محدودیت‌های امنیتی سطح بالا نمی‌باشد. در چنین موردی مسیریاب را میتوان طوری پیکربندی کرد که ترافیک بین سِرور اطلاعاتی و اینترنت را مستقیماً عبور دهد.

در پیکربندی دیوار آتش با یک دژ که در بالا ذکر گردید، اگر مسیریاب فیلترکنندهٔ بسته‌ها کاملاً در اختیار مهاجم قرار گیرد ترافیک خواهد توانست مستقیماً از درون مسیریاب، بین اینترنت و سایر میزبان‌های متصل به شبکهٔ خصوصی برقرار گردد. پیکربندی **دیوارآتش حفاظت‌کنندهٔ میزبان با دژ دوگانه** (screened host firewall, dual-homed bastion) بطور فیزیکی از چنین عمل ضدامنیتی جلوگیری می‌کند (شکل ۲-۱۱ب). محاسن لایهٔ دوگانهٔ امنیتی که در پیکربندی قبل وجود داشت در اینجا نیز بجای خود باقی است. بازهم سِرور اطلاعاتی و یا سایر میزبان‌ها را میتوان مجاز به ارتباط مستقیم با مسیریاب نمود، البته اگر این امر در راستای خط‌مشی‌های امنیتی سیستم باشد.

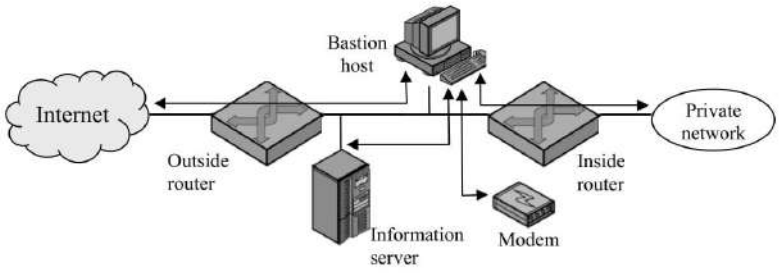
پیکربندی **دیوارآتش حفاظت‌کنندهٔ زیرشبکه** (screened subnet firewall) امن‌ترین نوع پیکربندی است (شکل ۲-۱۱ج). در این پیکربندی، از دو مسیریاب فیلترکنندهٔ بسته‌ها استفاده می‌شود که یکی بین میزبان دژدار و اینترنت و دیگری بین میزبان دژدار و شبکهٔ داخلی قرار دارد. این پیکربندی یک زیرشبکهٔ ایزوله را ایجاد می‌کند که ممکن است به سادگی شامل یک میزبان دژدار بوده ولی می‌تواند شامل یک یا چند سِرور اطلاعاتی و مُودم برای تماس تلفنی با اینترنت نیز باشد. معمولاً هم اینترنت و هم شبکهٔ داخلی به میزبان‌های روی زیرشبکهٔ حفاظت‌شده دسترسی داشته ولی ترافیک درعرض شبکهٔ حفاظت‌شده مسدود می‌شود. این پیکربندی محاسن متعددی دارد:



(الف) سیستم دیوار آتش Screened host (single-homed bastion host)



(ب) سیستم دیوار آتش Screened host (dual-homed bastion host)



(ج) سیستم دیوار آتش Screened-subnet

شکل ۲-۱۱ پیکربندی‌های دیوار آتش

- در اینجا از سه سدّ دفاعی در مقابل مهاجمین استفاده می‌شود.
- مسیریاب خارجی تنها حضور زیرشبکهٔ حفاظت‌شده را به اینترنت اعلام می‌دارد و بنابراین شبکهٔ داخلی برای اینترنت مرئی نمی‌باشد.
- بطریق مشابه، مسیریاب داخلی تنها حضور زیرشبکهٔ حفاظت‌شده را به شبکهٔ داخلی اطلاع می‌دهد و بنابراین سیستم‌های داخل شبکه نمی‌توانند ارتباط مستقیمی با اینترنت داشته باشند.

۱۱-۲ سیستم‌های معتمد (TRUSTED SYSTEMS)

یک روش برای بالابردن قابلیت دفاعی یک سیستم در مقابل مهاجمین و برنامه‌های بداندیش، پیاده‌سازی تکنولوژی سیستم معتمد است. در این بخش نگاه مختصری به این مقوله می‌اندازیم. مطالعه را با نگاهی به برخی مفاهیم اساسی کنترل دست‌یابی به داده‌ها آغاز می‌کنیم.

کنترل دست‌یابی به داده‌ها

پس از یک ورود موفق به سیستم، عملاً به کاربر اجازه داده شده است تا به یک یا چند میزبان و تعدادی برنامه‌های کاربردی دست یابد. این امر معمولاً برای سیستمی که شامل داده‌های گران‌قیمت در پایگاه داده خود می‌باشد، کافی نیست. از طریق روش‌های کنترل دست‌یابی کاربر، یک کاربر می‌تواند به سیستم معرفی گردد. در رابطه با هر کاربر، میتوان یک پروفایل از عملیات مجاز و دسترسی‌های ممکن به فایل‌ها را تعریف کرد. سیستم عامل آنگاه خواهد توانست بر اساس پروفایل کاربر، قوانینی را به نحوهٔ عملیات او اعمال نماید. با وجود این، سیستم مدیریت پایگاه داده بایستی دست‌یابی به رکوردهای بخصوص و حتی بخشی از رکوردها را کنترل کند. مثلاً ممکن است برای همهٔ افراد یک سازمان دست‌یابی به فایلی که نام پرسنل آن سازمان در آن ضبط شده است ممکن بوده ولی تنها افراد بخصوصی بتوانند به اطلاعات حقوق و دستمزد آنها دست یابند. این مقدار، بیش از یک سطح از جزئیات را دربر می‌گیرد. در حالی که سیستم عامل ممکن است به یک کاربر اجازهٔ دست‌یابی به یک فایل و یا استفاده از یک برنامه را اعطا نماید، که قاعدتاً پس از آن کنترل‌های امنیتی دیگری اعمال نخواهد شد، سیستم مدیریت پایگاه داده بایستی در مورد هر بار تلاش برای دست‌یابی فرد تصمیمی اتخاذ نماید. تصمیم این مدیریت نه تنها به مشخصات کاربر وابسته بوده بلکه بستگی به این دارد که او به کدام بخش دیتا علاقه داشته و یا حتی اینکه او قبلاً به کدام بخش دست یافته است.

یک مدل عمومی کنترل دست‌یابی که بتوسط سیستم مدیریت پایگاه داده و یا مدیریت فایل به اجرا گذاشته می‌شود، **ماتریس دست‌یابی (access matrix)** (شکل ۳-۱۱الف) است. مؤلفه‌های اساسی مدل بقرار زیراند:

- **سوژه (subject)**: واحدی که قادر به دست‌یابی به موضوعات است. معمولاً مفهوم سوژه معادل یک پردازش است. هر کاربر یا برنامهٔ کاربردی در واقع با کمک یک سری عملیات که نمایشگر آن کاربر یا برنامه کاربردی هستند، به موضوع دست می‌یابد.
- **موضوع (object)**: هر چیزی که دست‌یابی به آن باید کنترل شود. مثال‌های این مورد شامل فایل‌ها یا بخش‌هائی از یک فایل، برنامه‌ها و قسمت‌هائی از حافظه می‌باشند.
- **حق دست‌یابی**: روشی است که بتوسط آن یک سوژه به یک موضوع دسترسی پیدا می‌کند. خواندن، نوشتن و اجراکردن یک برنامه مثال‌هائی از این دست می‌باشند.

یک محور ماتریس شامل سوژه‌های شناخته شده‌ای است که ممکن است برای دست‌یابی به موضوعی تلاش نمایند. نوعاً این لیست شامل کاربران منفرد و یا گروه کاربران است ولی دست‌یابی را میتوان برای ترمینال‌ها، میزبان‌ها و یا برنامه‌ها نیز بجای کاربران کنترل نمود. محور دیگر موضوعاتی که می‌توان به آنها دست یافت را مشخص می‌سازد. در جزئی‌ترین سطح دست‌یابی، موضوعات می‌توانند میدان‌های منفرد دیتا باشند. گروه‌های متشکل‌تر مانند رکوردها، فایل‌ها و یا حتی کل پایگاه داده نیز می‌تواند موضوع قابل دست‌یابی را تشکیل دهند. در محل تلاقی هر سطر و هر ستون ماتریس یا جدول، حق دست‌یابی مجاز از طرف آن کاربر به آن موضوع تعیین شده است.

در عمل یک ماتریس دست‌یابی دارای خانه‌های خالی زیادی بوده و بنابراین برای اجرای عملیاتی آن به یکی از دو روش زیر عمل می‌گردد. ماتریس را می‌توان به مجموعهٔ ستون‌های آن تجزیه نمود که در نتیجه **لیست‌های کنترل دست‌یابی (access control lists)** (شکل ۳-۱۱ب) ایجاد می‌شود. بنابراین یک لیست کنترل دست‌یابی، برای هر موضوع، کاربران را لیست نموده و حقوق دست‌یابی آنان را مشخص می‌سازد. لیست کنترل دست‌یابی ممکن است شامل یک پیش‌فرض برای عموم باشد. این امر به کاربرانی که نام آنها در لیست ذکر نشده است اجازه می‌دهد تا به مجموعه‌ای از حقوق از پیش تعیین شده برسند. لیست ممکن است شامل اسامی کاربران منفرد و یا گروه‌های کاربران باشد.

تجزیهٔ لیست به ردیف‌ها، **بلیت‌های توانائی (capability tickets)** را ایجاد می‌کند (شکل۳-۱۱ج). یک بلیت از این نوع، موضوعات مطرح و عملیات مجاز هر کاربر روی آنها را تعیین می‌کند. هر کاربر تعدادی بلیت دارد و ممکن است مجاز باشد تا آنها را به دیگران نیز قرض داده و یا اهدا کند. چون این بلیت‌ها را می‌توان در جاهای مختلف سیستم خرج کرد، آنها می‌توانند مشکلات امنیتی جدی‌تری را نسبت به لیست‌های کنترل دست‌یابی ایجاد نمایند. علی‌الخصوص، بلیت بایستی غیرقابل جعل باشد. یکی از راه‌های اجرائی این است که از سیستم عاملی استفاده نمود که تمام بلیت‌ها را، بجای نزد کاربر، نزد خود نگاه دارد. این بلیت‌ها بایستی در ناحیه‌ای از حافظه نگهداری شوند که بتوسط کاربران قابل دسترس نباشد.

مفهوم سیستم‌های معتمد

بیشتر آنچه تا بحال مورد بحث قرار گرفته است، مربوط به حفاظت یک پیام و یا مقوله‌ای نظیر آن در برابر حملهٔ فعال و یا غیرفعال یک کاربر مشخص بوده است. نیاز متفاوت دیگری که بطور گسترده مورد استفاده است، حفاظت داده‌ها یا منابع بر اساس سطوح متفاوت امنیتی است. این امر را معمولاً در ارتش، که در آنجا اطلاعات به انواع طبقه‌بندی نشده (U)، محرمانه (C)، سری (S)، فوق سری (TS) و بالاتر از آن دسته‌بندی شده‌اند، می‌توان مشاهده کرد. این مفهوم در سایر زمینه‌ها نیز می‌تواند بکار گرفته شده و اطلاعات را به دسته‌های متفاوت تقسیم کرد. برای هر کاربر نیز می‌توان دست‌یابی مجاز به دسته‌ای از اطلاعات و ممنوعیت دست‌یابی به بخش‌های دیگر را تعیین نمود. بعنوان مثال، بالاترین سطح امنیت ممکن است متعلق به اسناد و داده‌های مربوط به سیاست‌های استراتژیک یک سازمان باشدکه فقط بایستی در دسترس هیئت مدیرهٔ سازمان و عوامل او قرار گیرند. در ردهٔ بعدی ممکن است اسناد حساس مالی و اطلاعات مربوط به پرسنل قرار داشته باشدکه فقط بایستی در دسترس پرسنل مدیریت امور اداری و مالی و عوامل آنها قرار گیرند

وقتی دسته‌ها و یا سطوح متفاوت داده‌ها مطرح می‌شوند، نیاز امنیتی آنها را **امنیت چند سطحه (multilevel security)** خوانند. بیان عمومی مساله امنیت چندسطحه بدین ترتیب است که یک کاربر سطح بالاتر ممکن نیست اطلاعات را به یک کاربر سطح پائین‌تر و یا سطح غیرقابل مقایسه با خود بدهد مگر اینکه این امر با دقت کامل از ارادهٔ یک کاربر معتبر ناشی شده باشد. برای اجرای این هدف، این نیاز به دو بخش تقسیم شده و بسادگی بیان گردیده است. یک سیستم امن چند سطحه بایستی دو قانون زیر را رعایت نماید:

	Program1	...	Segment A	Segment B
Process 1	Read Execute		Read Write	
Process 2				Read
.				
.				
.				

(الف) ماتریس دست‌یابی

Access control list for Program 1: Process1 (Read,Execute)
Access control list for segment A: Process1 (Read,Write)
Access control list for Segment B: Process2 (Read)

(ب) لیست کنترل دست‌یابی

Capability list for Process1: Program1 (Read,Execute) Segment A (Read,Write)
Capability list for Process2: Segment B (Read)

(ج) لیست توانائی‌ها

شکل ۳-۱۱ ساختار کنترل دست‌یابی

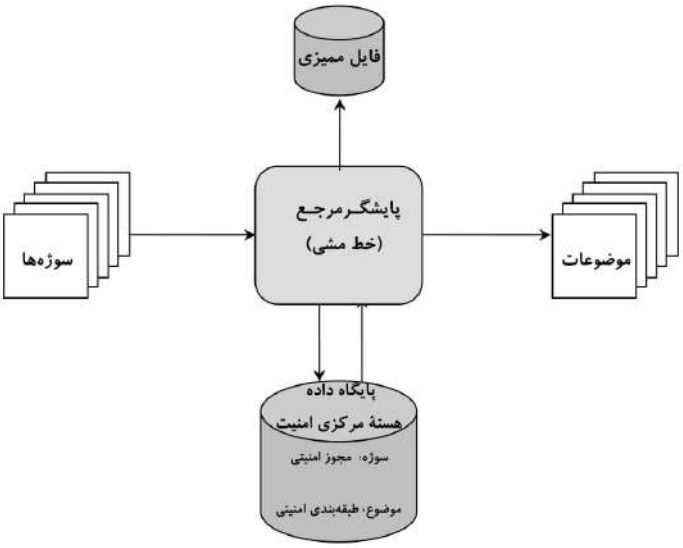
- **نخواندن سطح بالاتر**: یک سوژه تنها می‌تواند موضوعی را بخواند که در سطح او و یا پائین‌تر از سطح او قرار داشته باشد. این امر را معمولاً **خاصیت امنیتی ساده (Simple Security Property)** گویند.
- **ننوشتن در سطح پائین‌تر**: یک سوژه تنها می‌تواند در موضوعی دخل تصرف نماید که در سطح او و یا بالاتر از سطح او قرار داشته باشد. در متون امنیتی این امر را ***-Property** (star property) گویند.

اگر این دو قاعده بطور صحیح بکار گرفته شوند، امنیت چند سطحه ایجاد خواهد شد. برای یک سیستم پردازش داده‌ها، روشی که بکار گرفته شده و موضوع بسیاری از تحقیقات بوده است، استفاده از *پایشگر مرجع* (*reference monitor*) است. این روش برخورد با مسأله در شکل ۴-۱۱ نشان داده شده است. پایشگر مرجع یک واحد کنترل‌کنندهٔ در سخت‌افزار و سیستم عامل یک کامپیوتر بوده و نحوهٔ دست‌یابی کاربران به موضوعات را بر مبنای پارامترهای امنیتی سوژه و موضوع قانون‌مند می‌نماید. پایشگر مرجع به فایلی به نام *پایگاه داده هستهٔ مرکزی امنیت* (*security kernel database*) دسترسی داشته که در آن امتیازات دست‌یابی هر سوژه و سطوح حفاظتی هر موضوع لیست شده است. پایشگر مرجع قواعد امنیتی (نخواندن سطح بالاتر و ننوشتن در سطح پائین‌تر) را به اجرا گذاشته و دارای خصوصیات زیر است:

- **وساطت کامل**: قواعد امنیتی در هربار دست‌یابی، و نه مثلاً فقط زمان بازشدن یک فایل، به اجرا گذاشته می‌شوند.
- **ایزولاسیون**: پایشگر مرجع و پایگاه داده در مقابل دستکاری‌های غیرمسئولانه حفاظت می‌شوند.
- **قابلیت تائید**: صحت عمل پایشگر مرجع بایستی قابل اثبات باشد. یعنی بایستی بتوان از نظر ریاضی ثابت نمود که پایشگر مرجع قواعد امنیتی را به اجرا گذاشته و دخالت کامل و ایزولاسیون را بوجود می‌آورد.

شرایط بالا بسیار محکم‌اند. نیاز به وساطت کامل بدین معنی است که هر دست‌یابی به دیتا در حافظهٔ اصلی و روی دیسک و نوار، بایستی با واسطه باشد. اگر قرار باشد که این امر صرفاً از طریق نرم‌افزار انجام شود، پنالتی عملکرد قابل تحمل نخواهد بود و بنابراین حداقل بخشی از راه حل مساله را بایستی در سخت‌افزار اجرا نمود. نیاز به ایزولاسیون بدین معنی است که یک مهاجم هرچقدر زرنگ باشد نبایستی بتواند منطق پایشگر مرجع و یا محتویات پایگاه داده هستهٔ مرکزی امنیت را تغییر دهد. بالاخره نیاز به اثبات ریاضی مطلب برای سیستمی پیچیده همانند یک کامپیوتر با قابلیت‌های عام، کاری فوق تصور است. سیستمی که بتواند چنین قابلیت‌هائی را فراهم آورد یک **سیستم معتمد (trusted system)** خوانده می‌شود.

یک عنصر نهائی که در شکل ۴-۱۱ نشان داده شده است، یک فایل ممیزی (audit) است. پیشامدهای مهم امنیتی همانند نقض مقررات امنیتی و همچنین تغییرات مجاز در پایگاه داده هستهٔ مرکزی امنیت، در فایل ممیزی ذخیره می‌شود.



شکل ۴-۱۱ مفهوم پایشگر مرجع

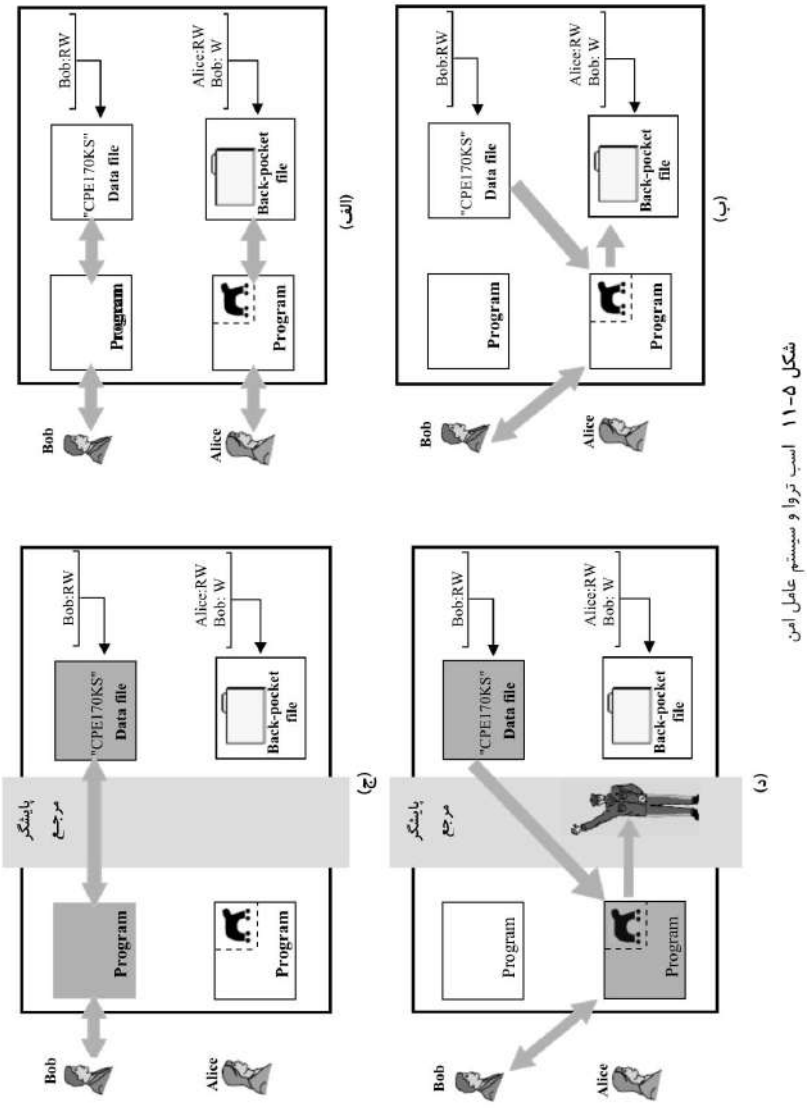
در تلاش برای تأمین نیازهای خود و همچنین برای ایجاد یک روش عام، وزارت دفاع امریکا در سال ۱۹۸۱ مرکز امنیت کامپیوتر(Computer Security Center) در آژانس امنیت ملی (NSA) را با هدف تشویق به ایجاد گستردهٔ سیستم‌های کامپیوتری معتمد تأسیس نمود. این هدف، از طریق خلق یک برنامهٔ ارزیابی محصولات تجاری دنبال گردید. بطور خلاصه، این مرکز تلاش میکند تا محصولات تجاری موجود را از نظر برآوردن لازمه‌های امنیتی فوق‌الذکر ارزیابی نماید. مرکز، محصولات ارزیابی شده را برحسب نوع مشخصه‌های امنیتی فراهم آمده طبقه‌بندی می‌نماید. این ارزیابی‌ها در راستای اهداف وزارت دفاع امریکا بوده ولی برای عموم هم انتشار یافته و در دسترس می‌باشند. بنابراین این اسناد می‌توانند برای خرید محصولات تجاری موجود در بازار مورد استفادهٔ مشتریان تجاری نیز قرار گیرند.

دفاع اسب تروا (Trojan Horse)

یکی از راه‌های ایجاد امنیت در مقابل حملات اسب تروا، استفاده از یک سیستم عامل امن و معتمد است. شکل ۵-۱۱ مثالی از این دست را نشان می‌دهد. در این مورد از یک اسب تروا برای دور زدن مکانیسم امنیتی استانداردی که بتوسط بیشتر مدیریت‌های فایل و سیستم‌های عامل بکار گرفته می‌شود، یعنی لیست کنترل دست‌یابی، استفاده شده است. در این مثال کاربری بنام Bob از طریق یک برنامه، به یک فایل داده که شامل دنباله‌های بسیار مهم و حساس "CPE170KS" است دسترسی دارد. Bob فایل را چنان خلق کرده است که اجازهٔ نوشتن/خواندن را تنها به برنامه‌هائی که بتوسط خودش اجرا می‌شود می‌دهد، یعنی تنها پردازش‌هائی که متعلق به Bob هستند ممکن است به فایل دست‌یابند.

حملهٔ اسب تروا زمانی آغاز می‌شود که یک کاربر مهاجم بنام Alice با دست‌یابی مجاز به سیستم راه یافته و یک برنامه اسب تروا به همراه یک فایل خصوصی که قرار است در حمله بعنوان جیب مخفی بکار رود را در سیستم نصب می‌کند. Alice برای این فایل بخودش اجازه نوشتن/خواندن را داده، در حالیکه برای Bob تنها اجازهٔ نوشتن را می‌دهد (شکل ۵-۱۱الف). حال Alice، Bob را وسوسه کرده تا برنامهٔ اسب تروا را بکار گیرد و شاید این کار را با تبلیغ اینکه این برنامه یک ابزار کاربردی خوب است انجام دهد. وقتی برنامه تشخیص می‌دهد که دارد بتوسط Bob اجرا می‌شود، دنبالهٔ کاراکترهای سرّی را از فایل Bob خوانده و آن را در فایل جیب مخفی Alice کپی می‌کند (شکل ۵-۱۱ب). هردو عملیات خواندن و نوشتن از محدودیت‌هائی که بتوسط لیست دست‌یابی تعیین شده است تبعیت می‌کنند. تنها کاری که برای Alice باقی مانده است این است که در زمان دیگری به فایل Bob دست یافته و دنباله را بخواند.

حال به استفاده از یک سیستم عامل امن در این سناریو توجه کنید (شکل ۵-۱۱ج). سطوح امنیتی در هنگام اتصال به سیستم به سوژه‌ها تخصیص می‌یابند و مبنای عمل، مواردی همچون نوع ترمینالی که بکار گرفته می‌شود و مشخصات کاربر بر اساس ID و کلمه عبور او است. در این مثال دو سطح امنیتی حساس و عمومی وجود دارد که بنحوی سطح‌بندی شده‌اند که سطح حساس بالاتر از سطح عمومی قرار دارد. به پردازش‌های مربوط به Bob و فایل دیتای او سطح امنیتی حساس تخصیص داده شده است. فایل Alice و پردازش‌های مربوط به او در سطح عمومی که پائین‌تر است قرار دارند. اگر Bob برنامهٔ بداندیش اسب تروا را بکار گیرد (شکل ۵-۱۱د)، آن برنامه سطح امنیتی Bob را استعلام می‌کند و بنابراین او قادر خواهد بود بر اساس خاصیت امنیتی ساده، دنبالهٔ کاراکترهای مهم را مشاهده نماید. وقتی برنامه برای ذخیرهٔ این دنباله در یک فایل عمومی تلاش کند (فایل جیب مخفی)، Property-* نقض گردیده و پایشگر مرجع جلوی این کار را می‌گیرد. بنابراین تلاش برای نوشتن دنباله در فایل جیب مخفی، حتی اگر لیست کنترل دست‌یابی آن را مجاز بشمارد، بی‌نتیجه می‌ماند. حاصل این است که خط‌مشی امنیتی بر مکانیسم کنترل دست‌یابی، توفق می‌یابد.



شکل ۵-۱۱ اسب تروا و سیستم عامل امن

۳-۱۱ معیارهای مشترک برای ارزیابی امنیت تکنولوژی اطلاعات

کارهای انجام شده بتوسط آژانس امنیت ملّی و سایر سازمان‌های دولتی امریکا برای تعیین لازمه‌ها و معیارهای ارزیابی سیستم‌های معتمد، همگام با کارهای مشابه در سایر کشورها بوده است. معیارهای مشترک(Common Criteria(CC برای تکنولوژی اطلاعات و ارزیابی امنیت، یک ابتکار بین‌المللی از سوی بخش‌های استاندارد کشورهای مختلف برای تعیین لازمه‌های امنیتی و تعریف معیارهای ارزیابی بوده است.

لازمه‌ها

CC یک مجموعه مشترک از لازمه‌های امنیتی برای استفاده در ارزیابی‌ها را تعریف می‌کند. اصطلاح **هدف ارزیابی** target of evaluation (TOE) به بخشی از محصول و یا سیستم که تحت ارزیابی قرار می‌گیرد اشاره می‌کند. لازمه‌ها در دو طبقه قرار دارند:

- **لازمه‌های عملیاتی**: اینها رفتارهای مطلوب امنیتی را تعریف می‌کنند. اسناد CC مجموعه‌ای از مؤلفه‌های عملیاتی امنیتی را ایجاد می‌نمایندکه روش استانداردی برای بیان لازمه‌های عملیاتی امنیت در یک TOE است.
- **لازمه‌های اطمینان‌بخشی**: اینها مبانی کسب اطمینان از این است که معیارهای امنیت مؤثر بوده و بطرز صحیحی پیاده‌سازی شده است. اسناد CC مجموعه‌ای از مؤلفه‌های اطمینان‌بخش را ایجاد می‌نمایند که روش استانداردی برای بیان لازمه‌های اطمینان‌بخشی در یک TOE است.

هم لازمه‌های عملیاتی و هم لازمه‌های اطمینان‌بخشی در طبقات مختلف سازمان‌دهی می‌شوند: یک **طبقه** عبارت از مجموعه‌ای از لازمه‌هاست که دارای تمرکز و یا منظور خاصی است. جداول ۳-۱۱ و ۴-۱۱ بطور مختصر طبقات مربوط به لازمه‌های عملیاتی و لازمه‌های اطمینان‌بخشی را تعریف می‌کنند. هریک از این طبقات شامل تعدادی خانواده‌اند. لازمه‌های درون هر **خانواده** اهداف امنیتی خاصی داشته ولی از جهت تأکید یا قوت با هم متفاوت‌اند. بعنوان مثال طبقه ممیزی شامل شش خانواده است که مربوط به جنبه‌های مختلف ممیزی است (مثل تولید اطلاعات ممیزی، تحلیل ممیزی و ذخیره کردن پیشامدهای مرتبط با ممیزی). هر خانواده بنوبهٔ خود شامل یک یا دو مؤلفه است. یک **مؤلفه**، مجموعه‌ای مشخص از لازمه‌های امنیتی را توصیف کرده و کوچک‌ترین مجموعهٔ قابل انتخاب از لازمه‌های امنیتی برای شمول در ساختار تعریف شده در CC است.

برای مثال، طبقهٔ پشتیبانی رمزنگاری از لازمه‌های عملیاتی، شامل دو خانواده است: مدیریت کلید رمزنگاری و تابع رمزنگاری. در زیر خانوادهٔ مدیریت کلید رمزنگاری، چهار مؤلفه قرار دارد که برای تعیین الگوریتم تولید کلید و اندازهٔ کلید، روش توزیع کلید، روش دست‌یابی به کلید و روش نابودی کلید بکار می‌رود. برای هر مؤلفه، ممکن است یک استاندارد تعیین نمود. در تحت خانوادهٔ تابع رمزنگاری تنها یک مؤلفهٔ منفرد قرار دارد که یک الگوریتم و اندازهٔ کلید آن را برحسب یک استاندارد مشخص می‌سازد.

مجموعه‌های مؤلفه‌های عملیاتی و اطمینان‌بخشی ممکن است با هم گرد آمده تا بسته‌های قابل استفاده مجدد جهت برآورده نمودن اهداف مشخصی را تشکیل دهند. مثالی از چنین بسته‌هائی می‌تواند مجموعه‌ای از مؤلفه‌های عملیاتی لازم برای کنترل‌های دست‌یابی منصفانه (Discretionary Access Control) باشد.

جدول ۳-۱۱ لازمه‌های عملیاتی امنیت CC

طبقه	توصیف
ممیزی	شامل شناسائی، ثبت، ذخیره‌سازی و تحلیل اطلاعات مرتبط با فعالیت‌های امنیتی است. سوابق ممیزی بتوسط این عملیات تولیدشده و می‌توانند از نظر ارتباط با مسائل امنیت مورد مطالعه قرار گیرند.
پشتیبانی رمزنگاری	زمانی که TOE توابع رمزنگاری را پیاده‌سازی می‌کند، مورد استفاده قرار می‌گیرند. اینها برای مثال ممکن است برای حمایت از ارتباطات، شناسائی و تصدیق هویت، یا جداسازی داده‌ها بکار روند.
ارتباطات	دو خانواده که مرتبط با عدم انکار بتوسط خلق‌کنندهٔ دیتا و دریافت‌کنندهٔ دیتا است را ایجاد می‌کند.
حفاظت داده‌های کاربر	لازمه‌های مرتبط با حفاظت از داده‌های کاربر در TOE درخلال ورود، خروج و ذخیره‌سازی را برآورده نموده و علاوه بر این مشخصه‌های امنیتی مرتبط با دیتای کاربر را تعیین می‌کند.
شناسائی و احراز هویت	شناسائی بدون ابهام کاربران معتبر و ارتباط صحیح مشخصه‌های امنیتی با کاربران و سوژه‌ها را اطمینان می‌دهد.
مدیریت امنیت	مدیریت مشخصه‌های امنیتی، داده‌ها و توابع را تعیین می‌کند.
سرّی بودن	یک کاربر را با حفاظت در برابر کشف و سوءاستفاده از هویت او از سوی دیگر کاربران تجهیز می‌کند.
حفاظت از عملیات امنیتی TOE	بر حفاظت از داده‌های TSF (عملیات امنیتی TOE)، و نه داده‌های کاربر، تمرکز دارد. این طبقه مرتبط با صحت و مدیریت مکانیسم‌ها و داده‌های TOE است.
بهره گیری از منابع	در دسترس بودن منابع لازم همچون توانائی پردازش و ظرفیت ذخیره‌سازی را پشتیبانی می‌کند. شامل لازمه‌های مربوط به تحمل خطا، اولویت سرویس و تخصیص منابع است.
دست‌یابی به TOE	لازمه‌های عملیاتی، علاوه بر آنهائی که برای شناسائی و احراز هویت تعیین شده‌اند، برای کنترل و برقراری یک اجلاس کاربر را تأمین می‌نماید. شامل لازمه‌های دست‌یابی TOE به اموری همچون محدودکردن تعداد و کیفیت اجلاس‌های کاربر، نمایش تاریخچهٔ دست‌یابی و دستکاری پارامترهای دست‌یابی است.
مسیرها/کانال‌های معتمد	در رابطه با مسیرهای ارتباطی مورد اعتماد بین کاربرها و TSFها، و بین خود TSFها عمل می‌کند.

پروفایل‌ها و هدف‌ها

CC همچنین دو نوع از اسناد که می‌توانند بر حسب لازمه‌های تعریف شده در CC تولید شوند، را تعریف می‌کند.

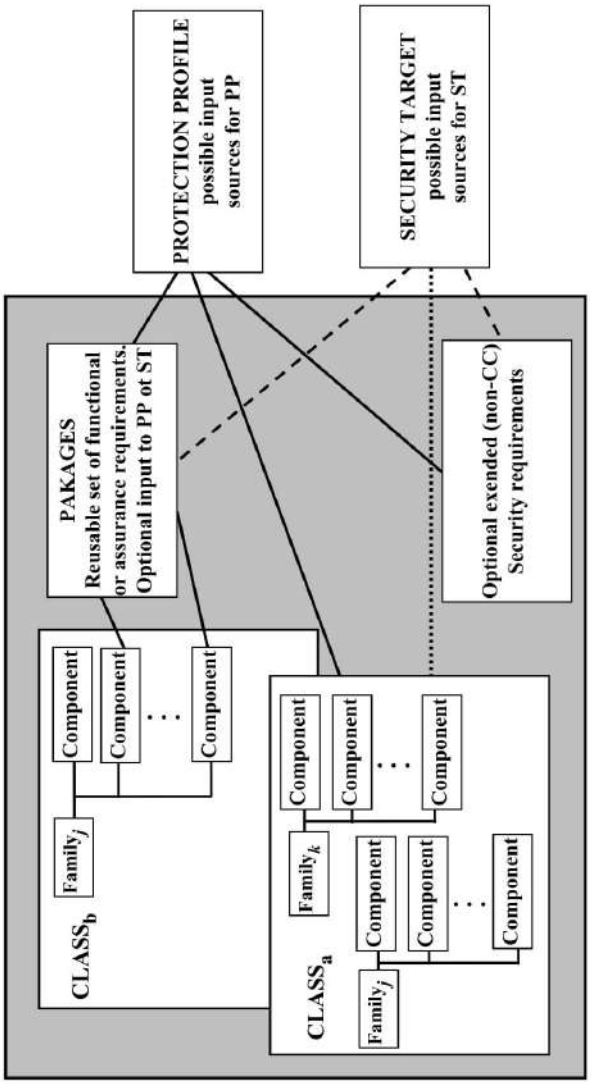
- **پروفایل‌های حفاظتی(PP)**: یک مجموعه از لازمه‌ها و اهداف امنیتی مستقل از پیاده‌سازی برای یک گروه از محصولات و یا سیستم‌ها که نیازهای مشابهی در زمینهٔ امنیت IT دارند را تعریف می‌کند. هدف از یک PP این است که قابل استفادهٔ مجدد بوده و لازمه‌هائی که از نظر برآوردن اهداف مشخصی مفید و مؤثر بنظر می‌رسند را تعریف کند. هدف از PP، پشتیبانی از تعریف استانداردهای عملیاتی و کمک به فرموله کردن مشخصات بوده است. PP منعکس‌کنندهٔ نیازهای امنیتی کاربر است.

جدول ۴-۱۱ لازمه‌های اطمینان بخشی امنیت CC

طبقه	توصیف
مدیریت پیکربندی	نیازمند به این است که سلامت TOE بطرز مناسبی حفظ گردد. علی‌الخصوص مدیریت پیکربندی این اطمینان را ایجاد می‌کند که TOE و اسناد استفاده شده برای ارزیابی همان‌هائی هستند که برای توزیع آماده شده‌اند.
تحویل و عملیات	مرتبط با معیارها، رَویه‌ها و استانداردها برای تحویل امن، نصب و استفادهٔ عملیاتی از TOE برای اطمینان‌بخشی از این مطلب است که حفاظت امنیتی ایجادشده بتوسط TOE در خلال این پیشامدها نقض نگردد.
توسعه	مرتبط با پالایش فرم TSF از فرم تعریف شده در ST تا مرحلهٔ پیاده‌سازی بوده و یک نگاشت از لازمه‌های امنیتی به پائین‌ترین سطح ارائه است.
اسناد راهبردی	مرتبط با استفادهٔ عملیاتی امن از TOE بتوسط کاربران و مدیران است.
پشتیبانی از دورهٔ حیات	مرتبط با دورهٔ حیات TOE بوده و شامل تعریف دورهٔ حیات، ابزارها و تکنیک‌ها، امنیت محیط پیاده‌سازی و رفع اشکالاتی است که بتوسط کاربران در TOE یافت می‌شود.
آزمایشات	مرتبط با نمایش این امر است که TOE لازمه‌های عملیاتی خود را اجرا می‌کند. خانواده‌ها در این کلاس مرتبط با اندازه‌گیری پوشش و عمق تست‌های اجرائی و تست‌های مستقل دیگرند.
ارزیابی آسیب‌پذیری	لازمه‌هائی را تعریف می‌کند که هدف آن شناسائی آسیب‌پذیری‌های قابل سوءاستفاده است که می‌تواند ناشی از ساخت، عملکرد و یا پیکربندی ناصحیح TOE باشد. خانواده‌هائی که در اینجا تعریف می‌شوند نگران شناسائی‌های آسیب‌پذیری‌ها در تحلیل یک کانال پنهان، تحلیل پیکربندی TOE، آزمایش توانائی مکانیسم‌های امنیتی، و شناسائی خطاهائی است که در خلال ایجاد TOE بوجود آمده‌اند. خانوادهٔ دوم، طبقه‌بندی امنیتی مؤلفه‌های TOE را پوشش می‌دهد. سومی و چهارمی تحلیل تغییرات در اثر یک ضربهٔ امنیتی و ایجاد شواهد برای رعایت رَویه‌ها است. این کلاس فراهم‌آورندهٔ بلوک‌های سازندهٔ استقرار روش‌های حفظ اطمینان‌بخشی است.
حفظ اطمینان‌بخشی	لازمه‌هائی که بایستی پس از تأئید یک TOE در برابر یک CC اعمال شوند را فراهم می‌آورد. این لازمه‌ها این هدف را دنبال می‌کنند که اطمینان دهند که TOE در صورت اعمال تغییراتی به آن و یا محیطش، عملیاتی باقی خواهد ماند.

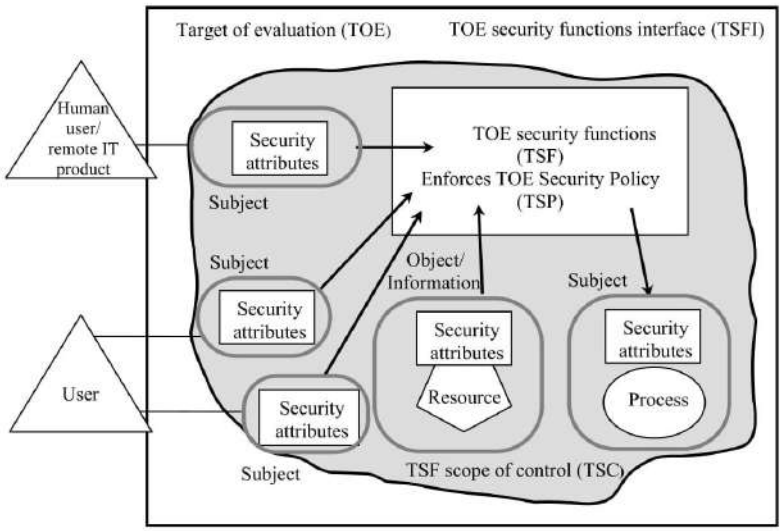
- **اهداف امنیتی (ST)**: شامل اهداف امنیتی IT و لازمه‌های یک TOE مشخص شناسائی شده بوده و معیارهای عملیاتی و اطمینان‌بخش پیشنهادشده بتوسط آن TOE برای دست‌یابی به لازمه‌های یادشده را تعریف می‌کند. ST ممکن است با یک یا چند PP مطابقت داشته و مبنای یک ارزیابی را تشکیل دهد. ST بتوسط یک فروشنده یا تولیدکننده فراهم می‌شود.

شکل ۶-۱۱ رابطهٔ بین لازمه‌ها از یک سو و پروفایل‌ها و هدف‌ها از سوی دیگر را نشان می‌دهد. برای یک PP، یک کاربر می‌تواند تعدادی از مؤلفه‌ها را انتخاب کرده تا لازمه‌های محصول مطلوب خود را تعریف نماید. کاربر همچنین می‌تواند به بسته‌های از قبل تعریف شده‌ای مراجعه کندکه تعدادی از لازمه‌های معمول یادشده در اسناد یک محصول را گردهم آورده است. بطریق مشابه، یک تولیدکننده یا طراح می‌تواند تعدادی از مؤلفه‌ها یا بسته‌ها را انتخاب کرده و یک ST را تعریف کند.



شکل ۶-۱۱ سازمان و ساختار لازمه‌های معیارهای مشترک

شکل ۷-۱۱ آنچه که در اسناد CC، پارادایم لازمه‌های عملیاتی امنیت خوانده می‌شود، را نشان می‌دهد. در واقع این شکل بر اساس مفهوم پایشگر مرجع بنا شده، اما از اصطلاحات و فلسفهٔ طراحی CC استفاده می‌کند.



شکل ۷-۱۱ پارادایم لازمه‌های عملیاتی امنیت

۴-۱۱ منابع مطالعاتی

یک منبع کلاسیک برای مطالعهٔ دیوارهای آتش [CHAP00] است. منبع کلاسیک دیگری که اخیراً بروزرسانی شده است [CHES03] است. [OPPL97]، [LODI98] و [BELL94b] مقالات خوبی برای مرور مطلب هستند. [WACK02] یک مرور عالی بر تکنولوژی دیوار آتش و سیاست‌های مرتبط با آن است. [AUDI04] و [WILS05] مباحث خوبی در مورد دیوارهای آتش دارند.

[GASS88] یک بررسی تفصیلی از سیستم‌های کامپیوتری معتمد را ارائه می‌دهد. [PFLE03] و [GOLL99] نیز این مقوله را پوشش می‌دهند. [FELT03] و [OPPL05] بحث‌های مفیدی در بارهٔ محاسبات مورد اعتماد دارند.

AUDI04 Audin, G. "Next-Gen Firewalls: What to Expect." *Business Communications Review*, June 2004.

BELL94b Bellovin, S., and Cheswick, W. "Network Firewalls." *IEEE Communications Magazine*, September 1994.

CHAP00 Chapman, D., and Zwicky, E. *Building Internet Firewalls*. Sebastopol, CA: O'Reilly,2000.

CHES03 Cheswick, W., and Bellovin, S. *Firewalls and Internet Security: Repelling the Wily Hacker*. Reading, MA: Addison-Wesley, 2003.

FELT03 Felten, E. "Understanding Trusted Computing: Will Its Benfits Outweigh Its Drawbacks?" *IEEE Society and Privacy*, May/June 2003.
GASS88 Gasser, M. *Building a Secure Computer System*. New York: Van Nostrand Reinhold, 1988.
GOLL99 Gollmann, D. *Computer Security*. New York: Wiley, 1999.
LODI98 Lodin, S., and Schuba, C. "Firewalls Fend Off Invasions from the Net." *IEEE Spectrum*, February 1998.
OPPL97 Oppliger, R. "Internet Security: Firewalls and Beyond." *Communications of the ACM*, May 1997.
OPPL05 Oppliger, R., and Rytz, R. "Does Trusted Computing Remedy Computer Security Problems?" *IEEE Security and Privacy*, March/April 2005.
PFLE03 Pfleeger, C. *Security in Computing*. Upper Saddle River, NJ: Prentice Hall, 2003.
WACK02 Wack, J.; Cutler, K.; and Pole, J. *Guidelines on Firewalls and Firewall Policy*. NIST Special Publication SP 800-41, January 2002.
WILS05 Wilson, J. "The Future of the Firewall."*Business Communications Review*, May 2005.

وِب سایت‌های مفید

- **Firewall.com**: لینک‌های بسیار به مراجع دیوارهای آتش و منابع نرم‌افزاری.
- **Trusted Computing Group**: گروه سازندگانی که درگیر ساخت و توسعهٔ استانداردهای کامپیوتری هستند. سایت شامل مقالات، مشخصه‌هاو لینک‌هائی به سازندگان است.
- **Common Criteria Portal**: وِب‌سایت رسمی پروژهٔ معیارهای مشترک.

۵-۱۱ واژه‌های کلیدی، سؤالات مرورکنندهٔ بحث و مسائل

واژه‌های کلیدی

دیوار آتش	firewall	لیست کنترل دست‌یابی	access control list (ACL)
امنیت چند سطحه	multilevel security	ماتریس دست‌یابی	access matrix
موضوع	object	حق دست‌یابی	access right
مسیریاب فیلترکنندهٔ بسته‌ها	packet- filtering router	دروازهٔ سطح کاربرد	application- level gateway
پایشگر مرجع	reference monitor	میزبان دژدار	bastion host
دیوارآتش تفتیش‌کننده حالت	stateful inspection firewall	بلیت توانائی	capability ticket
سوژه	subject	دروازهٔ سطح مدار	circuit- level gateway
سیستم معتمد	trusted system	معیارهای مشترک	common criteria(CC)

سؤالات مرورکننده بحث

۱۱-۱ سه هدف اصلی طراحی یک دیوار آتش را نام ببرید.

۱۱-۲ چهار تکنیک مختلف که بتوسط دیوارهای آتش برای کنترل دست‌یابی و استقرار نظام امنیتی استفاده می‌شود، را نام ببرید.

۱۱-۳ چه اطلاعاتی از سوی یک مسیریاب فیلترکنندهٔ بسته‌ها مورد استفاده قرار می‌گیرد؟

۱۱-۴ برخی از نقاط ضعف یک مسیریاب فیلترکنندهٔ بسته‌ها کدام است؟

۱۱-۵ فرق بین یک مسیریاب فیلترکنندهٔ بسته‌ها و یک دیوار آتش تفتیش‌کنندهٔ حالت چیست؟

۱۱-۶ یک دروازهٔ سطح مدار چیست؟

۱۱-۷ یک دروازهٔ سطح کاربرد چیست؟

۱۱-۸ تفاوت‌های سه پیکربندی شکل ۲-۱۱ کدام است؟

۱۱-۹ در مقولهٔ کنترل دست‌یابی، فرق بین یک سوژه و یک موضوع چیست؟

۱۱-۱۰ فرق بین یک لیست کنترل دست‌یابی و یک بلیت توانائی چیست؟

۱۱-۱۱ دو قانونی که یک پایشگر مرجع اعمال می‌کند، کدام‌اند؟

۱۱-۱۲ یک پایشگر مرجع چه خواصی باید داشته باشد؟

۱۱-۱۳ معیارهای مشترک چیستند؟

مسائل

۱۱-۱ همانطور که دربخش ۱-۱۱ ذکر گردید، یکی از روش‌های شکست دادن حملهٔ فِرگمنت‌های کوچک این است که یک طول حداقل از سرآیند حمل‌ونقل را مجبور سازیم تا در اولین فِرگمنت بستهٔ IP قرار گیرد. اگر اولین فِرگمنت پذیرفته نشود، همهٔ فِرگمنت‌های دیگر نیز می‌توانند پذیرفته نشوند. از سوی دیگر، ماهیت IP چنان است که فِرگمنت‌ها می‌توانند خارج از نظم دریافت شوند. بنابراین یک فِرگمنت میانی ممکن است قبل از اینکه فِرگمنت اولیه برگشت داده شود از فیلتر عبور کند. چگونه می‌توان این مشکل را رفع کرد؟

۱۱-۲ در یک بستهٔ IPv4، اندازهٔ محموله در اولین فِرگمنت برحسب اُکتت مساوی (Total Length − (4 × IHL است. اگر این اندازه کمتر از حداقل لازم (۸ اُکتت برای TCP) باشد، آنگاه این فِرگمنت و تمام بسته پذیرفته نمی‌شوند. روش دیگری برای برآورده نمودن این منظور پیشنهاد کنید که فقط از میدان Fragment Offset استفاده کند.

۱۱-۳ RFC 791 که مشخص‌کنندهٔ پروتکل IPv4 است، یک الگوریتم بازسازی مجدد(reassembly) را توصیف می‌کند که منجر به تولید یک فِرگمنت جدید می‌گردد که جای قسمت‌های هم‌پوشان فِرگمنت‌های قبل را پر می‌کند. با داشتن چنین پیاده‌سازی، یک حمله‌کننده می‌تواند یک سری بسته‌هائی را بسازد که در آنها پائین‌ترین فِرگمنت (zero-offset) شامل داده‌های بی‌آزار بوده (بنابراین بتوسط مدیریت فیلتر بسته‌ها عبور کند) و برخی بسته‌های بعد که دارای offset غیرصفربوده و روی اطلاعات سرآیند TCP (مثلاً پورت مقصد) افتاده و باعث تغییر آنها شود. بستهٔ دوم از بیشتر پیاده‌سازی‌ها عبور نموده زیرا دارای یک fragment offset صفر نیست. روشی را پیشنهاد کنید که بتواند از سوی یک فیلتر بسته‌ها بکار گرفته شده و با این حمله مقابله نماید.

۱۱-۴ ضرورت قانون « بالاتر را نخواند» برای یک سیستم امن چند سطحه کاملاً روشن است. اهمیت قانون « پائین‌تر را ننویسد» چیست؟

۱۱-۵ در شکل ۱۱-۵ یک لینک اسب تروا در زنجیرهٔ copy-and-observe-later پاره شده است. دو زاویهٔ ممکن دیگر برای حمله بتوسط Alice وجود دارد: Alice وارد سیستم شده و تلاش کند تا دنباله را مستقیماً بخواند و یا اینکه Alice یک سطح امنیتی حساس را به فایل جیب مخفی اعمال نماید. آیا پایشگر مرجع از این حملات جلوگیری می‌کند؟

پیوست (الف)

جنبه‌هائی از تئوری از اعداد

الف-۱ اعداد اول و اول نسبی

در این بخش تنها با اعداد صحیح غیرمنفی سروکار خواهیم داشت. استفاده از اعداد صحیح منفی تفاوت زیادی را در بحث بوجود نمی‌آورد.

مقسوم علیه‌ها

می‌گوئیم که $a, b \neq 0$ را می‌شمارد اگر برای مقداری از m، $a = mb$ باشد که درآن a ، b و m اعداد صحیح می‌باشند. یعنی a, b را می‌شمارد اگر تقسیم این دو برهم باقیمانده نداشته باشد. علامت $b \mid a$ اغلب به مفهوم این‌که a, b را می‌شمارد بکار می‌رود. همچنین $b \mid a$ به این مفهوم است که b *مقسوم‌علیه* a است. برای مثال، مقسوم‌علیه‌های عدد ۲۴، اعداد ۱، ۲، ۳، ۴، ۶، ۸، ۱۲ و ۲۴ هستند.

روابط زیر برقرارند:

- اگر $1 \mid a$ ، آنگاه $a = \pm 1$
- اگر $a \mid b$ و $b \mid a$ آنگاه $a = \pm b$
- هر $b \neq 0$ ، 0 را می‌شمارد
- اگر $g \mid b$ و $h \mid b$، آنگاه برای هر عدد صحیح اختیاری m و n، $b \mid (mg + nh)$

برای ملاحظهٔ آخرین نکته ذکرشده توجه کنید که

اگر $g \mid b$ ، آنگاه برای یک عدد صحیح g_1 ، $g = b \times g_1$ خواهد بود

اگر $h \mid b$ ، آنگاه برای یک عدد صحیح g_2 ، $h = b \times g_2$ خواهد بود

بنابراین

$$mg + nh = mbg_1 + nbh_1 = b \times (mg_1 + nh_1)$$

و در نتیجه b مقسوم علیه $mg + nh$ خواهد بود.

اعداد اول

هر عدد صحیح $p > 1$ یک عدد اول است اگر تنها مقسوم‌علیه‌های آن ± 1 و $\pm p$ باشند. اعداد اول نقش مهمی در تئوری اعداد و در تکنیک‌های مورد بحث در فصل ۳ این کتاب دارند.

هر عدد صحیح $a > 1$ را می‌توان به فرم یکتائی به صورت فاکتور زیر درآورد:

$$a = p_1^{a_1} p_2^{a_2} \dots p_t^{a_t}$$

که درآن $p_1 < p_2 < \dots < p_t$ اعداد اول بوده و هر a_i یک عدد صحیح مثبت است. برای مثال، ۱۳ × ۷ - ۹۱ و ۱۳ × ۱۱^۲ × ۷ - ۱۱۰۱۱ می‌باشد.

بیان مسأله به نحو دیگر نیز مفید است. اگر P مجموعهٔ همهٔ اعداد اول باشد، آنگاه هر عدد صحیح مثبت را می‌توان به فرم یکتائی به شکل زیر نوشت:

$$a = \prod_{p \in \mathrm{P}} p^{a_p}$$ که در آن هر $a_p \geq 0$ است

طرف سمت راست عبارت بالا، حاصلضرب تمام اعداد اول p ممکن است. برای هر مقدار a بیشتر توان‌های a_p صفر هستند.

اندازهٔ هر عدد مثبت داده شده را می‌توان با لیست نمودن تمام توان‌های غیرصفر فرمول قبل تعیین کرد. بنابراین، عدد صحیح ۱۲ بصورت $\{ a_2 = ۲ \text{ و } a_3 = ۱ \}$ و عدد صحیح ۱۸ بصورت $\{ a_2 = ۱ \text{ و } a_3 = ۲ \}$ نشان داده می‌شود. حاصلضرب دو عدد معادل جمع کردن توان‌های متناظر با هم است.

$$k = mn \quad \rightarrow \quad k_p = m_p + n_p \qquad \text{برای تمام مقادیر } p$$

ببینیم برحسب فاکتورهای اول ذکرشده $a \mid b$ چه معنی می‌دهد؟ هر عدد صحیح به فرم p^k را تنها می‌توان به یک عدد صحیح با توان کوچک‌تر یا مساوی عدد اول p^j $(j \leq k)$ تقسیم کرد. بنابراین می‌توان گفت

$$a \mid b \quad \rightarrow \quad a_p \leq b_p \qquad \text{برای تمام } p \text{ ها}$$

اعداد اول نسبی

نماد $\gcd(a, b)$ را برای نمایش **بزرگترین مقسوم‌علیه مشترک** a و b بکار می‌بریم. عدد صحیح مثبت c را بزرگترین مقسوم‌علیه مشترک a و b می‌نامیم اگر

۱- c یک مقسوم‌علیه a و b باشد.

۲- هر مقسوم‌علیه a و b یک مقسوم‌علیه c باشد.

بیان دیگر مطلب چنین است:

$$\gcd(a, b) = \text{ماکزیمم } k \text{ با شرط اینکه } k \mid a \text{ و } k \mid b$$

چون می‌خواهیم که بزرگترین مقسوم‌علیه مشترک مثبت باشد، $\gcd(a,b)=\gcd(-a,b)=\gcd(a,-b)=\gcd(-a,-b)$. در حالت کلی، $\gcd(a, b) = \gcd(|a|, |b|)$. برای مثال $\gcd(60,24) = \gcd(60,-24) = 12$. همچنین چون تمام اعداد صحیح 0 را می‌شمارند، $\gcd(a, 0) = |a|$.

تعیین بزرگترین مقسوم‌علیه مشترک دو عدد صحیح مثبت ساده است در صورتی که بتوان آن دو عدد را بصورت مضربی از اعداد اول نوشت. برای مثال

$$300 = 2^2 \times 3^1 \times 5^2$$

$$18 = 2^1 \times 3^2$$

$$\gcd(18,300) = 2^1 \times 3^1 \times 5^0 = 6$$

در حالت کلی

$$k = \gcd(a, b) \quad \rightarrow \quad k_p = \min(a_p, b_p) \qquad \text{برای تمام مقادیر } p$$

تعیین فاکتورهای اول یک عدد بزرگ کار آسانی نیست و بنابراین رابطهٔ قبل مستقیماً به محاسبهٔ بزرگترین مقسوم‌علیه مشترک منجر نخواهد شد.

اعداد صحیح a و b نسبت به هم اول‌اند اگر هیچ فاکتور اول مشترکی نداشته باشند، یعنی اگر تنها فاکتور مشترک آنها عدد ۱ باشد. این بیان معادل این است که بگوئیم a و b نسبت به هم اول‌اند اگر $\gcd(a, b) = 1$ باشد. برای مثال، ۸ و ۱۵ نسبت به هم اول‌اند زیرا مقسوم‌علیه‌های ۸ مساوی ۱ و ۲ و ۴ و ۸ بوده و مقسوم‌علیه‌های ۱۵، اعداد ۱ و ۳ و ۵ و ۱۵ هستند و بنابراین ۱ تنها عدد مشترک این دو لیست است.

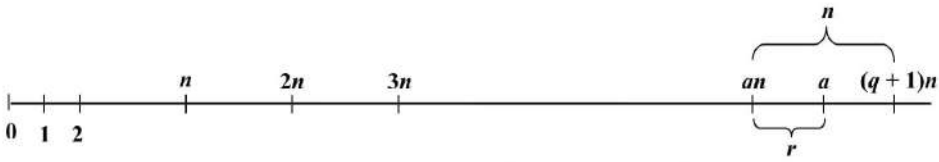
الف -۲ حساب پیمانه‌ای

برای هر عدد صحیح مثبت n و هر عدد صحیح غیرمنفی a، اگر a را بر n تقسیم کنیم، یک خارج قسمت صحیح q و یک باقیماندهٔ صحیح r بدست می‌آوریم که از رابطهٔ زیر تبعیت می‌کنند:

$$a = qn + r \qquad 0 \le r < n \; ; \; q = \lfloor a / n \rfloor$$

که در آن $\lfloor x \rfloor$ بزرگترین عدد صحیح کوچکتر یا مساوی x است.

شکل الف-۱ نشان می‌دهد که با داشتن a و n مثبت، همیشه می‌توان q و r را طوری پیدا کرد که شرط قبل را ارضاء کند. با نمایش اعداد صحیح روی یک محور اعداد، a در جائی از خط قرار خواهد گرفت (مقدار مثبت a نشان داده شده است. می‌توان عدد منفی a را نیز روی محور نشان داد). با شروع از 0 بسمت n، $2n$، تا qn طوری پیش می‌رویم که $qn \le a$ و $(q + 1)n > 1$ باشد. فاصلهٔ بین qn تا a برابر r بوده و ما اندازه‌های یکتای مقادیر q و r را پیدا کرده‌ایم. باقیماندهٔ r را اغلب **residue** گویند.



شکل الف-۱ رابطهٔ $a = qn + r \; ; \; 0 \le r < n$

اگر a یک عدد صحیح و n یک عدد صحیح مثبت باشد، باقیماندهٔ تقسیم a بر n را بصورت $a \bmod n$ تعریف می‌کنیم. بنابراین، برای هر عدد صحیح n همیشه می‌توان نوشت:

$$a = \lfloor a / n \rfloor \times n + (a \bmod n)$$

دو عدد صحیح a و b را **هم‌نهشت به پیمانهٔ n** گویند هرگاه $(a \bmod n) = (b \bmod n)$ باشد. این خاصیت را بصورت $a \equiv b \bmod n$ می‌نویسند. بعنوان مثال $73 \equiv 4 \bmod 23$ و $21 \equiv -9 \bmod 10$. توجه کنید که اگر $a \equiv 0 \bmod n$ باشد، آنگاه $n \mid a$.

عملگر پیمانه دارای خواص زیر است:

۱- اگر $n \mid (a-b)$ آنگاه $a \equiv b \bmod n$ است.

۲- $(a \bmod n) = (b \bmod n)$ به مفهوم $a \equiv b \bmod n$ است.

۳- اگر $a \equiv b \bmod n$ باشد، $b \equiv a \bmod n$ است.

۴- اگر $a \equiv b \bmod n$ و $b \equiv c \bmod n$ باشد، $a \equiv c \bmod n$ است.

برای اثبات خاصیت اول، اگر $n \mid (a-b)$، آنگاه برای مقداری از k، $(a - b) = kn$. بنابراین می‌توان نوشت $a = b + kn$. در نتیجه $(a \bmod n)$ = (باقیمانده تقسیم $b + kn$ بر n) = (باقیماندهٔ تقسیم b بر n) = $(b \bmod n)$ بقیه خواص نیز به آسانی اثبات می‌شوند.

عملگر $(\bmod n)$ تمام اعداد صحیح را به مجموعهٔ اعداد صحیح $\{0 , 1 , 2 , \ldots , (n-1)\}$ نگاشت می‌کند. در اینجا این سؤال مطرح می‌شود: آیا می‌توان عملیات حساب در این مجموعهٔ محدود را انجام داد؟ جواب سؤال مثبت بوده و تکنیک شناخته شده برای انجام این عمل **حساب پیمانه‌ای** (modular) است.

حساب پیمانه‌ای دارای خواص زیر است:

۱- $[(a \bmod n) + (b \bmod n)] \bmod n = (a + b) \bmod n$

۲- $[(a \bmod n) - (b \bmod n)] \bmod n = (a - b) \bmod n$

۳- $[(a \bmod n) \times (b \bmod n)] \bmod n = (a \times b) \bmod n$

خاصیت اول را ثابت می‌کنیم. اگر $(a \bmod n) = r_a$ و $(b \bmod n) = r_b$ تعریف شود، آنگاه برای یک عدد صحیح j، $a = ra + jn$ بوده و برای یک عدد صحیح k، $b = rb + kn$ خواهد بود. آنگاه

$$\begin{aligned}(a + b) \bmod n &= (ra + jn + rb + kn) \bmod n \\ &= (ra + rb + (k + j)n) \bmod n \\ &= (ra + rb) \bmod n \\ &= [(a \bmod n) + (b \bmod n)] \bmod n\end{aligned}$$

بقیهٔ خواص نیز به آسانی اثبات می‌شوند.

پیوست (ب)

واژه‌های امنیت شبکه

گرچه این واژه‌ها در طول فصول کتاب بطور مفصل تعریف شده و مورد بحث قرار گرفته‌اند ولی برای کسی که تنها بدنبال آشنائی با تعریف مختصر آنهاست، این واژه‌نامه می‌تواند مفید واقع شود.

asymmetric encription (رمزنگاری غیرمتقارن): نوعی سیستم رمزنگاری که در آن رمزنگاری و رمزگشائی با استفاده از دو کلید مختلف انجام می‌شود. نام یکی از کلیدها، کلیدعمومی و نام دیگری کلیدخصوصی است. این نوع رمزنگاری را رمزنگاری کلید- عمومی نیز می‌خوانند.

authentication (اعتبارسنجی): عمل تأئید هویت یک موجودیت سیستم است. در این مقوله اعتبارسنجی را می‌توان نوعی احراز هویت دانست.

authenticator (اعتبارسنج): نوعی اطلاعات اضافی است که به پیام متصل می‌شود تا گیرنده را قادر سازد تا معتبربودن پیام را آزمایش کند. اعتبارسنج ممکن است مستقل از محتوای پیام بوده (مثل یک nonce یا یک شناسهٔ منبع) و یا ممکن است تابعی از محتوای پیام باشد(مثل یک اندازهٔ hash یا یک parity).

avalanche effect (اثر بهمنی): یک خاصیت مثبت در یک الگوریتم رمزنگاری که بواسطهٔ آن یک تغییر کوچک در متن ساده و یا کلید، موجب تغییر بزرگی در متن رمزشده می‌گردد. برای یک کُد درهم‌ساز، اثر بهمنی باعث می‌گردد تا تغییر کوچکی در پیام موجب تغییر بزرگی در چکیدهٔ پیام شود.

bacteria (باکتری): برنامه‌ای که برای تولید کپی‌هائی از خود، منابع سیستم را به کار بیهوده مشغول می‌سازد.

birthday attack (حملهٔ روز تولد): این حمله که نوعی تلاش برای شکستن رمز است، سعی می‌کند تا دو مقدار مختلف در دامنهٔ یک تابع را که به یک مقدار یکسان در بُرد تابع نگاشت می‌شوند، پیدا کند.

block chaining (زنجیره‌کردن قالب‌ها): روشی در خلال رمزنگاری قالبی متقارن است که یک بلوک خروجی را نه تنها وابسته به متن سادهٔ جاری و کلید کرده، بلکه آن را وابسته به ورودی و/ یا خروجی مرحلهٔ قبل نیز می‌نماید. اثر زنجیره‌کردن قالب‌ها این است که دو بلوک متن سادهٔ مشابه، دو بلوک متن رمزشدهٔ متفاوت تولید کرده و در نتیجه شکستن رمز سخت‌تر خواهد شد.

block cipher (رمز قالبی): یک الگوریتم رمزنگاری متقارن است که در آن یک بلوک از بیت‌های متن ساده (معمولاً ۶۴ یا ۱۲۸ بیت)، کلاً به صورت یک بلوک از متن رمزشده با همان طول تبدیل می‌شود.

byte (بایت): یک ردیف ۸- تائی از بیت‌ها را گویند. به نام *اُکتت* نیز خوانده می‌شود.

cipher (رمز): یک الگوریتم برای رمزنگاری و رمزگشائی است. یک رمز، یک بخش از اطلاعات (یک عنصر از متن ساده) را با هدف پنهان کردن محتوای آن با عنصر دیگری عوض می‌کند. نوعاً نحوهٔ این تعویض از طریق یک کلید سرّی هدایت می‌شود.

ciphertext (متن رمزشده): خروجی یک الگوریتم رمزنگاری است که فرم رمزشدهٔ داده‌ها یا پیام می‌باشد.

code (کُد): یک قاعدهٔ تغییرناپذیر برای جایگزین کردن یک بخش از اطلاعات (مثل حرف، کلمه، جمله) با عنصر دیگری است که لزوماً از جنس اطلاعات اولیه نیست. معمولاً هدف این عمل پنهان کردن اطلاعات نیست. مثال‌هائی در این زمینه شامل کُد حرفی ASCII (هر علامت بتوسط یک ردیف ۷- بیتی نشان داده می‌شود) و کُد فرکانسی FSK (هر مقدار باینری با یک فرکانس خاص نشان داده می‌شود)، می‌باشند.

computaionally secure (ازنظر محاسباتی امن): از اینجهت امن است که زمان و/ یا هزینهٔ شکستن امنیت بقدری بالاست که انجام آن معقول نمی‌باشد.

confusion (گیج کردن): یک تکنیک رمزنگاری است که در جستجوی هرچه پیچیده‌تر کردن رابطهٔ بین خواص آماری متن رمزشده با کلید رمزنگاری است.این امر با استفاده از الگوریتم‌های مخلوط‌کنندهٔ پیچیده که وابسته به کلید و متن ورودی‌اند صورت می‌پذیرد.

conventional encryption (رمزنگاری رسمی): همان رمزنگاری متقارن است.

covert channel (کانال پنهان): یک کانال ارتباطی است که انتقال اطلاعات را بنحوی ممکن می‌سازد که خارج از اهداف طراحی آن تسهیلات بوده است.

cryptanalysis (شکستن رمز): شاخه‌ای از علم رمزنگاری است که مرتبط با کشف رمز برای استخراج اطلاعات، و یا جعل اطلاعات رمزشده بنحوی است که معتبر تلقی گردد.

cryptographic checksum (جمع کنترلی رمزی): یک اعتبارسنج است که تابعی هم از دیتائی که باید اعتبار آن سنجیده شود و هم از یک کلید سرّی می‌باشد. به آن کُد اعتبارسنجی پیام (MAC) نیز گفته می‌شود.

cryptography (رمزنگاری): شاخه‌ای از علم رمزشناسی است که مرتبط با طراحی الگوریتم‌های رمزنگاری و رمزگشائی بوده که هدف آنها سرّی نگاه داشتن و/ یا معتبرنگاه داشتن پیام است.

cryptology (رمزشناسی): بررسی ارتباطات امن است که هم شاخهٔ رمزنگاری و هم شاخهٔ کشف رمز را شامل می‌گردد.

decryption (رمزگشائی): تبدیل متن و یا دیتای رمزشده (که متن رمزشده نامیده می‌شود) به متن و یا دیتای اولیه (که متن ساده نامیده می‌شود) است.

differential cryptanalysis (شکستن تفاضلی رمز): روشی که در آن متون سادهٔ انتخاب شده با اُلگوهای تفاضلی XOR شده بخصوص رمزنگاری می‌شوند. اُلگوهای تفاضلی متن رمزشدهٔ منتجه، اطلاعاتی را به دست می‌دهند که می‌تواند برای کشف کلید رمز مفید باشد.

diffusion (پخش کردن): یک تکنیک رمزنگاری است که در پی ایجاد ابهام در ساختار آماری پیام، با پخش کردن اثر هر عنصر متن ساده به بسیاری از عناصر متن رمزشده، است.

digital signature (امضای دیجیتال): یک تکنیک رمزنگاری است که خلق‌کنندهٔ پیام را قادر می‌سازد تا کُدی که خاصیت یک امضاء را دارد به پیام متصل سازد. امضاء با محاسبهٔ hash پیام و رمزنگاری پیام با کلید خصوصی خلق‌کنندهٔ پیام انجام می‌شود. امضاء، منبع صدور و صحت پیام را تضمین می‌کند.

digram (دو- حرفی): یک ردیف دو- حرفی است. در زبان انگلیسی و سایر زبان‌ها، وقوع نسبی دو- حرفی‌ها در متن ساده می‌تواند در شکستن بعضی رمزها مورد استفاده قرار گیرد.

discretionary access control (کنترل دست‌یابی منصفانه): یک سرویس کنترل دست‌یابی که یک سیاست امنیتی، مبتنی بر هویت موجودیت‌های سیستم و حقوق دست‌یابی آنها به منابع سیستم، را پیاده می‌نماید. این سرویس از این‌جهت «منصفانه» نام دارد که یک موجودیت ممکن است دارای آنچنان حق دست‌یابی باشد که بتواند با صلاحدید خود به موجودیت دیگری حق دست‌یابی به بعضی منابع را تفویض نماید.

divisor (مقسوم علیه): یک عدد صحیح را مقسوم علیه عدد صحیح دیگر گویند در صورتی که باقیماندهٔ تقسیم این بر آن صفر باشد.

encryption (رمزنگاری): تبدیل یک متن ساده و یا دیتا به یک فرم نامفهوم را گویند که با استفاده از یک نگاشت برگشت‌پذیر مبتنی بر یک جدول تبدیل یا الگوریتم صورت می‌پذیرد.

firewall (دیوار آتش): یک کامپیوترکه بعنوان واسط ارتباط با کامپیوترهای خارج از یک شبکه تخصیص یافته و عوامل حفاظتی بخصوصی در آن تعبیه شده است تا فایل‌های حساس و یا کامپیوترهای درون شبکه را محافظت نماید. این دستگاه به شبکهٔ خارجی، بخصوص اینترنت، اتصالات و خطوط تلفنی ورودی سرویس می‌دهد.

greatest common divisor (بزرگترین مقسوم علیه مشترک): بزرگترین مقسوم علیه مشترک دو عدد صحیح a و b بزرگترین عدد صحیح مثبتی است که هم a و هم b را می‌شمارد. گویند یک عدد صحیح عدد صحیح دیگر را می‌شمارد در صورتی که تقسیم آن دو به هم باقیمانده نداشته باشد.

hash function (تابع درهم‌ساز): تابعی را گویندکه یک بلوک دیتا یا پیام با طول متغیر را به یک اندازهٔ ثابت، که کُد hash خوانده می‌شود، نگاشت می‌کند. این تابع طوری طراحی می‌شود که که وقتی مورد محافظت واقع گردد یک اعتبارسنج برای دیتا و یا پیام باشد. به آن چکیدهٔ پیام هم می‌گویند.

honeypot (طعمه) : یک سیستم دام است که برای فریب دادن حمله کننده‌ها و دور نگاه‌داشتن آنها از سیستم‌های اصلی بکار می‌رود. نوعی تشخیص تهاجم است.

initialization vector (بردار آغازگر): یک بلوک تصادفی از داده‌ها است که برای آغاز رمزنگاری بلوک‌های متعدد دیتا در هنگام استفاده از تکنیک رمزنگاری زنجیره‌ای قالبی از آن استفاده می‌شود. از IV برای خنثی نمودن حملات متن سادهٔ معلوم استفاده می‌شود.

intruder (مهاجم): فردی که بصورت غیرمجاز به یک سیستم کامپیوتری دست یافته و یا برای این منظور تلاش می‌کند. او می‌تواند یک کاربر معتبر بوده که برای کسب امتیازاتی بیشتر از حقوق خود تلاش می‌نماید.

intrusion detection system (سیستم تشخیص تهاجم): مجموعه‌ای از ابزارهای خودکار که برای تشخیص دست‌یابی‌های غیرمجاز به یک سیستم میزبان بکار گرفته می‌شود.

Kerberos: نامی است که به برنامهٔ سرویس اعتبارسنجی پروژهٔ Athena داده شده است.

key distribution center (مرکز توزیع کلید): یک سیستم معتبر برای ارسال کلیدهای موقت اجلاس به رؤسای ارتباط است. هر کلید اجلاس با استفاده از یک کلید اصلی، که بین مرکز توزیع کلید و رئیس موردنظر به اشتراک گذاشته شده است، رمزنگاری می‌شود.

logic bomb (بمب لاجیک): منطقی که در یک برنامهٔ کامپیوتری طوری جاسازی شده است که وقوع شرایط خاصی در سیستم را کنترل کند. وقتی این شرایط حاصل شوند، عملی اجرا خواهد شد که منجر به فعالیت‌های غیرمجاز می‌گردد.

mandatory access control (کنترل دست‌یابی اجباری): وسیله‌ای برای محدودکردن دست‌یابی به موضوعات است که مبتنی بر تخصیص صفاتی به یک کاربر، یک فایل و سایر موضوعات است. این کنترل‌ها به این مفهوم اجباری هستند که نمی‌توانند بتوسط کاربر و یا برنامه‌های او دستکاری شوند.

man-in-the-middle attack (حملهٔ واسطه‌گرانه): نوعی حملهٔ استراق سمع فعال است که در آن حمله کننده در جریان مخابرهٔ داده‌ها وارد شده و بطور انتخابی دیتای انتقال داده شده را طوری دستکاری نماید که خود را بجای یک یا چند طرف درگیر ارتباط جا بزند.

master key (کلید اصلی): یک کلید پردوام که بین یک مرکز توزیع کلید و یک رئیس ارتباط به اشتراک گذاشته می‌شود تا از آن برای رمزکردن انتقال کلیدهای اجلاس استفاده شود. . معمولاً کلیدهای اصلی به فرمی خارج از قواعد رمزنگاری توزیع می‌شوند. با آن کلید رمزنگاری- کلید هم می‌گویند.

meet-in-the-middle attack : این یک حملهٔ شکستن رمز است که تلاش می‌کند تا اندازه‌ای در برد و دامنه ترکیب دو پیام را بنحوی پیدا نماید که نگاشت مستقیم تابع اول برابر تصویر معکوس تابع دوم باشد. بعبارت دیگر ملاقات دو مقدار در وسط دو تابع ترکیبی انجام شود.

message authentication (اعتبارسنجی پیام): عملی که برای تأئید صحت پیام از آن استفاده می‌شود.

Message authentication code (MAC): سرجمع مرتبط با رمزنگاری

message digest (چکیدهٔ پیام): تابع درهم ساز (hash)

modular arithmetic (حساب پیمانه‌ای): نوعی از حساب اعداد صحیح که برای عددی مانند n ، همهٔ اعداد صحیح را به یک مجموعهٔ متناهی $[0, 1,\ldots,n\text{-}1]$ کاهش می‌دهد. هر عدد صحیح خارج از این محدوده با انتخاب باقیماندهٔ تقسیم آن بر n به یک عدد داخل این محدوده تبدیل می‌شود.

mode of operation (مُود عملیاتی): تکنیکی برای ارتقاء اثر یک الگوریتم رمزنگاری و یا وفق دادن یک الگوریتم به کاربردی بخصوص، همچون اعمال یک رمز قالبی به ردیفی از بلوک‌های دیتا و یا یک جریان داده‌هاست.

multilevel security (امنیت چند لایه): قابلیتی است که کنترل دست‌یابی را به سطوح چندگانهٔ طبقه‌بندی داده‌ها اعمال می‌کند.

multiple encryption (رمزنگاری چندگانه): استفادهٔ مکرر از یک تابع رمزنگاری، با کلیدهای مختلف، برای تولید یک نگاشت پیچیده‌تر از متن ساده به متن رمزشده است.

nibble: ردیفی از چهار بیت را گویند

nonce: یک شناسه و یا یک عدد که تنها یک‌بار بکار می‌رود.

one-way function (تابع یک- طرفه): تابعی که محاسبهٔ آن آسان بوده ولی محاسبهٔ معکوس آن غیرممکن است.

password (کلمهٔ عبور): یک اندازهٔ سرّی، معمولاً ردیفی از کاراکترها، که از آن بعنوان یک اطلاعات اعتبارسنجی استفاده می‌شود. یک کلمهٔ عبور معمولاً با یک شناسهٔ کاربر جفت بوده که این شناسهٔ در عمل اعتبارسنجی بصورت واضح ارائه می‌شود. در بعضی موارد این شناسه ممکن است ضمنی باشد.

plaintext (متن ساده): ورودی یک تابع رمزنگاری و یا خروجی یک تابع رمزگشائی است.

primitive root (ریشهٔ اولیه): اگر r و n نسبت به هم اول باشند، $n > 0$ و اگر $\Phi(n)$ کوچک‌ترین توان مثبت m بنحوی باشد که $r^m \equiv 1 \bmod n$ باشد. آنگاه r را ریشهٔ اولیهٔ با پیمانهٔ n خوانند.

private key (کلید خصوصی): یکی از دو کلیدی که در سیستم رمزنگاری نامتقارن از آن استفاده می‌شود. بمنظور ارتباطات امن، کلید خصوصی بایستی تنها برای خلق‌کنندهٔ آن معلوم باشد.

pseudorandom number generator (تولیدکنندهٔ اعداد شبه تصادفی): تابعی که بصورت یقینی ردیفی از اعداد را درست می‌کند که ظاهراً از نظر آماری تصادفی هستند.

public key (کلید عمومی): یکی از دو کلیدی است که در سیستم رمزنگاری نامتقارن از آن استفاده می‌شود. کلید عمومی، در معرض استفادهٔ عموم قرار می‌گیرد تا بهمراه کلید خصوصی جفت آن مورد استفاده قرار گیرد.

public-key certificate (گواهی‌نامهٔ کلید- عمومی): شامل یک کلید عمومی باضافه User ID صاحب آن است، که کل آن بتوسط یک شخص ثالث معتبر امضاء شده است. شخص ثالث معمولاً یک مقام مسئول صدورگواهی‌نامه (CA) است که همچون یک سازمان دولتی و یا یک مؤسسهٔ اعتباری مورد اعتماد یک جمعیت از کاربران است.

public-key encryption (رمزنگاری کلید- عمومی): رمزنگاری نامتقارن.

public-key infrastructure(PKI) (زیرساخت کلید- عمومی): مجموعهٔ سخت‌افزارها، نرم‌افزارها، مردم، سیاست‌ها و رَویه‌های لازم برای خلق، مدیریت، ذخیره سازی، توزیع و ابطال گواهی‌نامه‌های دیجیتال مبتنی بر رمزنگاری نامتقارن می‌باشد.

relatively prime (اول نسبی): دو عدد در صورتی نسبت به هم اول هستند که هیچ فاکتور اول مشترکی با هم نداشته باشند، یعنی مقسوم علیه مشترک آنها یک باشد.

replay attack (حملات بازخوانی): حمله‌ای که در آن یک سرویس که قبلاً اعتبارسنجی و کامل شده است با جعل تقاضای مجددی برای کسب فرامین معتبر تلاش می‌کند.

residue: وقتی عدد صحیح a بر عدد صحیح n تقسیم می‌شود، باقیماندهٔ r را residue گویند. بطور معادل $r = a \bmod n$

residue class: تمام اعداد صحیحی که وقتی بر n تقسیم شوند دارای باقیماندهٔ یکسانی باشند یک residue class به پیمانه n را تشکیل می‌دهند. بنابراین، برای یک باقیماندهٔ داده شده r اعداد صحیح r $r+n$ $r \pm 2n$ و ... متعلق به residue class (mod n) هستند.

RSA algorithm: یک الگوریتم رمزنگاری کلید- عمومی است که بر مبنای بتوان رساندن یک عدد در حساب پیمانه‌ای قرار دارد. این الگوریتم تنها الگوریتم پذیرفته شده عمومی برای یک رمزنگاری کلید- عمومی امن و عملی است.

secret key (کلید سرّی): کلیدی است که از آن در یک سیستم رمزنگاری متقارن استفاده می‌شود.

security attack (حملهٔ امنیتی): یک ضربه بر امنیت سیستم که از یک تهدید هوشمندانه ناشی می‌شود. یعنی یک عمل هوشمندانه برای یک تلاش هوشمندانه در جهت شکست سرویس‌های امنیتی و نقض سیاست امنیتی یک سیستم است.

security mechanism (مکانیسم امنیتی): یک پردازش (یا دستگاهی که این پردازش را فراهم می‌کند) که برای تشخیص، جلوگیری و یا بازیابی یک حملهٔ امنیتی طراحی شده است.

security service (سرویس امنیتی): یک پردازش و یا یک سرویس ارتباطی است که امنیت سیستم‌های پردازش دیتا و انتقال اطلاعات یک سازمان را ارتقاء می‌دهد. سرویس‌ها به منظور مقابله با حملات امنیتی طراحی شده و از یک یا چند مکانیسم امنیتی برای ایجاد سرویس استفاده می‌کنند.

security threat (تهدید امنیتی): خطر بالقوه‌ای برای نقض امنیت است که وقتی وجود دارد که شرایط، قابلیت، عمل و یا رویدادی بتواند امنیت را نقض کرده و ایجاد اخلال نماید. بعبارت دیگر یک تهدید یک خطر بالقوه است که ممکن است از یک نقطهٔ ضعف استفاده کند.

session key (کلید اجلاس): یک کلید رمزنگاری موقت است که بین دو رئیس ارتباط از آن برای رمزنگاری استفاده می‌شود.

steganography (پنهان نگاری): روش‌های پنهان کردن وجود یک پیام و یا داده‌های دیگر است. این مقوله با رمزنگاری که در آن مفهوم پیام، و نه وجود پیام، پنهان می‌شود متفاوت است.

stream cipher (رمز دنباله‌ای): یک الگوریتم رمزنگاری متقارن است که در آن متن رمز شدهٔ خروجی، بیت- به- بیت و یا بایت- به- بایت، از روی دنبالهٔ متن سادهٔ ورودی تولید می‌شود.

symmetric encryption (رمزنگاری متقارن): یک روش رمزنگاری که در آن رمزنگاری و رمزگشائی با استفاده از یک کلید صورت می‌پذیرد. به آن رمزنگاری رسمی و یا سنّتی نیز گویند.

trapdoor (درب مخفی): نقطهٔ ورودی مخفی و غیرمستندی که دست‌یابی به برنامه یا سیستم، بدون عبور از مراحل معمول کنترلی، را امکان‌پذیر می‌نماید.

trapdoor one-way function (تابع یک- طرفه با درب مخفی): تابعی که محاسبهٔ آن ساده بوده و محاسبهٔ معکوس آن مقدور نیست مگر اینکه اطلاعات ویژه‌ای در دست باشد.

Trojan horse (اسب تروا): یک برنامهٔ کامپیوتری که در ظاهر مفید و قابل استفاده بوده ولی شامل یک تابع بالقوه بداندیش نیز هست که مکانیسم‌های امنیتی را شکست می‌دهد. این کار، گاهی با استثمار اعتبارهای قانونی موجودیتی که برنامه را بکار گرفته است، صورت می‌پذیرد.

trusted system (سیستم مورد اعتماد): یک کامپیوتر و سیستم عامل که برای اجرای یک سیاست امنیتی مورد تأئید است.

unconditionally secure (بطور غیرمشروط امن): در برابر دشمنی که زمان نامحدود و منابع محاسباتی نامحدود دارد، امن است

virtual private network(VPN) (شبکهٔ خصوصی مجازی): شامل مجموعه‌ای از کامپیوترهاست که بتوسط یک شبکهٔ نسبتاً ناامن بهم متصل شده و از رمزنگاری و پروتکل‌های مخصوص برای ایجاد امنیت استفاده می‌کند.

virus (ویروس): کُد برنامه‌ای که در درون یک برنامهٔ دیگر جاسازی شده و باعث می‌شود تا یک کپی از ویروس در برنامه یا برنامه‌های دیگر وارد شود. علاوه بر انتشار، ویروس معمولاً عملیات ناخواسته‌ای را انجام می‌دهد.

worm (کِرم): برنامه‌ای است که می‌تواند خود را تکثیر کرده و کپی‌های خود را در عرض شبکه از یک کامپیوتر به کامپیوتر دیگر ارسال دارد. پس از ورود، یک کِرم ممکن است فعال شده و مجدداً تکثیر و انتشار یابد. علاوه بر انتشار، کِرم معمولاً کارهای ناخواسته‌ای را نیز انجام می‌دهد.

zombie (زامبی): برنامه‌ای است که بطور مخفیانه کنترل کامپیوتر دیگری،که به اینترنت متصل است، را بدست گرفته و از آن کامپیوتر مبادرت به حملاتی می‌کند که دنبال کردن آن برای خلق‌کنندهٔ زامبی نیز مشکل است.

پیوست (ج)

مراجع

علائم اختصاری

ACM Association for Computing Machinery
IEEE Institute of Electrical and Electronics Engineers
NIST National Institute of Standards and Technology

ALVA90 Alvare,A. "How Crackers Crack Passwords or What Passwords to Avoid." *Proceedings, UNIX Security Workshop II*, August 1990.

ANDE80 Anderson, J. *Computer Security Threat Monitoring and Surveillance*. Fort Washington, PA: James P. Anderson Co., April 1980.

AUDI04 Audin, G. "Next-Gen Firewalls: What to expect." *Business Communications Review*, June 2004.

AXEL00 Axelsson, S. "The Base-Rate Fallacy and the Difficulty of Intrusion Detection." *ACM Transactions and Information and system Security*, August 2000.

BACE00 Bace, R. *Intrusion Detection*. Indianapolis, IN: Macmillan Technical Publishing, 2000.

BACE01 Bace, R., and Mell, P. *Intrusion Detection Systems*. NIST Special Publication SP 800-31 , November 2000.

BARR03 Barreto, P., and Rijmen, V. "The Whirlpool Hashing Function." *Submitted to NESSIE*, September 2000, revised May 2003.

BAUE88 Bauer, D., and Koblentz, M."NIDX- An Expert System for Real-Time Network Intrusion Detection." *Proceedings, Computer Networking Symposium*, April 1988.

BELL90 Bellovin, S. and Merritt, M. " Limitations of the Kerberos Authentication System." *Computer Communications Review*, October 1990.

BELL92 Bellovin, S., "There Be Dragons." *Proceedings, UNIX Security Symposium III*, September 1992.

BELL93 Bellovin, S. "Packets Found on an Internet." *Computer Communications Review*, July 1993.

BELL94 Bellovin, S., and Cheswick, W. "Network Firewalls." *IEEE Communications Magazine*, September 1994.

BELL96a Bellare, M.; Canetti, R.; and Krawczyk, H. "Keying Hash Functions for Message Authentication." *Proceedings, CRYPTO '96*, August 1996; published by Springer-Verlag. An expanded version is available at http://www-cse.ucsd.edu/users/mihir.

BELL96b Bellare, M.; Canetti, R.; and Krawczyk, H. "The HMAC Construction." *CryptoBytes*, Spring 1996.

BERS92 Berson, T."Differential Cryptoanalysis Mod 2^{32} with Applications to MD5." *Proceedings, EUROCRYPT '92*, May 1992; published by Springer-Verlag.

BISH03 Bishop, M. *Computer Security: Art and Science*. Boston: Addison-Wesley,2003.

BISH05 Bishop, M. *Introduction to Computer Security*. Boston: Addison-Wesley,2005.

BLOO70 Bloom, B. "Space/time Trade-offs in Hash Coding with Allowable Errors." *Communications of the ACM*, July 1970.

BLUM97a Bluementhal, U.; Hien, N.; and Wijnen, B. "Key Derivation for Network Management Applications." *IEEE Network*, May/June, 1997.

BLUM97b Blumenthal, U, and Wijnen, B. "Security Features for SNMPv3." *The Simple Times*, December 1997.

BOER93 Boer, B., and Bosselaers, A. "Collisions for the Compression Function of MD5." *Proceedings, EUROCRYPT '93*, 1993; published by Springer-Verlag.

BRYA88 Bryant, W. *Designing an Authentication System: A Dialog in Four Scenes*. Project Athena document, February 1988. Available at http://web.mit.edu/kerberos/www/dialogue.html.

CASS01 Cass, S."Anatomy of Malice." *IEEE Spectrum*, November 2001.

CERT01 CERT Coordination Center. "Denial of Service Attacks." June 2001. http://www.cert.org/tech_tips/denial_of_service.html

CERT02 CERT Coordination Center. "Multiple vulnerabilities in Many Implementations of the Simple Network Management Protocol." CERT Advisory CA-2002-03, 25 June 2002. www.cert.org/advisories/CA-2002-03.html

CHAN02 Chang, R. "Defending Against Flooding-Based Distributed Denial - of – Service Attacks: A Tutorial." *IEEE Communications Magazine*, October 2002.

CHAP00 Chapman, D., and Zwicky, E. *Building Internet Firewalls*. Sebastopol, CA: O'Reilly, 2000.

CHEN98 Cheng, P., et al. "A Security Architecture for the Internet Protocol." *IBM Systems Journal*, Number 1, 1998.

CHES97 Chess, D. "The Future of Viruses on the Internet." *Proceedings, Virus Bulletin International Conference*, October 1997.

CHES03 Cheswick, W., and Bellovin, S. *Firewalls and Internet Security:* Repelling the wily Hacker. Reading, MA: Addison-Wesley, 2003.

COHE94 Cohen, F. *A Short Course on Computer Viruses*. New York: Wiley, 1994.

CORM01 Cormen, T.; Leiserson, C.; Rivest. R.; and Stein. C. *Introduction to Algorithms*. Cambridge, MA: Addison-Wesley, 2003.

DAVI89 Davies, D., and Price, W. *Security for Computer Networks*. New York: Wiley, 1994.

DAMG89 Damgard, I. "A Design Principle for Hash Functions." *Proceedings, CRYPTO '89*, 1989; published by Springer-Verlag.

DAVI93 Davies, C., and Ganesan, R. "BApassed: A New Proactive Password Checker." *Proceedings, 16th National Computer Security Conference*, September 1993.

DAWS96 Dawson, E., and Nielsen, L. "Automated Cryptoanalysis of XOR Plaintext Strings." *Cryptologia*, April 1996.

DENN87 Denning, D. "An Intrusion-Detection Model." *IEEE Transaction on Software Engineering*. February 1987.

DIFF76 Diffie, W., and Hellman, M. "Multiuser Cryptographic Techniques." *IEEE Transactions on Information Theory*, November 1976.

DIFF88 Diffie, W., "The First Ten Years of Public-Key Cryptography." *Proceedings of the IEEE*, May 1988. Reprinted in [SIMM92]

DOBB96 Dobbertin, H. "The Status of MD5 After a Recent Attack." *CryptoBytes*, Summer 1996.

DORA03 Doraswamy, N., and Harkins, D. *IPSec*, Upper Saddle River, NJ: Prentice Hall, 2003.

DREW99 Drew, G. *Using SET for Secure Electronic Commerce*. Upper Saddle River, NJ: Prentice Hall, 1999.

EFF98 Electronic Frontier Foundation. *Cracking DES: Secrets of Encryption Research, Wire-tap Politics, and Chip Design*. Sebastopol, CA: O'Reilly, 1998.

ENGE80 Enger, N., and Howerton, P. *Computer Security*. New York: Amacom, 1980.

FEIS73 Feistel, H. "Cryptography and Computer Privacy." *Scientific American*, May 1973.

FELT03 Felten, E. "Understanding Trusted Computing: Will Its Benefits Outweigh its Drawbacks?" *IEEE Security and Privacy*, May/June 2003.

FLUH00 Fluhrer, S., and McGrew, D. "Statistical Analysis of the Alleged RC4 Key Stream Generator." Proceedings, *Fast Software Encryption 2000*, 2000.

FLUH01 Fluhrer, S.; Mantin, I.; and Shamir, A. "Weakness in the Key Scheduling Algorithm of RC4." *Proceedings, Workshop in Selected Areas of Cryptography*, 2001.

FORD95 Ford, W. "Advances in Public-Key Certificate Standards." *ACM SIGSAC Review*, July 1995.

FORR97 Forrest, S.; Hofmeyr, S.; and Somayaji, A. "Computer Immunology." *Communications of the ACM*, October 1997.

FRAN01 Frankel, S. *Demystifying the IPSec Puzzle*. Boston: Artech House, 2001.

GARD77 Gardner, M. "A New Kind of Cipher That Would Take Millions of Years to Break." *Scientific American*, August 1977.

GARF97 Garfinkel, S., and Spafford, G. *Web Security & Commerce*. Cambridge, MA: O'Reilly and Associates, 1997.

GASS88 Gasser, M. *Building a Secure Computer System*. New York: Van Nostrand Reinhold, 1988.

GAUD00 Gaudin, S. "The Omega Files." *Network World*, June 26, 2000.

GOLL99 Gollmann, D. *Computer Security*. New York: Wiley, 1999.

GUTM02 Gutmann, P. "PKI: It's Not Dead, Just Resting." *Computer*, August 2002.

HARL01 Harley, D.; Slade, R.; and Gattiker, U. *Viruses Revealed*. New York: Osborne/McGraw Hill, 2001.

HEBE92 Heberlein, L.; Mukherjee, B.; and Levitt, K. "Internetwork Security Monitor: An Intrusion Detection System for Large-Scale Networks." *Proceedings, 15th National Computer Security Conference*, October 1992.

HELD96 Held, G. *Data and Image Compression: Tools and Techniques*. New York: Wiley, 1996.

HONE01 The Honeynet Project. *Know Your Enemy: Revealing the Security Tools, Tactics, and Motives of the Blackhat Community*. Reading, MA: Addison-Wesley, 2001.

HUIT98 Huitema, C. *IPv6: The New Internet Protocol*. Upper Saddle River, NJ: Prentice Hall, 1998.

IANS90 I'Anson, C., and Mitchell, C. "Security Defects in CCITT Recommendation X.509 – The Directory Authentication Framework." *Computer Communications Review*, April 1990.

ILGU93 Ilgun, K. "USTAT: A Real-Time Intrusion Detection System for UNIX." *Proceedings, 1993 IEEE Computer Society Symposium on Research in Security and Privacy*, May 1993.

JAVI91 Javitz, H., and Valdes, A. "The SRI IDES Statistical Anomaly Detector." *Proceedings, 1991 IEEE Computer Society Symposium on Research in Security and Privacy*, May 1991.

JIAN02 Jiang, G. "Multiple Vulnerabilities in SNMP." *Security and Privacy Supplement to Computer Magazine*, 2002.

JUEN85 Jueneman, R.; Matyas, S.; and Meyer, C. "Message Authentication." *IEEE Communications Magazine*, September 1988.

KENT00 Kent, S. "On the Trail of Intrusions into Information Systems." *IEEE Spectrum*, December 2000.

KEPH97a Kephart, J.; Sorkin, G.; Chess, D.; and White, S. "Fighting Computer Viruses." *Scientific American*, November 1997.

KEPH97b Kephart, J.; Sorkin, G,; Swimmer, B.; and White, S. "Blueprint for a Computer Immune System." *Proceedings, Virus Bulletin International Conference*, October 1997.

KLEI90 Klein, D. "Foiling the Cracker: A Survey of, and Improvements to, Password Security." *Proceedings, UNIX Security Workshop II*, August 1990.

KNUD98 Knudsen, L., et al. "Analysis Method for Alleged RC4." *Proceedings, ASIACRYPT '98*, 1998.

KOBL92 Koblas, D., and Koblas, M. "SOCKS." *Proceedings, UNIX Security Symposium III*, September 1992.

KOHL89 Kohl, J. "The Use of Encryption in Kerberos for Network Authentication." *Proceedings, Crypto '89*, 1989; published by Springer-Verlag.

KOHL94 Kohl, J.; Neuman, B.; and Ts'o, T. "The Evolution of the Kerberos Authentication Service." In Brazier, F., and Johansen, D. *Distributed Open Systems*. Los Alamitos, CA: IEEE Computer Society Press, 1994. Available at http://web.mit.edu/kerberos/www/papers.html.

KUMA97 Kumar, I. *Cryptology*. Laguana Hills, CA: Aegean Park Press, 1997.

LEUT94 Leutwyler, K. "Superhack." *Scientific American*, July 1994.

LODI98 Lodin, S., and Schuba, C. "Firewalls Fend Off Invasions from the Net." *IEEE Spectrum*, February 1998.

LUNT88 Lunt, T., and Jagannathan, R. " A Prototype Real-Time Intrusion-Detection Expert System." *Proceedings, 1988 IEEE Computer Society Symposium on Research in Security and Privacy*, April 1988.

MACG97 Macgregor, R.; Ezvan, C.; Liguori, L.; and Han, J. *Secure Electronic Transactions: Credit Card Payment on the Web in Theory and Practice*. IBM RedBook SG24-4978-00, 1997. Available at www.redbooks.ibm.com.

MADS93 Madsen, J. "World Record in Password Checking." *Usenet, comp.security.misc news-group*, August 18, 1993.

MANT01 Mantin, I., Shamir, A. "A Practical Attack on Broadcast RC4." *Proceedings, Fast Software Encryption*, 2001.

MARK97 Markham, T. "Internet Security Protocol." *Dr. Dobb's Journal*, June 1997.

MCHU00 McHugh, J.; Christie, A.; and Allen, J. "The Role of Intrusion Detection Systems." *IEEE Software*, September/October 2000.

MEIN01 Meinel, C. "Code Red for the Web." *Scientific American*, October 2001.

MENE97 Menezes, A.; van Oorschot, P.; and Vanstone, S. *Handbook of Applied Cryptography*. Boca Raton, FL: CRC Press, 1997.

MERK97 Merkle, R. Secrecy, *Authentication, and Public Key Systems*. PHD Thesis, Stanford University, June 1979.

MERK89 Merkle, R. "One Way Hash Functions and DES." *Proceedings, CRYPTO '89, 1989*;published by Springer-Verlag.

MEYE82 Meyer, C., and Matyas, S. *Cryptography: A New Dimension in Computer Data Security*. New York: Wiley, 1982.

MILL88 Miller, S.; Neuman, B.; Schiller, J.; and Saltzer, J. "Kerberos Authentication and Authorization System." *Section E.2.1, Project Athena Technical Plan*, M.I.T. Project Athena, Cambridge, MA. 27 October 1988.

MIRK04 Mirkovic, J., and Relher, P. "A Taxonomy of DDoS Attack and DDoS Defense Mechanisms." ACM S*IGCOMM Computer Communications Review*, April 2004.

MIST98 Mister, S., and Tavares, S. "Cryptoanalysis of RC4-Like Ciphers." *Proceedings, Workshop in Selected Areas of Cryptography, SAC' 98*. 1998.

MITC90 Mitchell, C.; Walker, M.; and Rush, D. "CCITT/ISO Standards for Secure Message Handling." *IEEE Journal on Selected Areas in Communications*, May 1989.

MOOR01 Moore, M. "Inferring Internet Denial-of-Service Activity." *Proceedings of the 10th USENIX Security Symposium*, 2001.

NACH97 Nachenberg, C. "Computer Virus-Antivirus Coevolution." *Communications of the ACM*, January 1997.

NEED78 Needham, R., and Schroeder, M. "Using Encryption for Authentication in Large Networks of Computers." *Communications of the ACM*, December 1978.

NING04 Ning, P., et al. "Techniques and Tools for Analyzing Intrusion Alerts." *ACM Transactions on Information and System Security*, May 2004.

OPPL97 Oppliger, R. "Internet Security: Firewalls and Beyond." *Communications of the ACM*, May 1997.

OPPL05 Oppliger, R., and Rytz, R. "Does Trusted Computing Remedy Computer Security Problems?" *IEEE Security and Privacy*, March/April 2005.

PATR04 Patrikakis, C.; Masikos, M.; and Zouraraki, O. "Distributed Denial of Service Attacks." *The Internet Protocol Journal*, December 2004.

PAUL03 Paul, S., and Preneel, B. "Analysis of Non-fortuitous Predictive States of the RC4 Keystream Generator." *Proceedings, INDOCRYPT '03*, 2003.

PAUL04 Paul, S., and Preneel, B. "A New Weakness in the RC4 Keystream Generator and an Approach to Improve the Secutity of the Cipher." *Proceedings, Fast Software Encryption*, 2004.

PERL99 Perlman, R. "An Overview of PKI Trust Models." *IEEE Network*, November/December 1999.

PFLE03 Pfleeger, C. *Security in Computing*. Upper Saddle River, NJ: Prentice Hall, 2003.

PIAT91 Piattelli-Palmarini, M. "Probability: Neither Rational nor Capricious." *Bostonia*, March 1991.

PIEP03 Pieprzyk, J.; Hardjono, T.; and Seberry, J. *Fundamentals of Computer Security*. New York: Springer-Verlag, 2003.

PORR92 Porras, P. *STAT: A State Transition Analysis Tool for Intrusion Detection*. Master's Thesis, University of California at Santa Barbara, July 1992.

PREN02 Preenel, B. "New European Schemes for Signature, Integrity and Encryption (NESSIE): A Status Report." *Proceedings of the 5th International Workshop on Practice and Theory in Public Key Cryptosystems: Public Key Cryptography*. 2002.

PROC01 Proctor, P., *The Practical Intrusion Detection Handbook*. Upper Saddle River, NJ: Prentice Hall, 2001.

PUDO02 Pudovkina, M. "Statistical Weaknesses in the Alleged RC4 Keystream Generator." *Proceedings, 4th International Workshop on Computer Science and Information Technologies*, 2002.

RESC01 Rescorla, E. SSL and TLS: *Designing and Building Secure Systems*. Reading, MA: Addison-Wesley, 2001.

RIVE78 Rivest, R.; Shamir, A.; and Adleman, L. "A Method for Obtaining Digital Signatures and Public Key Cryptosystems." *Communications of the ACM*, February 1978.

ROBS95a Robshaw, M. *Stream Ciphers*. RSA Laboratories Technical Report TR-701 , July 1995. http://www.rsasecurity.com/rsalabs

ROBS95b Robshaw, M. *Block Ciphers*. RSA Laboratories Technical Report TR-601, August 1995. http://www.rsasecurity.com/rsalabs

RODR02 Rodriguez, A.,A., et al. *TCP/IP Tutorial and Technical Overview*. Upper Saddle River: NJ: Prentice Hall, 2002.

SAFF93 Safford, D.; Schales, D.; and Hess. D. "The TAMU Security Package: An Ongoing Response to Internet Intruders in an Academic Environment." *Proceedings, UNIX Security Symposium IV*, October 1993.

SCHN00 Schneier, B. *Secrets and Lies: Digital Security in a Networked World*. New York: Wiley 2000.

SCHN96 Schneier, B. *Applied Cryptography*. New York: Wiley, 1996.

SIMM92 Simmons, G., ed. *Contemporary Cryptology: The Science of Information Integrity*. Piscataway, NJ; IEEE Press 1992.

SING99 Singh, S. The Code Book: *The Science of Secrecy from Ancient Egypt to Quantum Cryptography*. New York: Anchor Books, 1999.

SMIT97 Smith, R. *Internet Cryptography*. Reading, MA: Addison-Wesley, 1997.

SNAP91 Snapp, S., et al. "A System for Distributed Intrusion Detection." *Proceedings, COMPCON Spring '91*, 1991.

SPAF92a Spafford, E. "Observing Reusable Password Choices." *Proceedings, UNIX Security Symposium III*, September 1992.

SPAF92b Spafford, E. "OPUS: Preventing Weak Password Choices." *Computers and Security*, No. 3, 1992.

STAL99 Stallings, W. *SNMP, SNMPv2, SNMPv3, and RMON 1 and 2*. Reading, MA: Addison-Wesley, 1999.

STAL04 Stallings, W. *Computer Networking with Internet Protocols and Technology*. Upper Saddle River, NJ: Prentice Hall, 2004.

STAL06a Stallings, W. *Cryptography and Network Security: Principles and Practice, Fourth Edition*. Upper Saddle River, NJ: Prentice Hall, 2006.

STAL06b Stallings, W. "The Whirlpool Secure Hash Function." *Cryptologia*, January 2006.

STEI88 Steiner, J.; Neuman, C.; and Schiller, J. "Kerberos: An Authentication Service for Open Networked Systems." *Proceedings of the Winter 1988 USENIX Conference*, February 1988.

STEP93 Stephenson, P. "Preventive Medicine." *LAN Magazine*, November 1993.

STER92 Sterling, B. *The Hacker Crackdown: Law and Disorder on the Electronic Frontier*. New York: Bantam, 1992.

STIN06 Stinson, D. *Cryptography: Theory and Practice*. Boca Raton, FL: CRC Press, 2006.

STOL88 Stoll, C. "Stalking the Wiley Hacker." *Communications of the ACM*, May 1988.

STOL89 Stoll, C. *The Cuckoo's Egg*. New York: Doubleday, 1989.

SZOR05 Szor, P., *The Art of Computer Virus Research and Defense*. Reading, MA: Addison-Wesley, 2005.

THOM84 Thompson, K. "Reflections on Trusting Trust (Deliberate Software Bugs)." *Communications of the ACM*, August 1984.

TIME90 Time, Inc. *Computer Security, Understanding Computers Series*. Alexandria, VA: Time-Life Books, 1990.

TSUD92 Tsudik, G. "Message Authentication with One-Way Hash Functions." *Proceedings, INFOCOM '92*, May 1992.

TUNG99 Tung, B. Kerberos: *A Network Authentication System*. Reading, MA:Addison-Wesley, 1999.

VACC89 Vaccaro, H., and Liepins, G. "Detection of Anomalous Computer Session Activity." *Proceedings of the IEEE Symposium on Research in Security and Privacy*, May 1989.

VIJA02 Vijayan, J. "Denial-of-Service Attacks Still a Threat." *Computer World*, April 8, 2002.

WACK02 Wack, J.; Cutler, K.; and Pole, J. *Guidelines on Firewalls and Firewall Policy*. NIST Special Publications SP 800-41, January 2002.

WAGN00 Wagner, D., and Goldberg, I. "Proofs of Security for the UNIX Password Hashing Algorithm." *Proceedings, ASIACRYPT '00*, 2000.

WANG05 Wang, X.; Yin, Y.;and Yu, H. "Finding Collisions in the Full SHA-1. *Proceedings, Crypto'05*, 2005; published by Springer-Verlag.

WILS05 Wilson, J. "The Future of the Firewall." *Business Communications Review*, May 2005.

YUVA79 Yuval, G. "How to Swindle Rabin." *Cryptologia*, July 1979.

ZIV77 Ziv, J., and Lempel, A. "A Universal Algorithm for Sequential Data Compression." *IEEE Transactions on Information Theory*, May 1977.

واژه‌نامه انگلیسی– فارسی

abort	نادیده گرفتن
access control	کنترل دست یابی
acquirer	مباشر - فراهم‌کننده
active attack	حملهٔ فعال
affiliation	پیوستگی
agent	عامل
aggregate	تراکم
alarm	هشدار
algorithm	الگوریتم
anomaly	ناهنجاری
append	وصل کردن
arbiter	داور
association	اتحاد
asymmetric	نامتقارن
audit	بازرسی، ممیزی
authentic	معتبر
authentication	اعتبارسنجی
authentication header	سرآیند اعتبارسنجی
authenticator	اعتبارسنج
autoexecute	خوداجرا
availibility service	قابلیت دسترسی
backdoor	درب مخفی
bait	طعمه
bastion	دژ
batch	دسته
benign	بی خطر
block	بلوک
block cipher	رمز قالبی
body	بدنه
bog	باتلاق
bogus	ساختگی
boot-sector virus	ویروس بخش راه اندازی
bottleneck	گلوگاه
browser	مرورگر
brute-force attack	حملهٔ همه جانبه
bug	اشکال
byte	بایت
canonical	قانونی

CAST-128	نام یک الگوریتم رمزنگاری
certificate	گواهی‌نامه
chaining	زنجیر کردن
checksum	جمع کنترلی
cipher	رمز
ciphertext	متن رمزشده
ciphertext-only	فقط- متن رمزشده
clandestine	خفیه
client	کلاینت
clogging	انسداد
codebook	کتاب کُد
community	جامعه
compatibility	سازگاری
compiler	کامپایلر
component	مؤلفه
compression	فشرده سازی
compromise	لو رفتن
computationally secure	از نظر محاسباتی امن
computer	رایانه
computer security	امنیت رایانه
concatenation	جمع رشته ای
concievable	قابل تصور
confidentiality	محرمانگی
conformance	مطابقت
confusion	سردرگمی
connection	اتصال
connection oriented	اتصال گرا
connectionless	بدون اتصال
connection-request	درخواست ارتباط
consensus	مطابقت
context	مقوله
conventional	قراردادی
cookie	کوکی
counter	شمارنده
covert	پنهان
cracker	شکننده
cryptoanalysis	شکستن رمز
cryptoanalyst	شکنندهٔ رمز
cryptography	علم رمزنگاری
cryptology	علم رمزشناسی
daemon	دیو
data	داده
decode	کُدگشائی
decoy	دام
decryption	رمزگشائی
detached	جداشده- مجزا

diffusion	انتشار
digest	چکیده
digital signature	امضای دیجیتال
digram	دو- حرفی
directory	فهرست راهنما
discretionary access cotrol	کنترل دست‌یابی منصفانه
dispatcher	حمل‌کننده
disruptive	مخل
distributed enviroments	محیط های توزیع شده
domain	دامنه- قلمرو
dual	دوگانه
eavesdropping	استراق سمع- شنود
editor	ویرایش‌گر
elliptic curve	خَم بیضوی
email	پست الکترونیک
emoticon	احساس‌نما
encryption	رمزنگاری
encryption devices	تجهیزات رمزنگاری
end system	سیستم انتهائی
end-to-end	سر- به - سر
ephermal	یکبارمصرف
error-detection	تشخیص خطا
expert system	سیستم خبره
exploit	بهره برداری
extension	الحاقیه
external	خارجی
extranet	اکسترانت
fallacy	خطا
fax-back	فاکس برگردان
feasible	مقدور
feedback	بازخورد
Feistel	نام یک ساختار رمزنگاری متقارن
field	میدان
fingerprint	اثر انگشت
firewall	دیوارآتش
flaw	خطا
flooding	بمباران
foil	خنثی کردن
forgery	تقلب- جعل
fragmentation	قطعه قطعه کردن
framework	چهارچوب
frond-end processor	پردازشگر خط اول
gateway	دروازه
gauge	پیمانه
genuine	دست اول
hacker	هَکِر- نفوذگر

handshaking	دستداد
hash function	تابع درهم ساز
header	سَرآیند
heuristic	تاریخی
hostile	خصمانه
hub	هاب
IDEA	نام یک الگوریتم رمزنگاری
identifier	شناسه
immune	مصون
impersonation	جعل هویت
indispensible	ضروری
information security	امنیت اطلاعات
initiator	آغازگر
inner	درونی
innocuous	بی آزار
integrity	صحت-اصالت
intercept	قطع کردن
internal	داخلی
internet	بین شبکه ای
Internet	اینترنت
Internet Association	انجمن اینترنت
Internet Draft	پیش‌نویس اینترنت
Internet Publication	انتشارات اینترنت
internet security	امنیت اینترنت
Internet society	جامعهٔ اینترنت
Internet Standard	استاندارد اینترنت
intranet	اینترانت
intruder	مهاجم
intrusion	تهاجم
Kerberos	نام یک سرویس اعتبارسنجی
key	کلید
key distribution	توزیع کلید
key exchange	مبادلهٔ کلید
key length	طول کلید
key management	مدیریت کلید
key ring	دسته کلید
key space	فضای کلید
keystone	سنگ بنا
label	برچسب
legitimacy	مشروعیت
legitimate	مشروع- قانونی
lifetime	طول عمر
link	پیوند
loading	بارگذاری
logic bomb	بمب لاجیک
logical connection	اتصال منطقی

logon	اتصال به سیستم
macro	ماکرو
macro virus	ویروس ماکرو
mailbox	صندوق پستی
malicious	بداندیش
malware	بدافزار
mapping	نگاشت
masquerade	بالماسکه
masquerader	نقاب دار
master key	کلید اصلی
mediation	وساطت
memory-resident virus	ویروس مستقر در حافظه
merchant	تاجر
message	پیام
metmorphic virus	ویروس دگردیس
metric	معیار
misfeasor	سوءاستفاده‌کننده
mode	مُود
modification	دستکاری
modular	پیمانه ای
modulo	پیمانه
monitoring	پایش- نظارت
motivation	انگیزش
multiplicative inverse	معکوس ضربی
mutation engine	موتور تغییر
mutual	متقابل
native	بومی
nesting	لانه سازی
network security	امنیت شبکه
newsgroup	گروه خبری
node	گره
nonce	حال فعلی
nonrepudiation	عدم انکار
notation	علامت اختصاری
notification	تذکر
notion	آگاهی
obfuscation	ابهام زائی
object	موضوع
octet	اُکتت
offline	خارج از خط
one-time key	کلید یکبارمصرف
one-way	یکطرفه
online	برخط
orphan	یتیم
outer	بیرونی
overhead	سرباره

overlap	هم‌پوشانی
overt	آشکار
packet	بسته
packet switch	سوئیچ بسته‌ای
pad	لائی
padding	لائی‌گذاری
paradigm	پارادایم
parasitic virus	ویروس انگلی
passive attack	حملهٔ غیرفعال
passphrase	جملهٔ عبور
password	کلمهٔ عبور
patent	ثبت اختراع
pattern	الگو
payload	محموله
peer	نظیر
peer-to-peer	نظیر- به- نظیر
penetration	نفوذ
permutation	جایگشت
plaintext	متن ساده
platform	سیستم عامل - کامپیوتر
polymorphic virus	ویروس چندچهره
port	درگاه
precedence	حق تقدم
primitive root	ریشهٔ اولیه
principal	رئیس
private key	کلید خصوصی
procedure	رَویه
processor	پردازش گر
product system	سیستم ترکیبی
promulgate	اعلام کردن
protocol	پروتکل
proxy	پروکسی
pseudorandom	شبه تصادفی
public key	کلید عمومی
radix-64	تبدیلradix-64
random	تصادفی
range	بُرد
realm	قلمرو
receipt	رسید
recovery	بازیابی
reference monitor	پایشگر مرجع
reliability	قابلیت اعتماد
reliable	قابل اعتماد
replay	بازخوانی
resource allocation	تخصیص منابع
reversible	برگشت‌پذیر

revokation	ابطال- لغو
round	دُور
round-function	تابع دُور
router	مسیریاب
rudimentary	مقدماتی
rule-based	مبتنی بر قاعده
scalable	مقیاس‌پذیر
scrambled	درهم ریخته
secret	سرّی
secret key	کلید سرّی
secure	امن
security	امنیت
security architecture	معماری امنیت
security association	اتحاد امنیتی
security attack	حملهٔ امنیتی
security flaw	نقص امنیتی
security mechanism	سازوکار امنیتی
security service	سرویس امنیتی
seed	بذر
segment	سگمنت
sequence number	شمارهٔ ردیف
server	سرور
session	اجلاس
session key	کلید اجلاس
shareware	اشتراک‌افزار
signatory	امضاءکننده
signature	امضاء
smiley	خندانک
sniffer	بو کشنده
socket	سوکت
spam	نامه الکترونیکی ناخواسته
spoofing	جعل - تقلید
spurious	ساختگی- نادرست
stack	پشته
stateful inspection firewall	نوعی دیوار آتش
stealth virus	ویروس پنهان‌شونده
stream	دنباله
stream cipher	رمز دنباله‌ای
subkey	زیرکلید
substitution	جایگزینی
suite	مجموعه
symmetric encryption	رمزنگاری متقارن
tag	دنباله
terminal	پایانه
theft	دزدی
thread	رشته

threat	تهدید
threshold	آستانه
throughput	توان عملیاتی
ticket	بلیت
timely	بهنگام
time-sharing	اشتراک زمانی
timestamp	برچسب زمانی
traditional	سنّتی
traffic analysis	تحلیل ترافیک
trailer	تَه آیند
transaction	سند
transformation	تبدیل
transparent	شفاف
transport	حمل‌ونقل- ترابری
transposition	جابجائی
trapdoor	درب مخفی
trend	رَوَند
trespass	تعرض
trigger	ماشه‌کشی
Trojan Horse	اسب تروا
trust	اعتماد
unique	یکتا
update	به روز درآوردن
utility program	برنامهٔ کمکی
utilization	بهره گیری
vendor	فروشنده
version	نسخه
view	منظر
virtual	مجازی
virtual circuit	مدار مجازی
virus	ویروس
vulneribility	آسیب‌پذیری
web	وِب
webmail	پست مبتنی بر وِب
website	وِب سایت
wildcard	عام - عمومی
worm	کرم
zombie	زامبی

واژه‌نامه فارسی- انگلیسی

آستانه	threshold
آسیب‌پذیری	vulneribility
آشکار	overt
آغازگر	initiator
آگاهی	notion
ابطال- لغو	revokation
ابهام زائی	obfuscation
اتحاد	association
اتحاد امنیتی	security association
اتصال	connection
اتصال به سیستم	logon
اتصال‌گرا	connection oriented
اتصال منطقی	logical connection
اثر انگشت	fingerprint
اجلاس	session
احساس نما	emoticon
از نظر محاسباتی امن	computationally secure
اسب تروا	Trojan Horse
استاندارد اینترنت	Internet Standard
استراق سمع- شنود	eavesdropping
اشتراک‌افزار	shareware
اشتراک زمانی	time-sharing
اشکال	bug
اعتبارسنج	authenticator
اعتبارسنجی	authentication
اعتماد	trust
اعلام کردن	promulgate
اُکتت	octet
اکسترانت	extranet
الحاقیه	extension
الگو	pattern
الگوریتم	algorithm
امضاء	signature
امضاءکننده	signatory
امضای دیجیتال	digital signature
امن	secure
امنیت	security
امنیت اطلاعات	information security

internet security	امنیت اینترنت
computer security	امنیت رایانه
network security	امنیت شبکه
diffusion	انتشار
Internet Publication	انتشارات اینترنت
Internet Association	انجمن اینترنت
clogging	انسداد
motivation	انگیزش
intranet	اینترانت
Internet	اینترنت
loading	بارگذاری
replay	بازخوانی
feedback	بازخورد
audit	بازرسی ، ممیزی
recovery	بازیابی
bog	باتلاق
masquerade	بالماسکه
byte	بایت
malware	بدافزار
malicious	بداندیش
body	بدنه
connectionless	بدون اتصال
seed	بذر
label	برچسب
timestamp	برچسب زمانی
online	برخط
range	بُرد
reversible	برگشت‌پذیر
utility program	برنامهٔ کمکی
packet	بسته
block	بلوک
ticket	بلیت
logic bomb	بمب لاجیک
flooding	بمباران
update	به روز درآوردن
exploit	بهره برداری
utilization	بهره گیری
timely	بهنگام
sniffer	بو کشنده
native	بومی
innocuous	بی آزار
benign	بی خطر
outer	بیرونی
internet	بین شبکه‌ای
paradigm	پارادایم
terminal	پایانه

monitoring	پایش- نظارت
reference monitor	پایشگر مرجع
processor	پردازش‌گر
frond-end processor	پردازشگر خط اول
protocol	پروتکل
proxy	پروکسی
email	پست الکترونیک
webmail	پست مبتنی بر وِب
stack	پشته
covert	پنهان
message	پیام
Internet Draft	پیش نویس اینترنت
gauge	پیمانه
modulo	پیمانه
modular	پیمانه ای
affiliation	پیوستگی
link	پیوند
hash function	تابع درهم ساز
round-function	تابع دُور
merchant	تاجر
heuristic	تاریخی
transformation	تبدیل
radix-64	تبدیلradix-64
encryption devices	تجهیزات رمزنگاری
traffic analysis	تحلیل ترافیک
resource allocation	تخصیص منابع
notification	تذکر
aggregate	تراکم
error-detection	تشخیص خطا
random	تصادفی
trespass	تعرض
forgery	تقلب- جعل
trailer	تَه‌آیند
intrusion	تهاجم
threat	تهدید
throughput	توان عملیاتی
key distribution	توزیع کلید
patent	ثبت اختراع
transposition	جابجائی
Internet society	جامعهٔ اینترنت
community	جامعه
substitution	جایگزینی
permutation	جایگشت
detached	جداشده- مجزا
spoofing	جعل - تقلید
impersonation	جعل هویت

concatenation	جمع رشته‌ای
checksum	جمع کنترلی
passphrase	جملهٔ عبور
digest	چکیده
framework	چهارچوب
nonce	حال فعلی
precedence	حق تقدم
dispatcher	حمل‌کننده
transport	حمل‌ونقل- ترابری
security attack	حملهٔ امنیتی
passive attack	حملهٔ غیرفعال
active attack	حملهٔ فعال
brute-force attack	حملهٔ همه جانبه
offline	خارج از خط
external	خارجی
hostile	خصمانه
flaw	خطا
fallacy	خطا
clandestine	خفیه
elliptic curve	خَم بیضوی
foil	خنثی کردن
smiley	خندانک
autoexecute	خوداجرا
internal	داخلی
data	داده
decoy	دام
domain	دامنه- قلمرو
arbiter	داور
trapdoor	درب مخفی
backdoor	درب مخفی
connection-request	درخواست ارتباط
port	درگاه
scrambled	درهم ریخته
gateway	دروازه
inner	درونی
theft	دزدی
bastion	دژ
genuine	دست اول
handshaking	دستداد
modification	دستکاری
batch	دسته
key ring	دسته کلید
tag	دنباله
stream	دنباله
digram	دو- حرفی
round	دُور

dual	دوگانه
daemon	دیو
firewall	دیوارآتش
principal	رئیس
computer	رایانه
receipt	رسید
thread	رشته
cipher	رمز
stream cipher	رمز دنباله‌ای
block cipher	رمز قالبی
decryption	رمزگشائی
encryption	رمزنگاری
symmetric encryption	رمزنگاری متقارن
trend	رَوَند
procedure	رَویه
primitive root	ریشهٔ اولیه
zombie	زامبی
chaining	زنجیرکردن
subkey	زیرکلید
bogus	ساختگی
spurious	ساختگی- نادرست
compatibility	سازگاری
security mechanism	سازوکار امنیتی
end-to-end	سر- به - سر
header	سَرآیند
authentication header	سرآیند اعتبارسنجی
overhead	سرباره
confusion	سردرگُمی
secret	سرّی
server	سِرور
security service	سِرویس امنیتی
segment	سِگمنت
transaction	سَند
keystone	سنگ بنا
traditional	سنّتی
misfeasor	سوءاستفاده‌کننده
packet switch	سوئیچ بسته‌ای
socket	سوکت
end system	سیستم انتهائی
product system	سیستم ترکیبی
expert system	سیستم خبره
platform	سیستم عامل - کامپیوتر
pseudorandom	شبه تصادفی
transparent	شفاف
cryptoanalysis	شکستن رمز
cryptoanalyst	شکنندهٔ رمز

cracker	شکننده
sequence number	شمارهٔ ردیف
counter	شمارنده
identifier	شناسه
integrity	صحت-اصالت
mailbox	صندوق پستی
indispensible	ضروری
bait	طعمه
lifetime	طول عمر
key length	طول کلید
wildcard	عام - عمومی
agent	عامل
nonrepudiation	عدم انکار
notation	علامت اختصاری
cryptology	علم رمزشناسی
cryptography	علم رمزنگاری
fax-back	فاکس برگردان
vendor	فروشنده
compression	فشرده‌سازی
key space	فضای کلید
ciphertext-only	فقط- متن رمزشده
directory	فهرست راهنما
reliable	قابل اعتماد
concievable	قابل تصور
reliability	قابلیت اعتماد
availibility service	قابلیت دسترسی
canonical	قانونی
Conventional	قراردادی
intercept	قطع کردن
fragmentation	قطعه قطعه کردن
realm	قلمرو
compiler	کامپایلر
codebook	کتاب کُد
decode	کُدگشائی
worm	کرم
client	کَلاینت
password	کلمهٔ عبور
key	کلید
session key	کلید اجلاس
master key	کلید اصلی
private key	کلید خصوصی
secret key	کلید سرّی
public key	کلید عمومی
one-time key	کلید یکبارمصرف
access control	کنترل دست‌یابی
discretionary access cotrol	کنترل دست‌یابی منصفانه

cookie	کوکی
node	گره
newsgroup	گروه خبری
bottleneck	گلوگاه
certificate	گواهی نامه
pad	لائی
padding	لائی‌گذاری
nesting	لانه سازی
compromise	لو رفتن
component	مؤلفه
trigger	ماشه کشی
macro	ماکرو
key exchange	مبادلهٔ کلید
acquirer	مباشر- فراهم کننده
rule-based	مبتنی بر قاعده
mutual	متقابل
ciphertext	متن رمزشده
plaintext	متن ساده
virtual	مجازی
suite	مجموعه
confidentiality	محرمانگی
payload	محموله
distributed enviroments	محیط‌های توزیع‌شده
disruptive	مخل
virtual circuit	مدار مجازی
key management	مدیریت کلید
browser	مرورگر
router	مسیریاب
legitimate	مشروع- قانونی
legitimacy	مشروعیت
immune	مصون
consensus	مطابقت
conformance	مطابقت
authentic	معتبر
multiplicative inverse	معکوس ضربی
security architecture	معماری امنیت
metric	معیار
rudimentary	مقدماتی
feasible	مقدور
context	مقوله
scalable	مقیاس‌پذیر
view	منظر
intruder	مهاجم
mutation engine	موتور تغییر
mode	مُود
object	موضوع

میدان	field
نادیده گرفتن	abort
نام یک ساختار رمزنگاری متقارن	Feistel
نام یک سرویس اعتبارسنجی	Kerberos
نام یک سیستم رمزنگاری	IDEA
نام یک سیستم رمزنگاری	CAST-128
نامتقارن	Asymmetric
نامهٔ الکترونیکی ناخواسته	spam
ناهنجاری	anomaly
نسخه	version
نظیر	peer
نظیر- به- نظیر	peer-to-peer
نفوذ	penetration
نقاب دار	masquerader
نقص امنیتی	security flaw
نگاشت	mapping
نوعی دیوار آتش	stateful inspection firewall
هاب	hub
هشدار	alarm
هَکِر- نفوذگر	hacker
هم پوشانی	overlap
وِب	web
وِب سایت	website
وِساطت	mediation
وصل کردن	append
ویرایش‌گر	editor
ویروس	virus
ویروس انگلی	parasitic virus
ویروس بخش راه اندازی	boot-sector virus
ویروس پنهان شونده	stealth virus
ویروس چندچهره	polymorphic virus
ویروس دگردیس	metmorphic virus
ویروس ماکرو	macro virus
ویروس مستقر در حافظه	memory-resident virus
یتیم	orphan
یک الگوریتم رمزنگاری نامتقارن	RSA
یکبارمصرف	ephermal
یکتا	unique
یکطرفه	one-way

علائم اختصاری

3DES	Triple Data Encryption Standard
AES	Advanced Encryption Standard
AH	Authentication Header
ANSI	American National Standards Institute
AS	Authentication Server
BBS	Bulletin Board System
BCP	Best Current Practice
CBC	Cipher Block Chaining
CC	Common Criteria
CERT	Computer Emergency Response Team
CESG	Communications-Electronics Security group
CFB	Cipher Feedback
CMAC	Cipher-Based Message Authentication Code
CRT	Chinese Remainder Theorem
DDoS	Distributed Denial of Service
DEA	Data Encryption Algorithm
DES	Data Encryption Standard
DH	Diffie-Hellman
DOI	Domain of Interpretation
DoS	Denial of Service
DSA	Digital Signature Algorithm
DSS	Digital Signature Standard
ECB	Electronic Codebook
ESP	Encapsulating Security Payload
FCS	Frame Check Sequence
FIPS	Federal Information Processing Standard
FTP	File Transfer Protocol
HAR	Host Audit Protocol
HTTP	Hyper Text Transfer Protocol
IAB	Internet Architecture Board
IESG	Internet Engineering Steering Group
IETF	Internet Engineering Task Force
IMAP	Internet Mail Access Protocol
IP	Internet Protocol
IPSec	IP Security
ISO	International Organization for Standardization
ISP	Internet Service Provider
ITU	International Telecommunication Union
ITU-T	ITU Telecommunication Standardization Sector
IV	Initialization Vector
KDC	Key Distribution Center
LAN	Local Area Network
MAC	Message Authentication Code
MIB	Management Information Base
MIC	Message Integrity Code
MIME	Multipurpose Internet Mail Extension
MD5	Message Digest, Version 5
MTA	Mail Transfer Agent
MTU	Maximum Transmission Unit
MUA	Mail User Agent
NIST	National Institute of Standards and Technology
NSA	National Security Agency
OFB	Output Feedback
OSI	Open System Interconnection
OSPF	Open Shortest Path First
PCBC	Propagating Cipher Block Chaining
PDU	Protocol Data Unit
PGP	Pretty Good Privacy
PKI	Public Key Infrastructure
POP3	Post Office Protocol, Version 3
PRNG	Pseudorandom Number Generator
RFC	Request for Comments
RNG	Random Number Generator
RSA	Rivest-Shamir-Adelman
SET	Secure Electronic Transaction
SHA	Secure Hash Algorithm
SHS	Secure Hash Standard
S/MIME	Secure MIME
SMTP	Simple Mail Transfer Protocol
SNMP	Simple Network Management Protocol
SNMPv3	Simple Network Management Protocol,Version 3
SPI	Security Parameters Index
SPD	Security Policy Database
SSL	Secure Sockets Layer
ST	Security Target
TCP	Transmission Control Protocol
TFTP	Trivial File Transfer Protocol
TGS	Ticket Granting Service
TLS	Transport Layer Security
TOE	Target of Evaluation
TS	Technical Specification
UDP	User Datagram Protocol
USM	User Security Model
VACM	View-Based Access Control Mode
VPN	Virtual Private Network
WAN	Wide Area Network