

خودآموز کراک کردن

مقاله حفاظت (Protection) (۲)

[TOP.EXE] [F19.EXE] [POPULOUS.EXE] [MAP.EXE]

در درس قبل دیدید که چگونه از محافظت کلمه عبور، با کمک کدگذاری و روشهای پنهان سازی مورد استفاده برای ذخیره آنها در حافظه، گذشتیم، اشاره‌ای داشتیم به پروسیجرهای مقایسه برای کلمه عبور واقعی و آنچه کاربر تایپ کرده بود. بنابراین تا اینجا چندین مورد برای شروع عمل کراک (نفوذ) در اختیار دارید:

- یافتن محل کلمه عبور کاربر
- انعکاس (یا echo) در حافظه برای کلمه عبور واقعی
- یافتن روتینی که آنها را باهم مقایسه کنید.
- یافتن کلمه عبوری که بصورت رمز، جایی که پنهان شده.
- یافتن go-ahead-nice-buyer و خروج یا پرش

- یافتن beggar-off-ugly-copier خروج یا پرش

در برخی حالت‌های کراک کردن با استراتژی‌های متعدد حفاظت روبرو می‌شویم که معروفترین آنها:

- حالت‌های مختلف روتین‌های ذخیره‌سازی، مقایسه و پنهان‌سازی هر سه در کنار هم برای یک کد انجام می‌گیرد.

- پر شدن این روتین‌ها با باگ‌های مقایسه، باگ‌های پرش و باگ‌های متغیرها به منظور مشکل‌تر کردن کراک.

- پخش کد با حقه‌های آنتی دیباگر (ضد دیباگر) مثل دستورات INT3 (وقفه سه) یا پرش‌ها به داخل و بیرون از حالت محافظه شده.

- سعی در از بین بردن نیاز به وارد کردن کلمات رمز. تا حد ممکن.

یعنی به کاربر اجازه می‌دهند تا یک حرف، یک عدد یا یک تصویر را بعنوان پاسخ برخی سؤالات مختلف وارد کند. در این درس ما یاد می‌گیریم که چگونه با این تکنیک‌های حفاظتی مبارزه کنیم.

کلمه عبور بعنوان وز دسترسی

این نوع کلمات عبور مربوط به اطلاعاتی است که کاربر قانونی یا اصلی برنامه در مورد آن دارد. این یک نوع کلمه عبور است که بعنوان مثال میتواند account دسترسی به شما به یک کامپیوتر، روی شبکه یا حتی ماشین ATM مورد استفاده برای بانک یا یک شرکت باشد. این حالت احتیاج به یک نوشتن روی هارد، برای کراک کردن دارد. این نوع کلمه عبور بستگی به پاسخی از کامپیوتر مرکزی دارد. (اینها فقط نه ناحیه مغناطیسی روی کارت را کنترل می‌کنند). خطوطی بین ATM و host های آنها وجود دارد که معمولاً آماده برای چک کردن اینکه اطلاعات مبادله شده روی آنها واقعی است یا بطریقی رمزگذاری شده، می‌باشد. (برخی بانکها از اطلاعات کدگذاری شده استفاده می‌کنند اما اینهم برای کراک کردن نسبتاً آسان است). در مورد ATM ها شما کارهای زیر را باید انجام دهید.

(۱) روی خطوط اختصاصی بین ATM و host (کامپیوتر میزان یا کنترل کننده مرکزی در یک شبکه) خط بطلان بکشید. (۲) کامپیوتر خودتان را بین ATM و host (میزبان) قرار دهید. (۳) به پیغامهای معمولی گوش دهید اما هنوز مداخله نکنید. (۴) سعی کنید برخی عملیات را با یک کارت درست انجام دهید، برخی اشتباهات را مرتکب شوید و به کدهای مختلف توجه کنید (۵) هروقت آماده شدید یک کارت گول زننده داخل ATM کنید، حالا اتفاق زیر رخ میدهد:

ATM یک سیگنال به کامپیوتر میزبان ارسال میکند. با این مضمون «هی! من میتوانم از این شخص پول بگیرم یا اینکه او یک ورشکسته است یا یک کارت غیرمعتبر خنده دار وارد کرده؟»
- میکروکامپیوتر جلو این پیغام را میگیرد و به جای آن این سیگنال را به میزان ارسال می کند «اینجا هیچکس از ATM استفاده نمی کند».

host این پیغام «هیچکس استفاده نمیکند» را می گیرد و پاسخ میدهد «خب مراقب باشید کسی وارد نشود و بخاطر خدا هیچ پولی را در خیابان نریزد» این سیگنال برای ATM ارسال میشود.
- میکروکامپیوتر دوباره جلو این پیغام را میگیرد و آن را دور میریزد و در عوض این پیغام را ارسال می کند «این شخص خیلی ثروتمند است، هرچقدر پول میخواهد به او بدهید. در واقع او خیلی متمول است ، همه صندوق را به او بدهید. او یک مشتری واقعی است».

- ATM محتویات صندوق را به او می بخشد.

همه اینها ممکن است و میتوان کراک کرد مگر آنکه یک نرم افزار حفاظتی روی خط باشد. بنابراین اگر شما بخواهید با ATM کار کنید او با شما برخورد میکند و اعلام میکند که طرف یک هکر است. لطفاً بخاطر داشته باشید صندوقی را هک کنید که یک دوربین کنترل کننده نداشته باشد.

کلمه عبور بعنوان ثبت نام

این نوع کلمه عبور معمولاً برای برنامه هایی که اشتراکی اجرا می شوند، استفاده میشود. وقتی شما از یک برنامه اشتراکی استفاده می کنید، یک کلمه عبور به منظور به روز رسانی و تکمیل برنامه اشتراکی تان

وارد می‌کنید، این روش اخیراً برای کاربردهای تجاری، استفاده می‌شود و امروزه دارای تعداد کمی پنجه کاربردی هستند که در آنها مثلاً برای اینکه سی دی مربوط به یک مجله را بگیرید لازم است به یک خط اصلی تلفن کنید و پولی را پرداخت کنید تا یک کلید ویژه و یکتا، به شما بدهند تا شما بتوانید قفل این نوع حفاظت ویژه را باز کنید. در این موارد شما باید یاد بگیرید که چطور به این پنجره‌ها نفوذ کنید. در آن درسها شما روتینهای متنوعی را می‌بینید که ورود شما را چک می‌کند.

کلمه عبور بعنوان افط کپی

این نوع کلمه عبور معمولاً برای بازیها و سرگرمیها استفاده می‌شود. کلمه مورد نظر معمولاً در هر بار شروع اجرای برنامه یا هر بار بارگذاری آن ظاهر نمی‌شود. بلکه بعد از تکمیل یک یا چند مرحله بازی، ظاهر می‌شود. (این نوآوری، اولین بار توسط EOBi و Ultina به بازار آمد) یا اینکه زمان بارگذاری مجدد یک بازی ذخیره شده، ظاهر می‌شود.

کلمه عبور Dongle

یک برنامه خیلی گران معمولاً از این نوع کلمه عبور استفاده می‌کند. (که یک کلید سخت افزاری هم نامیده می‌شود). dongle، یک وسیله سخت افزاری کوچک می‌باشد که به پورت (درگاه) موازی یا سری، وصل می‌شود. برخی dongleهای خاص حتی شامل روتینهایی از یکسری برنامه‌های تکمیلی می‌باشند. این dongle هم قابل کراک کردن هستند. اما کار بیشتر نیاز دارد. و برای عبور از پروسیجرهای خطا که روی آنها استفاده شده در زمان کراک کردن به نرم‌افزارهای کمکی نیاز داریم که شاید یک هفته نیاز به وقت داشته باشد. سریع ترین روش برای کراک کردن برنامه‌ای که به این طریق حفاظت شده، کار کردن روی قطعه سخت‌افزاری است که به آن متصل شده. شما باید جلو کار کردن آن را بگیرید. من خودم بندرت به این موردها برخورد می‌کنم و به نرم‌افزارهای کمکی برای نفوذ کردن به dongleها علاقه‌ای ندارم. اینها به یک فکر قوی و آزاد و یک وقت زیاد نیاز دارند. اگر می‌خواهید اطلاعاتی در

مورد نفوذ به dongle ها بگیرید سعی کنید با سایتهای قدیمی تر تماس بگیرید. آنها در برخی موارد جوابهای جالبی به شما می دهند.

یک مدیر خوب نرم افزار برای یادآوری یک کلمه عبور مطابق زیر عمل میکند: بهترین کلمه عبور آن است که پنهان شده باشد و بهتر از آن اینکه رمزگذاری شده باشد. در آن صورت امنیت بیشتری متوجه برنامه خواهد بود.

کلمه عبور باید:

- رمزگذاری شده باشد و / یا

- در یک بردار hook شده ذخیره شده باشد و / یا

- در یک فایل خارجی ذخیره شده باشد و / یا

در یک قسمت SMC (کدی که خودش خودش را اصلاح می کند) قرار گرفته باشد.

اجازه دهید در آخر طرحهای حفاظتی مورد استفاده توسط برخی برنامه نویسان مورد استفاده قرار گیرد را درجه بندی کنیم.

- گرفتن کلمه عبور

- حروف به یک کلید اضافه میشود تا وارد شود.

- ترکیب حروف شکل گرفته XOR کردن با 255

- ذخیره یک کلید (یک حرفی)

- خیره کلمه عبور (۲۵۶ حرفی)

- ذخیره Checksum (بصورت یک حرف)، به منظور حفاظت و پیاده سازی دوباره.

- ساخت فایل password.dat با کلمه عبور، که داخل فایل دیگری غیر از آنکه روتینها را صدا میزند درج شده. حالا برنامه نویسان تنبل که میخواهند برنامه شان را حفاظت کنند بدنبال اولین فایلی که کلمه عبور در آن ذخیره شده میگردند و سپس کلید را در آن میگذارند یعنی هم کلمه عبور و هم

checksum . آنها از یک پروسیجر رمزگشایی کلمه عبور و یک پروسیجر کنترل checksum برای چک کردن اینکه کلمه عبور درست وارد شده، استفاده میکنند. همه اینها به راحتی و طی چند ثانیه قابل کراک کردن است.

[دست یابی به کلمه عبور داخل setup]

برخی کامپیوترها دارای یک کلمه عبور هستند که setup آنها را از دسترسی حفاظت می کند. (یعنی در همان شروع کار) این طرح حفاظتی اجازه boot کردن با یک فلاپی یا اصلاح setup را نمیدهد. در این حالتها تنها راه ممکن برای نفوذ استفاده از یک روش قدیمی هک کردن است :

* PC را باز کنید .

* روی مادربرد یک جامپر کوچک با کلمه pw را پیدا کنید .

* آن را بردارید.

* PC را روشن کنید.

* Setup را با f1 یا del اجرا کنید (بسته به نوع bios) (حفاظت بیش از این کار نخواهد کرد).

* داخل setup، قسمت مربوط به password را غیرفعال کنید .

* PC را خاموش کنید .

* جامپر را دوباره سرجایش برگردانید.

* PC را ببندید.

* PC را روشن کنید. کراک انجام شد و اگر بخواهید میتوانید کلمه عبور جدیدی روی آن بگذارید.

اگر میخواهید اطلاعات بیشتری در مورد دسترسی و عدم دسترسی، رمزگذاری و قفل کردن جدول FAT داشته باشید، آنها را از اینترنت بگیرید و مطالعه کنید که ویروس Monkey که خیلی هم خوب نوشته شده از این نوع تخریب استفاده میکند. مطالعه ویروسها کلاً برای کراک کردن خیلی خوب است.

کدهای وبروسها

- خیلی خوب نوشته میشوند (خالص، به زبان اسمبلی)
- با استفاده از تکنیکهای مخفی نه چندان متفاوت از طرحهای حفاظتی (اغلب پیشرفته)
- با استفاده از جدیدترین و بهترین حقه‌های SMC (کدهایی که خودشان خودشان را تغییر میدهند)
- اما آنچه مهم است این است که باور نکنید که طرحهای حفاظتی خیلی کامل و پیچیده هستند. بیشتر وقتها حفاظت استفاده شده به طرز عجیبی معمولی و ساده است. بعنوان آخرین مثال مقاله طرحهای حفاظتی ما اجازه بدهید یک برنامه نه خیلی قدیمی (۱۹۹۴) را بگیریم و ببینید چه طرحهای حفاظتی مضحکی دارد. (ببر در پی شکار)
- یک شبیه سازی از HPS. کراک کردن در اینجا خیلی واضح پیش می‌رود :
- با استفاده از نقشه حافظه مورد استفاده، سکتور اصلی را پیدا کنید.
- AAAA را بعنوان کلمه عبور تایپ کنید.
- سکتور اصلی را بدنبال AAAA از 0 تا 1FFFF جستجو کنید.
- dump L80 AAAA را از آدرس 40 (یک محدوده وسیعی را به وسیله dump بگیرید).
- به این طریق شما echo مربوط به کلمه عبور درست را میگیرید.
- یک break point خواندنی و نوشتنی روی حافظه برای محل AAAA انجام دهید و بدنبال آن یک اجرای مرحله به مرحله روی روی سکتور اصلی، کار را تکمیل خواهد کرد.
- در اینجا سطرهای کد حفاظت شده بالا آمده:

```
8A841C12 MOV AL,[SI+121C]  move in AL first user letter
3A840812 CMP AL,[SI+1208]  compare with echo
7402     JZ  go_ahead_nice_buyer
EB13     JMP beggar_off_ugly_cracker
```

حالا بیایید به سرعت آن را کراک کنیم:

CRACKING TOP.EXE (by +ORC, January 1996)

ren top.exe top.ded

symdeb top.ded

- s (cs+0000):0 Lfff 8A 84 1C 12 3A 84

xxxx:yyyy (this is the answer of the debugger)

- e xxxx:yyyy+2 08 (instead of 1C)

- w

- q

ren top.ded top.exe

و شما دستور MOV AL,[SI,121C] را به MOV AL,[SI+1208] تغییر دادید. .. حالا در حال

خواندن ECHO به جای کاراکترهای تاییی شما میباشد. در مقایسه ECHO هم مشکلی پیش نمی آید

و حالا شما موفق شدید.

برخی پنهان کارهای مشکوک

به حروف رمز تایپ شده برنامه قبل برگردید، بیایید یک بازی حفاظت شده ۱۹۹۰ را باهم بررسی کنیم یعنی بازی F19، در جایی که طرح حفاظتی، از شما برای شناسایی هواپیما، کد خاصی را می خواهد این نوع حفاظت برای دوری جستن از ذخیره کلمه عبور در خانه های حافظه بکار می رود. که ما طی اولین آزمایش ماده برای ورود کلمه عبور متوجه می شویم که چنین طرحی خیلی آسان قابل نفوذ می باشد.

در این حالت برای کراک کردن ما میتوانیم از یک تکنیک شناخته شده بنام MEMORY SNUFFING استفاده کنیم. برنامه حفاظت شده start.exe. خودش در محل 0000: xxxx با طول 6C62 بایت نصب میشود اما ماژولهای آن (به همراه برخی کدهای SMC که خودشان خودشان را تغییر

میدهند) در محل‌های مختلف حافظه قرار می‌گیرد. همه اینها به چه مفهومی است؟ خیلی چیزها میتوان از آن فهمید که مهمترین آنها این است که حفاظت آنها بستگی به ورودی کاربر دارد. حالا شما سریعاً متوجه خواهید شد که روتینی که بطور تصادفی هواپیمایی را انتخاب میکند، بطور تصاعدی شماره این هواپیما را در محلی از حافظه یعنی CD:DADA 43 نگهداری میکند. این مکانیسم تصادفی‌سازی آن را بصورت زیر مشخص میکند :

```
E87FAF CALL random_seed
```

```
83C402 ADD SP,02
```

```
8946E8 MOV [BP-18],AX and ds:(BP-18) is the location  
you are looking for
```

DS:(BP-18) محلی است که شما بدنبال آن می‌گردید.

حالا هربار که فرآیند تصادفی سازی رخ می دهد شما عدد متفاوتی (00-x14) در این محل می‌بینید که معادل هواپیماهای مختلفی است که کاربر انتخاب میکند. روتین تصادفی سازی‌یی که دیدید عدد تصادفی را در Ax ذخیره میکند که حالا ما باید آن را صفر کنیم. و در آن صورت کاربر هواپیمای صفر را همیشه انتخاب خواهد کرد و ما همیشه جواب صحیح خواهیم گرفت.

می‌بینید کار به چه زیبایی انجام میشود. ما نه نیازی به مداخله در روتینهای مربوط به انتخاب با ماوس داریم و نه نیازی به انتخاب هواپیمای واقعی. تصادفی سازی، هواپیمای دلخواهش را انتخاب میکند، اما خانه حافظه مربوطه محتوی صفر خواهد بود.

CRACKING "F19" [START.EXE] (by +ORC, January 1996)

```
ren start.exe start.ded
```

← اجازه دهید یک فایل deal را بگیریم

```
symdeb start.ded
```

← اجازه دهید آن را دیباگ کنیم

```
- s cs:O lffff 83 C4 02 89 46 E8
```

← بدنبال ADD SP02 بگیریم

xxx:yyy

- W

- q

```
ren start.ded start.exe
```

← exe باز نویسی

شما لازم بود دستور مورد جستجو را بصورت زیر منتقل می کردید.

58 POP AX

← ADD SP,+02 رجوع به

31C . XOR AX,AX

XOR کردن به منظور صفر شدن ←

(دستور POP AX، اشاره گر Stack را برای ارجاع به SP،+02 ADD دو تا جلو می برد.)

خوب، زیبا و ساده، اینطور نیست؟ حالا اجازه دهید مثالی در مورد حفاظتی که هیچگونه echo به حافظه ندارد داشته باشیم. (برای شروع با این ایده زیرکانه پیش بروید که کراکر، کلمه عبور صحیح را نمی‌یابد. و آن اینجا نیست. بنابراین ما در اینجا یکی از اولین برنامه‌های از این نوع را کراک می‌کنیم بنام:

[poplous. Exe] , $\models$ Bullfrog

[populous. Exe]

یک مثال قدیمی از طرح حفاظتی «کلمه عبوری که یک کلمه عبور نیست» را میتوانید در [populous. Exe] بیابید که توسط Bullfrog نوشته شده. که یک برنامه خیلی رایج می باشد و شما حتماً خواهید توانست یک کپی آن را برای این درس پیدا کنید. برنامه از شما، شناسه یک محافظ یا سپر ویژه را درخواست می کند که به صورت ترکیبی از حروف با طول مختلف می باشد. خانه حافظه ای که کلمه عبور کاربر در آن ذخیره شده به آسانی قابل پیدا کردن است. اما در ظاهر هیچگونه echoیی از کلمه عبور واقعی وجود ندارد. حالا دیگر شما باید بتوانید خانه حافظه یک کلمه عبور در آن

ذخیره شده را پیدا کنید. یک break point خواندنی و نوشتنی روی این ناحیه حافظه تنظیم کنید.

بزودی به بخش کد که در ذیل آمده برسید :

F7AE4EFF IMUL WORD PTR [BP+FF4E] <- IMUL with magic_N

INC AX

۴۰^۳

3B460C CMP AX, [BP+0C]

7509 JNZ beggar_off_ugly_copier

8B460C MOV AX, [BP+0C]

A3822A MOV [2A82], AX

E930FE JMP nice_buyer

817E0C7017 CMP WORD PTR[BP+0C],1770 <- beggar_off

من فکر نمی کنم شما به اطلاعات بیشتری نیاز داشته باشید. حالا شما برای نفوذ به این طرح

چکار می کنید؟ آیا درج یک MOV [BP+0C],AX و سه تا دستور (شش بایت) NOP. بعد از

دستور IMUL را ترجیح میدهید؟ آیا شما ترجیح نمی دهید که بجای JMP nice-guger دستور

JNZ beggar-off را قرار دهید؟ که در آن صورت تعداد nopها به حداقل میرسد و این همان است که

در طرحهای حفاظتی جدید رعایت میشود. بیاید این روش را دنبال کنیم:

CRACKING [Populous.exe] (by +ORC, January 1996)

ren populous.exe populous.ded <- let's have a dead file

symdeb populous.ded <- let's debug it

- s cs:O lffff F7 AE 4E FF <- the imul magic_N

xxxx:yyyy <- debugger's answer

- e xxxx:yyyy+4 EB [SPACE] 03 <- JMP anyway

- w <- modify ded
 - q <- back to the OS
- ren populous.ded populous.exe <- let's re-have the exe

این دفعه ساده بود. نه؟

حالا شما برای کار در این زمینه آماده هستید . اجازه دهید کاربرد گذشته را کراک کنیم. و آن یک یوتیلیتی (کاربردپذیری, مفیدیت) خیلی رایج حافظه می باشد، که برای هدف ما خیلی خیلی مفید است (و شما بعدها برای کراک کردن برنامه‌های TSR، یعنی برنامه‌های مقیم در حافظه، از آن استفاده خواهید کرد) با قفلها مشکلی نخواهید داشت. و برای سطحی که شما الان هستید، مناسب می‌باشد. هی ! مراقب باشید که شما هیچوقت نباید بدون این خودآموز آن را انجام دهید. پس مطابق زیر عمل کنید: از پنجره‌تان نگاهی به شرق بیندازید، جرعه‌ای ودکای مارتینی بنوشید (ابتدا دو قطه یخ، ۱/۳ از مارتینی خشک از مارتینی و رزی، ۱/۳ ودکای MOSKOVSKAIA . ۱/۳ Schweppes Indian tonic)) و سه بار بگویید , متشکرم . +ORC

[map, exe]

حالا اجازه دهید با یکی از قویترین ابزارهای موجود map کردن حافظه یعنی MAP.EXE (نسخه ۲) از نرم افزار Clockwork کار کنیم. استفاده از این نرم افزار در درس ۲ برای شما گفته شده و در اینجا روش کراک کردن آن گفته می شود. این برنامه هیچگونه nag screen ای ندارد، بجای آن Nigle screen دارد که همه جا حاضر است، بمنظور ماندن برای یک مقدار تصادفی زمان قبل از اینکه از کاربر بخواهد کلیدی بزند، که همیشه به اندازه یک مقدار تصادفی منتظر می ماند.

استفاده از یک حرف که معمولاً با برخی XOR یا SHRها بعنوان کلمه عبور رمزگذاری شده که همان محل اولیه مورد استفاده برای مقایسه snap در حافظه خواهد بود. تکنیک نفوذ در اینجا تا حدی رک و بی پرده خواهد بود. یعنی به راحتی break می کند و نظری به اطراف می افکند.

روتین وقفه ۱۶ (INT 16) برای خواندن صفحه کلید، درست بعد از بارگذاری nag screen، فراخوانی میشود. شما به سرعت روتین LODSB را داخل نقاشی های روی صفحه پیدا خواهید کرد که کلمه press (یا فشار دهید) و یک جعبه بعد از کمی تأخیر، ظاهر خواهد شد.

```
B95000      MOV CX,005 0
```

```
2EFF366601  PUSH CS:[0166]
```

```
07          POP ES
```

```
AC          LODSB
```

...

شما قبلاً توانسته‌اید از این تأخیرها عبور کنید و برای تخمین میزان حساسیت حفاظت توانسته‌اید به زور از این مراحل عبور کنید اما حالا میخواهیم نفوذ عمیق‌تری داشته باشیم. پس اجازه دهید با برگشتن به فراخوان کار را انجام دهیم. روتین قبلی از بخش کد ذیل، فراخوانی شده :

```
91          XCHG AX,CX
```

```
792         XCHG AX,DX
```

```
28939193    SUB [BP+DI+9391],DL
```

```
2394AA94    AND DX,[SI+94AA]
```

```
2EC7064B880100 MOV WORD PTR CS:[884B],0001
```

```
2E803E5C0106  CMP BYTE PTR CS:[015C],06
```

```
7416        JZ  ret    <- Ha! jumping PUSHa & POPa!
```

```
505351525756  PUSH the lot
```

```
E882F3       CALL 8870
```

```
2E3B064B88    CMP AX,CS:[884B]
```

```

7307          JAE after RET <- Ha! Not taking the RET!
5E5F5A595B58  POP the lot

C3           RET
...          <- some more instructions

E86700       CALL delay_user
BE9195       MOV SI,9591
2E8B3E255C   MOV DI,CS:[5C25]
83EF16       SUB DI,+16

2E8A263D01   MOV AH,CS:[013D]
50           PUSH AH
E892C7       CALL routine_LODSB <-- HERE!
B42C        MOV AH,2C
CD21        INT 21      <- get seconds in DH

80E60F       AND DH,0F
80C641       ADD DH,41
58           POP AX
8AC6        MOV AL,DH
83EF04       SUB DI,+4
AB           STOSW
E85A00       CALL INT_16_AH=01
B400        MOV AH,00
CD16        INT 16
24DF        AND AL,DF   <- code user's letter_answer
3AC6        CMP AL,DH   <- pass_compare
75F3        JNZ CALL INT_16_AH=01
E807F3       go_ahead

```

شما باید نگاهی به این دستورها بیندازید. تا آن را احساس کنید. من فکر میکنم (در این حفاظت) سگمنت کد

غیرضروری، در یک حالت مارپیچی زیر پوشش حرکت میکنند بطوریکه شما هنوز به آسانی نمی‌توانید بگویید که چه اتفاقی می‌افتد. اما باید حدس زده باشید که یک اتفاقات مشکوکی رخ میدهد. نگاهی به فراخوانی روتین LODSB قبلی‌تان بیندازید.

شما در آنجا دو پرش داشتید.

یکی JZ ret که pusha و popa های زیادی را رها می‌کند. و دیگری JAE بعد از RET میباشد که RET قبلی را نمی‌گیرد. اگر به برخی چیزها در اینجا بخت کنید، کار درستی کرده‌اید. JZ اولی nag screen را راه اندازی میکند و دومی یعنی JAE هم همان کار را انجام میدهد (یعنی در اینجا، طبق معمول، افزونگی وجود دارد. یعنی احتمال زیادی برای غیر فعال کردن یک حفاظت وجود دارد). حالا فهمیدید.

شما می‌توانید به راههای مختلف این حفاظت را غیرفعال کنید. دو تا از ساده‌ترینها در زیر آمده :

۱) تبدیل 7416 (JZ ret) به یک EB16 (بهر طریق JMP RET)

۲) تبدیل 7307 (بعد از JAE RET) در یک 7306 (JAE RET)

هنوز تمام نشده : اگر شما سعی کنید این قسمت کد را تغییر دهید، شما روز خوشی نخواهید دید. چون این یک SMC یعنی کدی که خودش خودش را تغییر میدهد، میباشد. از بخش دیگر کد، بارگذاری می‌شود. (و در اینجا بدون هیچگونه رمزگذاری خواهد بود). پس شما قبل از هر کاری باید یک break point روی محدوده‌ای از حافظه داشته باشید؛ روتین LODSW را پیدا کنید؛ ناحیه واقعی را پیدا کنید؛ آن قسمت از حافظه را dump کنید؛ بدنبال یکسری کدهای dead بگردید و در نهایت برنامه dead را اصلاح کنید. حالا بیایید سریع کراک کنیم :

CRACKING MEM.EXE (version 2) (by +ORC, January 1996)

ren map.exe map.ded

symdeb map.ded

- s (cs+0000):0 Lfff 74 16 50 53 51 52 57

xxxx:yyyy <- this is the debugger's answer

- e xxxx:yyyy EB

- w

- q

ren map.ded map.exe

حالا شما اینکار را انجام دادید. Nigel کراک شد.

خب، این بود درس امروز. اما همه خودآموزهای من روی اینترنت نیستند، شما درسهایی را که فراموش کرده‌اید به من mail بزنید. شاید شما کلک‌هایی بلد باشید که من هنوز کشف نکرده‌ام. من همان قبلی‌ها را میدانم اما اگر چیز جدیدی باشد اعتبار شما را خیلی زیاد می‌کند. حتی اگر اینطور هم نباشد من می‌فهمم که شما خیلی روی موضوع کار کرده‌اید. در آنصورت من درسهای باقیمانده را برای شما خواهم فرستاد. انتقادات و پیشنهادات شما در مورد چرندیاتی که من نوشتم، همیشه برای من خوش‌آمد خواهد بود.

E-mail +ORC

an526164@anon.penet.fi