¡¡Mis saludos, pequeñas sabandijas, gusanos de la programación, amebas de las ciénagas del ASM!! Este tutorial va dedicado a vosotros, los malditos, los hijos bastardos de un dios menor, los desarrapados, perseguidos por lo "políticamente correcto",los odiados, los temidos, los nefandos y oscuros delincuentes que maquinan confabulaciones contra Gates y su prole de refulgentes dioses del dinero y nuevas tecnologías.

Este tutorial va dedicado a mis amigos, los crackers, ingenieros inversos, aquellos desheredados que solo quieren aprender, a los que se les niega el conocimiento, a los frekies feos y gordos, bebedores de coca cola y guarradas con alcohol, a los deshechos de un mundo que se ha vuelto loco y codicioso, a los que realmente son seres humanos, con afán de superación y evolución, con la suficiente paciencia como para leer ASM y modificarlo, a los nauseabundos piratas cuyo delito es preguntar y curiosear.

Este tutorial va dedicado a vosotros...¡¡mis maestros!!.

Y después de esta agresiva y romántica introducción pasemos a...

KEYGEN PARA BATTLEFLEET 1.0 FOR PALM

INTRODUCCION:

Apenas decir que este tuto es la continuación del anterior en el que parcheamos el programita y por el camino nos encontramos el serial y toda la rutina de checksum. Me remito a lo que dije sobre las herramientas necesarias para llevar a buen término este barco pirata...

BATTLEFLEET 1.0 SEGUNDA PARTE:

La película nos había dejado intrigados con un bonito parche que hacíamos una vez desensamblado el programita y por el camino nos había dado por utilizar PalmDeMon y el debugger tan simpático que nos regala PalmOs (son tan buenos y nos quieren tanto que regalan programas... por si acaso, yo no me fio mucho... ahí hay truco seguro...). Pero bueno, salimos del cine, con los ojos rojos y medio zumbados después de tres horas de peli y directos al McMierda a comer una hamburguesa, un colega nos mira a nuestra cara de pobres diablos y nos espeta a la cara: "tío, no me he enterado de la mitad del argumento" y nosotros, almas cándidas, suspiramos mirando al cielo y armados de paciencia le explicamos por qué Norman Bates se disfrazaba de su madre para cargarse chicas que caían en su hotel: "Manu, macho, es que era drag-queen, una locaza...". A ver, te explico...

Y CUANDO ESTAMOS DEBUGGEANDO

Si mal no recordáis, en el último tuto habíamos hecho una serie de pasos hasta llegar al debugger. Como temo repetirme y aburriros, os los resumo a grandes rasgos:

1- Desensamblado el programita, buscamos el Talt del mensaje malo y feo y lo localizamos. Para hacer bien este punto uno, leeos mis otros dos tutos.

2- Tras echar un vistazo, notamos una llamada a la label 56, que es la del checksum. En PalmDeMOn también lo vemos claro. Lástima que Carpathia haya desaparecido de la arena, necesitábamos un desarrollo más profundo de esa herramienta :o(Los buenos desaparecen, mueren en las inmensidades del mar del olvido, bailando y bebiendo con las sirenas por la eternidad. Ya se me salta una lagrimita... debo ir viejo...

3- Fieles a nuestro desmedido afán de conocimiento e insaciable sed de sabiduría, ejecutamos PalmOs emulator y cargamos el programa. Aquí me voy a parar un poco y retomamos la milonga esta...

ENTONCES ES CUANDO DESCUBRE SU LADO FEMENINO, MIRANDO EL CUCHILLO DE COCINA

Hasta ahora el PalmOs emulator lo he mencionado de pasada. Es muy bueno, creedme, y lo podéis descargar gratuitamente desde la web de Palm. Es la nueva versión del antiguo Pilot, y juro que es de los mejores emuladores que he visto en mi vida. Os recomiendo echarle un ojo.
(Hummmm... a veces me pregunto si alguien leerá mis tutos o si serán útiles).

Bajaos también PalmDebugger de la web de PalmOs, si no, difícilmente podremos meternos con listado vivo.
Ejecutamos el Emulator de marras y vemos algo asi como que nos sale una ventanita pidiéndonos una ROM y tamaño del archivo que se va a crear virtualmente. Las ROMs las podemos obtener de nuestra Palm, de alguna web de warez (yo miraré a otra parte, os lo juro) o descargándolas de PalmOs pidiendo permiso y firmando una licencia y eso. Bueno, elegimos nuestra ROM y cualquier tamaño de RAM para la supuesta capacidad de memoria de nuestra Palm virtual. Poner una carátula bonita hace más agradable nuestro trabajo.

Acto seguido, arrancamos la máquina y tras tocar con el ratón diversos puntos de la pantalla para calibrarlo y precisarlo, nos vemos de lleno en la ventana inicial de la Palm (¿he dicho "ventana"... oh, Dios, ¡¡soy carne de Gates!!). Botón secundario del ratón y le damos a instalar aplicación, donde escogeremos BattleFleet. Esto va por gustos, pero si escogéis otra aplicación de poco os sirve este tuto, je je... Luego le damos a HotSync y pulsando el icono de la casita de abajo volvemos a la pantalla principal. Juas, ahora empieza el tiroteo.

Cargamos el debugger y tras ejecutarlo con:

att (y luego enter.Fijaos bien que en el menú de Connection esté seleccionado el Emulator)

tecleamos:

atb "FrmAlert"

que en lengua no-hereje significa algo así como:

bpx "MessageBox"

que para los que estudiaron Humanidades se traduce como:

párate cuando aparezca un aviso

Tras escribir eso, ponemos:

g

que es el comando que nos permitirá volver a nuestro amado emulador. Para los curiosos, la g es de "go". ¡Uau!, apabullante razonamiento ¿verdad?.

Y sin comerlo ni beberlo estamos en el PalmOs Emulator como quien no quiere la cosa. Ya tenemos una systrap husmeando el aire para saltar de un momento a otro. Tocamos con el raton encima del juego-objetivo de nuestro estudio para que se ejecute y vemos un horrible dibujo de un gorila (el John Love este creo yo que andaba un poco pasado de LSD cuando programó la mierda esta...) y se nos carga el BattleFleet sin problemas. Vamos a registration, metemos un número fake ("basura" en lengua cristiana) y le damos a aceptar.
NOTA: si en algún momento de este proceso saltase el debugger, basta con poner

g

y continuar como quien no quiere la cosa. Fácil...

Bueno, llegados a este punto, chis-pum, bloqueo de emulador y debugger sonriente, peinado con raya al medio y reluciente de gomina. Buen cazador, el chico, ha dado de lleno con un FrmAlert interesante. Tres anotaciones en este punto:

1- Poned el atb antes de cargar el programa objetivo.
2- El PalmDeMOn abierto a la vez, en listado muerto, os ayudará mucho a no perderos. Veis perfectamente dónde andáis.
3- Ummm, éste se me olvidó... x)

Y NO VEAS, TIO, LA COCHINADA QUE HIZO CON EL CUCHILLO DE COCINA Y LA DE SANGRE QUE SALTABA

Ahora, al fin, estamos propiamente en el tema de este tutorial. Vamos a localizar nuestro número fake (esta palabreja la aprendí en el ircnet en #palmwarez, pero no os recomiendo ir ahí, son muy desagradables. Más majos son los chicos del irc-hispano en #crackers y si no, como ejemplo pongo a SunEvil, cybdan, TGILITO, DS, jonas_, juasss o a eSn-mIn). Digo que buscaremos nuestro serial basura, veremos cómo lo halla y haremos un keygen. Como dijo Jack el Destripador, "Vayamos por partes".

1- El numerito:

Con el atb que pusimos aparecemos en la siguiente sección de código:

```
0000801c  6314                          BLS     L872
0000801e  3f3c076c                      MOVE.W  #1900!$76c,-(A7)
00008022  4e4fa192                      sysTrapFrmAlert
```

y si observamos el listado muerto, no se nos escapa otro FrmAlert más arriba en

```
00008004  3f3c07d0                      MOVE.W  #2000!$7d0,-(A7)
00008008  4e4fa192                      sysTrapFrmAlert
0000800c  38bc0016                      MOVE.W  #22!$16,(A4)
```

ese es el mensaje de bad boy, el de "Demo Mode for 7 days". Ahora os pongo el código que nos interesa, que es

```
00007fca  4e4fa153                      sysTrapFldGe tTextHandle
00007fce  2448                          MOVEA.L A0,A2
00007fd0  200a                          MOVE.L  A2,D0
00007fd2  584f                          ADDQ.W  #4,A7
00007fd4  671c                          BEQ     L870
00007fd6  2f0a                          MOVE.L  A2,-(A7)
00007fd8  4e4fa021                      sysTrapMemHandleLock
00007fdc  584f                          ADDQ.W  #4,A7
00007fde  2f08                          MOVE.L  A0,-(A7)
00007fe0  486dfbca                      PEA     -1078(A5)
00007fe4  4e4fa0c5                      sysTrapStrCopy
00007fe8  2f0a                          MOVE.L  A2,-(A7)
00007fea  4e4fa022                      sysTrapMemHandleUnlock
00007fee  4fef000c                      LEA     12(A7),A7
00007ff2  4eba8570              L870    JSR     L56
00007ff6  2b40fd66                      MOVE.L  D0,-666(A5)
00007ffa  0cad0000014afd66              CMPI.L  #330!$14a,-666(A5)
00008002  6410                          BCC     L871
00008004  3f3c07d0                      MOVE.W  #2000!$7d0,-(A7)
00008008  4e4fa192                      sysTrapFrmAlert
0000800c  38bc0016                      MOVE.W  #22!$16,(A4)
00008010  544f                          ADDQ.W  #2,A7
00008012  6074                          BRA     L876
00008014  0cad00000198fd66      L871    CMPI.L  #408!$198,-666(A5)
0000801c  6314                          BLS     L872
0000801e  3f3c076c                      MOVE.W  #1900!$76c,-(A7)
00008022  4e4fa192                      sysTrapFrmAlert
```

de ahí abajo ya no llega la ejecución del programa. Os he resaltado las sentencias que nos saltamos siempre que estemos en el trial de los 7 días para probarlo. Mirando tan solo las cosas que nos interesan aquí, tenemos que

```
00007fca  4e4fa153                      sysTrapFldGetTextHandle
```

lee nuestro serial basura por primera vez. Luego, ya a la altura de

```
00007fd6  2f0a                    MOVE.L      A2,-(A7)
```

nuestro fake serial anda por el registro de datos A0 y el de direcciones A0 (recordad que A7 es la pila). ESo lo sabemos tecleando

dm A0

y nos enseña la memoria del registro A0. otros comandos útiles son

il

que nos muestra las 10 siguientes líneas de código,

s

de "step" que nos permite ejecutar una línea de código en el debugger y

t

que ejecuta una línea pero saltando por encima de las funciones (de "trace").

Bueno, entonces "¿ese Norman Bates realmente crees que estaba loco?". No entendisteis nada... ¡arggggggg!.

Esta es la línea de código en la que aparezco

```
0000801e  3f3c076c                MOVE.W      #1900!$76c,-(A7)
```

a lo que llego después de que

```
00007ffa  0cad0000014afd66        CMPI.L      #330!$14a,-666(A5)
00008002  6410                    BCC      L871
```

una comparación me lleva a la Label 871 si el registro A5-666 (en hexa sería la dirección A5-29a) no contiene nada. Significa Branch if Carry Clear.
Así que si en 00007fe8 vemos mi serial basura en A0 y D0, en 00007ff2 me salta obligatoriamente a la Label 56 y en 00007ffa hay una comparación (que como dice mi madre son siempre odiosas) entonces... ¿¿qué diablos pasa con la Label 56??. Sólo vosotros, Dios y John Love lo saben. Es nuestro Checkregistration. Bueno, ejem, el debugger también lo dice, pero eso es otra historia X)

2- Cómo lo halla:

Para saber dónde pararnos debemos utilizar el código muerto a la vez que el vivo, asi que miraremos posibles SysTrap del codigo muerto para poner atb estratégicos que nos vayan cantando lo que tienen los registros, así que ni cortos ni perezosos, le introducimos

g

al debugger para salir del proceso, borramos los atb con

atc

(atd nos muestra todos los breakpoints) y volvemos a la ventanita principal de Palm saliendo antes del programa. Si hubiese algún problema, basta con re-conectar el debugger con att o saliendo y cargando de nuevo todo desde el principio. Pero ahora los atb serán más astutos y no perseguiremos la usada SysTrapFrmAlert (nota importante: el debugger es "case sensitive". Un atb mal puesto y no parará. Me llevó

mucho averiguarlo). Ahora miramos bien el listado muerto y escribimos estos atb como mero sondeo del cálculo del número, a fin de ir viendo los registros:

atb "DlkGetSyncInfo"

éste es bueno, nos para la ejecución en el momento en que se obtiene el User con el que HotSync se entiende con la PDA. En dos líneas, sería como un login al iniciar una sesión Linux o un usuario en Windows. Ese User ID se guarda en el programa HotSync que coordina la PDA con el Pc. Por lo tanto, vemos que nuestro número se genera al partir de nuestro nombre de User... adiós, esto atufa a keygen.
Antes de poner otro atb, debemos analizar un poco el código muerto para ver lo que nos vamos a encontrar, ya que ejecutando el debugger con g tras poner ese atb y accediendo a BattleFleet, nos vamos a dar de morros con

```
00000610   4e4fa2a9                    sysTrapDlkGetSyncInfo
```

que es lo que queríamos hacer, justo donde se lee nuestro User. Para no entrar en todo ese proceso, pulsamos t en el debugger y vamos directos fuera de esa llamada a API Palm. Luego un il nos mostrará las 10 líneas que vienen por delante.
Al llegar a

```
00000614   486dfba1                    PEA       -1119(A5)
```

tenemos algo importante. Un PEA carga algo en memoria, en este caso, el registro de dirección A5-1119. Para verlo, le damos a s hasta llegar a la siguiente línea de código y una vez ahi, tecleamos

```
dm A5-45f
```

que es lo mismo pero en valores hexa (sería igual a ver lo que tiene la dirección A5-1119). Otras veces está más adelante, en cuyo caso sería, por ejemplo, A5+1, y que se expresaría como PEA 1(A5). Pues bien, miramos ahí y ¡oh sorpresa! aparece sonriente mi nombre de User, que en mi caso es "PalmOs emulator". Luego vemos en

```
00000618   4e4fa0c7                    sysTrapStrLen
```

una llamada a API que computa la longitud de un string, devolviendo tal longitud en bytes. Vaya, vaya, esto promete.
"La tía rubia que se cargó en la bañera no estaba nada mal ¿verdad?". Pues la verdad es que no...
Bueno, dejamos aparcado un momento el listado vivo y volvemos a leer el muerto. Nos marea un poco el User ID de un lado a otro hasta que llegamos a lo realmente interesante

```
00000652   41eeffa8                    LEA       -88(A6),A0
```

nos carga el User ID en A6-58 (hexa) que pasa a A0 letra a letra y ATENCIÓN CHICOS

```
00000658   3d700000ff7c                MOVE.W    0(A0,D0.W),-132(A6)
0000065e   3006                        MOVE.W    D6,D0
00000660   0640014d                    ADDI.W    #333!$14d,D0
00000664   7200                        MOVEQ     #0,D1
00000666   3200                        MOVE.W    D0,D1
00000668   82eeff7c                    DIVU.W    -132(A6),D1
0000066c   3d41ff7a                    MOVE.W    D1,-134(A6)
00000670   3d46ff74                    MOVE.W    D6,-140(A6)
00000674   3d6effa8ff78                MOVE.W    -88(A6),-136(A6)
0000067a   302eff78                    MOVE.W    -136(A6),D0
0000067e   c0eeff7a                    MULU.W    -134(A6),D0
```

Ahora es cuando nos preparamos un vodkila, una pizza de esas baratas que hemos dejado en la nevera desde ayer por la noche, está fría y nos ha granjeado el apodo de "Carroñero" entre los amigos, un poco de música de agrado (últimamente a mí me ha dado por un grupo llamado Lacrimosa. Excelentes los germanos esos) y una infinita paciencia.

X-Grimator, del grupo La Tristeza de Moldavia (único componente, jeje) os va a explicar todo ese rollo y por qué la madre de Norman Bates tenía un vestido tan feo y una voz tan ronca (sospecho que era un problema de alcoholismo).

```
00000658  3d700000ff7c                    MOVE.W      0(A0,D0.W),-132(A6)
```

ahí tenemos el primer número del serial. Os cuento (y aquí le debo un favor al dr_funk que me ayudó con esto): la estructura del serial es la siguiente:

UBF#1#2-#3#4#5

donde UBF lo pone siempre, y las variables #1, #2, #3, #4, #5 se calculan. El guión entre la segunda y la tercera va siempre.
Os decía que en esa línea de código se calcula el valor de la **primera variable**, y será el valor ASCII del último carácter del User ID, en decimal. Después, en

```
0000066c  3d41ff7a                    MOVE.W      D1,-134(A6)
```

obtenemos la **segunda variable (#2)** con la siguiente fórmula (#4+333)/#1.
Luego os explico como llegué a esto. Y en

```
00000670  3d46ff74                    MOVE.W      D6,-140(A6)
```

nos calcula la **variable #4** sumando en ASCII todos los caracteres que componen el User Id (y que se recoge en D6). Después tenemos en

```
00000674  3d6effa8ff78                MOVE.W      -88(A6),-136(A6)
```

la **variable #3**, que es el valor ASCII en decimal del primer caracter del User Id, y por último en

```
0000067e  c0eeff7a                    MULU.W      -134(A6),D0
```

sacamos la **variable #5** multiplicando la #3 por la #2.

Bueno, así vemos muy bonito todo pero tuvo que haber algo que nos ayudase en todo eso ¿verdad?. El secreto de todo esto está en una llamada a API que se repite cinco veces:

sysTrapStrCat

que convierte a string un número calculado. La encontramos en

000006ba
000006e4
00000728
00000752
0000077c

de ahí que deducimos que nos calcula cinco variables, a las que llamamos #1, #2, #3, #4, #5. Es decir, calcula un número y lo pasa a string en ASCII. Y en cuanto a la estructura que va a tener el serial la encontramos en

```
00000686  41fa022e                    LEA     L67,A0
0000068a  4850                        PEA     (A0)
```

donde vemos en A0 algo así como "UBF.....-.....", que para nuestro caso pone un punto (.) por cada numero necesario para el User "PalmOs emulator" (nota: os recuerdo que aquí una variable puede tener varios números, repasad unas líneas arriba cómo se calculan. Por ejemplo, veremos que #1 es 114). Pues bien, ya sabemos cómo es la estructura del serial, dónde calcula las variables y con qué llamada a API pasa el número calculado a número "entendible". Francamente, Norman Bates está como una regadera. Sólo él podría protagonizar "Psicosis".

Si observamos justo debajo de las líneas de código donde se hace una llamada a la API SysTrapStrCat y que os señalé arriba, vemos que después de ellas hay siempre un PEA, que os recuerdo que cargaba algo en una dirección. De este modo vemos

```
000006be   486efffa                PEA      -6(A6)

000006e8   486efffa                PEA      -6(A6)

0000072c   486efffa                PEA      -6(A6)

00000756   486efffa                PEA      -6(A6)

00000780   486efffa                PEA      -6(A6)
```

Santo Dios, pero esto ¡va viento en popa!. Con sólo observar A6-6 en cada uno de ellos vamos sacando los numeritos:

114
15
80
1461
1200

y en

```
0000079e   486eff7e                PEA      -130(A6)
```

tenemos nuestro serial entero, para un HotSync UserID de "PalmOs emulator", que sería

UBF11415-8014611200

Y la magia está en

```
0000079a   486dfbca                PEA      -1078(A5)
0000079e   486eff7e                PEA      -130(A6)
000007a2   4e4fa2ee                sysTrapStrNCaselessCompare
```

donde la primera línea tiene mi serial fake y la segunda el correcto, de manera que la llamada a API compara los dos strings de n caracteres sin ser case sensitive ni accent sensitive. En este punto intenté modificar la llamada a API por otra de FrmAlert, de manera que me presentase una Talt en pantalla (lo que llamamos messagebox en Windows) con el número correcto. Pero al pobre X-Grimator siempre le salían errores y no entiende por qué... seguiré en el tema, a ver qué ocurre.

Bueno, entonces hemos seguido esta táctica:

1- Os dije cómo se calcula cada variable
2- Vimos dónde calcula cada una y el numerito que nos da a cambio
3- Ahora vamos a repasar el cálculo de cada variable.

Tenemos que observar dos cosas, en primer lugar estas líneas de código, justo al principio del cálculo de cada numerito:

```
000006b0   302eff7c                MOVE.W   -132(A6),D0

000006da   302eff7a                MOVE.W   -134(A6),D0

0000071e   302eff78                MOVE.W   -136(A6),D0
```

```
00000748  302eff74                    MOVE.W      -140(A6),D0

00000772  302eff72                    MOVE.W      -142(A6),D0
```

Son interesantes porque van detrás de una llamada a API llamada StrCopy, que copia un string a otro, y porque luego el programa nos da un número a cambio, como vimos (el 114, el 15, etc...).

Y en segundo lugar, esas llamadas a memoria coinciden con lo que pone en la línea 00000658 y siguientes, que es el algoritmo de keygen, cada una con su variable.
Echadle un vistazo con calma a las líneas 00000652 (donde carga nuestro User) hasta 0000068a (donde carga la palabra UBF) y veréis como el cálculo lo hace como os señalé. Una vez más, el ASM de Motorola muestra todo su potencial y claridad de lectura, y personalmente me cae más simpático que el de x86.

3- El keygen:

Bueno, llegado a este punto y con el algoritmo hallado, el keygen lo programáis en vuestro lenguaje de programación favorito y con los adornos y dibujitos que queráis, u os bajáis de alguna web horrorosa el que os hice yo, que hasta tiene un huevo de pascua para que os entretengáis buscándolo.

...Y POR ESO YO CREO QUE AL BATES LO QUE LE FALTA ES UN BUEN POLVO

Esto es el final ¿a que ahora ya entendéis mejor la película?. El vodkila no estuvo mal, la McMierda cara e insana, como siempre, y Lacrimosa, sublimes haciendo música.
Lo lastimoso es la protección del John Love éste... cielos, eso de sumar 333 es el típico caso de zombi-programador. En fin, así vamos...

No quiero acabar mi tercer tuto sobre Palm sin agradecer especialmente la ayuda del dr_funk, único de #palmwarez con ganas de ayudar y que trabajamos juntos en un momento en que yo andaba atascado. Por supuesto, y lo hago siempre, un saludo muy grande al irc-hispano en #crackers al que le debo mucho, casi todo lo que aprendí y grandes amigos como los que menciono siempre, khanete, daimon, seven, Zeven_eT, el gran Pope (casi nuestro padre en esto junto a atomraM), Buba_luba, VIKTORY etc, etc... y aprovecho a ver si por fin me ponen un op fijo, que nunca sé cuándo lo tengo y cuándo no XD.

Ups, olvidaba a un amigo programador que me corrigió el keygen, Pataphalo, y al que me enseña Delphi a cambio de que yo le enseñe a crackear, DruidaElAdivino.

A todos ellos, mis amigos, los sabios, los que ayudan, los que saben, los que no se dejan manipular, los que piensan.

Usando mi lema cracker: "Pensad, pensad y siempre pensad"

En algún lugar de Europa, 19/06/02

By X-Grimator