

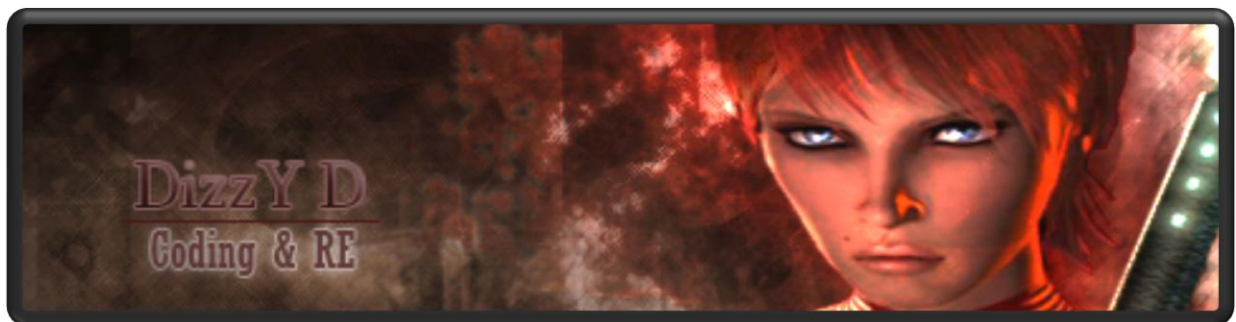
Made for:



&

SceneCoderz

Tutorial by



Hallo F-H und SC Community!

Ein Paar User waren an einem Tutorial zum Thema Filemodding interessiert.

Hier ist es :)

Im Forum sieht man immer wieder Threads mit Titeln wie "Avira weghexxen aber wie?" u.s.w.

Was mit "weghexxen" gemeint ist sollte jeder der dieses Tutorial sieht wissen ;)

Wer sich mit dem Aufbau und der Funktionsweise von PE Dateien ein wenig auskennt sollte schnell merken, dass diese Methode mehr als "unsauber" ist.

Warum das so ist und wie man es besser lösen kann möchte ich euch in diesem Tut zeigen.

Unser Ziel ist es eine Crypterstub "FUD" zu machen.

Als Target habe ich mir den Area51 Crypter 2.0 ausgesucht.

Ich habe diese Wahl getroffen weil er erstens schön Detected ist und zweitens weil er, wie fast alle Crypter heute, in VB6 gecodet ist.

Dieses Thema heisst nicht umsonst "Advanced EXE Modding", deshalb solltet ihr schon einige Kenntnisse mitbringen:

- OllyDbg bedienen können
- Grundlagen des PE Formats
- Grundbefehle ASM (ich werde den ASM Code kurz erklären)
- Zeit und Interesse ;)

Folgende Tools verwende ich in diesem Tutorial:

- OllyDbg 1.10
- Hex Workshop 6
- LordPE / CFF Explorer
- ResHacker
- AV Devil
- A-Squared Kommandozeilenscanner (DL Link: <http://download1.emsisoft.com/a2cmd.zip>)
- PE Detective
- Brain.exe (Die, die jetzt danach googlen sollten spätestens jetzt das Tut schließen :D)

Okay dann kanns losgehen...

Wie man die Stub aus dem Crypter ausbaut, darauf werde ich hier nicht eingehen.

Es ist kein Sonderfall das heisst, dass ihr es einfach mit ResHack ausbauen könnt.

So, wenn wir die Stub nun in diesem Zustand mal Scannen lassen sieht das Ganze so aus:

Results from the virus scan of uploaded sample

[Return to the Virus.Org Scanning Service](#)

The following represents the test results from the virus scanners used by the Virus.Org scanning service when it performed the scan on the file 'crypt.exe'.

File: crypt.exe
SHA-1 Digest: 7250df7d4cf210617a183a69a08434d5896eb403
Size: 65536 bytes
Detected Packer: Microsoft Visual Basic v5.0/v6.0
Status: Infected or Malware (Confidence 65.22%)
Date Scanned: Tue Apr 14 20:26:07 +0100 2009

Scanner	Scanner Version	Scanner Engine	Scanner Signatures	Result	Scan Time
A-Squared	4.0.0.32	N/A	1239735604	Riskware.Win32.Vbinder	9.13 secs
Arcavir	1.0.5	N/A	12:20 20-03-2009	Clean	4.58 secs
avast!	1.0.8	N/A	090414-0	Win32:VB-KLY	13.99 secs
AVG Anti Virus	7.5.52	442	270.11.57/2059	Crypt.APK	15.88 secs
Avira AntiVir	2.1.12-151	7.9.0.143	7.1.3.50	TR/VB.GSY	15.35 secs
BitDefender	7.81008	7.24800	2846471	Trojan.VB.NWZ	4.33 secs
CA eTrust	N/A	31.06.00	31.06.6435	Clean	4.08 secs
CAT QuickHeal	10.00	N/A	14 April, 2009	Win32.Trojan.VB.gsy.3	14.80 secs
ClamAV	0.94.2	N/A	9236	Trojan.VB-5652	0.09 secs
Comodo	3.8	3.8	1113	Clean	8.82 secs
CPSecure	1.15	1.1.0.715	14/04/2009 11:40AM	Clean	6.58 secs
Dr. Web	4.44.0.10060	4.44.0.9170	566216	Trojan.Siggen.1548	35.82 secs
F-PROT 6	6.2.1.4252	4.4.4.56	200904141034724	W32/Trojan2.FGPA	5.97 secs
F-Secure	1.10	6392	2009-04-14_08	Trojan.Win32.VB.gsy [AVP]	29.19 secs
Ikarus T3SCAN	1.32.4.0	1.01.49	2009-04-14 17:01:13	VirTool.Win32.Vbinder	14.69 secs
Kaspersky	5.7.13	1848517	14-04-2009	Trojan.Win32.VB.gsy	34.62 secs
McAfee Virusscan	5.30.0	5.3.00	v5579	Clean	14.54 secs
Norman Virus Control	7.00.00	6.00.06	6.00.00	Clean	56.03 secs
Panda	9.04.03.0001	1848106	06/04/2009	Clean	6.68 secs
Sophos Sweep	4.40.0	2.85.0	4.40	Troj/VBDrop-J	15.38 secs
Trend Micro	N/A	8.700-1004	961	BKDR_KOLOBOT.A	3.29 secs
VBA32	3.12.10.2	N/A	2009.04.13	Trojan.Win32.VB.ljk	13.73 secs
VirusBuster 2005	1.4.5	4.6.5	10.102.32	Clean	10.54 secs

Zimlich detected, oder? Na umso besser ;)

Als Erstes sollten unbenutzte Strings aus dem Programm entfernt werden, wie ihr das macht solltet ihr aus anderen Tutorials wissen, deshalb gehe ich auch hier nicht weiter drauf ein. Danach müssen FileVersion etc. geändert werden und das Icon gechanged werden. (Ja, manche AVs haben Signaturen auf Icons gesetzt) Beides ist mit Reshack machbar.

Nachdem ihr das gemacht habt sollte es ca. so aussehen:

Results from the virus scan of uploaded sample

[Return to the Virus.Org Scanning Service](#)

The following represents the test results from the virus scanners used by the Virus.Org scanning service when it performed the scan on the file 'Data_1.exe'.

File: Data_1.exe
SHA-1 Digest: 15935d7465a203cc78e955e1800700460dab6c69
Size: 61440 bytes
Detected Packer: Microsoft Visual Basic v5.0/v6.0
Status: Infected or Malware (Confidence 36.36%)
Date Scanned: Tue Apr 14 22:40:20 +0100 2009

Scanner	Scanner Version	Scanner Engine	Scanner Signatures	Result	Scan Time
A-Squared	4.0.0.32	N/A	1239742805	Riskware.Win32.Vbinder	6.76 secs
Arcavir	1.0.5	N/A	12:20 20-03-2009	Clean	9.32 secs
avast!	1.0.8	N/A	090414-0	Clean	34.79 secs
AVG Anti Virus	7.5.52	442	270.11.57/2059	Crypt.APK	32.78 secs
Avira AntiVir	2.1.12-151	7.9.0.143	7.1.3.50	TR/VB.GSY	32.40 secs
BitDefender	7.81008	7.24802	2846478	Clean	42.23 secs
CA eTrust	N/A	31.06.00	31.06.6435	Clean	36.15 secs
CAT QuickHeal	10.00	N/A	14 April, 2009	Clean	32.91 secs
ClamAV	0.94.2	N/A	9236	Clean	0.08 secs
Comodo	3.8	3.8	1113	Clean	2.16 secs
CPSecure	1.15	1.1.0.715	14/04/2009 11:40AM	Clean	4.18 secs
Dr. Web	4.44.0.10060	4.44.0.9170	566259	Trojan.Siggen.1548	55.69 secs
F-PROT 6	6.2.1.4252	4.4.4.56	2009041418523	Clean	34.35 secs
F-Secure	1.10	6392	2009-04-14_08	Trojan.Win32.VB.gsy [AVP]	0.34 secs
Ikarus T3SCAN	1.32.4.0	1.01.49	2009-04-14 17:01:13	VirTool.Win32.Vbinder	89.05 secs
Kaspersky	5.7.13	1848694	14-04-2009	Trojan.Win32.VB.gsy	76.86 secs
McAfee Virusscan	5.30.0	5.3.00	v5579	Clean	30.53 secs
Norman Virus Control	7.00.00	6.00.06	6.00.00	Clean	87.53 secs
Panda	9.04.03.0001	1848106	06/04/2009	Clean	15.60 secs
Sophos Sweep	4.40.0	2.85.0	4.40	Troj/VBDrop-J	37.53 secs
Trend Micro	N/A	8.700-1004	966	Clean	5.10 secs
VirusBuster 2005	1.4.5	4.6.5	10.102.32	Clean	21.94 secs

Schonmal fast die Hälfte weg :)

Okay jetzt fügen wir eine gefälschte Signatur in die EXE ein um die AVs zu verwirren.

Ich werde hierfür eine Armadillo Signatur nehmen.

Signaturen werden meist ab dem EP eingelesen. Daher müssen wir zuerst eine Stelle in der exe mit genügend Platz für unsere Signatur finden.

Also suchen wir einfach nach einer Stelle mit vielen Nullbytes. Man sollte am besten eine Stelle finden, die mitten in der exe liegt und nicht am Ende. Denn manche AVs erkennen das.

00401546	· 0000	ADD	BYTE PTR [EAX], AL
00401548	· 0000	ADD	BYTE PTR [EAX], AL
0040154A	· 0000	ADD	BYTE PTR [EAX], AL
0040154C	· 0000	ADD	BYTE PTR [EAX], AL
0040154E	· 0000	ADD	BYTE PTR [EAX], AL
00401550	· 0000	ADD	BYTE PTR [EAX], AL
00401552	· 0000	ADD	BYTE PTR [EAX], AL
00401554	· 0000	ADD	BYTE PTR [EAX], AL
00401556	· 0000	ADD	BYTE PTR [EAX], AL
00401558	· 0000	ADD	BYTE PTR [EAX], AL
0040155A	· 0000	ADD	BYTE PTR [EAX], AL
0040155C	· 0000	ADD	BYTE PTR [EAX], AL
0040155E	· 0000	ADD	BYTE PTR [EAX], AL
00401560	· 0000	ADD	BYTE PTR [EAX], AL
00401562	· 0000	ADD	BYTE PTR [EAX], AL
00401564	· 0000	ADD	BYTE PTR [EAX], AL
00401566	· 0000	ADD	BYTE PTR [EAX], AL
00401568	· 0000	ADD	BYTE PTR [EAX], AL
0040156A	· 0000	ADD	BYTE PTR [EAX], AL
0040156C	· 0000	ADD	BYTE PTR [EAX], AL
0040156E	· 0000	ADD	BYTE PTR [EAX], AL
00401570	· 0000	ADD	BYTE PTR [EAX], AL
00401572	· 0000	ADD	BYTE PTR [EAX], AL
00401574	· 0000	ADD	BYTE PTR [EAX], AL
00401576	· 0000	ADD	BYTE PTR [EAX], AL
00401578	· 0000	ADD	BYTE PTR [EAX], AL
0040157A	· 0000	ADD	BYTE PTR [EAX], AL
0040157C	· 0000	ADD	BYTE PTR [EAX], AL
0040157E	· 0000	ADD	BYTE PTR [EAX], AL
00401580	· 0000	ADD	BYTE PTR [EAX], AL
00401582	· 0000	ADD	BYTE PTR [EAX], AL
00401584	· 0000	ADD	BYTE PTR [EAX], AL
00401586	· 0000	ADD	BYTE PTR [EAX], AL
00401588	· 0000	ADD	BYTE PTR [EAX], AL
0040158A	· 0000	ADD	BYTE PTR [EAX], AL
0040158C	· 0000	ADD	BYTE PTR [EAX], AL
0040158E	· 0000	ADD	BYTE PTR [EAX], AL
00401590	· 0000	ADD	BYTE PTR [EAX], AL
00401592	· 0000	ADD	BYTE PTR [EAX], AL
00401594	· 0000	ADD	BYTE PTR [EAX], AL
00401596	· 0000	ADD	BYTE PTR [EAX], AL
00401598	· 0000	ADD	BYTE PTR [EAX], AL
0040159A	· 0000	ADD	BYTE PTR [EAX], AL
0040159C	· 0000	ADD	BYTE PTR [EAX], AL
0040159E	· 0000	ADD	BYTE PTR [EAX], AL
004015A0	· 0000	ADD	BYTE PTR [EAX], AL
004015A2	· 0000	ADD	BYTE PTR [EAX], AL
004015A4	· 0000	ADD	BYTE PTR [EAX], AL
004015A6	· 0000	ADD	BYTE PTR [EAX], AL
004015A8	· 0000	ADD	BYTE PTR [EAX], AL
004015AA	· 0000	ADD	BYTE PTR [EAX], AL
004015AC	· 0000	ADD	BYTE PTR [EAX], AL
004015AE	· 0000	ADD	BYTE PTR [EAX], AL
004015B0	· 0000	ADD	BYTE PTR [EAX], AL
004015B2	· 0000	ADD	BYTE PTR [EAX], AL
004015B4	· 0000	ADD	BYTE PTR [EAX], AL
004015B6	· 0000	ADD	BYTE PTR [EAX], AL
004015B8	· 0000	ADD	BYTE PTR [EAX], AL
004015BA	· 0000	ADD	BYTE PTR [EAX], AL
004015BC	· 0000	ADD	BYTE PTR [EAX], AL
004015BE	· 0000	ADD	BYTE PTR [EAX], AL
004015C0	· 0000	ADD	BYTE PTR [EAX], AL
004015C2	· 0000	ADD	BYTE PTR [EAX], AL
004015C4	· 0000	ADD	BYTE PTR [EAX], AL
004015C6	· 0000	ADD	BYTE PTR [EAX], AL
004015C8	· 0000	ADD	BYTE PTR [EAX], AL
004015CA	· 0000	ADD	BYTE PTR [EAX], AL
004015CC	· 0000	ADD	BYTE PTR [EAX], AL

Die Stelle hier sieht doch gut aus. Nicht am Ende und genügend Platz. :)

Hier können wir nun unsere Signatur einfügen. Aber, was ist überhaupt eine Signatur? Die meisten kennen den Begriff wahrscheinlich von den AVs.

Diese Scannen unter Anderem auch nach Signaturen. Eine Signatur ist nichts anderes als eine Abfolge von festgelegten Bytes.

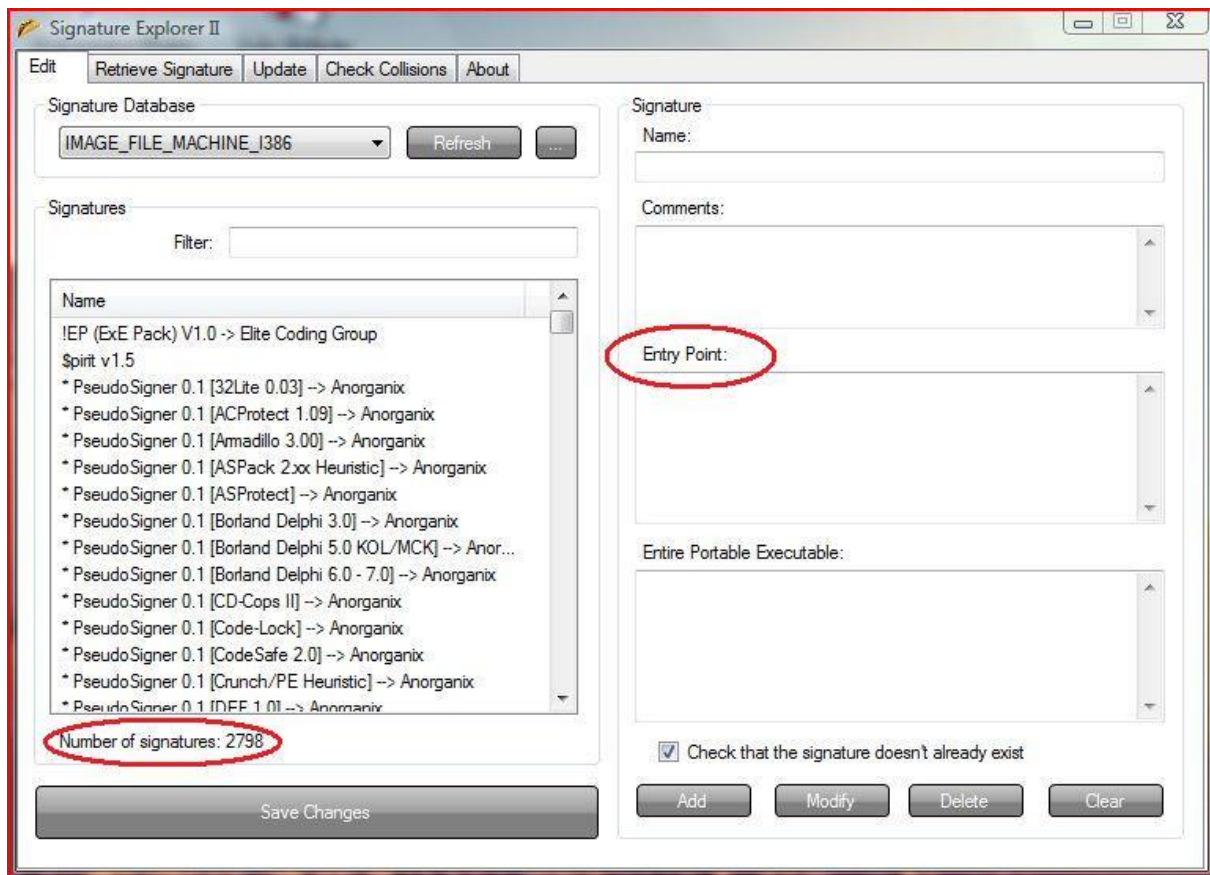
AVs gehen meistens so vor, dass sie in einem bestimmten Abstand zum EP nach einer Signatur Scannen. Jetzt wird auch klar, warum das EP verschieben überhaupt UD macht.

Wenn der EP verschoben wurde passen die Abstände nicht mehr und das AV erkennt nichts. Wenn man allerdings einfach einen JMP zum OEP (OriginalEntryPoint) verwendet, können die AVs dies zurückverfolgen. Um das zu verhindern reicht es momentan noch aus mehrere JMPs zu verwenden.

Das ist auch einer der Gründe warum ich eine Armadillo Signatur verwenden werde. Sie besteht nämlich, wie ihr sehen werdet, schon aus vielen JMPs.

Die Signatur habe ich aus dem Programm "PE Detective". Diesem Programm liegt ein Signature Explorer bei, mit dem man sehr schnell Signaturen suchen kann.

Ich zeig euch das Programm mal eben.



Ihr habt hier 2700 Signaturen zur Auswahl. Das reicht doch erstmal, oder?

Im Fenster „EntryPoint“ seht ihr die Bytes der Signatur, die am Entrypoint liegen.

Hier die Armadillo Sig die ich verwenden werde :

```
60 E8 00 00 00 00 5D 50 51 EB 0F 90 EB 0F 90 EB 07 90 EB 0F 90 EB 08 FD EB 0B F2 EB F5 EB F6 F2 EB
08 FD EB E9 F3 EB E4 FC 90 59 58 50 51 EB 0F 90 EB 0F 90 EB 07 90 EB 0F 90 EB 08 FD EB 0B F2 EB F5
EB F6 F2 EB 08 FD EB E9 F3 EB E4 FC 90 59 58 50 51 EB 0F 00 00 00 00 00 00 00 00 00 00 00 00 00
00 61
```

Und so kann man sie verwenden:

Bytes markieren -> kopieren -> in Olly die leeren Bytes markieren -> rechtecklick -> Binary paste

OK wir merken uns die Adresse, andie wir Sig eingefügt haben.

```
00401546 60 PUSHAD
```

Jetzt abspeichern.

Das macht ihr wie folgt:

Rechtsklick -> copy to executable -> All Modifications -> Copy All ->Rechtsklick -> Save to File

So damit wir auch beim Start der exe an unserer Sig starten müssen wir den EP anpassen.

Das macht ihr mit LordPE oder CFF Explorer.

00401546 <--- Aus dieser Adresse müssen wir nun die RVA (Relative Virtual Address) berechnen.

Das Hört sich komplizierter an als es ist.

In jedem PE Explorer ist ein Rechner drin, der das kann.

Im Grunde macht dieser nichts anderes als $00401546 - \text{Imagebase}$.

Die Imagebase können wir auch in unserem PE Explorer ablesen.

In unserem Fall ist 00400000 also die Imagebase.

$00401546 - 00400000$ ist 1546 (welch Überraschung)

Also als EP 1546 eingeben, speichern.

Gut, mal sehen ob wir in Olly nun auch am neuen EP landen.

```

00401546 $ 60 PUSHAD
00401547 . E8 00000000 CALL Fake_Sig.0040154C
0040154C . 5D POP EBP
0040154D . 59 POP ECX
0040154E . 51 PUSH ECX
0040154F .> EB 0F JMP SHORT Fake_Sig.00401560
00401551 . 90 NOP
00401552 .> EB 0F JMP SHORT Fake_Sig.00401563
00401554 .> 90 NOP
00401555 .> EB 07 JMP SHORT Fake_Sig.0040155E
00401557 . 90 NOP
00401558 .> EB 0F JMP SHORT Fake_Sig.00401563
0040155A . 90 NOP
0040155B .> EB 08 JMP SHORT Fake_Sig.00401565
0040155D . FD STD
0040155E .> EB 0B JMP SHORT Fake_Sig.0040156B
00401560 .> F2: PREFIX REPNE: Superfluous prefix
00401561 .> EB F5 JMP SHORT Fake_Sig.00401568
00401563 .> EB F6 JMP SHORT Fake_Sig.0040156B
00401565 .> F2: PREFIX REPNE: Superfluous prefix
00401566 .> EB 08 JMP SHORT Fake_Sig.00401570
00401568 . FD STD
00401569 .> EB E9 JMP SHORT Fake_Sig.00401554
0040156B .> EB E4 PREFIX REP: Superfluous prefix
0040156C . FC CLD
0040156E . 90 NOP
00401570 . 59 POP ECX
00401571 . 58 POP EAX
00401572 . 50 PUSH ECX
00401573 . 51 PUSH ECX
00401574 .> EB 0F JMP SHORT Fake_Sig.00401585
00401576 . 90 NOP
00401577 .> EB 0F JMP SHORT Fake_Sig.00401588
00401579 .> 90 NOP
0040157A .> EB 07 JMP SHORT Fake_Sig.00401583
0040157C . 90 NOP
0040157D .> EB 0F JMP SHORT Fake_Sig.0040158E
0040157F . 90 NOP
00401580 .> EB 08 JMP SHORT Fake_Sig.0040158A
00401582 . FD STD
00401583 .> EB 0B JMP SHORT Fake_Sig.00401590
00401585 .> F2: PREFIX REPNE: Superfluous prefix
00401586 .> EB F5 JMP SHORT Fake_Sig.00401570
00401588 .> EB F6 JMP SHORT Fake_Sig.00401580
0040158A .> F2: PREFIX REPNE: Superfluous prefix
0040158B .> EB 08 JMP SHORT Fake_Sig.00401595
0040158D . FD STD
0040158E .> EB E9 JMP SHORT Fake_Sig.00401579
00401590 .> F3: PREFIX REP: Superfluous prefix
00401591 .> EB E4 JMP SHORT Fake_Sig.00401577
00401593 . FC CLD
00401594 . 90 NOP
00401595 . 59 POP ECX
00401596 . 58 POP EAX
00401597 . 50 PUSH ECX
00401598 . 51 PUSH ECX
00401599 .> EB 0F JMP SHORT Fake_Sig.004015A9
0040159B . 00 DB 00
0040159C . 00 DB 00
0040159D . 00 DB 00
0040159E . 00 DB 00
0040159F . 00 DB 00
004015A0 . 00 DB 00
004015A1 . 00 DB 00
004015A2 . 00 DB 00
004015A3 . 00 DB 00
004015A4 . 00 DB 00
004015A5 . 00 DB 00
004015A6 . 00 DB 00
004015A7 . 00 DB 00
004015A8 . 00 DB 00
004015A9 . 00 DB 00
004015AA .> 51 POPAD

```

Siehe da, wir sind an unserer Fake Sig!

Wie wir sehen können: Viieeele JMPs ☺

Mal sehen was user PE Detective nun zu dieser Datei sagt!

Best Match		
Armadillo 3.00a -> Silicon Realms Toolworks		
All Matches		
Signature	Matches	Comments
Armadillo 3.00a -> Silicon Realms Toolworks	75	
Microsoft Visual Basic v5.0 - v6.0	9	

75 Matches für Armadillo :D

Er fällt also auch auf die Fälschung rein.

Um es jetzt für die AVs noch komplizierter zu machen fügen wir noch ein Paar JMPs hinzu.

Um zu sehen, wo wir am Ende der Sig unseren JMP zu platzieren, tracen wir mal ein bisschen durch die Sig mit F8

Der Letzte Befehl ist ein POPAD. Nach diesem Befehl können wir noch ein Paar JMPs einfügen und zu guter Letzt dann den JMP zum OEP.

```

> EB 0F      JMP     SHORT Fake_Sig.004015AA
> 90        NOP
> 90        NOP
> 90        NOP
> 90        NOP
> 90        NOP
> 90        NOP
> 90        NOP
> 90        NOP
> 90        NOP
> 90        NOP
> 90        NOP
> EB 06      JMP     SHORT Fake_Sig.004015AD
00         DB     00
00         DB     00
00         DB     00
61         POPAD
> EB EE      JMP     SHORT Fake_Sig.0040159B
> E9 06FBFFF JMP     Fake_Sig.004010B8

```

So jetzt tracen wir mal weiter bis wir am OEP sind um zu gucken ob alles klappt.

Okay, das ist er OEP. Funtzt also :D. Direkt erstmal speichern. Mal sehen wies jetzt mit den Detections aussieht.

Results from the virus scan of uploaded sample

[Return to the Virus.Org Scanning Service](#)

The following represents the test results from the virus scanners used by the Virus.Org scanning service when it performed the scan on the file 'Data_3.exe'.

File: Data_3.exe
SHA-1 Digest: 2e754eec125cae95a28c223cbc4ca799a8558b7f
Size: 24576 bytes
Detected Packer: Themida
Status: Infected or Malware (Confidence 22.73%)
Date Scanned: Tue Apr 14 23:07:10 +0100 2009

Scanner	Scanner Version	Scanner Engine	Scanner Signatures	Result	Scan Time
A-Squared	4.0.0.32	N/A	1239746405	Riskware.Win32.Vbinder	8.44 secs
Arcavir	1.0.5	N/A	12:20 20-03-2009	Clean	4.44 secs
avast!	1.0.8	N/A	090414-0	Clean	16.44 secs
AVG Anti Virus	7.5.52	442	270.11.57/2059	Clean	15.70 secs
Avira AntiVir	2.1.12-151	7.9.0.143	7.1.3.50	TR/VB.GSY	15.75 secs
BitDefender	7.81008	7.24803	2846480	Gen:Trojan.Heur.VB.1024DB989B	5.20 secs
CA eTrust	N/A	31.06.00	31.06.6435	Clean	4.79 secs
CAT QuickHeal	10.00	N/A	14 April, 2009	Clean	16.75 secs
ClamAV	0.94.2	N/A	9236	Clean	0.08 secs
Comodo	3.8	3.8	1113	Clean	7.25 secs
CPSecure	1.15	1.1.0.715	14/04/2009 11:40AM	Clean	7.18 secs
Dr. Web	4.44.0.10060	4.44.0.9170	566271	Trojan.Siggen.1548	34.84 secs
F-PROT 6	6.2.1.4252	4.4.4.56	2009041418523	Clean	9.42 secs
F-Secure	1.10	6392	2009-04-14_08	Clean	26.65 secs
Ikarus T3SCAN	1.32.4.0	1.01.49	2009-04-14 17:01:13	VirTool.Win32.Vbinder	17.24 secs
Kaspersky	5.7.13	1848805	14-04-2009	Clean	31.63 secs
McAfee Virusscan	5.30.0	5.3.00	v5579	Clean	15.92 secs
Norman Virus Control	7.00.00	6.00.06	6.00.00	Clean	56.11 secs
Panda	9.04.03.0001	1848106	06/04/2009	Clean	6.69 secs
Sophos Sweep	4.40.0	2.85.0	4.40	Clean	18.13 secs
Trend Micro	N/A	8.700-1004	966	Clean	2.60 secs
VirusBuster 2005	1.4.5	4.6.5	10.102.32	Clean	11.36 secs

Hat sich doch gelohnt, oder?

Stört euch bitte nicht daran, dass da „Themida“ steht. Diese Sig habe ich vorher benutzt und ich wollte es deswegen nicht nochmal uploaden. Das Ergebnis ist das Selbe.

Natürlich reicht uns das noch nicht.

Da man Avira ja mit AVDevil leicht wegbekommt machen wir das als Nächstes.

Folgende Offsets sind Detected:

1067 - 10E4

Die Meisten würde jetzt mit dem Hexeditor versuchen 00 zu FF zu ändern.

An der richtigen Stelle hätte man auch in diesem Fall Erfolg. Ich will euch aber von dieser Methode abraten und zeige euch jetzt wie es besser geht.

Um eine Entscheidung zu treffen, was wir im Code verändern, müssen wir ihn erstmal verstehen.

Um diesen Bereich hier geht es:

00401068	.-FF25 14104000	JMP	DWORD PTR [<&MSUBUM60.#526>]	MSUBUM60.rtcSpaceVar
0040106E	.-FF25 24104000	JMP	DWORD PTR [<&MSUBUM60.#711>]	MSUBUM60.rtcSplit
00401074	.-FF25 08104000	JMP	DWORD PTR [<&MSUBUM60.#666>]	MSUBUM60.rtcEnvironVar
0040107A	.-FF25 1C104000	JMP	DWORD PTR [<&MSUBUM60.#600>]	MSUBUM60.rtcShell
00401080	.-FF25 2C104000	JMP	DWORD PTR [<&MSUBUM60.#716>]	MSUBUM60.rtcCreateObject2
00401086	.-FF25 28104000	JMP	DWORD PTR [<&MSUBUM60.#608>]	MSUBUM60.rtcVarBstrFromAnsi
0040108C	.-FF25 0C104000	JMP	DWORD PTR [<&MSUBUM60.#595>]	MSUBUM60.rtcMsgBox
00401092	.-FF25 30104000	JMP	DWORD PTR [<&MSUBUM60.#717>]	MSUBUM60.rtcStrConvVar2
00401098	.-FF25 04104000	JMP	DWORD PTR [<&MSUBUM60.#626>]	MSUBUM60.rtcGetObject
0040109E	.-FF25 38104000	JMP	DWORD PTR [<&MSUBUM60.#644>]	MSUBUM60.VarPtr
004010A4	.-FF25 20104000	JMP	DWORD PTR [<&MSUBUM60.__vbaExcept	MSUBUM60.__vbaExceptionHandler
004010AA	.-FF25 34104000	JMP	DWORD PTR [<&MSUBUM60.ProcCallE	MSUBUM60.ProcCallEngine
004010B0	-\$-FF25 40104000	JMP	DWORD PTR [<&MSUBUM60.#100>]	MSUBUM60.ThunRTMain
004010B6	00	DB	00	
004010B7	00	DB	00	
004010B8	\$. 68 00114000	PUSH	Data_1.00401100	
004010BD	. E8 EFFFFFFF	CALL	<JMP.&MSUBUM60.#100>	
004010C2	. 0000	ADD	BYTE PTR [EAX], AL	
004010C4	. 0000	ADD	BYTE PTR [EAX], AL	
004010C6	. 0000	ADD	BYTE PTR [EAX], AL	
004010C8	. 3000	XOR	BYTE PTR [EAX], AL	
004010CA	. 0000	ADD	BYTE PTR [EAX], AL	
004010CC	. 3800	CMP	BYTE PTR [EAX], AL	
004010CE	. 0000	ADD	BYTE PTR [EAX], AL	
004010D0	. 0000	ADD	BYTE PTR [EAX], AL	
004010D2	. 0000	ADD	BYTE PTR [EAX], AL	
004010D4	. 61	POPAD		
004010D5	. 6F	OUTS	DX, DWORD PTR ES:[EDI]	I/O command
004010D6	. 107A 85	ADC	BYTE PTR [EDX-7B], BH	
004010D9	. 2F	DAS		
004010DA	. A2	DB	A2	
004010DB	. 41	DB	41	CHAR 'A'
004010DC	. AD	DB	AD	
004010DD	. 33	DB	33	CHAR '3'
004010DE	. D1	DB	D1	
004010DF	. 3D	DB	3D	CHAR '='
004010E0	. A2	DB	A2	
004010E1	. 2F	DB	2F	CHAR '/'
004010E2	. CF	DB	CF	
004010E3	. 7A	DB	7A	CHAR 'z'
004010E4	. 00	DB	00	

Um die Signatur zu bypassen setzen wir bei den JMPs an.

0040106E JMP DWORD PTR [401024]

Dieser JMP springt an die Adresse, die wiederum an der Adresse 401024 steht.

Mal sehen welche Adresse wir dort vorfinden. Olly hat es für uns schon zurückverfolgt: Dieser Sprung springt zur VB Split Funktion.

```
0040108C JMP   DWORD PTR [40100C]
```

Dieser hier Springt zu VB MsgBox Funktion, denn die Adresse dieser Funktion steht an 40100C.

Um die Signatur von Avira zu umgehen müssen wir nur ein Paar Bytes ändern. Es gibt viele Methoden wie man hier vorgehen kann. Meine Variante zeige ich euch jetzt.

Ich suche zuerst eine kleine, freie Stelle um den Sprung dort einzufügen.

0040110E	00	DB	00	
0040110F	00	DB	00	
004011E0	-FF25 24104000	JMP	DWORD PTR [<&MSUBUM60.#711>]	MSUBUM60.rtcSplit
004011E6	00	DB	00	
004011E7	00	DB	00	
004011E8	00	DB	00	

Danach überschreibe ich den eigentlichen Sprung zur Split Funktion mit einem Sprung zu unserem neuen JMP.

0040106E	.\vE9 6E010000	JMP	<JMP.&MSUBUM60.#711>
----------	----------------	-----	----------------------

Wenn irgendwo im Stub Code nun die Split Funktion aufgerufen wird, springen wir nicht mehr direkt zur Funktion, sondern werden erstmal über den, von uns eingefügten JMP, umgeleitet.

Durch diesen weiteren Sprung haben sich die Bytes an dieser Adresse komplett geändert.

Trotzdem wird diese neue Methode funktionieren, da wir ja genau wissen, was wir tun, im Gegensatz zum "Weghexxen".

Das werde ich jetzt auch noch mit ein Paar anderen Funktionen machen. Wer will kann auch Alle JMPs umleiten.

00401050	.\vFF25 18104000	JMP	DWORD PTR [<&MSUBUM60.DllFunctionCall	MSUBUM60.DllFunctionCall
00401056	.\vFF25 10104000	JMP	DWORD PTR [<&MSUBUM60.#632>]	MSUBUM60.rtcMidCharVar
0040105C	.\vFF25 00104000	JMP	DWORD PTR [<&MSUBUM60.#516>]	MSUBUM60.rtcAnsiValueBstr
00401062	.\vFF25 3C104000	JMP	DWORD PTR [<&MSUBUM60.#570>]	MSUBUM60.rtcFileLength
00401068	.\vFF25 14104000	JMP	DWORD PTR [<&MSUBUM60.#526>]	MSUBUM60.rtcSpaceVar
0040106E	.\vE9 6E010000	JMP	<JMP.&MSUBUM60.#711>	
00401073	90	NOP		
00401074	.\vFF25 08104000	JMP	DWORD PTR [<&MSUBUM60.#666>]	MSUBUM60.rtcEnvironVar
0040107A	.\vFF25 1C104000	JMP	DWORD PTR [<&MSUBUM60.#600>]	MSUBUM60.rtcShell
00401080	.\vFF25 2C104000	JMP	DWORD PTR [<&MSUBUM60.#716>]	MSUBUM60.rtcCreateObject2
00401086	.\vFF25 28104000	JMP	DWORD PTR [<&MSUBUM60.#608>]	MSUBUM60.rtcVarBstrFromAnsi
0040108C	.\vE9 56010000	JMP	<JMP.&MSUBUM60.#595>	
00401091	90	NOP		
00401092	.\vFF25 30104000	JMP	DWORD PTR [<&MSUBUM60.#717>]	MSUBUM60.rtcStrConvVar2
00401098	.\vFF25 04104000	JMP	DWORD PTR [<&MSUBUM60.#626>]	MSUBUM60.rtcGetObject
0040109E	.\vFF25 38104000	JMP	DWORD PTR [<&MSUBUM60.#644>]	MSUBUM60.VarPtr
004010A4	.\vFF25 20104000	JMP	DWORD PTR [<&MSUBUM60.__vbaExcept	MSUBUM60.__vbaExceptionHandler
004010AA	.\vFF25 34104000	JMP	DWORD PTR [<&MSUBUM60.ProcCallEn	MSUBUM60.ProcCallEngine
004010B0	.\vE9 38010000	JMP	<JMP.&MSUBUM60.#100>	
004010B5	90	NOP		

Okay, wieder abspeichern.

Und so siehts aus:

Results from the virus scan of uploaded sample

[Return to the Virus.Org Scanning Service](#)

The following represents the test results from the virus scanners used by the Virus.Org scanning service when it performed the scan on the file 'Data_4.exe'.

File: Data_4.exe
SHA-1 Digest: c4952b6683e70681786ead0a116f27b453693967
Size: 24576 bytes
Detected Packer: Themida
Status: Infected or Malware (Confidence 13.64%)
Date Scanned: Tue Apr 14 23:13:29 +0100 2009

Scanner	Scanner Version	Scanner Engine	Scanner Signatures	Result	Scan Time
A-Squared	4.0.0.32	N/A	1239746405	Riskware.Win32.Vbinder	6.42 secs
Arcavir	1.0.5	N/A	12:20 20-03-2009	Clean	15.18 secs
avast!	1.0.8	N/A	090414-0	Clean	37.42 secs
AVG Anti Virus	7.5.52	442	270.11.57/2059	Clean	59.13 secs
Avira AntiVir	2.1.12-151	7.9.0.143	7.1.3.50	Clean	46.21 secs
BitDefender	7.81008	7.24803	2846480	Gen:Trojan.Heur.VB.1024DB9B9B	3.43 secs
CA eTrust	N/A	31.06.00	31.06.6435	Clean	5.90 secs
CAT QuickHeal	10.00	N/A	14 April, 2009	Clean	17.12 secs
ClamAV	0.94.2	N/A	9236	Clean	0.29 secs
Comodo	3.8	3.8	1113	Clean	2.12 secs
CPSecure	1.15	1.1.0.715	14/04/2009 11:40AM	Clean	8.19 secs
Dr. Web	4.44.0.10060	4.44.0.9170	566271	Clean	82.24 secs
F-PROT 6	6.2.1.4252	4.4.4.56	2009041418523	Clean	17.68 secs
F-Secure	1.10	6392	2009-04-14_08	Clean	30.44 secs
Ikarus T3SCAN	1.32.4.0	1.01.49	2009-04-14 17:01:13	VirTool.Win32.Vbinder	35.67 secs
Kaspersky	5.7.13	1848805	14-04-2009	Clean	60.49 secs
McAfee Virusscan	5.30.0	5.3.00	v5579	Clean	51.42 secs
Norman Virus Control	7.00.00	6.00.06	6.00.00	Clean	117.43 secs
Panda	9.04.03.0001	1848106	06/04/2009	Clean	22.39 secs
Sophos Sweep	4.40.0	2.85.0	4.40	Clean	50.16 secs
Trend Micro	N/A	8.700-1004	966	Clean	6.96 secs
VirusBuster 2005	1.4.5	4.6.5	10.102.32	Clean	46.59 secs

Wir kommen unserem Ziel immer näher ;)

Allerdings liegt unser größter Feind noch vor uns: A-Squared

Dieses AV wird uns mit 3 verschiedenen Detections "nerven". Um diese zu bypassen brauchen wir eine Möglichkeit oft zu scannen. Online Scanner sind dafürzu Zeitaufwändig und überflüssig. Deswegen habe ich mir eine Kommandozeilen-basierte Version on A-Squared besorgt. Diese verlangsamt den PC nicht, da sie nicht im Hintergrund läuft.

Ein Downloadlink liegt dem Tut bei.

Wie man schon auf den Screenie sieht erkennt A-Squared unsere Stub als "Riskware.Win32.Vbinder"

Um das Offset zu finden habe ich die Datei gesplitted. Wie das geht müsset ihr aus anderen Tuts kennen. Ansonsten postet die Frage einfach in den Thread.

Das Problem liegt an Offset 2248.

2248 + Imagebase = Adresse in Olly. Also 2248 + 400000 = 402248. Wenn wir uns diese Prossition in Olly angucken sehen wirdirekt das Problem:

CallWindowProcA

Was ist das? Alle die schon mal einen Crypter in VB gecodet haben, werden diese API kennen. Sie wird benutzt um APIs Dynamisch aufzurufen.

Das Problem ist, dass dieser String nicht verändert werden darf. Was nun? Ganz einfach: Hier sind einige Nullbytes zwischen dem Namen und dem Code der die API aufruft.

00402239	00	DB	00	
0040223A	00	DB	00	
0040223B	00	DB	00	
0040223C	. 75 73 65 72 3:	ASCII	"user32",0	
00402243	00	DB	00	
00402244	10	DB	10	
00402245	00	DB	00	
00402246	00	DB	00	
00402247	00	DB	00	
00402248	. 43 61 6C 6C 5:	ASCII	"CallWindowProcA",0	
00402258	3C224000	DD	Fake_Sig.0040223C	ASCII "user32"
0040225C	48224000	DD	Fake_Sig.00402248	ASCII "CallWindowProcA"
00402260	00	DB	00	
00402261	00	DB	00	
00402262	04	DB	04	
00402263	00	DB	00	
00402264	D4444000	DD	Fake_Sig.004044D4	
00402268	00	DB	00	
00402269	00	DB	00	
0040226A	00	DB	00	
0040226B	00	DB	00	
0040226C	00	DB	00	
0040226D	00	DB	00	
0040226E	00	DB	00	
0040226F	00	DB	00	
00402270	. A1 DC444000	MOV	EAX, DWORD PTR [4044DC]	
00402275	. 0BC0	OR	EAX, EAX	
00402277	. v 74 02	JE	SHORT Fake_Sig.0040227B	
00402279	. FFE0	JMP	EAX	
0040227B	> 68 58224000	PUSH	Fake_Sig.00402258	
00402280	. B8 50104000	MOV	EAX, <JMP.&MSUBUM60.DllFunction	
00402285	. FFD0	CALL	EAX	
00402287	. FFE0	JMP	EAX	
00402289	00	DB	00	
0040228A	00	DB	00	
0040228B	00	DB	00	
0040228C	0F	DB	0F	
0040228D	00	DB	00	
0040228E	00	DB	00	

Der Code Teil darf ebenfalls nicht modifiziert werden. Wir können allerdings die Bytes dazwischen verändern. Ob ihr da nun FF oder 90 oder whatever einsetzt ist völlig egal. Was ihr beachten müsst ist, dass ihr nur Nullen ändert. Die Werte die bereits da stehen dürfen nicht verändert werden.

Okay nun scannen wir das Teil mal mit unserem CMDScanner.

```
C:\Users\DizzY_D\Desktop\AU Scanner\AU\S-A-Squared>a2cmd C:\Users\DizzY_D\Desktop\Fake_Sig4.exe /r /h

a-squared Command Line Scanner v. 4.0.0.32
(C) 2003-2008 Emsi Software GmbH - www.emsisoft.com

a-squared Command Line Scanner - Version 4.0
Last update: 14.04.2009 21:44:49

Scan settings:

Objects:          C:\USERS\DIZZY_D\DESKTOP\FAKE_SIG4.EXE
Scan archives:   Off
Heuristics:      On
ADS Scan:        Off

Scan start:      17.04.2009 21:37:31

C:\USERS\DIZZY_D\DESKTOP\FAKE_SIG4.EXE detected: Trojan-Dropper!IK
```

Wie gesagt, es ist immer noch detected, aber nun als Trojan-Dropper!IK.

Um dies zu bypassen kommen wir wieder mit Splitten weiter.

Nach einiger Zeit hat man folgendes Offset gefunden:

2926 also in Olly 402926.

Wie wir sehen ist dort kein Code vorhanden. Zumindest erkennt Olly ihn nicht als solchen.

00402926	00	DB	00	
00402927	18	DB	18	
00402928	00	DB	00	
00402929	3C	DB	3C	CHAR '<'
0040292A	6C	DB	6C	CHAR 'L'
0040292B	48	DB	48	CHAR 'H'
0040292C	FF	DB	FF	
0040292D	04	DB	04	
0040292E	74	DB	74	CHAR 't'
0040292F	FF	DB	FF	
00402930	. FC	CLD		
00402931	. 58	POP	EAX	
00402932	. 2F	DAS		
00402933	. 48	DEC	EAX	
00402934	. FF6C74 FF	JMP	FAR FWORD PTR [ESP+ESI*2-1]	
00402938	43	DB	43	CHAR 'C'
00402939	78	DB	78	CHAR 'x'
0040293A	FF	DB	FF	
0040293B	14	DB	14	
0040293C	84114000	DD	Fake_Sig.00401184	
00402940	10	DB	10	
00402941	00	DB	00	
00402942	58	DB	58	CHAR 'X'
00402943	00	DB	00	
00402944	94	DB	94	
00402945	01	DB	01	
00402946	3C	DB	3C	CHAR '<'
00402947	00	DB	00	
00402948	00	DB	00	
00402949	00	DB	00	
0040294A	00	DB	00	
0040294B	00	DB	00	
0040294C	00	DB	00	
0040294D	00	DB	00	
0040294E	13	DB	13	
0040294F	00	DB	00	
00402950	00	DB	00	
00402951	00	DB	00	
00402952	00	DB	00	
00402953	00	DB	00	
00402954	24	DB	24	CHAR '\$'

Um dies genauer zu prüfen machen wir folgendes:

Bereich markieren -> Analysis -> Remove Analysis from Selection

Nun zeigt Olly uns diese Daten als Code an. Wie man unschwer erkennen kann ist das kein ausführbarer Code. Doch wozu ist es dann da?

Diese Frage führt uns tiefer in die Funktionsweise von VB6 exen. Hier ist es so, dass der eigentliche Code in der VB Runtime ausgeführt wird.

Die Runtime ließt also die Daten die in der exe stehen aus, interpretiert sie und nimmt dann die nötigen Veränderungen etc. vor.

Was heißt das für uns?

Hexxen absolut unmöglich und Code umschreiben geht auch net, da ja überhaupt kein Code vorhanden ist!

Um die Detection zu umgehen ist es aber notwendig die Bytes an dieser Stelle zu verändern.

Das Stichwort heisst Laufzeimentschlüsselung!

Wir schreiben uns also einen kleinen Algorithmus, der diese Stelle verschlüsselt, und beim Start der Datei, noch bevor die VB Runtime überhaupt ausgeführt

wird, diese Offsets wieder entschlüsselt und die exe ausgeführt werden kann.

Als Verschlüsselung wähle ich die einfache Xor Verschlüsselung, die ja in ASM mit einem Befehl realisierbar ist ☺

So sieht der Algo aus:

```
004015A0 > B8 26294000 MOV EAX, Fake_Sig.00402926
004015B2 > 8030 0F XOR BYTE PTR [EAX], 0F
004015B5 . 40 INC EAX
004015B6 . 3D 54294000 CMP EAX, Fake_Sig.00402954
004015B8 . ^7E F5 JLE SHORT Fake_Sig.004015B2
```

Okay das wars schon. Ich gehe jeden Befehl einzeln durch damits verständlicher wird. Die, die schonmal "Manual Packing" verwendet haben werden den Algo ja

schon kennen. Aber ob sie ihn auch verstehen? ;)

```
MOV EAX, 402926 // Startoffset in EAX Speichern
ABCD: XOR BYTE PTR [EAX], 0F // Das Byte an der Adresse von EAX "XORen"
INC EAX // Eax um 1 erhöhen
CMP EAX, 402954 // Vergleichen, ob wir schon am Ende sind (In dem Fall
verschlüsseln wir bis 402954)
JLE ABCD // Springe wenn EAX kleiner oder gleich 402954 ist zur
Sprungmarke "ABCD" (XOR Befehl)
```

Diese Schleife läuft so lange durch bis wir am Ende (402954) sind.

Unter dieser Schleife kommt dann der JMP zum OEP, sodass die EXE danach wieder normal ausgeführt wird.

Okay speichern und reinladen.

Wir setzen jetzt einen Breakpoint (F2) unter unsere XOR Schleife und starten das Programm (F9)

```
Access violation when writing to [00402926] - use Shift+F7/F8/F9 to pass exception to program
```

Hmm was hat das da unten zu bedeuten?

Access Violation...

Anscheinend haben wir keine Schreiberlaubnis auf diese Adresse. Wie können wir das ändern? Ganz einfach: Section Characteristics anpassen.

Dazu laden wir unsere Stub in LordPE.

So hier unsere 3 Sections. Das Offset 2926 liegt in der Code Section. Das kann man ganz einfach daran ablesen, indem man Prüft ob das Offset zwischen VirtualAddress und VirtualAddress + VirtualSize liegt. Das trifft in unserem Fall, ohne groß nachzurechnen, zu.

Gut also passen wir es an indem wir auf Edit Section Handler -> Falgs gehen und bei "Writable" nen Haken rein machen.



Save...

In Olly laden, der BP sollte noch da sein, wenn nicht -> neu setzen

Starten (F9), jetzt sollten wir ohne Violation breaken.

Da unser Code ja noch nicht verschlüsselt ist, müssen wir ihn erst verschlüsselt abspeichern.

Dazu gehen wir zum Startoffset (402926) und makieren die Zeilen bis zum Verschlüsselungsende (402954).

Danach Copy to executable -> Selection und speichern...

In Olly laden, BP ans Ende des Algos setzen, starten, wir breaken.

Mal schauen wies nun an unserem Offset aussieht.

00402926	00	DB	00	
00402927	18	DB	18	
00402928	00	DB	00	
00402929	3C	DB	3C	CHAR '<'
0040292A	6C	DB	6C	CHAR 'l'
0040292B	48	DB	48	CHAR 'H'
0040292C	FF	DB	FF	
0040292D	04	DB	04	
0040292E	74	DB	74	CHAR 't'
0040292F	FF	DB	FF	
00402930	. FC	CLD		
00402931	. 58	POP	EAX	
00402932	. 2F	DAS		
00402933	. 48	DEC	EAX	
00402934	. FF6C74 FF	JMP	FAR FWORD PTR [ESP+ESI*2-1]	
00402938	43	DB	43	CHAR 'C'
00402939	78	DB	78	CHAR 'x'
0040293A	FF	DB	FF	
0040293B	14	DB	14	
0040293C	84114000	DD	Fake_Sig.00401184	
00402940	10	DB	10	
00402941	00	DB	00	
00402942	58	DB	58	CHAR 'X'
00402943	00	DB	00	
00402944	94	DB	94	
00402945	01	DB	01	
00402946	3C	DB	3C	CHAR '<'
00402947	00	DB	00	
00402948	00	DB	00	
00402949	00	DB	00	
0040294A	00	DB	00	
0040294B	00	DB	00	
0040294C	00	DB	00	
0040294D	00	DB	00	
0040294E	13	DB	13	
0040294F	00	DB	00	
00402950	00	DB	00	
00402951	00	DB	00	
00402952	00	DB	00	
00402953	00	DB	00	
00402954	24	DB	24	CHAR '\$'

Wisst ihr noch wie das Erste Byte war? Es war 00. Und, welches ist es jetzt? 00! Das heißt es funktioniert!

Gut... Jetzt Scannen wir es mal. Aha kein Dropper mehr :)

Trojan.Win32.Buzus!IK

Eins kann ich euch verraten: Die größte Hürde ist geschafft :D

Durch nochmaliges splitten (Jetzt merkt ihr vll. wie viel Zeit ich investiert habe), habe ich rausgefunden, dass die Nullbytes am Ende der Code Section detected sind.

Wir laden die Exe in den Hexeditor und überschreiben diese Nullbytes mit irgendwelchen Werten. Auch hier habt ihr wieder freie Wahl :)

Das wars schon. Speichern, Scannen UD :) :)


```

00003924 60 02 00 80 CC 02 00 80 CD 02 00 80 7A 39 00 00 84 02 00 80 3A 02 00 `.....z9.....
0000393B 80 64 00 00 80 00 00 00 00 4D 53 56 42 56 4D 36 30 2E 44 4C 4C 00 00
00003952 00 00 44 6C 6C 46 75 6E 63 74 69 6F 6E 43 61 6C 6C 00 00 00 5F 5F 76
00003969 62 61 45 78 63 65 70 74 48 61 6E 64 6C 65 72 00 00 00 00 50 72 6F 63
00003980 43 61 6C 6C 45 6E 67 69 6E 65 00 00 00 00 00 00 00 00 00 00 00 00
00003997 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000039AE 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000039C5 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36
000039DC 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36
000039F3 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36
00003A0A 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36
00003A21 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36
00003A38 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36
00003A4F 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36
00003A66 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36
00003A7D 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36
00003A94 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36
00003AAB 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36
00003AC2 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36
00003AD9 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36
00003AF0 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36
00003B07 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36
00003B1E 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36
00003B35 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36
00003B4C 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36
00003B63 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36
00003B7A 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36
00003B91 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36
00003BA8 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36
00003BBF 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36
00003BD6 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36
    
```

Und so siehts insgesamt aus:

Results from the virus scan of uploaded sample

Return to the Virus.Org Scanning Service

The following represents the test results from the virus scanners used by the Virus.Org scanning service when it performed the scan on the file 'Fake_Sig6.exe'.

File: Fake_Sig6.exe

SHA-1 Digest: e75cdef77b9a3df8fad8c026a136cdd9bb4a7227

Size: 20480 bytes

Detected Packer: Armadillo 3.00a

Status: Infected or Malware (Confidence 4.55%)

Date Scanned: Fri Apr 17 21:33:52 +0100 2009

Scanner	Scanner Version	Scanner Engine	Scanner Signatures	Result	Scan Time
A-Squared	4.0.0.32	N/A	1239998403	Clean	15.23 secs
Arcavir	1.0.5	N/A	12:20 20-03-2009	Clean	4.57 secs
avast!	1.0.8	N/A	090417-0	Clean	15.14 secs
AVG Anti Virus	7.5.52	442	270.12.0/2065	Clean	17.42 secs
Avira AntiVir	2.1.12-152	7.9.0.148	7.1.3.72	Clean	17.77 secs
BitDefender	7.81008	7.24859	2848894	Gen: Trojan.Heur.VB.1024DB9B9B	4.22 secs
CA eTrust	N/A	31.06.00	31.06.6435	Clean	4.78 secs
CAT QuickHeal	10.00	N/A	17 April, 2009	Clean	16.21 secs
ClamAV	0.94.2	N/A	9252	Clean	0.02 secs
Comodo	3.8	3.8	1117	Clean	10.65 secs
CPSecure	1.15	1.1.0.715	17/04/2009 11:40AM	Clean	6.35 secs
Dr. Web	4.44.0.10060	4.44.0.9170	569276	Clean	31.93 secs
F-PROT 6	6.2.1.4252	4.4.4.56	200904171529	Clean	9.36 secs
F-Secure	1.10	6392	2009-04-17_09	Clean	31.42 secs
Ikarus T3SCAN	1.32.4.0	1.01.49	2009-04-17 07:16:41	Clean	12.80 secs
Kaspersky	5.7.13	1859034	17-04-2009	Clean	27.53 secs
McAfee Virusscan	5.30.0	5.3.00	v5579	Clean	14.96 secs
Norman Virus Control	7.00.00	6.00.06	6.00.00	Clean	58.14 secs
Panda	9.04.03.0001	1848106	06/04/2009	Clean	7.01 secs
Sophos Sweep	4.40.0	2.85.0	4.40	Clean	19.10 secs
Trend Micro	N/A	8.700-1004	974	Clean	3.79 secs
VirusBuster 2005	1.4.5	4.6.5	10.102.32	Clean	11.26 secs

Nurnoch BitDefender...

Bei diesem AV stehen wir vor einem großen Problem. Den Emulator.

Dieser Emuliert unsere exe. Das heißt, er startet unsere Exe in sicherer Umgebung und analysiert das Verhalten

der Datei. Gegen diese Methode ist ohne Sourcecode nicht viel zu machen. Wir könnten zwar einige Anti-Emulator

Codes in ASM nachcoden, jedoch wirft das wieder neue Detections auf.

Wenn Interesse besteht sowas trotzdem zu erklären bin ich gerne bereit dafür. Meldet euch dann ggf. einfach im

Thread.

An dieser Stelle endet dieses Tutorial, da wir hier, wie gesagt nicht ohne Weiteres weiterkommen.

Aber ich denke Das Ergebnis ist Akzeptabel ;)

Outro

In diesem Tutorial wollte ich neue und saubere Methoden zeigen um AVs zu bypassen.

Damit auch Anfänger wissen, wie man hier vorzugehen hat, wollte ich es möglichst einfach rüberbringen.

Zuerst wollte ich daraus ein V-Tut machen, das wäre allerdings zu 60% nur das mitlesen von Text gewesen, wofür diese Form von Tutorial einfach besser geeignet ist. Durch die vielen Screenshots soll die Übersichtlichkeit eines V-Tuts hergestellt werden.

Sehr würde ich mich über zahlreiches Feedback im Forum freuen. So ein Tutorial zu machen ist viel Vorbereitung und man möchte ja wissen wie es ankommt ;)

Wenn es noch Fragen gibt steht dafür der Thread auf Free-Hack offen. Ich werde alle Fragen weitestgehend beantworten.

Wer mit mir einen Smalltalk zum Thema halten möchte, ich stehe gerne unter folgender ICQ Nummer zur Verfügung:

296119081

Ich hoffe es hat euch gefallen und ihr könnt was damit anfangen.

Greetz & Shouts

So erstmal n dickes **DANKESCHÖN** an den „AV Bypasser Freak“ aka **Slayer616**, der mir wichtige Tipps fürs Tut gegeben hat.

Ich grüße außerdem das Komplette **SceneCoderz Team**. Freu mich sehr auf den Relaunch!

Ebenfalls grüße ich **f0Gx**, der mir schon so manche (dämliche *hust*) Frage beantwortet hat.

Naja ich will jetzt hier keine langen Listen machen, ich würde sowieso welche vergessen.

Ich grüße einfach alle, Reverser da draußen und die, die es werden wollen ☺

Das größte **DANKE** gebürt aber dir, für das Lesen meines Tutorials!