Brief Introduction	SSEXY	SSEXY Internals	Obstacles	Improvements	Last notes
00		0	0000	0	
0	0	0		0	
0	0	0000		000	
00					
0					

SSEXY Binary Obfuscation using SSE

Jurriaan Bremer @skier_t

Hack in the Box Amsterdam 2012

May 24, 2012

Jurriaan Bremer @skier_t

Binary Obfuscation using SSE

May 24, 2012 1 / 31

◆□▶ ◆□▶ ◆臣▶ ◆臣▶ □臣 = のへで

Brief Introduction	SSEXY	SSEXY Internals	Obstacles	Improvements	Last notes
00		0	0000	0	
0	0	0		0	
0	0	0000		000	
00					

Table of Contents

Brief Introduction

SSEXY

SSEXY Internals

Obstacles

Improvements

Last notes

Jurriaan Bremer @skier_t

Binary Obfuscation using SSE

May 24, 2012 2 / 31

Brief Introduction	SSEXY	SSEXY Internals	Obstacles	Improvements	Last notes
• • •	000	0	0000	0	00
000	0	0000		000	
-					



Student at the University of Amsterdam Interested in Low-Level Stuff Hack in the Box CTF De Eindbazen

Jurriaan Bremer @skier_t

Binary Obfuscation using SSE

May 24, 2012 3 / 31

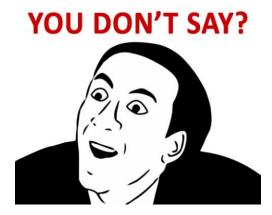
◆□▶ ◆□▶ ◆臣▶ ◆臣▶ □臣 = のへで

Brief Introduction	SSEXY	SSEXY Internals	Obstacles	Improvements	Last notes
•0	000	0	0000	0	00
0	0	0		0	
0	0	0000		000	
00					
0					
0 00 0	0	0000		000	

Me?

Student at the University of Amsterdam Interested in Low-Level Stuff Hack in the Box CTF De Eindbazen Giving this Presentation!

Brief Introduction	SSEXY	SSEXY Internals	Obstacles	Improvements	Last notes
0.	000	0	0000	0	00
0	0	0		0	
0	0	0000		000	
00					
0					



Jurriaan Bremer @skier_t

Binary Obfuscation using SSE

৳▶ ৰ≣▶ ≣ ∽ি৭ে May 24, 2012 4 / 31

(日) (同) (日) (日)

Brief Introduction	SSEXY	SSEXY Internals	Obstacles	Improvements	Last notes
00	000	0	0000	0	00
•	0	0		0	
0	0	0000		000	
00					
0					

PE File - Windows Binary

Bunch of Headers

Defines:

```
Data (e.g. strings, "hello world")
Imported Functions (e.g. printf)
Metadata (e.g. Relocations)
Code
```

Brief Introduction	SSEXY	SSEXY Internals	Obstacles	Improvements	Last notes
00		0	0000	0	
•	0	0		0	
0	0	0000		000	
00					
0					

PE File - Windows Binary

Bunch of Headers

Defines:

```
Data (e.g. strings, "hello world")
Imported Functions (e.g. printf)
Metadata (e.g. Relocations)
Code
```

Brief Introduction	SSEXY	SSEXY Internals	Obstacles	Improvements	Last notes
00	000	0	0000	0	00
•	0	0		0	
0	0	0000		000	
00					
0					

PE File - Windows Binary

Bunch of Headers

Defines:

Data (e.g. strings, "hello world") Imported Functions (e.g. printf) Metadata (e.g. Relocations) Code

Brief Introduction	SSEXY	SSEXY Internals	Obstacles	Improvements	Last notes
00	000	0	0000	0	00
•	0	0		0	
0	0	0000		000	
00					
0					

PE File - Windows Binary

Bunch of Headers

Defines:

```
Data (e.g. strings, "hello world")
Imported Functions (e.g. printf)
Metadata (e.g. Relocations)
Code
```

Brief Introduction	SSEXY	SSEXY Internals	Obstacles	Improvements	Last notes
00		0	0000	0	
•	0	0		0	
0	0	0000		000	
00					
0					

PE File - Windows Binary

Bunch of Headers

Defines:

Data (e.g. strings, "hello world") Imported Functions (e.g. printf) Metadata (e.g. Relocations) Code

Brief Introduction	SSEXY	SSEXY Internals	Obstacles	Improvements	Last notes
00		0	0000	0	
•	0	0		0	
0	0	0000		000	
00					
0					

PE File - Windows Binary

Bunch of Headers

Defines:

Data (e.g. strings, "hello world") Imported Functions (e.g. printf) Metadata (e.g. Relocations) Code

Brief Introduction	SSEXY	SSEXY Internals	Obstacles	Improvements	Last notes
00		0	0000	0	
0	0	0		0	
•	0	0000		000	
00					
0					

x86 Instruction Set

Variable Instruction Size Eight 32bit General Purpose Registers Approximately 100 "normal" Instructions

Brief Introduction	SSEXY	SSEXY Internals	Obstacles	Improvements	Last notes
00		0	0000	0	
0	0	0		0	
•	0	0000		000	
00					
0					

x86 Instruction Set

Variable Instruction Size Eight 32bit General Purpose Registers Approximately 100 "normal" Instructions SIMD (Single Instruction, Multiple Data) SSE and SSE2 (Streaming SIMD Extensions) Pentium 3 (SSE), Pentium 4 (SSE2) Eight 128bit XMM Registers Few dozen Instructions

Brief Introduction	SSEXY	SSEXY Internals	Obstacles	Improvements	Last notes
00		0	0000	0	
0	0	0		0	
0	0	0000		000	
•0					
0					

Traditional Virtual Machines (1)

Custom Bytecode, Custom Context Main Execution Loop

Jurriaan Bremer @skier_t

Binary Obfuscation using SSE

May 24, 2012 7 / 31

Brief Introduction	SSEXY	SSEXY Internals	Obstacles	Improvements	Last notes
00		0	0000	0	
0	0	0		0	
0	0	0000		000	
•0					
0					

Traditional Virtual Machines (1)

Custom Bytecode, Custom Context Main Execution Loop

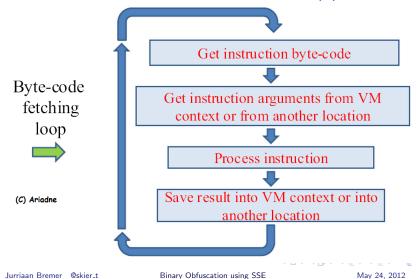
Jurriaan Bremer @skier_t

Binary Obfuscation using SSE

May 24, 2012 7 / 31







8 / 31

Brief Introduction	SSEXY	SSEXY Internals	Obstacles	Improvements	Last notes
00		0	0000	0	
0	0	0		0	
0	0	0000		000	
00					

Metamorphic Code

Listing	1:	Before
---------	----	--------

mov eax, 0x100 mov ebx, 0x200 Listing 2: After

push 0x2100

pop eax

sub eax, 0x2000

mov ebx, eax xor ebx, 0x300

Jurriaan Bremer @skier_t

Binary Obfuscation using SSE

May 24, 2012 9 / 31

◆□▶ ◆□▶ ◆臣▶ ◆臣▶ 三臣 - のへで

Brief Introduction	SSEXY	SSEXY Internals	Obstacles	Improvements	Last notes
00	000	0	0000	0	
0	0	0		0	
0	0	0000		000	
00					

Why SSE? (1)

Uncommon Instructions

Jurriaan Bremer @skier_t

Binary Obfuscation using SSE

May 24, 2012 10 / 31

◆□▶ ◆□▶ ◆臣▶ ◆臣▶ □臣 = のへで

Brief Introduction	SSEXY	SSEXY Internals	Obstacles	Improvements	Last notes
00	000	0	0000	0	
0	0	0		0	
0	0	0000		000	
00					
0					

Why SSE? (1)

Uncommon Instructions

Did you ever encounter SSE during Reverse Engineering?

Jurriaan Bremer @skier_t

Binary Obfuscation using SSE

May 24, 2012 10 / 31

◆□▶ ◆□▶ ◆臣▶ ◆臣▶ □臣 = のへで

Brief Introduction	SSEXY	SSEXY Internals	Obstacles	Improvements	Last notes
00	000	0	0000	0	
0	0	0		0	
0	0	0000		000	
00					
0					

Why SSE? (1)

Uncommon Instructions

Did you ever encounter SSE during Reverse Engineering?



Jurriaan Bremer @skier_t

Binary Obfuscation using SSE

May 24, 2012 10 / 31

∃ <\0<</p>

(日)、

Brief Introduction	SSEXY	SSEXY Internals	Obstacles	Improvements	Last notes
00	000	0	0000	0	00
0	0	0		0	
0	0	0000		000	
00					
0					

Why SSE? (2)

Obscure Instruction Names

Jurriaan Bremer Øskier_t

Binary Obfuscation using SSE

May 24, 2012 11 / 31

◆□▶ ◆□▶ ◆臣▶ ◆臣▶ □臣 = のへで

Brief Introduction	SSEXY	SSEXY Internals	Obstacles	Improvements	Last notes
00	000	0	0000	0	00
0	0	0		0	
0	0	0000		000	
00					
0					

Why SSE? (2)

Obscure Instruction Names

What do you think 'CVTTPD2DQ' does?

Jurriaan Bremer @skier_t

Binary Obfuscation using SSE

May 24, 2012 11 / 31

Brief Introduction	SSEXY	SSEXY Internals	Obstacles	Improvements	Last notes
00	000	0	0000	0	00
0	0	0		0	
0	0	0000		000	
00					
0					

Why SSE? (2)

Obscure Instruction Names

What do you think 'CVTTPD2DQ' does?

Convert with Truncation Packed Double-Precision FP Values to Packed Dword Integers

Jurriaan Bremer @skier_t

Binary Obfuscation using SSE

May 24, 2012 11 / 31

Brief Introduction	SSEXY	SSEXY Internals	Obstacles	Improvements	Last notes
00	000	0	0000	0	00
0	0	0		0	
0	0	0000		000	
00					
0					

Why SSE? (3)

Enormous Registers

Jurriaan Bremer Øskier_t

Binary Obfuscation using SSE

May 24, 2012 12 / 31

◆□▶ ◆□▶ ◆臣▶ ◆臣▶ □臣 = のへで

Brief Introduction	SSEXY	SSEXY Internals	Obstacles	Improvements	Last notes
00	000	0	0000	0	
0	0	0		0	
0	0	0000		000	
00					
0					

Why SSE? (3)

Enormous Registers

How many General Purpose Registers fit in an XMM Register? (This is not a trick question)

Jurriaan Bremer @skier_t

Binary Obfuscation using SSE

May 24, 2012 12 / 31

Brief Introduction	SSEXY	SSEXY Internals	Obstacles	Improvements	Last notes
00	000	0	0000	0	
0	0	0		0	
0	0	0000		000	
00					
0					

Why SSE? (3)

Enormous Registers

How many General Purpose Registers fit in an XMM Register? (This is not a trick question)

Four..

Jurriaan Bremer @skier_t

Binary Obfuscation using SSE

May 24, 2012 12 / 31

Brief Introduction	SSEXY	SSEXY Internals	Obstacles	Improvements	Last notes
00		0	0000	0	
0	•	0		0	
0	0	0000		000	
00					
0					

Load a PE File as Input (*pefile*)

Parse the PE Headers Disassemble all Instructions (*distorm3*, !) Analyze Metadata (!) Needs Relocations At the moment not very generic :(

Generate a new Binary

Translate all Instructions (*pyasm2* + *ssexy*, !) Craft a PE Binary (*gcc*)

Jurriaan Bremer @skier_t

Binary Obfuscation using SSE

May 24, 2012 13 / 31

Brief Introduction	SSEXY	SSEXY Internals	Obstacles	Improvements	Last notes
00		0	0000	0	
0	•	0		0	
0	0	0000		000	
00					
0					

Load a PE File as Input (*pefile*) Parse the PE Headers Disassemble all Instructions (*distorm3*, !) Analyze Metadata (!) Needs Relocations At the moment not very generic :(Generate a new Binary

Translate all Instructions (*pyasm2* + *ssexy*, !) Craft a PE Binary (*gcc*)

Jurriaan Bremer @skier_t

Binary Obfuscation using SSE

May 24, 2012 13 / 31

Brief Introduction	SSEXY	SSEXY Internals	Obstacles	Improvements	Last notes
00		0	0000	0	
0	•	0		0	
0	0	0000		000	
00					
0					

Load a PE File as Input (*pefile*)

Parse the PE Headers

Disassemble all Instructions (distorm3, !)

Analyze Metadata (!)

Needs Relocations

At the moment not very generic :(

Generate a new Binary

Translate all Instructions (*pyasm2* + *ssexy*, !) Craft a PE Binary (*gcc*)

Jurriaan Bremer @skier_t

Binary Obfuscation using SSE

May 24, 2012 13 / 31

Brief Introduction	SSEXY	SSEXY Internals	Obstacles	Improvements	Last notes
00		0	0000	0	
0	•	0		0	
0	0	0000		000	
00					
0					

Load a PE File as Input (*pefile*)

Parse the PE Headers Disassemble all Instructions (*distorm3*, !) Analyze Metadata (!) Needs Relocations

At the moment not very generic :(

Generate a new Binary

Translate all Instructions (*pyasm2* + *ssexy*, !) Craft a PE Binary (*gcc*)

Jurriaan Bremer @skier_t

Binary Obfuscation using SSE

May 24, 2012 13 / 31

Brief Introduction	SSEXY	SSEXY Internals	Obstacles	Improvements	Last notes
00		0	0000	0	
0	•	0		0	
0	0	0000		000	
00					
0					

Load a PE File as Input (*pefile*)

Parse the PE Headers Disassemble all Instructions (*distorm3*, !) Analyze Metadata (!)

Needs Relocations

At the moment not very generic :(

Generate a new Binary

Translate all Instructions (*pyasm2* + *ssexy*, !) Craft a PE Binary (*gcc*)

Jurriaan Bremer @skier_t

Binary Obfuscation using SSE

May 24, 2012 13 / 31

Brief Introduction	SSEXY	SSEXY Internals	Obstacles	Improvements	Last notes
00		0	0000	0	
0	•	0		0	
0	0	0000		000	
00					
0					

Load a PE File as Input (*pefile*)

Parse the PE Headers Disassemble all Instructions (*distorm3*, !) Analyze Metadata (!) Needs Relocations

At the moment not very generic :(

Generate a new Binary

Translate all Instructions (*pyasm2* + *ssexy*, !) Craft a PE Binary (*gcc*)

Jurriaan Bremer @skier_t

Binary Obfuscation using SSE

May 24, 2012 13 / 31

Brief Introduction	SSEXY	SSEXY Internals	Obstacles	Improvements	Last notes
00		0	0000	0	
0	•	0		0	
0	0	0000		000	
00					
0					

Load a PE File as Input (*pefile*) Parse the PE Headers Disassemble all Instructions (*distorm3*, !) Analyze Metadata (!) Needs Relocations At the moment not very generic :(

Generate a new Binary

Translate all Instructions (*pyasm2* + *ssexy*, !) Craft a PE Binary (*gcc*)

Jurriaan Bremer @skier_t

Binary Obfuscation using SSE

May 24, 2012 13 / 31

Brief Introduction	SSEXY	SSEXY Internals	Obstacles	Improvements	Last notes
00		0	0000	0	
0	•	0		0	
0	0	0000		000	
00					
0					

Load a PE File as Input (*pefile*) Parse the PE Headers Disassemble all Instructions (*distorm3*, !) Analyze Metadata (!) Needs Relocations At the moment not very generic :(

Generate a new Binary

Translate all Instructions (*pyasm2* + *ssexy*, !) Craft a PE Binary (*gcc*)

Jurriaan Bremer @skier_t

Binary Obfuscation using SSE

May 24, 2012 13 / 31

Brief Introduction	SSEXY	SSEXY Internals	Obstacles	Improvements	Last notes
00		0	0000	0	
0	•	0		0	
0	0	0000		000	
00					
0					

Load a PE File as Input (*pefile*) Parse the PE Headers Disassemble all Instructions (*distorm3*, !) Analyze Metadata (!) Needs Relocations At the moment not very generic :(

Generate a new Binary

Translate all Instructions (*pyasm2* + *ssexy*, !) Craft a PE Binary (*gcc*)

Jurriaan Bremer @skier_t

Binary Obfuscation using SSE

May 24, 2012 13 / 31

Brief Introduction	SSEXY	SSEXY Internals	Obstacles	Improvements	Last notes
00		0	0000	0	
0	0	0		0	
0	•	0000		000	
00					
0					

pyasm2

Intel syntax: mov ecx, dword [ebp+8]

>>> from pyasm2 import *
>>> mov(ecx, dword[ebp+8])
<pyasm2.mov instance at 0x....>

>>> str(mov(ecx, dword[ebp+8])) 'mov ecx, dword [ebp+0x8]'

```
>>> mov(ecx, dword[ebp+8]).encode() '\x8bM\x08'
```

Brief Introduction	SSEXY	SSEXY Internals	Obstacles	Improvements	Last notes
00		•	0000	0	00
0	0	0		0	
0	0	0000		000	
00					
0					

General Purpose Register stored in XMM Registers:

Eight 32bit GPRs fit in 2 XMM Registers eax, ecx, edx, ebx in xmm6 esp, ebp, esi, edi in xmm7 xmm0, ..., xmm3 for intermediate values

General Purpose Instructions:

Written as a sequence of SSE Instructions; Source operand(s) are loaded Some operation is performed Destination operand(s) are written

Jurriaan Bremer @skier_t

Binary Obfuscation using SSE

불 ▶ < 불 ▶ 불 ∽ ९. May 24, 2012 15 / 31

Brief Introduction	SSEXY	SSEXY Internals	Obstacles	Improvements	Last notes
00		•	0000	0	00
0	0	0		0	
0	0	0000		000	
00					
0					

General Purpose Register stored in XMM Registers: Eight 32bit GPRs fit in 2 XMM Registers eax, ecx, edx, ebx in xmm6 esp, ebp, esi, edi in xmm7 xmm0, ..., xmm3 for intermediate values General Purpose Instructions:

Written as a sequence of SSE Instructions; Source operand(s) are loaded Some operation is performed Destination operand(s) are written

Jurriaan Bremer @skier_t

Binary Obfuscation using SSE

불▶ < 불▶ 불 ∽ Q @ May 24, 2012 15 / 31

Brief Introduction	SSEXY	SSEXY Internals	Obstacles	Improvements	Last notes
00		•	0000	0	00
0	0	0		0	
0	0	0000		000	
00					
0					

General Purpose Register stored in XMM Registers:

Eight 32bit GPRs fit in 2 XMM Registers eax, ecx, edx, ebx in xmm6 esp, ebp, esi, edi in xmm7 xmm0, ..., xmm3 for intermediate values

General Purpose Instructions:

Written as a sequence of SSE Instructions; Source operand(s) are loaded Some operation is performed Destination operand(s) are written

Jurriaan Bremer @skier_t

Binary Obfuscation using SSE

불 ▶ ◀ 불 ▶ 불 ∽ ९. May 24, 2012 15 / 31

Brief Introduction	SSEXY	SSEXY Internals	Obstacles	Improvements	Last notes
00		•	0000	0	00
0	0	0		0	
0	0	0000		000	
00					
0					

General Purpose Register stored in XMM Registers:

Eight 32bit GPRs fit in 2 XMM Registers eax, ecx, edx, ebx in xmm6 esp, ebp, esi, edi in xmm7 xmm0, ..., xmm3 for intermediate values

General Purpose Instructions:

Written as a sequence of SSE Instructions; Source operand(s) are loaded Some operation is performed Destination operand(s) are written

Jurriaan Bremer @skier_t

Binary Obfuscation using SSE

불 ▶ ◀ 불 ▶ 불 ∽ ९. May 24, 2012 15 / 31

Brief Introduction	SSEXY	SSEXY Internals	Obstacles	Improvements	Last notes
00		•	0000	0	00
0	0	0		0	
0	0	0000		000	
00					
0					

General Purpose Register stored in XMM Registers: Eight 32bit GPRs fit in 2 XMM Registers eax, ecx, edx, ebx in xmm6 esp, ebp, esi, edi in xmm7 xmm0, ..., xmm3 for intermediate values

General Purpose Instructions:

Written as a sequence of SSE Instructions; Source operand(s) are loaded Some operation is performed Destination operand(s) are written

Jurriaan Bremer @skier_t

Binary Obfuscation using SSE

E ► 4 E ► E ∽ Q ⊂ May 24, 2012 15 / 31

Brief Introduction	SSEXY	SSEXY Internals	Obstacles	Improvements	Last notes
00		•	0000	0	00
0	0	0		0	
0	0	0000		000	
00					
0					

General Purpose Register stored in XMM Registers: Eight 32bit GPRs fit in 2 XMM Registers eax, ecx, edx, ebx in xmm6 esp, ebp, esi, edi in xmm7 xmm0, ..., xmm3 for intermediate values

General Purpose Instructions:

Written as a sequence of SSE Instructions; Source operand(s) are loaded Some operation is performed Destination operand(s) are written

Jurriaan Bremer @skier_t

Binary Obfuscation using SSE

E ► < E ► E ∽ Q C May 24, 2012 15 / 31

Brief Introduction	SSEXY	SSEXY Internals	Obstacles	Improvements	Last notes
00		•	0000	0	00
0	0	0		0	
0	0	0000		000	
00					
0					

General Purpose Register stored in XMM Registers: Eight 32bit GPRs fit in 2 XMM Registers eax, ecx, edx, ebx in xmm6 esp, ebp, esi, edi in xmm7 xmm0, ..., xmm3 for intermediate values

General Purpose Instructions:

Written as a sequence of SSE Instructions; Source operand(s) are loaded Some operation is performed Destination operand(s) are written

Jurriaan Bremer @skier_t

Binary Obfuscation using SSE

E ► < E ► E ∽ Q C May 24, 2012 15 / 31

Brief Introduction	SSEXY	SSEXY Internals	Obstacles	Improvements	Last notes
00		•	0000	0	00
0	0	0		0	
0	0	0000		000	
00					
0					

General Purpose Register stored in XMM Registers: Eight 32bit GPRs fit in 2 XMM Registers eax, ecx, edx, ebx in xmm6 esp, ebp, esi, edi in xmm7 xmm0, ..., xmm3 for intermediate values

General Purpose Instructions:

Written as a sequence of SSE Instructions; Source operand(s) are loaded Some operation is performed Destination operand(s) are written

Jurriaan Bremer @skier_t

Binary Obfuscation using SSE

불 ▶ < 불 ▶ 불 ∽ ९. May 24, 2012 15 / 31

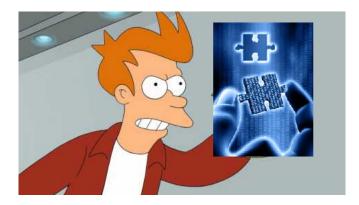
Brief Introduction	SSEXY	SSEXY Internals	Obstacles	Improvements	Last notes
00		0	0000	0	
0	0	•		0	
0	0	0000		000	
00					

Practical

Listing 4: After movd xmm3, [__m32_0] pshufd xmm2, xmm7, 1 paddd xmm3, xmm2 movd eax, xmm3 Listing 3: Before mov eax, [eax] mov ecx, [ebp+8]movd xmm0, eax pshufd xmm0, xmm0, 0 pand xmm0, [__m128_0] pand xmm6, [__m128_1] pxor xmm6, xmm0

Brief Introduction	SSEXY	SSEXY Internals	Obstacles	Improvements	Last notes
00		0	0000	0	
0	0	0		0	
0	0	0000		000	
00					
0					

Shut up and take my Binary! (1)



Jurriaan Bremer @skier_t

Binary Obfuscation using SSE

≣ ▶ < ≣ ▶ ≣ ∽ ९ (° May 24, 2012 17 / 31

・ロト ・ 同ト ・ ヨト ・ ヨ

Brief Introduction	SSEXY	SSEXY Internals	Obstacles	Improvements	Last notes
00		0	0000	0	
0	0	0		0	
0	0	0000		000	
00					
0					

Shut up and take my Binary! (2)

Listening daemon which checks the input against a hardcoded hash and executes it when the hash matches.

Jurriaan Bremer @skier_t

Binary Obfuscation using SSE

May 24, 2012 18 / 31

Brief Introduction	SSEXY	SSEXY Internals
00		0
0	0	0
0	0	0000
00		

rnals		(

Obstacles 0000 Improvemen 0 0 000 Last notes

Shut up and take my Binary! (3)

0700	(HOV	LEHAJ, EUA	ØF1ØDB	MOUUPS	XMM3, XMM3
660F70C7 00	PSHUFD	XMM0, XMM7, 00		HUVUFS	A1110, A1110
440E70C0 00	Deulien	XMMO XMMO OO	660F7ED8	MOVD	EAX, XMM3 EAX, [EAX]
CC050000 001	DOND	XMM0, [401500]	8B00	MOU	EAX. [EAX]
660FD805 001	PHNU	Anno, 14015001	660F6ED8	MOUD	XMM3, EAX
660FDB3D 101	PAND	XMM7 [401510]	0F10C3	MOVUPS	XMMO, XMM3
660EEEE8	PX0R	XMM7 XMM0	0F10U3	novues	Anno, Anns
660FDB05 001 660FDB3D 101 660FEFF8 660F70DF 00	Deulien	XMM3, XMM7, 00	660F70C0 00	PSHUFD	XMMO, XMMO, OO
660FFA1D 201	POHOPD		660FDB05 701	PAND PAND PXOR	XMM0 [401570]
660FFHID 201	PSUBD	XMM3, E40152<u>01</u>	660FDB35 801	DOND	XMM6. [401580]
660F70DB 00	PSHUFD	XMM3 XMM3 00	660FEFF0	6000	XMM6, XMM0
660FDB1D 301	PAND	XMM3 E4015301	660FEFF0	PAUR	2006, 2000
220EDDOD 401	DOND	XMM7, [401540]	660F70C6 01	PSHUFD	XMMØ, XMM6, 01
660FDB3D 401 660FEFFB 660F6E1D 941	PHOD	AUD7, 19010401	660F6E1D A81	MOVD	XMM3. E4013081
660FEFFB	PXUR	XMM7, XMM3	ØF10CB	MOUUPS	XMM1 XMM3
660F6E1D 941	MOVD	XMM3, E4013941	660FFAC1	PSUBD	SHHO. SHHI
660F70D7 00	PSHUED	XMM2 XMM7 00	000FFHU1	FOUDD	Anno, Anni
660FFEDA	PADDD	XMM3, XMM2	660F70C0 00	PSHUFD	XMM0, XMM0, 00
OCCOPPEDA	MOUUDO		660FDB05 901	PAND	XMM0 [401590]
ØF10DB	MOVUPS	XMM3, XMM2 XMM3, XMM3 EAX, XMM3 XMM0, XMM6, 01 ECX XMM6, 01	660F70C0 00 660FDB05 901 660FDB35 A01	POND	XMM6. [4015A0]
660F7ED8	MOVD	EAX, XMM3	220000000000 H011	6006	XMM6 XMM0
660F70C6 01	PSHUFD	XMMA, XMM6, M1	660FEFF0	PAUK	Anno, Anno
660F7EC1	MOVD	ECX. XMM0	660FEFF0 660F70C6 01 660F6E1D AC1:	PSHUFD	XMMO, XMM6, 01
COOPTECT	HOVD	ECO, OHIO	660E6E1D 8C11	MOUD	XMM3 E4013ACI
8908	nuv	LEHAJ, EUA	660F70D7 01	PSHUED	XMM2 XMM7 01
660F6E1D 981	MOVD	XMM3. [401398]		PADDD	XMM3 XMM2
0E10C3	MOULES	CEAXI, ECX XMM3, C4013981 XMM0, XMM3	660FFEDA	PHUUU	Anns, Ann2
660F6E1D 9C1 660F70D7 01	MOUD	XMM3, E40139C1	ØF1ØDB	MOVUPS	XMM3, XMM3
COPPOEID SCI	DOUDED .		660F7ED8	MOVD MOVD MOV	EAX, XMM3 ECX, XMM0 LEAXI , ECX
660F70D7 01	PSHUED	XMM2, XMM7, 01	660F7EC1	MOUD	ECY YMMO
660FFEDH	PADDD	XMM3, XMM2	8908	MOUL	FEOUR FOU
ØF1ØDB	MOVUPS	XMM3, XMM3	0700	nov	LEHOJ, EUO
660F7ED8	MOUD	EOV VMMO	660F70DE 00	PSHUFD	XMM3, XMM6, 📴
660F7EC1	MOVD MOVD MOV	XMM3, XMM2 XMM3, XMM3 EAX, XMM3 ECX, XMM0 (EAX1 , ECX	660F7ED8	MOVD	EAX. XMM3
	NUOD	EUX, XNN0	660F70DE 00	PSHUFD	XMM3, XMM6, 🙋
8908	I MOV	CERXI, ECX	660F7EDB	MOUD	EBX, XMM3
660F6E1D A01	I MOVD	XMM3. [4013A0]	000F/EDD	1000	EBO, 0009
660EZ0DZ 01	PCHIER	VMM2 VMM2 D1	85D8	TEST	ERX, EBX
660FFFF00	DODDD	XMM3, E4013A01 XMM2, XMM7, 01 XMM3, XMM2	0F84 B501000	JE	EAX, EBX 00402470
DOULLEDH	PHUDU	XMM3, XMM2 XMM3, XMM3 EAX, XMM3 FAX, TFAX1	660F6E1D B01:	JE MOVD	XMM3. [4013B0]
0F10DB	MOUUPS	XMM3, XMM3	660F70D7 01	PSHUFD	XMM2, XMM7, 01
660F7ED8	MOUD	EAX. XMM3	00001001 01	ronuru	0006 0006 🛤
8B00	MOU	FOX FFOX1	660FFEDA	PADDD	XMM3, XMM2
660F6ED8	MOUD	EAX, [EAX] XMM3, EAX	ØF1ØDB	MOUUPS	XMM3. XMM3
ODUFOEDO	HOUDE	0000, 5000	660F7ED8	MOVD	EAX, XMM3 EAX, LEAXI
0F10C3	I NUVUPS	XMM0, XMM3	8B00	MOLL	EOV FEOVI
660F70C0 00	PSHUFD	XMMO XMMO DO		MOU MOVD	LUCA FEOD
660FDB05 501	PAND	XMM0, E4015503	600F0ED8	HOVD	XMM3, EAX
660FDB35 601	POND	XMM6. [401560]	0F10C3	MOVUPS	XMM0, XMM3
000000000000000	PAND PXOR		660F70C0 00	PSHUED	XMM0, XMM0, 00
660FEFF0	PAUR	XMM6, XMM0	CONTRAC DOLL	POND	XMM0. [4015B0]
660F6E1D A41	MOVD	XMM3. [4013A4]	CC050005 0011	6666	UMM2 14015001
660F70D7 01	PSHUED	XMM3, [4013A4] XMM2, XMM7, 01	660FDB35 C01	PHNU	XMM6, [4015C0]
660FFEDA	Ponnn	XMM3 XMM2	660F6ED8 0F10C3 660F70C0 00 660FDB05 B01! 660FDB35 C01! 660FEFF0 660FEFF0	PXOR	XMM6, XMM0
COOFFEDH	1 4000	01119, 01116	22002010 D411	MOUD	VMMO FARIODAT

3 lines of C

불▶ < 불▶ 불 ∽ Q @ May 24, 2012 19 / 31

Brief Introduction	SSEXY	SSEXY
00		0
0	0	0
0	0	0000
00		

SSEXY	Internals
0	
0	
~ ~ ~ ~	

Obstacles 0000 Improvemer 0 0 000 Last notes

Shut up and take my Binary! (3)

0700	THOV	LEHAJ, EUA	ØF1ØDB	MOUUPS	XMM3, XMM3
660F70C7 00	PSHUFD	XMM0 XMM7 00			AHH5, AHH5
66057000 00	DOLUTED	XMMÖ, XMMÖ, ÖÖ	660F7ED8	MOVD	EAX, XMM3 EAX, CEAX]
660F70C0 00	PSHUFU	Anine, Anine, ee	8800	MOLL	FOY FEOVI
660FDB05 001	PAND	XMM0. E401500]		HOVE	5002 5500
660EDDOD 1011	DOND	VMM7 FAGIE1GT	660F6ED8	MOVD	XMM3, EAX
000FDD3D 101	CHUD	XMM0, [401500] XMM7, [401510] XMM7, XMM0	0E10C2	MOUUPS	XMM0, XMM3
660FEFF8	I PXOR	XMM7. XMM0	001000000000	DOUUED.	
660F70DF 00	PSHUED	XMM3, XMM7, DO	0F10C3 660F70C0 00 660FDB05 701	PSHUFD	XMMO, XMMO, 🙆
00011001 00	POHOF D		660FDB05 701	PAND PAND PXOR	XMM0 [401570]
660FFA1D 201	PSUBD	XMM3, [401520]	660FDB35 8011	DOND	XMM6, [401580]
660EZ0DB 00	PSHUED	XMM3, XMM3, 00	000000000 0011	CHUD	VIIIO, 14010001
CC000010 0011	DOMD	VMMO EXOTECOD	660FEFF0	PXOR	XMM6, XMM0
660FDBID 3013	PHND	Anns, 14015301	660F70C6 01	PSHUFD	XMMO, XMM6, 01
660FDB3D 401	PAND	XMM7. [401540]	0000110000101	MOVD	
660FFFFB	PYOP	VMM2 VMM2	660F6E1D A81	novu	XMM3, E4013A8]
CODECTION ON A	HOUR	XMM3, [401530] XMM7, [401540] XMM7, XMM3 XMM7, XMM3 XMM3, [401394]	ØF10CB	MOVUPS	XMM1 XMM3
660F6EID 941	novo	Xnn3, 14013941	660FFAC1	PSUBD	XMM0 XMM1
660F70D7 00			COOFFHUI	FOUDD	0000, 0001
660FFEDA	PADDD	VMMO VMMO	660F70C0 00	PSHUFD	XMMO XMMO DO
OCULLEDH	FHUDD	01010, 01012	660EDB05 901	PAND PAND	XMM0 [401590]
ØF1ØDB	MOUUPS	XMM3. XMM3	0000000000 0011		AU10, 14010201
660F7ED8	MOVD	EOV VMM9	660F70C0 00 660FDB05 901 660FDB35 A01	PHNU	XMM6, [4015A0]
0000112000	DOVD		660FEFF0	PXOR	XMM6, XMM0
660F70C6 01	PSHUED	XMM3, XMM2 XMM3, XMM3 EAX, XMM3 XMM0, XMM3, B1	660FEFF0 660F70C6 01 660F6E1D AC1:	PXOR PSHUFD	XMMØ, XMM6, 01
660F7EC1			00001000001	FONORD	
0000	MOUL	FEOVA FOY	660F6E1D AC11	MOVD	XMM3, E4013ACI
0700	1100	LEHAJ, EUA	660F70D7 01	PSHUFD	XMM2, XMM7, 01
660F6E1D 981	MOVD	XMM3. [401398]	660FFEDA	PADDD	
ØF10C3	MOLIUPS	[EAX] , ECX XMM3, [401398] XMM0, XMM3	660FFEDH	PHUUU	XMM3, XMM2 XMM3, XMM3
COPCETE OCT	MOUD	X1110, X1113 X1113, 440139C1 X1112, X1117, 01 X1113, X1112, X1113, X1113 EAX, X1113 ECX, X1113 ECX, X1110 EEAX1, ECX X1113, EC4013A03	ØF1ØDB	MOUUPS	XMM3. XMM3
BEBEELD ACT	NUVD	XNN3, 14013901	660F7ED8	MOVD MOVD MOV PSHUFD	EAX, XMM3
660F70D7 01	I PSHUFD	XMM2 XMM7 01	000011200	HOVE	- E201 0002
660FFEDA 0F10DB 660F7ED8	PODDD	VMM2 VMM2	660F7EC1	novu	ECX, XMMØ LEAXI, ECX
COOTTEDA	HODD	ATTE ATTE	8908	MOU	FEGX1. ECX
0F10DB	I NUVUPS	ANNS, ANNS	660F70DE 00	DOLUTED	XMM3 XMM6 00
660EZED8	I MOUD	FOX. XMM3	000F70DE 00	ranuru	ATTIO, ATTIO, MM
660E7E01	MOUD	ECV VMMO	660F7ED8	MOVD	EAX. XMM3
CONFRENT	1000	EUA, AHHØ	660F70DE 00	PSHUED	XMM3, XMM6, 🙋
8908	I MOV	CERXI, ECX	660F7EDB	MOVD TEST	EDV VMMO
660E6E1D 001:	MOUD	XMM3 F4013001	660F7EDB	novo	EBX, XMM3
CC057057 01	DOUUED.	UMMO UMM7 OI	85D8	I TEST	EAX. EBX
666F76D7 61	FORUED	onnz, onnc, ei	0E94 E501000		EAX, EBX 00402470
660FFEDA	I PADDD	XMM3. XMM2	8508 0F84 B501000 660F6E1D B01	Mour	UNING FLORODOT
ØF10DB	MOLUPS	VMM3 VMM3	660F6EID 801	novo	XMM3, <u>[4013B0]</u>
66053500	MOUD	FOUL VMMO	660F70D7 01	PSHUED	XMM2 XMM7 M1
660F7ED8	MUVU	EHX, XMM3	660FFEDA	PADDD	VMMO VMMO
SBOO	I MOU	EAX. FEAX1	ODULLEDH	FHUUU	XMM3, XMM2 XMM3, XMM3
CCOFCEDO	MOUD	VMMO EOV	ØF1ØDB	MOUUPS	XMM3, XMM3
CONFOLDO	11000	ALLIO, EHA	660F7ED8	MOUD	FOX XMM3
0F10C3	MOVUPS	Xnm3; Xnm3; ERX; Xnm3; ECX; Xnm6; ECX; Xnm6; Xnm3; C4013A0; Xnm3; Xnm7; Xnm3; Xnm2; Xnm3; Xnm3; Xnm3; Xnm3; Xnm3; Xnm3; Enx; CERN1; Xnm3; Xnm3; Xnm3; Xnm3; Xnm3; Xnm3; Xnm3; Xnm3;	8B00	MOUL	EAX, XMM3 EAX, [EAX]
660F70C0 00	PSHUFD	XMMO, XMMO, DO	0000	HOV	CHO, LEHOJ
660FDB05 501	DOND	XMM0 [401550]	660F6ED8	MOU MOVD	XMM3. EAX
000FDB05 501	EHUD	Anno, 14015501	0F10C3	MOUUPS	XMM0 XMM3
660FDB35 601	PAND	XMM6. [401560]	000000000000000000000000000000000000000	DOUUED	
660FEFF0	PYOR	XMM6 XMM0	660F70C0 00	PSHUED	XMMO, XMMO, 00
COPECTIP 041	MOUD	VMMO E4010047	660FDB05 B01	PAND	XMM0. [4015B0]
660F6E1D A41	I NOVD	Anns, 14013H41	660F70C0 00 660FDB05 B01 660FDB35 C01	POND	XMM6, [4015C0]
660F70D7 01	PSHUED.	XMM2, XMM7, 01	000F0835 101	CHOU	ATTIO, 14015001
660FFEDA	PODDD	VMMO VMMO	660FEFF0	PXOR	XMM6, XMM0
COULLEDH	FHODD	Xmm6, E4015601 Xmm6, Xmm0 Xmm3, E4013641 Xmm2, Xmm7, 01 Xmm2, Xmm7, 01 Xmm2, Xmm2	CORECTIO DATE	MOUD	VMMO FAGIODAT

3 lines of C 20 lines of x86

Jurriaan Bremer @skier_t

Binary Obfuscation using SSE

불▶ < 불▶ 불 ∽ Q @ May 24, 2012 19 / 31

Brief Introduction	SSEXY	SSEXY Int
00		0
0	0	0
0	0	0000
00		

ternals	

Obstacles 0000 O O O OOO Last notes

Shut up and take my Binary! (3)

0700 [100	LEHAJ, EUA	ØF1ØDB	MOUUPS	VMMO VMMO
660F70C7 00 PSHUFD) XMM0, XMM7, 00			XMM3, XMM3
660F70C0 00 PSHUFD	i XMMO XMMO <mark>OO</mark>	660F7ED8	MOVD	EAX, XMM3 EAX, [EAX]
operrece ee ranuru	ATTEN ATTEN DE	8800	MOU	FOX FOX1
660FDB05 001 PAND	XMM0, E4015003	660F6ED8	MOUD	UMMO FOU
660EDB3D 1011 POND	XMM7, [401510]			ANNS, EHA
660FDB3D 101 PAND 660FEFF8 PXOR	XMM7, XMM0	0F10C3	MOVUPS	XMM3, EAX XMM0, XMM3
ODULELLO EVOR	ANNY, ANN9	660F70C0 00	PSHUED	XMMO XMMO DO
660F70DF 00 PSHUFD	I XMM3, XMM7, 00	660FDB05 701	PAND	XMM0 [401570]
660FFA1D 201 PSUBD	XMM3 [401520]		PHND	XNN0, 14015701
660F70DB 00 PSHUFD	XMM3, XMM3, 00	660FDB35 801	PAND PXOR	XMM6 [401580]
COULLED OF LESULT		660FEFF0	PX0R	XMM6 XMM0
660FDB1D_301 PHND	XMM3, [401530]	660F70C6 01	PSHUFD	XMMØ, XMM6, Ø1
660FDB1D 301 PAND 660FDB3D 401 PAND 660FEFFB PXOR	XMM7, [401540]	00001000001	FORUED	VILLE VILLE
COFFEED DVOD	XMM7, XMM3	660F6E1D A81	MOVD	XMM3, E4013A83
CODECETED OTTAINOUD		ØF10CB	MOVUPS	XMM1 XMM3
660F6E1D 941 MOVD	XMM3, E40139<u>41</u>	660FFAC1	PSUBD	XMMØ XMM1
660F70D7 00 PSHUFD) XMM2, XMM7, 00		PSHUED	0000 0000
660FFEDA PADDD	XMM3, XMM2	660F70C0 00	PSHUED	XMM0 XMM0 00
ØF10DB MOVUPS	XMM3, XMM5	660FDB05 901	PAND	XMM0 [401590]
OFTODE INDODES	Anns, Anns	660FDB35 A01!	PAND	XMM6. [4015A0]
660F7ED8 MOVD	EAX, XMM3			Anno, Labranos
660F70C6 01 PSHUFD	NMMO, XMM6, 01	660FEFF0	PXOR	XMM6, XMM0
CORRECT MOUD	ECV VMMO	660F70C6 01	PSHUFD	XMMØ, XMM6, Ø1
ODUE/ECI HUVD	EUA, ANNO	660F6E1D AC1	MOVD	XMM3, E4013ACI
8908 (MOV	CEAXI, ECX		PSHUED	UMMO UMM7 DI
660E6E1D 9811 MOUD	XMM3. [401398]	660F70D7 01		XMM2, XMM7, 01
ACTACO MOULIDO	VMMO VMMO	660FFEDA	PADDD	XMM3 XMM2
PERCENT PROVES		ØF1ØDB	MOUUPS	XMM3 XMM3
660F7EC1 HOVD 8908 MOU 660F6E1D 9811 MOUD 0F10C3 MOUDPS 660F6E1D 9C11 MOVD 660F76D7 01 PSHUFD 660F76D7 01 PSHUFD	XMM3, E40139C1	660F7ED8	MOUD	EAX, XMM3
660EZ0DZ 01 PSHUED	VXMM2 XMM7 01	660F7ED0	1000	EHO, OUUS
660FFEDA PADDD	XMM3, XMM2	660F7EC1	MOUD	ECX, XMMØ LEAXI, ECX
OCCOPTION HOUSE	SMMS XMMS	8908	MOU	FEAX1. ECX
0F10DB MOVUPS	Anna, Anna	660F70DE 00	PSHUFD	XMM3 XMM6 00
660F7ED8 MOVD	EAX, XMM3		MOVD	
660F7EC1 MOUD	ECX XMM0	660F7ED8	11000	EAX, XMM3
660F7EC1 MOUD 8908 MOU	ECX, XMM0 CEAXI, ECX	660F70DE 00	PSHUED	XMM3, XMM6, DO
8968 1000	LEHAJ, EUA	660F7EDB	MOVD	EBX, XMM3
660F6E1D A01 MOVD	XMM3, [4013H0]	8508	TEST	EOU EDU
660F70D7 01 PSHUF	XMM3, C4013A01 D XMM2, XMM7, 01 XMM3, XMM2		IESI	EAX, EBX 00402470
660FFEDA PADDD	VMMO VMMO	0F84 B501000	JE	00402470
ØF100B MOUUP		660F6E1D B011	MOVD	XMM3. [4013B0]
	S XMM3 XMM3	660F70D7 01	PSHUED	XMM2, XMM7, 01
660F7ED8 MOUD	EAX, XMM3			
8800 MOU	FOX FEOX1	660FFEDA	PADDD	XMM3, XMM2
660F6ED8 MOVD	XMM3. EAX	ØF10DB	MOUUPS	XMM3, XMM3
660F6ED8 1100D	ANNIS, EHA	660F7ED8	MOUD	EAX, XMM3
ØF10C3 MOVUP	S XMM0, XMM3	8800	MOUL	EAX, CEAXI
660F70C0 00 PSHUF	D XMM0, XMM0, 00		MOU MOVD	CHO, LEHOJ
660FDB05 5011 PAND	XMM0 [401550]	660F6ED8	MUVD	XMM3, EAX
COSCIDENCE COLL DOND	01110, L4015501	0F10C3	MOUUPS	XMMØ XMM3
660FDB35 601 PAND	XMM6, [401560] XMM6, XMM0 XMM3, [4013A4]	660F70C0 00	PSHUFD	XMMÖ, XMMÖ, DO
660FEFF0 PXOR	XMM6. XMM0	000071000 00	POHOPD.	
660F6E1D A411 MOUD	XMM3 F4013041	660FD805 801	PHNU	XMM0, [4015B0]
660F70D7 01 PSHUF	TO UMMO UMM7 D4	660FDB05 B01 660FDB35 C01	PAND	XMM6 [4015C0]
		660FEFF0	PXOR	XMM6 XMM0
660FFEDA PADDD	I XMM3 XMM2	22052510 DA11	MOUD	
	and the second se			

3 lines of C 20 lines of x86 200 lines of SSE

(日)、

Jurriaan Bremer @skier_t

Binary Obfuscation using SSE

E ► < E ► E ∽ Q @ May 24, 2012 19 / 31

Brief Introduction	SSEXY	SSEXY Internals	Obstacles	Improvements	Last notes
00		0	0000	0	
0	0	0		0	
0	0	0000		000	
00					
0					

Shut up and take my Binary! (4)

Demonstration

What I will demonstrate:

Start ssexified listening daemon

Send a few invalid and valid packets to the daemon (netcat)

Show that only the valid ones are executed

▲□▶ ▲圖▶ ▲臣▶ ▲臣▶ ―臣 – のへで

Brief Introduction	SSEXY	SSEXY Internals	Obstacles	Improvements	Last notes
00		0	0000	0	
0	0	0		0	
0	0	0000		000	
00					
0					

x86 supports memory addresses

'effective address', 'scale index' and 'displacement' For example: dword ptr [eax+ebx*4+32] effective address: *eax* scale index: *ebx*, multiplied by *four* displacement: *32* SSEXY has to emulate this

Jurriaan Bremer @skier_t

Binary Obfuscation using SSE

May 24, 2012 21 / 31

Brief Introduction	SSEXY	SSEXY Internals	Obstacles	Improvements	Last notes
00		0	0000	0	
0	0	0		0	
0	0	0000		000	
00					
0					

x86 supports memory addresses 'effective address', 'scale index' and 'displacement' For example: dword ptr [eax+ebx*4+32] effective address: *eax* scale index: *ebx*, multiplied by *four* displacement: *32* SSEXY has to emulate this

Jurriaan Bremer @skier_t

Binary Obfuscation using SSE

May 24, 2012 21 / 31

Brief Introduction	SSEXY	SSEXY Internals	Obstacles	Improvements	Last notes
00		0	0000	0	
0	0	0		0	
0	0	0000		000	
00					
0					

x86 supports memory addresses

'effective address', 'scale index' and 'displacement'

For example: dword ptr [eax+ebx*4+32]

effective address: *eax* scale index: *ebx*, multiplied by *four*

displacement: 32

SSEXY has to emulate this

Brief Introduction	SSEXY	SSEXY Internals	Obstacles	Improvements	Last notes
00		0	0000	0	
0	0	0		0	
0	0	0000		000	
00					
0					

x86 supports memory addresses 'effective address', 'scale index' and 'displacement' For example: dword ptr [eax+ebx*4+32] effective address: *eax* scale index: *ebx*, multiplied by *four* displacement: *32* SSEXY has to emulate this

Binary Obfuscation using SSE

May 24, 2012 21 / 31

Brief Introduction	SSEXY	SSEXY Internals	Obstacles	Improvements	Last notes
00		0	0000	0	
0	0	0		0	
0	0	0000		000	
00					
0					

x86 supports memory addresses 'effective address', 'scale index' and 'displacement' For example: dword ptr [eax+ebx*4+32] effective address: *eax* scale index: *ebx*, multiplied by *four* displacement: *32* SSEXY has to emulate this

Binary Obfuscation using SSE

May 24, 2012 21 / 31

Brief Introduction	SSEXY	SSEXY Internals	Obstacles	Improvements	Last notes
00		0	0000	0	
0	0	0		0	
0	0	0000		000	
00					
0					

x86 supports memory addresses

'effective address', 'scale index' and 'displacement'

For example: dword ptr [eax+ebx*4+32]

effective address: *eax*

scale index: ebx, multiplied by four

displacement: 32

SSEXY has to emulate this

Jurriaan Bremer @skier_t

Binary Obfuscation using SSE

May 24, 2012 21 / 31

Brief Introduction	SSEXY	SSEXY Internals	Obstacles	Improvements	Last notes
00		0	0000	0	
0	0	0		0	
0	0	0000		000	
00					
0					

x86 supports memory addresses 'effective address', 'scale index' and 'displacement' For example: dword ptr [eax+ebx*4+32] effective address: *eax* scale index: *ebx*, multiplied by *four* displacement: *32*

SSEXY has to emulate this

Jurriaan Bremer @skier_t

Binary Obfuscation using SSE

May 24, 2012 21 / 31

Brief Introduction	SSEXY	SSEXY Internals	Obstacles	Improvements	Last notes
00	000	0	0000	0	00
0	0	0		0	
0	0	0000		000	
00					
0					

Remember the 'Practical' slide?

movd xmm3, [m32_0]	;	load the displacement
pshufd xmm2, xmm6, 0	;	load effective address
paddd xmm3, xmm2		
pshufd xmm2, xmm6, 3	;	load the scale index
pslld xmm2, 2	;	multiply by four
paddd xmm3, xmm2		
movd eax, xmm3	;	store address to a gpr
mov ecx, dword [eax]	;	read from address

Memory address is now in *xmm3* and *eax* Using *eax* we can read/write from/to this address

Jurriaan Bremer @skier_t

Binary Obfuscation using SSE

May 24, 2012 22 / 31

Brief Introduction	SSEXY	SSEXY Internals	Obstacles	Improvements	Last notes
00	000	0	0000	0	00
0	0	0		0	
0	0	0000		000	
00					
0					

Remember the 'Practical' slide?

movd xmm3, [m32_0]	;	load the displacement
pshufd xmm2, xmm6, 0	;	load effective address
paddd xmm3, xmm2		
pshufd xmm2, xmm6, 3	;	load the scale index
pslld xmm2, 2	;	multiply by four
paddd xmm3, xmm2		
movd eax, xmm3	;	store address to a gpr
mov ecx, dword [eax]	;	read from address

Memory address is now in xmm3 and eax

Using *eax* we can read/write from/to this address

Jurriaan Bremer @skier_t

Binary Obfuscation using SSE

May 24, 2012 22 / 31

▲□▶ ▲圖▶ ▲臣▶ ▲臣▶ ―臣 – のへで

Brief Introduction	SSEXY	SSEXY Internals	Obstacles	Improvements	Last notes
00	000	0	0000	0	00
0	0	0		0	
0	0	0000		000	
00					
0					

Remember the 'Practical' slide?

movd xmm3, [m32_0]	;	load the displacement
pshufd xmm2, xmm6, 0	;	load effective address
paddd xmm3, xmm2		
pshufd xmm2, xmm6, 3	;	load the scale index
pslld xmm2, 2	;	multiply by four
paddd xmm3, xmm2		
movd eax, xmm3	;	store address to a gpr
mov ecx, dword [eax]	;	read from address

Memory address is now in xmm3 and eax

Using eax we can read/write from/to this address

Jurriaan Bremer @skier_t

Binary Obfuscation using SSE

May 24, 2012 22 / 31

Brief In	troduction	SSEXY	SSEXY Internals	Obstacles	Improvements	Last notes
00			0	0000	0	
0		0	0		0	
0		0	0000		000	
00						
0						

Obstacles - Conditional Jumps

No usable comparison instructions

Use normal x86 instructions instead

Jurriaan Bremer @skier_t

Binary Obfuscation using SSE

May 24, 2012 23 / 31

Brief Introduction	SSEXY	SSEXY Internals	Obstacles	Improvements	Last notes
00		0	0000	0	
0	0	0		0	
0	0	0000		000	
00					
0					

Obstacles - Conditional Jumps

No usable comparison instructions Use normal x86 instructions instead

Jurriaan Bremer @skier_t

Binary Obfuscation using SSE

May 24, 2012 23 / 31

Brief Introduction	SSEXY	SSEXY Internals	Obstacles	Improvements	Last notes
00		0	0000	0	00
0	0	0		0	
0	0	0000		000	
00					
0					

Two types; internal and external

internal: function calls within the application external: function calls to 3rd party libraries (e.g. windows API)

internal: these are generally fine.. :) external: need extra precautions;

stack pointer has to be set

Funky stuff happens, e.g. MessageBoxA resets xmm registers

Jurriaan Bremer @skier_t

Binary Obfuscation using SSE

May 24, 2012 24 / 31

Brief Introduction	SSEXY	SSEXY Internals	Obstacles	Improvements	Last notes
00		0	0000	0	
0	0	0		0	
0	0	0000		000	
00					
0					

Two types; *internal* and *external* internal: function calls within the application external: function calls to 3rd party libraries (e.g. windows API)

internal: these are generally fine.. :)
external: need extra precautions;
stack pointer has to be set
Funky stuff happens, e.g. MessageBoxA resets xmm registers

Jurriaan Bremer @skier_t

Binary Obfuscation using SSE

May 24, 2012 24 / 31

Brief Introduction	SSEXY	SSEXY Internals	Obstacles	Improvements	Last notes
00		0	0000	0	
0	0	0		0	
0	0	0000		000	
00					
0					

```
Two types; internal and external
internal: function calls within the application
external: function calls to 3rd party libraries (e.g. windows
API)
```

internal: these are generally fine.. :)
external: need extra precautions;
stack pointer has to be set
Funky stuff happens, e.g. MessageBoxA resets xmm registers

Jurriaan Bremer @skier_t

Binary Obfuscation using SSE

May 24, 2012 24 / 31

Brief Introduction	SSEXY	SSEXY Internals	Obstacles	Improvements	Last notes
00		0	0000	0	00
0	0	0		0	
0	0	0000		000	
00					
0					

Two types; *internal* and *external* internal: function calls within the application external: function calls to 3rd party libraries (e.g. windows API)

internal: these are generally fine.. :)
external: need extra precautions;
stack pointer has to be set
Funky stuff happens, e.g. MessageBoxA resets xmm registers

Jurriaan Bremer @skier_t

Binary Obfuscation using SSE

May 24, 2012 24 / 31

Brief Introduction	SSEXY	SSEXY Internals	Obstacles	Improvements	Last notes
00		0	0000	0	00
0	0	0		0	
0	0	0000		000	
00					
0					

Two types; *internal* and *external* internal: function calls within the application external: function calls to 3rd party libraries (e.g. windows API)

internal: these are generally fine.. :)
external: need extra precautions;
stack pointer has to be set
Funky stuff happens, e.g. MessageBoxA resets xmm registers

Jurriaan Bremer @skier_t

Binary Obfuscation using SSE

May 24, 2012 24 / 31

Brief Introduction	SSEXY	SSEXY Internals	Obstacles	Improvements	Last notes
00		0	0000	0	00
0	0	0		0	
0	0	0000		000	
00					
0					

Two types; *internal* and *external* internal: function calls within the application external: function calls to 3rd party libraries (e.g. windows API)

internal: these are generally fine.. :)
external: need extra precautions;
stack pointer has to be set
Funky stuff happens, e.g. MessageBoxA resets xmm registers

Jurriaan Bremer @skier_t

Binary Obfuscation using SSE

May 24, 2012 24 / 31

Brief Introduction	SSEXY	SSEXY Internals	Obstacles	Improvements	Last notes
00		0	0000	0	00
0	0	0		0	
0	0	0000		000	
00					
0					

```
Two types; internal and external
internal: function calls within the application
external: function calls to 3rd party libraries (e.g. windows
API)
```

internal: these are generally fine.. :)

external: need extra precautions;

stack pointer has to be set

Funky stuff happens, e.g. MessageBoxA resets xmm registers

Jurriaan Bremer @skier_t

Binary Obfuscation using SSE

May 24, 2012 24 / 31

Brief Introduction	SSEXY	SSEXY Internals	Obstacles	Improvements	Last notes
00		0	0000	•	00
0	0	0		0	
0	0	0000		000	
00					
0					

Shuffling stored GPRs

GPRs	eax	ecx	edx	ebx
Before	a0 a1 a2 a3	c0 c1 c2 c3	d0 d1 d2 d3	b0 b1 b2 b3
After	a0 c0 d0 b0	a1 c1 d1 b1	a2 c2 d2 b2	a3 c3 <mark>d3 b3</mark>

The values of GPRs are much harder to read

Jurriaan Bremer @skier_t

Binary Obfuscation using SSE

May 24, 2012 25 / 31

◆□▶ ◆□▶ ◆臣▶ ◆臣▶ □臣 = のへで

Brief Introduction	SSEXY	SSEXY Internals	Obstacles	Improvements	Last notes
00		0	0000	•	
0	0	0		0	
0	0	0000		000	
00					
0					

Shuffling stored GPRs

GPRs	eax	ecx	edx	ebx
Before	a0 a1 a2 a3	c0 c1 c2 c3	d0 d1 d2 d3	b0 b1 b2 b3
After	a0 c0 d0 b0	a1 c1 d1 b1	a2 c2 d2 b2	a3 c3 d3 b3

The values of GPRs are much harder to read Every read/write operation to GPRs need (de-)shuffling code

Jurriaan Bremer @skier_t

Binary Obfuscation using SSE

May 24, 2012 25 / 31

Brief Introduction	SSEXY	SSEXY Internals	Obstacles	Improvements	Last notes
00		0	0000	•	00
0	0	0		0	
0	0	0000		000	
00					
0					

Shuffling stored GPRs

GPRs	eax	ecx	edx	ebx
Before	a0 a1 a2 a3	c0 c1 c2 c3	d0 d1 d2 d3	b0 b1 b2 b3
After	a0 c0 d0 b0	a1 c1 d1 b1	a2 c2 d2 b2	a3 <mark>c3 d3 b3</mark>

The values of GPRs are much harder to read Every read/write operation to GPRs need (de-)shuffling code Possibly use different encodings per function (Needs translation when calling another function)

Jurriaan Bremer @skier_t

Binary Obfuscation using SSE

May 24, 2012 25 / 31

Brief Introduction	SSEXY	SSEXY Internals	Obstacles	Improvements	Last notes
00		0	0000	0	00
0	0	0		•	
0	0	0000		000	
00					
0					

Encrypting stored GPRs

Each GPR is encrypted, e.g. using a unique xor key

Jurriaan Bremer @skier_t

Binary Obfuscation using SSE

May 24, 2012 26 / 31

▲□▶ ▲□▶ ▲□▶ ▲□▶ □ ○ ○ ○

Brief Introduction	SSEXY	SSEXY Internals	Obstacles	Improvements	Last notes
00		0	0000	0	00
0	0	0		•	
0	0	0000		000	
00					
0					

Encrypting stored GPRs

Each GPR is encrypted, e.g. using a unique xor key Every read/write to GPRs need decryption/encryption code

Jurriaan Bremer @skier_t

Binary Obfuscation using SSE

May 24, 2012 26 / 31

Brief Introduction	SSEXY	SSEXY Internals	Obstacles	Improvements	Last notes
00		0	0000	0	00
0	0	0		•	
0	0	0000		000	
00					
0					

Encrypting stored GPRs

Each GPR is encrypted, e.g. using a unique xor key Every read/write to GPRs need decryption/encryption code Again, can be function-specific

Jurriaan Bremer @skier_t

Binary Obfuscation using SSE

May 24, 2012 26 / 31

Brief Introduction	SSEXY	SSEXY Internals	Obstacles	Improvements	Last notes
00		0	0000	0	
0	0	0		0	
0	0	0000		000	
00					
0					

Did you notice the *mov* instruction earlier? (in the Obstacles - Memory Addresses slide)

mov is an x86 general purpose instruction SSEXY doesn't like those

Jurriaan Bremer @skier_t

Binary Obfuscation using SSE

May 24, 2012 27 / 31

Brief Introduction	SSEXY	SSEXY Internals	Obstacles	Improvements	Last notes
00		0	0000	0	00
0	0	0		0	
0	0	0000		000	
00					
0					

Did you notice the *mov* instruction earlier? (in the Obstacles -Memory Addresses slide) *mov* is an x86 general purpose instruction SSEXY doesn't like those

Jurriaan Bremer @skier_t

Binary Obfuscation using SSE

May 24, 2012 27 / 31

Brief Introduction	SSEXY	SSEXY Internals	Obstacles	Improvements	Last notes
00		0	0000	0	00
0	0	0		0	
0	0	0000		000	
00					
0					

Did you notice the *mov* instruction earlier? (in the Obstacles -Memory Addresses slide) *mov* is an x86 general purpose instruction SSEXY doesn't like those

Jurriaan Bremer @skier_t

Binary Obfuscation using SSE

May 24, 2012 27 / 31

Brief Introduction	SSEXY	SSEXY Internals	Obstacles	Improvements	Last notes
00		0	0000	0	
0	0	0		0	
0	0	0000		000	
00					
0					

To refresh your memory

movd xmm3, [m32_0]	;	load the displacement
pshufd xmm2, xmm6, 0	;	load effective address
paddd xmm3, xmm2		
pshufd xmm2, xmm6, 3	;	load the scale index
pslld xmm2, 2	;	multiply by four
paddd xmm3, xmm2		
movd eax, xmm3	;	store address to a gpr
mov ecx, dword [eax]	;	read from address

Jurriaan Bremer @skier_t

Binary Obfuscation using SSE

イロト 不同ト 不同ト 不同ト

Brief Introduction	SSEXY	SSEXY Internals	Obstacles	Improvements	Last notes
00		0	0000	0	00
0	0	0		0	
0	0	0000		000	
00					
0					

movd eax, xmm3	; still refreshing
mov ecx, dword [eax]	; your memory ;)

movd takes the lowest 32bits of *xmm3* and stores them into *eax* Now if we rewrite the *mov* instruction

Jurriaan Bremer @skier_t

Binary Obfuscation using SSE

E ► 4 E ► E ∽ Q C May 24, 2012 29 / 31

(日)、

Brief Introduction	SSEXY	SSEXY Internals	Obstacles	Improvements	Last notes
00		0	0000	0	00
0	0	0		0	
0	0	0000		000	
00					
0					

movd eax, xmm3	; still refreshing
mov ecx, dword [eax]	; your memory ;)

movd takes the lowest 32bits of *xmm3* and stores them into *eax* Now if we rewrite the *mov* instruction

movd dword [next_instr+4], xmm3
next_instr:
movd xmm2, dword [0x11223344]

Jurriaan Bremer @skier_t

Binary Obfuscation using SSE

May 24, 2012 29 / 31

= 900

(日)、

Brief Introduction	SSEXY	SSEXY Internals	Obstacles	Improvements	Last notes
00		0	0000	0	00
0	0	0		0	
0	0	0000		000	
00					
0					

movd eax, xmm3	; still refreshing
mov ecx, dword [eax]	; your memory ;)

movd takes the lowest 32bits of *xmm3* and stores them into *eax* Now if we rewrite the *mov* instruction

movd	dword	[next	[instr+4],	xmm3
next	_instr			
movd	xmm2,	dword	[0×1122334	14]

Single-threaded only (we overwrite machine code)

Jurriaan Bremer @skier_t

Binary Obfuscation using SSE

■ ▶ < ■ ▶ ■ つへの May 24, 2012 29 / 31

(日)、

Brief Introduction	SSEXY	SSEXY Internals	Obstacles	Improvements	Last notes
00		0	0000	0	00
0	0	0		0	
0	0	0000		000	
00					
0					

movd eax, xmm3	; still refreshing
mov ecx, dword [eax]	; your memory ;)

movd takes the lowest 32bits of *xmm3* and stores them into *eax* Now if we rewrite the *mov* instruction

```
movd dword [next_instr+4], xmm3
```

```
next_instr:
```

```
movd xmm2, dword [0x11223344]
```

Single-threaded only (we overwrite machine code) Doesn't matter, had SSE.

Jurriaan Bremer @skier_t

Binary Obfuscation using SSE

May 24, 2012 29 / 31

Brief Introduction 00 0	SSEXY 000 0 0	SSEXY Internals 0 0000	Obstacles 0000	Improvements O O OOO	Last notes ●○
00					

Performance

Running the hash() algorithm a few million times, showed me a performance decrease of 5 times. Sounds reasonable, since it takes a Reverse Engineer probably five times longer to analyze the binary..;)

Binary Obfuscation using SSE

Brief Introduction	SSEXY	SSEXY Internals	Obstacles	Improvements	Last notes
00	000	0	0000	0	0.
0	0	0		0	
0	0	0000		000	
00					
0					

Source?-s

http://jbremer.org/ http://github.com/jbremer/ssexy jurriaanbremer@gmail.com

Jurriaan Bremer @skier_t

Binary Obfuscation using SSE

May 24, 2012 31 / 31

◆□▶ ◆□▶ ◆臣▶ ◆臣▶ □臣 = のへで