

You can't stop us: latest trends on exploit techniques

CONFidence 2010, Krakow

Alexey Sintsov
Security Researcher
Digital Security

Russia?

 **• WANTED •** 

by
THE NEW YORK COUNTY DISTRICT ATTORNEY'S OFFICE,
THE UNITED STATES SECRET SERVICE,
and internationally by **INTERPOL**,

FOR CYBERCRIMES
INCLUDING MONEY LAUNDERING, SCHEME TO DEFRAUD, AND CONSPIRACY,
IN VIOLATION OF NEW YORK STATE LAW

OLEG COVELIN



\\ Actually they are **non-russian** citizen

#whoami

Digital Security:

- Audit/Pentest (ISO/PCI/PA–DSS and blah-blah-blah)
- ERP Assesment/Pentest
- Software development

XAKEP magazine:

- Leading “Exploit-Review” column
- Writing articles about exploit dev.

DSecRG – white hats:

- Finding vulnerabilities in customers software and systems
- Finding ways to exploit them all
- Giving report to the vendor and making the world more secure

RDBMS
ERP systems

WEB Applications
Internet-Bank systems

Attacks

BUG	IMPACT	SHELLCODE
BoF in stack	<ul style="list-style-type: none">• RET• SEH• CALL/JMP REG	<ul style="list-style-type: none">• Stack• Heap
BoF in heap	<ul style="list-style-type: none">• Flink	<ul style="list-style-type: none">• Heap
Format string bug	<ul style="list-style-type: none">• RET• SEH	<ul style="list-style-type: none">• Stack• Heap
Memory bugs (for example Use After Free)	<ul style="list-style-type: none">• Bad pointer	<ul style="list-style-type: none">• Heap

Mitigations

BUG	IMPACT	SHELLCODE
BoF in stack	<ul style="list-style-type: none"> • RET • SEH • CALL/JMP REG 	<ul style="list-style-type: none"> • Stack • Heap
BoF in heap	<ul style="list-style-type: none"> • RET 	<ul style="list-style-type: none"> • Heap
Format string bug	<ul style="list-style-type: none"> • RET • SEH 	<ul style="list-style-type: none"> • Stack • Heap
Memory bugs (for example Use After Free)	<ul style="list-style-type: none"> • Bad pointer 	<ul style="list-style-type: none"> • Heap

- **Stack cookies**
- **Save unlinking**
- **Heap cookies**

Mitigations

BUG	IMPACT	SHELLCODE
BoF in stack	<ul style="list-style-type: none"> RET SEH CALL/JMP REG 	<ul style="list-style-type: none"> Stack Heap
BoF in heap	<ul style="list-style-type: none"> RET SEH 	<ul style="list-style-type: none"> Heap
Format string bug	<ul style="list-style-type: none"> RET SEH 	<ul style="list-style-type: none"> Stack Heap
Memory bugs (for example Use After Free)	<ul style="list-style-type: none"> Bad pointer 	<ul style="list-style-type: none"> Heap

- Stack cookies
- Save unlinking
- SEH handler validation
- Heap cookies
- SEH chain validation

Mitigations

BUG	IMPACT	SHELLCODE
BoF in stack	<ul style="list-style-type: none"> • RET • SEH • CALL/JMP REG 	<ul style="list-style-type: none"> • Stack • Heap
BoF in heap	<ul style="list-style-type: none"> • RET • SEH 	<ul style="list-style-type: none"> • Heap
Format string bug	<ul style="list-style-type: none"> • RET • SEH 	<ul style="list-style-type: none"> • Stack • Heap
Memory bugs (for example Use After Free)	<ul style="list-style-type: none"> • Bad pointer 	<ul style="list-style-type: none"> • Heap

- Stack cookies
- Save unlinking
- SEH handler validation
- DEP
- ASLR
- Heap cookies
- SEH chain validation

Mitigations

BUG	IMPACT	SHELLCODE
BoF in stack	<ul style="list-style-type: none"> • RET • SEH • CALL/JMP REG 	<ul style="list-style-type: none"> • Stack • Heap
BoF in heap	<ul style="list-style-type: none"> • Flink 	<ul style="list-style-type: none"> • Heap
Format string bug	<ul style="list-style-type: none"> • RET • SEH 	<ul style="list-style-type: none"> • Stack • Heap
Memory bugs (for example Use After Free)	<ul style="list-style-type: none"> • Bad pointer 	<ul style="list-style-type: none"> • Heap

[1] use SEH
[2] [3] many ways
[4] [5] use none safeSEH fake SEH handler
• DEP
• ASLR

DEP bypass – retn2libc

Code reuse - disable DEP code:

- NtSetInformationProcess ^[6]
- SetProcessDEPPolicy ^[7]

Or just make memory executable:

- VirtualAlloc and memcpy ^[8]
- VirtualProtect

Or copy shellcode to .code section:

- WriteProcessMemory ^[9]

DEP bypass – ret2libc

Code reuse - disable DEP code:

- ~~NtSetInformationProcess~~
- ~~SetProcessDEPPolicy~~

Or just disable DEP:

- VirtualMemoryControlSpace
- VirtualMemoryControlSpace

Or copy code to writable memory:

- WriteProcessMemory

**Permanent DEP / AlwaysOn
ASLR**

ASLR bypass

- Use non-ASLR modules
 - static base address – call functions from modules with unknown address
- Address disclosure
 - @WTFuzz on pwn2own bypass ASLR with two bugs in IE8 ^[10]
- Address bruteforce
 - PHP 6.0 DEV Exploit brute VirtualProtect address 0xFFFFSSSS
XXXX – value for bruteforce, SSSS – static offset from base addr ^[11]
- Heap spraying
 - Browser – javascript ^[12]
java heap spray
.NET

DEP bypass – ret2libc

Code reuse - disable DEP code:

- ~~NtSetInformationProcess~~
- ~~SetProcessDEPPolicy~~

Or just make memory executable:

- VirtualAlloc and memcpy
- VirtualProtect

Or copy shellcode to .code section:

- WriteProcessMemory

But more problems:

- Need to calc parameters for these functions
- Avoid null bytes or non-ASCII bytes

Permanent DEP / AlwaysOn

DEP
bypassed

[13]

Return-Oriented Programming

- We can change stack frame
- We can calc and save values

CODE

```
POP EDI
MOV EAX, 0x10
MOV [EDI], EAX
```

```
MOV EAX, 0x10
```

CPU

```
0x04001111: CALL EDI
0x7C010101: XCHG EAX, ESP
0x7C010102: RETN
0x8C010103: POP EDI
0x8C010104: RETN
0x8C020105: POP EAX
0x8C020106: RETN
0x8C030107: NEG EAX
0x8C030108: RETN
0x8C040109: MOV [EDI], EAX
0x8C05010B: RETN
```

Vuln.

STACK

```
0xFFFFFFFF
0xFFFFFFFF
0xFFFFFFFF
```

```
0x0D0D0D0D
0x8C010103
0x0A0A0A0A
0x8C020104
0xFFFFFFFF0
0x8C030105
0x8C040106
```

REGS

```
EAX=0x0D0D0D0D
EDI=0x7C010101
```

R
O
P

ROP example

ProSSHD exploit – does not work against Win7^[14]
Add ROP for ASLR/DEP bypass:

- Non-ASLR DLL's – find a call of VirtualProtect
- Prepare place for VP parameters
- Calc and save parameters via ROP (no NULL bytes in addr)
- Call VP from step 1 – make stack executable
- Give control to shellcode from stack

Freeware plugin for ImmunityDebugger – **pvefindaddr**^[15] can help:

- !pvefindaddr nonaslr
- !pvefindaddr rop <module name>

EXPLOIT DEMO

ProSSHD is under attack

Clients are under attack

Software:

- Browsers
 - Plugins/ActiveX
 - Bank-Client
 - ERP/Business
- And more:
 - MS Office
 - Adobe Acrobat Reader
 - Adobe Flash

Format

- HTML/JS
- SWF
- PDF
- DOC*

Exploit



* More features by third party software

JIT-SPRAY

Instruction code injection via **ActionScript**

[16]

```
var ret=(0x3C909090^0x3C909090^0x3C909090^0x3C909090);
```



```
0x1A1A0100: B89090903C MOV EAX, 3C909090  
0x1A1A0105: 359090903C XOR EAX, 3C909090  
0x1A1A010A: 359090903C XOR EAX, 3C909090  
0x1A1A010F: 359090903C XOR EAX, 3C909090
```



Executable

JIT SPRAY: DEP bypass

0x1A1A0100: B89090903C MOV EAX, 3C909090

0x1A1A0105: 359090903C XOR EAX, 3C909090

0x1A1A010A: 359090903C XOR EAX, 3C909090

0x1A1A010F: 359090903C XOR EAX, 3C909090



+ 0x01 to address

0x1A1A0101: 90

NOP

0x1A1A0102: 90

NOP

0x1A1A0103: 90

NOP

0x1A1A0104: 3C35 CMP AL, 35

0x1A1A0106: 90

NOP

0x1A1A0107: 90

NOP

0x1A1A0108: 90

NOP

0x1A1A0109: 3C35 CMP AL, 35

As I said – executable

JIT Shellcode

Size

- 0xXXYY0000 – base address of every block with JITed shellcode ^[17]
- Intro Flash code – from beginning with the size of ~ 0xD3
- Offset between blocks 0x00010000 (If block size less than 0x1000)
- We can guess address of block: 0xXXYY0101

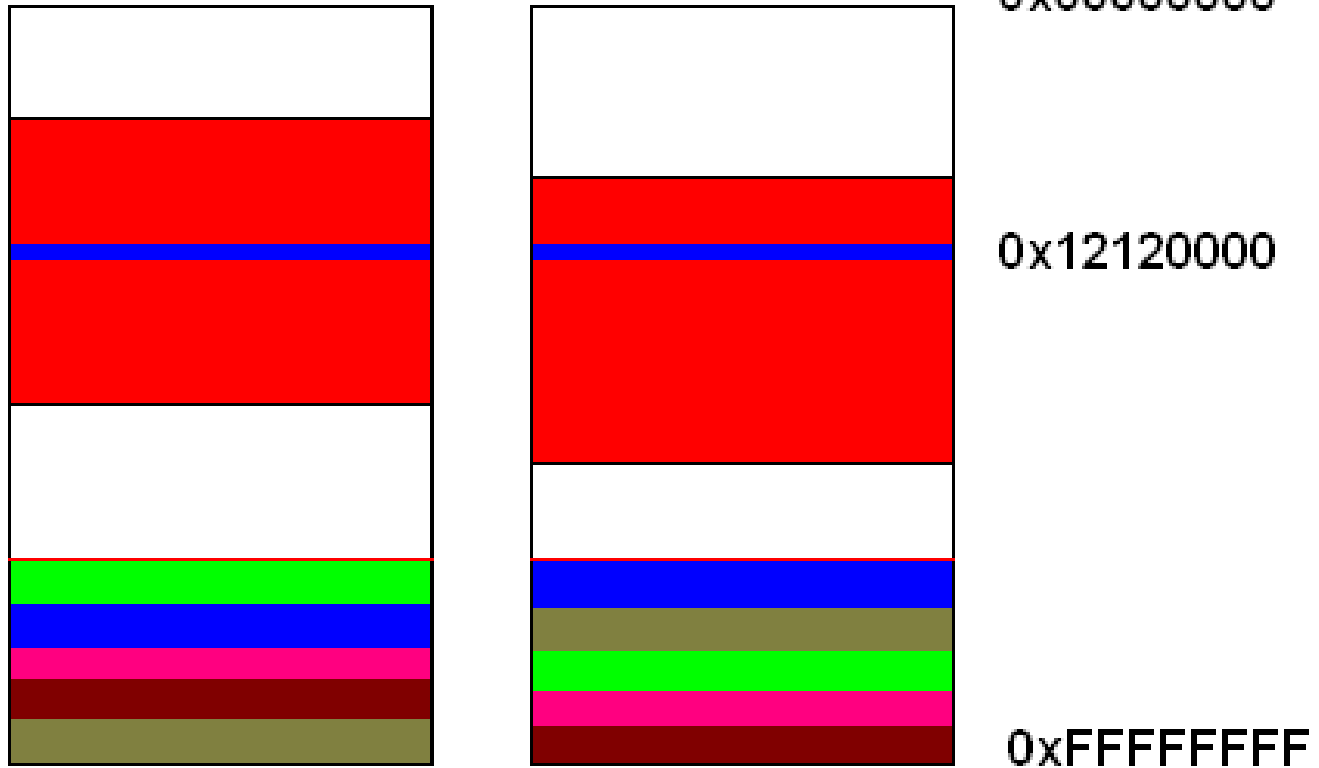
Is this enough for ASLR bypass ?

JIT spray beats ASLR+DEP

Memory map

Address	Size	Owner	Section	Contains	Type	Access
12630000	00002000	JIT SPRAY + ASLR			Priv	R E
12634000	00001000		Priv	RW		
12640000	00002000		Priv	R E		
12644000	00001000		Priv	RW		
12650000	00002000		Priv	R E		
12654000	00001000		Priv	RW		
12660000	00002000		Priv	R E		
12664000	00001000		Priv	RW		
12670000	00002000		Priv	R E		
12674000	00001000		Priv	RW		
12680000	00002000		Priv	R E		
12684000	00001000		Priv	RW		
12690000	00002000		Priv	R E		
12694000	00001000		Priv	RW		
126A0000	00002000		Priv	R E		
126A4000	00001000		Priv	RW		
126B0000	00002000		Priv	R E		
126B4000	00001000		Priv	RW		
126C0000	00002000		Priv	R E		
126C4000	00001000		Priv	RW		
126D0000	00002000		Priv	R E		
126D4000	00001000		Priv	RW		
126E0000	00002000		Priv	R E		
126E4000	00001000		Priv	RW		
126F0000	00002000	Priv	R E			
126F4000	00001000	Priv	RW			
12700000	00002000	Priv	R E			
12704000	00001000	Priv	RW			
12710000	00002000	Priv	R E			
12714000	00001000	Priv	RW			
12720000	00002000	Priv	R E			
12724000	00001000	Priv	RW			
12730000	00002000	Priv	R E			
12734000	00001000	Priv	RW			

JIT spray beats ASLR+DEP



Guess address with ASLR

```
Dump - 091F0000..091F1FFF
091F0000 81 FC 08 00 EC 01 0F 82 95 00 00 00 55 8B EC 81
091F0010 EC 10 00 00 00 8B 45 10 8B 00 8B 00 D8 A0 18 04
091F0020 85 C9 0F 85 5B 00 00 00 8B 4D 08 89 0D 50 A3 18
091F0030 04 8B 11 8B 4A 14 8D 49 04 89 0D 04 A0 18 04 8B
091F0040 4A 1C 00 10 10 00 4E 5F 05 5F 89 5D F8 8B D8
091F0050 52 6A 00 00 00 00 00 00 00 00 00 8B 43 08 8B
091F0060 88 88 00 00 00 00 00 00 00 00 00 50 6A 00 51 0F
091F0070 77 FF 51 0C 83 C4 0C 0F 77 B8 04 00 00 00 E9 0A
091F0080 00 00 00 8B 4D 08 0F 77 E8 B3 E3 C5 FA 8B 5D F8
091F0090 C9 C3 FF 74 24 08 B9 00 A0 18 04 E8 E0 75 C7 FA
091F00A0 C3 E8 EC FF FF FF E9 61 FF FF FF 00 00 00 00
091F00B0 81 FC 00 00 EC 01 0F 82 66 03 00 00 55 8B EC 81
091F00C0 EC 00 00 00 00 8B 05 D8 A0 18 04 85 D0 0F 85 43
091F00D0 03 00 00 B8 90 90 90 3C 35 90 90 90 3C 35 90 90
091F00E0 90 3C 35 90 90 90 3C 35 90 90 90 3C 35 90 90
091F00F0 3C 35 90 90 90 3C 35 90 90 90 3C 35 90 90 3C
091F0100 35 90 90 90 3C 35 90 90 90 3C 35 90 90 3C 35
091F0110 90 90 90 3C 35 90 90 90 3C 35 90 90 90 3C 35
091F0120 90 90 3C 35 90 90 90 3C 35 90 90 90 3C 35 90
091F0130 90 90 3C 35 90 90 90 3C 35 90 90 90 3C 35 90
091F0140 3C 35 90 90 90 3C 35 90 90 90 3C 35 90 90 3C
091F0150 35 90 90 90 90 3C 35 90 90 90 3C 35 90 90 3C
091F0160 90 90 90 3C 35 90 90 90 3C 35 90 90 90 3C 35 90
091F0170 90 90 3C 35 90 90 90 3C 35 90 90 90 3C 35 90 90
091F0180 90 3C 35 90 90 90 3C 35 90 90 90 3C 35 90 90 90
091F0190 3C 35 90 90 90 3C 35 90 90 90 3C 35 90 90 3C
091F01A0 35 90 90 90 3C 35 90 90 90 3C 35 90 90 3C 35
091F01B0 90 90 90 3C 35 90 90 90 3C 35 90 90 90 3C 35 31
091F01C0 D2 58 3C 35 80 CA FF 3C 35 80 CE 0F 3C 35 90 90
091F01D0 42 3C 35 52 6A 43 3C 35 58 CD 2E 3C 35 3C 05 90
091F01E0 6A 35 5A 5A 90 6A 35 74 D8 90 3C 35 59 59 B8 31
091F01F0 35 33 07 90 3C 35 8B FA AF 6A 35 75 D1 AF 6A 35
091F0200 90 59 59 6A 35 75 8F 57 33 35 83 EC 44 3C 35 33
091F0210 C0 90 3C 35 B0 30 90 3C 35 64 8B 00 3C 35 8B 40
091F0220 0C 3C 35 90 90 90 00 08 3C 35 8B 78 20
091F0230 3C 35 90 90 90 3C 35 90 90 90 3C 35 75 EA 90 3C
091F0240 35 47 47 90 3C 35 80 3F 65 6A 35 75 EF 90 3C 35
091F0250 47 47 90 3C 35 80 3F 72 6A 35 75 EF 90 3C 35 47
091F0260 47 90 3C 35 80 3F 6E 6A 35 75 EF 90 3C 35 90 90
091F0270 52 3C 35 83 C2 3C 3C 35 8B 3A 90 3C 35 8B 14 24
091F0280 3C 35 03 D7 90 3C 35 83 C2 78 3C 35 8B 3A 90 3C
091F0290 35 8B 14 24 3C 35 03 D7 90 3C 35 83 C2 18 3C 35
091F02A0 8B 30 90 3C 35 83 C2 04 3C 35 8B 1A 90 3C 35 03
```

FLASH INTRO CODE

JIT NOP SLICE

JIT SHELL CODE

JIT payload

Egg-Hunter – the best decision

- Metasploit shellcode in Flash string or heap (with tag)
- JIT sellcode will try to find tag
- When found: call VirtualProtect, and JMP.

What we get:

- Universal (can used in BoF, memory corruptions)
- Safe (no 15 sec Flash time-out)
- **Faster** (we can find tag with known offset)

JIT EXPLOIT example

Safari memory corruption ^[18] – does not work against Win7

Remove HEAP-SPRAY

Add JIT-SPRAY for ASLR/DEP bypass

- JIT SPRAY
- Pop-up, pop-up...
- parent.close() – exact address
 - Guess address – beat **ASLR**
- JIT-shellcode – beat **DEP**

EXPLOIT DEMO

Safari is under attack

U can't stop THEM...

1. <http://www.ngssoftware.com/papers/defeating-w2k3-stack-protection.pdf> (Defeating the Stack Based Buffer Overflow Prevention Mechanism of Microsoft Windows 2003 Server by **David Litchfield**)
2. <http://cybertech.net/~sh0ksh0k/projects/winheap/XPSP2%20Heap%20Exploitation.ppt> (Windows Heap Exploitation by **Matt Conover**)
3. <http://www.orkspace.net/secdocs/Windows/Protection/Bypass/Exploiting%20Freelist%5B0%5D%20On%20XP%20Service%20Pack%202.pdf> (Exploiting Freelist[0] On XP Service Pack 2 by **Brett Moore**)
4. <http://www.uninformed.org/?v=5&a=2&t=txt> (Preventing the Exploitation of SEH Overwrites by **Matt Miller**)
5. http://www.sysdream.com/articles/sehop_en.pdf (Bypassing SEHOP by **Stefan Le Berre Damien Cauquil**)
6. <http://www.uninformed.org/?v=2&a=4&t=pdf> (Bypassing hardware-enforced DEP by **Matt Miller** and **Ken Johnson**)
7. <http://bernardodamele.blogspot.com/2009/12/dep-bypass-with-setprocessdeppolicy.html> (blog by **Bernardo Damele A. G.**)
8. <http://woct-blog.blogspot.com/2005/01/dep-evasion-technique.html> (blog by **John**)
9. <http://www.exploit-db.com/papers/11988> (Exploitation With WriteProcessMemory()/Yet Another DEP Trick by **Spencer Pratt**)
10. <http://vreugdenhilresearch.nl/Pwn2Own-2010-Windows7-InternetExplorer8.pdf> (Pwn2Own 2010 Windows 7 IE 8 exploit by **Peter Vreugdenhil**)
11. <http://www.exploit-db.com/exploits/12189> (PHP 6.0 DEV exploit with NX+ASLR bypass by **Matteo Memelli**)
12. <http://taossa.com/archive/bh08sotirovdowd.pdf> (Bypassing Browser memory Protections by **Alexander Sotirov** and **Mark Dowd**)
13. <http://cseweb.ucsd.edu/~hovav/dist/blackhat08.pdf> (Return oriented programming Exploitation without Code Injection by **Erik Buchanan, Ryan Roemer, Stefan Savage and Hovav Shacham**)
14. <http://www.exploit-db.com/exploits/11618> (ProSSHD v1.2 20090726 Buffer Overflow Exploit by **S2 Crew**)
15. <http://www.corelan.be:8800/index.php/security/pvefindaddr-py-immunity-debugger-pycommand> (pvefindaddr.py ImmDbg Plugin by **Peter Van Eeckhoutte**)
16. <http://www.semanticscope.com/research/BHDC2010/BHDC-2010-Paper.pdf> (Interpreter Exploitation : Pointer Inference and JIT Spraying by **Dionysus Blazakis**)
17. <http://dsecrg.com/pages/pub/show.php?id=22> (Writing JIT-Spray Shellcode for fun and profit by **me**)
18. <http://www.exploit-db.com/exploits/12573> (Safari parent.close() exploit by **Krystian Kloskowski**)



Questions?

www.dsecrg.com

www.twitter.com/asintsov
a.sintsov@dsec.ru