Digital Security

ConfidencE 2010

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

CONFidence 2010, Krakow

Alexey Sintsov

Security Researcher

Digital Security

Russia?

\\ 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	**WEB Applications**
ERP systems	**Internet-Bank systems**

Attacks

BUG	IMPACT	SHELLCODE
BoF in stack	• RET • SEH • CALL/JMP REG	• Stack • Heap
BoF in heap	• Flink	• Heap
Format string bug	• RET • SEH	• Stack • Heap
Memory bugs (for example Use After Free)	• Bad pointer	• Heap

Mitigations

BUG	IMPACT	SHELLCODE
BoF in stack	~~• RET~~ • SEH • CALL/JMP REG	• Stack • Heap
BoF in heap	~~• Slink~~	• Heap
Format string bug	• RET • SEH	• Stack • Heap
Memory bugs (for example Use After Free)	• Bad pointer	• Heap

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

Mitigations

BUG	IMPACT	SHELLCODE
BoF in stack	~~• RET~~ ~~• SEH~~ • CALL/JMP REG	• Stack • Heap
BoF in heap	~~• Flink~~	• Heap
Format string bug	• RET ~~• SEH~~	• Stack • Heap
Memory bugs (for example Use After Free)	• Bad pointer	• Heap

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

Mitigations

BUG	IMPACT	SHELLCODE
BoF in stack	• RET • SEH • CALL/JMP REG	• Stack • Heap
BoF in heap	• Flink	• Heap
Format string bug	• RET • SEH	• Stack • Heap
Memory bugs (for example Use After Free)	• Bad pointer	• Heap

• **Stack cookies** • **Save unlinking** • **Heap cookies** • **SEH handler validation** • **SEH chain validation** • **DEP** • **ASLR**

Mitigations

BUG	IMPACT	SHELLCODE
BoF in stack	• RET • SEH • CALL/JMP REG	• Stack • Heap
BoF in heap	• Flink	• Heap
Format string bug	• RET • SEH	• Stack • Heap
Memory bugs (for example Use After Free)	• Bad pointer	• Heap

use SEH [1]

many ways [2] [3]

use none safeSEH [4] [5]

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 – retn2libc

Code reuse - disable DEP code:

- NtSetinformationProcess
- SetProcessDEPPolicy

Or just disable:

- VirtualProtect
- VirtualProtectEx

Or copy code to executable section:

- 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 0xXXXXSSSS [11]

  XXXX – value for bruteforce, SSSS – static offset from base addr
- Heap spraying
  - Browser – javascript [12]

    java heap spray

    .NET

DEP bypass – retn2libc

Code reuse - disable DEP code:

- NtSetInformationProcess
- SetProcessDEPPolicy

Or just make memory executable:

- VirtualAlloc and memcpy
- VirtualProtect

Or copy shellcode to .code section:

- WriteProcessMemory

Permanent DEP / AlwaysOn

bypassed

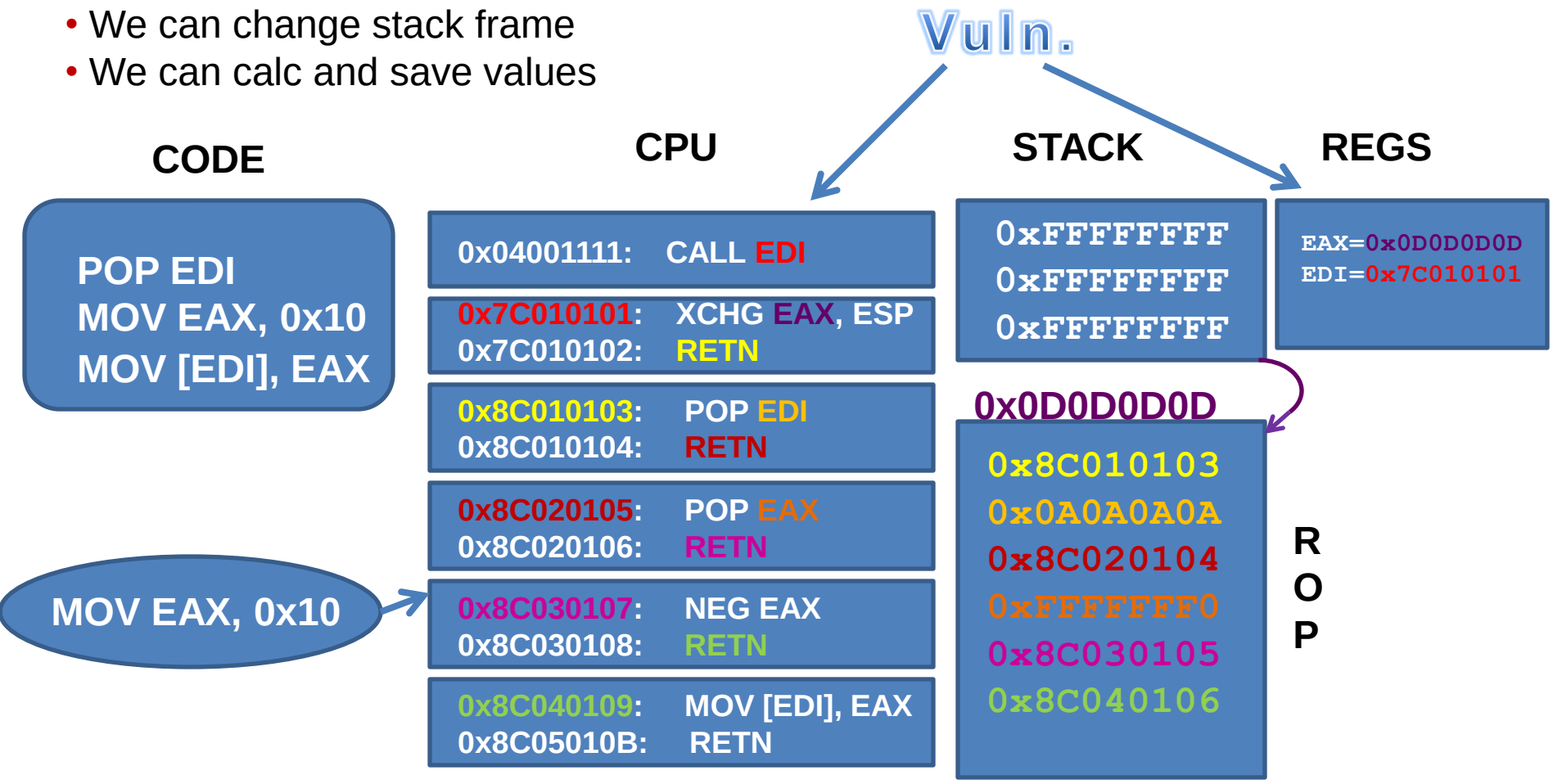
But more problems:

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

Return-Oriented Programming

[13]

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



ROP example

ProSSHD exploit[14] **–** does not work against Win7
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>

Digital Security

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: B8**90**90903C MOV EAX, 3C**90**9090

0x1A1A0105: **35**9090903C **XOR EAX**, 3C909090

0x1A1A010A: **35**9090903C **XOR EAX**, 3C909090

0x1A1A010F: **35**9090903C **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

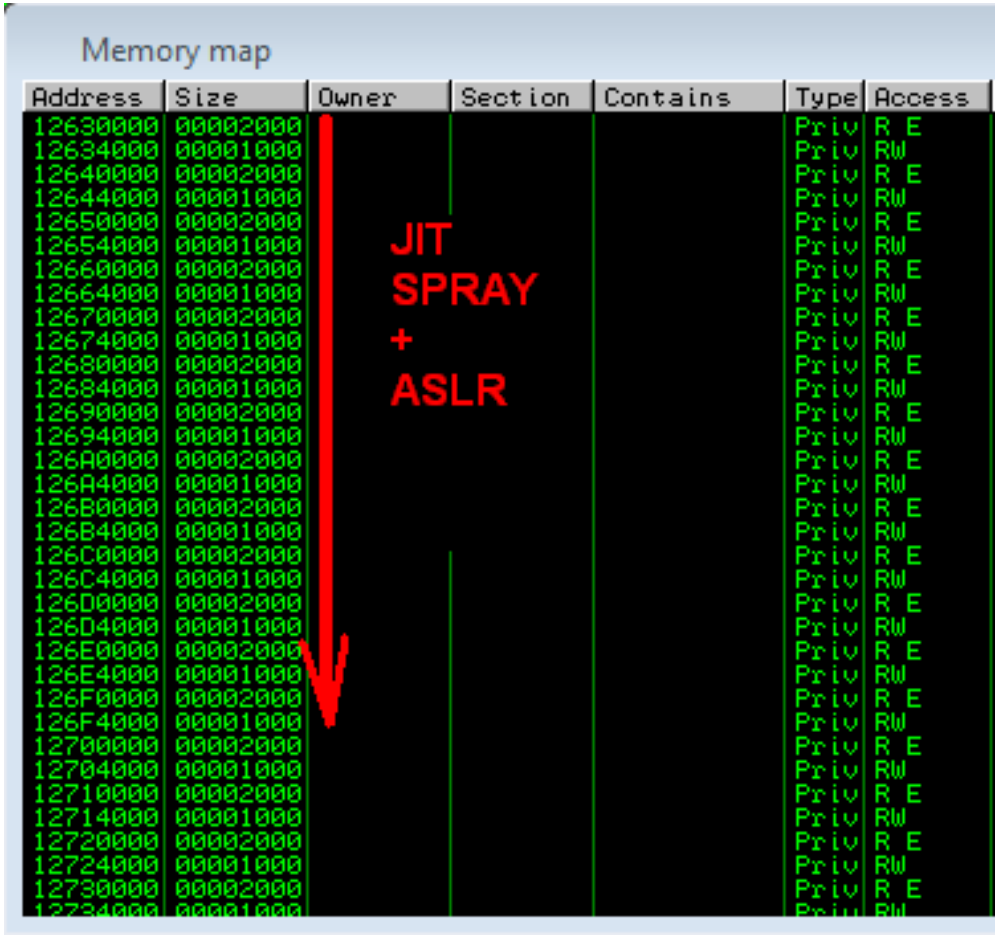
JIT Shellcode

Size

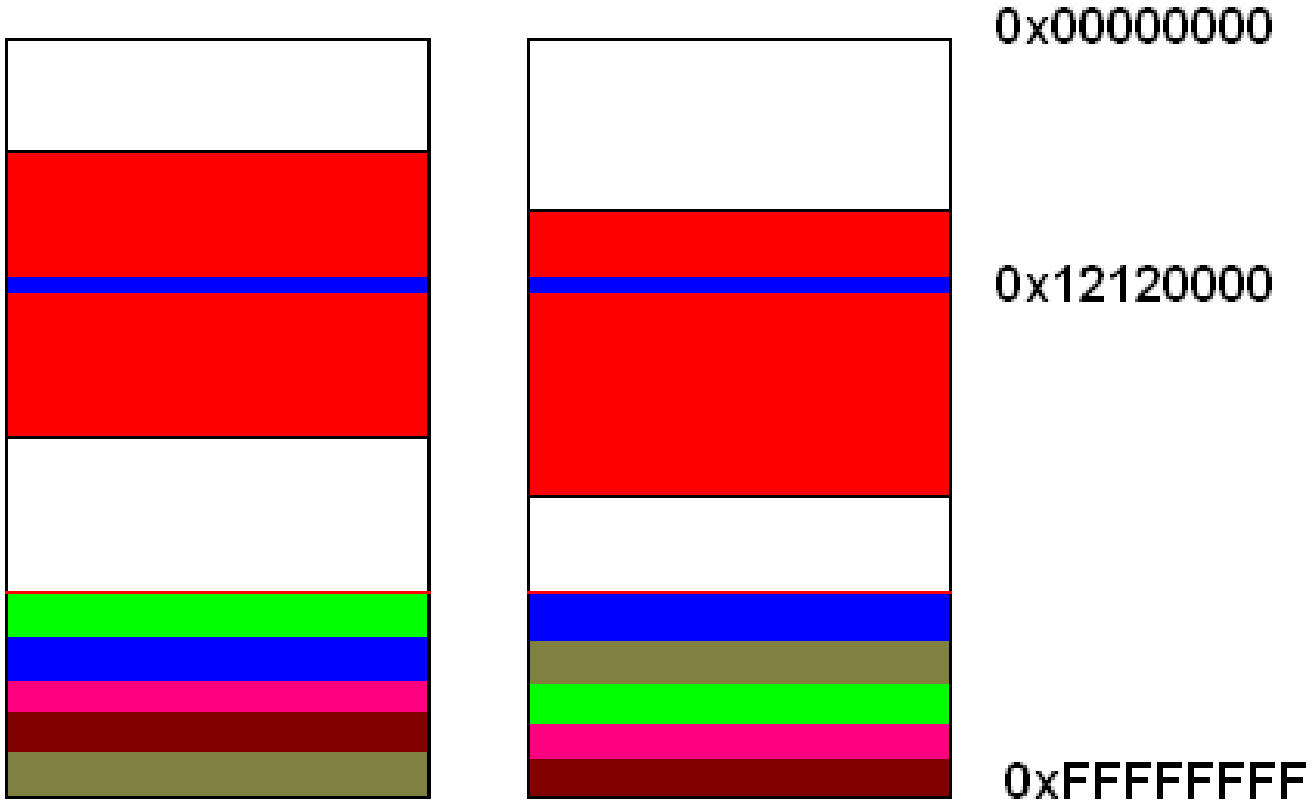
- 0xXXYY0000 – base address of every block with JITed shellcode [17]
- Intro Flash code – from beginning with the size of ~ 0xD3
- Offset between blocks 0x00010000 (If block size less than 0x1000)
- We can guess address of block: 0xXXYY0101

Is this enough for ASLR bypass ?

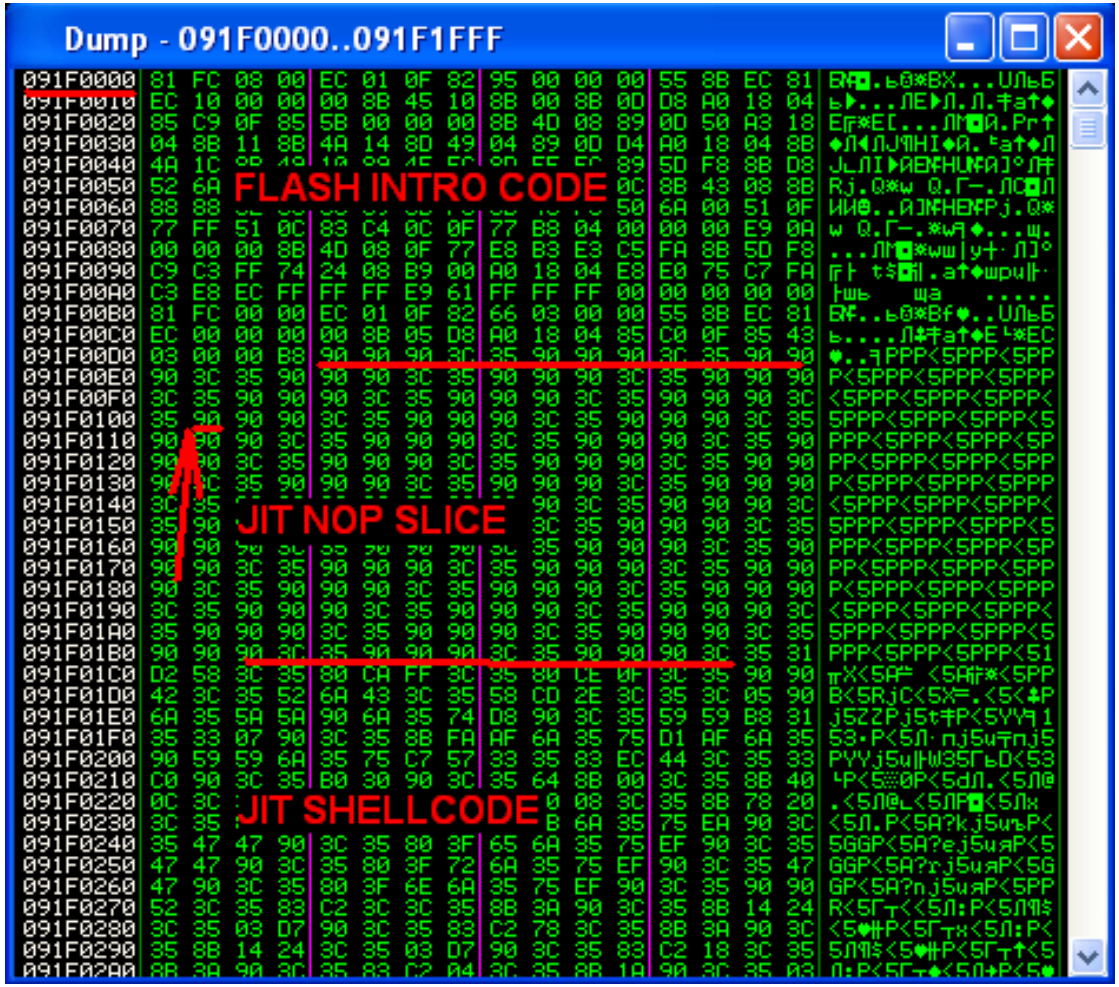
JIT spray beats ASLR+DEP



JIT spray beats ASLR+DEP



Guess address with ASLR



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**

Digital Security

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.semantiscope.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 paremt.close() exploit by **Krystian Kloskowski**)

Digital Security

Questions?

www.dsecrg.com

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