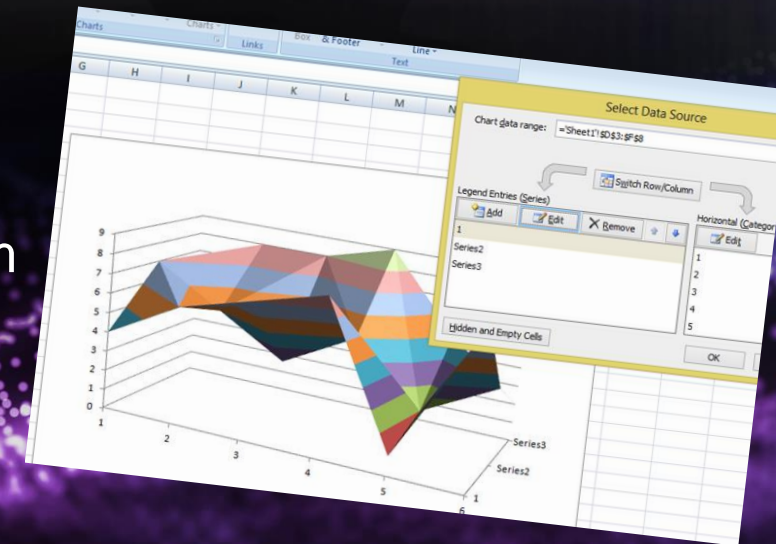
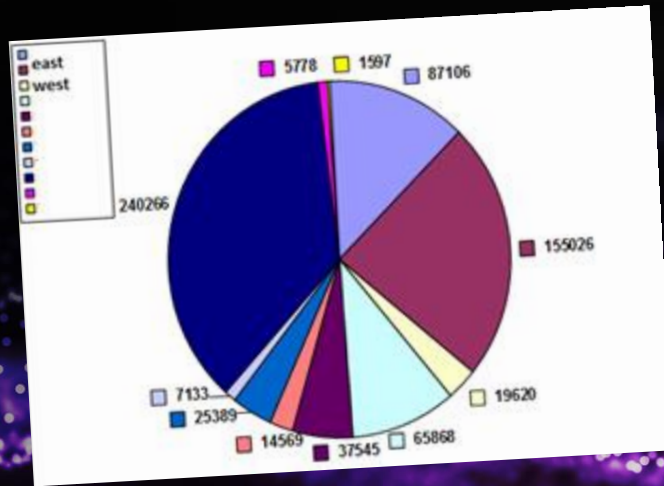


I calc'd Calc - Exploiting Excel Online

Nicolas Joly - @n_joly

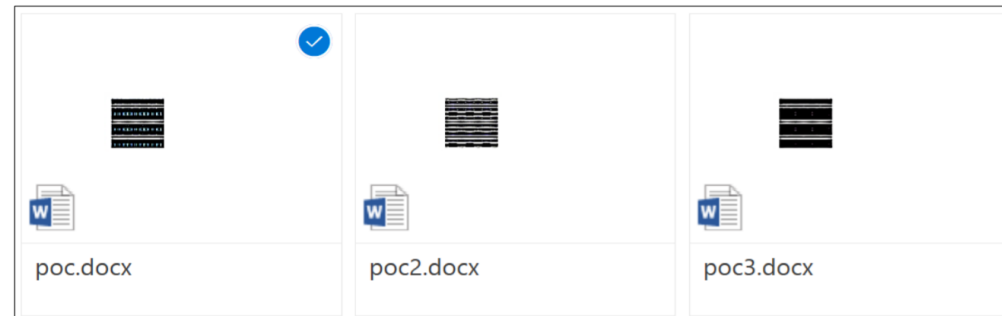
MSRC Vulnerabilities and Mitigations Team



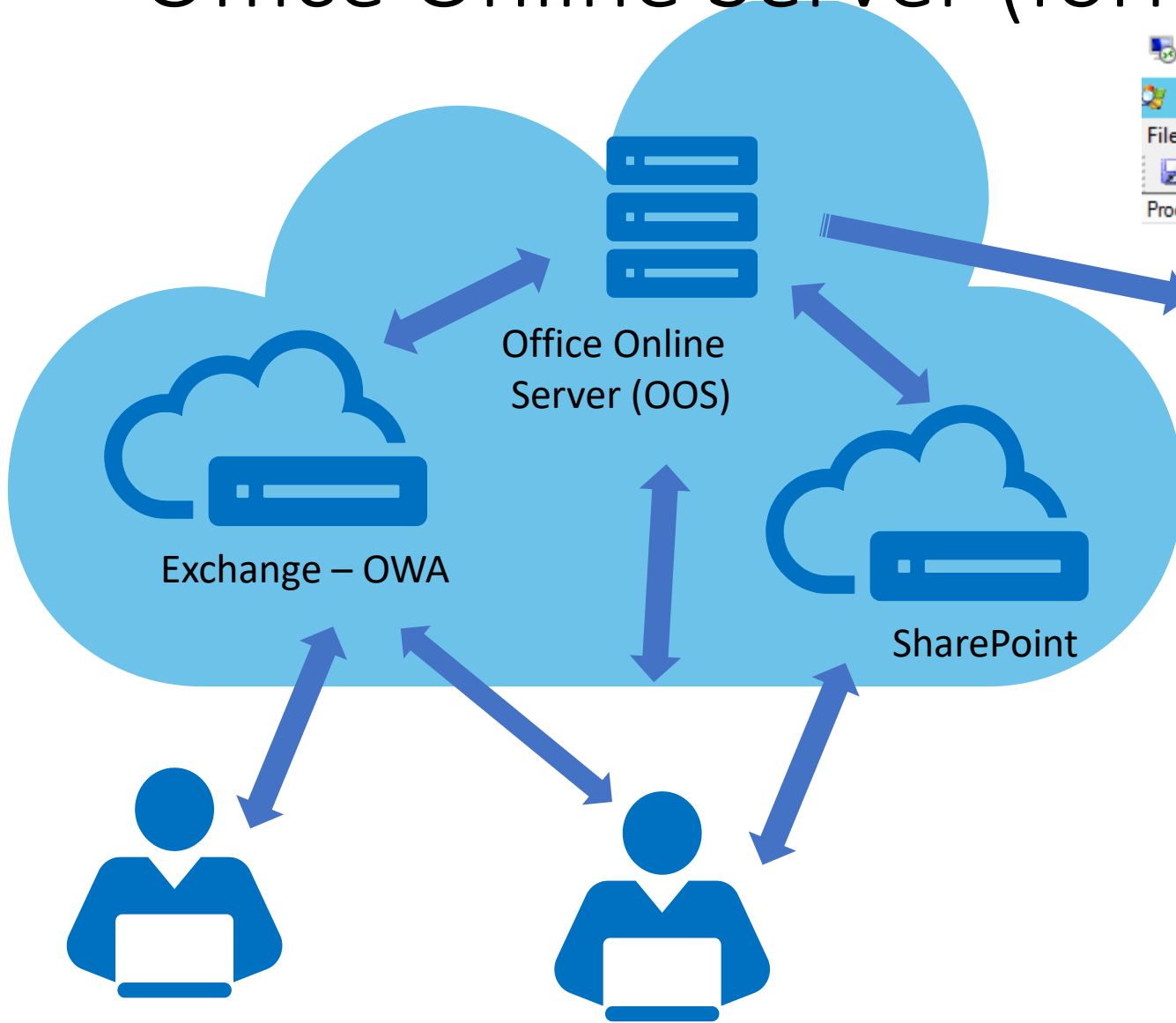


Office exploits?

- Several in the past years, essentially logic issues
- No exploit for memory corruption involving core Office features seen recently
 - CVE-2015-2545 a bug in the EPS font parser exploited in Word
 - CVE-2017-11882 stack overflow in Equation Editor
- What about Office Online?
 - Some issues found in the past
 - CVE-2016-3263 found by Mateusz “j00ru” Jurczyk affecting GDI
 - Uninitialized memory
 - Triggerable in Office Online



Office Online Server (formerly WAC)



oos - Remote Desktop Connection

File Options View Process Find Handle Users Help

Process	CPU	Private Bytes	Working Set	PID	D
rdpclip.exe		2,040 K	9,008 K	756 R	
svchost.exe		1,308 K	4,788 K	1260 H	
svchost.exe	< 0.01	2,916 K	10,496 K	1372 H	
svchost.exe		6,092 K	10,884 K	2064 H	
w3wp.exe	0.02	96,512 K	90,268 K	3580 IIS	
w3wp.exe	0.10	325,840 K	263,636 K	3588 IIS	
w3wp.exe					
w3wp.exe					
w3wp.exe					
w3wp.exe					
w3wp.exe					
w3wp.exe					
w3wp.exe					
W3WP.exe					
EditAppServerHostSlim.exe	< 0.01	76,768 K	99,396 K	5872 M	
w3wp.exe	0.09	225,612 K	158,364 K	6000 IIS	
w3wp.exe	0.01	216,956 K	161,552 K	6112 IIS	
AppServerHost.exe		38,936 K	57,372 K	14464 A	
AppServerHost.exe		30,744 K	30,876 K	12408 A	
AppServerHost.exe		38,620 K	57,372 K	16224 A	
AppServerHost.exe		30,736 K	30,856 K	3244 A	
AppServerHost.exe		30,716 K	30,820 K	14444 A	
AppServerHost.exe		30,684 K	30,860 K	15152 A	
AppServerHost.exe		30,736 K	30,880 K	13308 A	
AppServerHost.exe		30,740 K	30,884 K	14204 A	
AppServerHost.exe		30,700 K	30,860 K	12556 A	
AppServerHost.exe		30,736 K	30,900 K	9832 A	
AppServerHost.exe		30,736 K	30,892 K	16712 A	
AppServerHost.exe		30,752 K	30,856 K	1568 A	
AppServerHost.exe		30,732 K	30,868 K	14820 A	
AppServerHost.exe		34,160 K	52,460 K	9440 A	

Command Line:
 c:\windows\system32\inetpub\w3wp.exe -ap "HTTP80" -v "v4.0" -l "webser
 2f65b6-753d-43b0-8092-bb46966233e9 -h "C:\inetpub\temp\appools\HT
 20 ta 0
 Path:
 c:\Windows\System32\inetpub\w3wp.exe

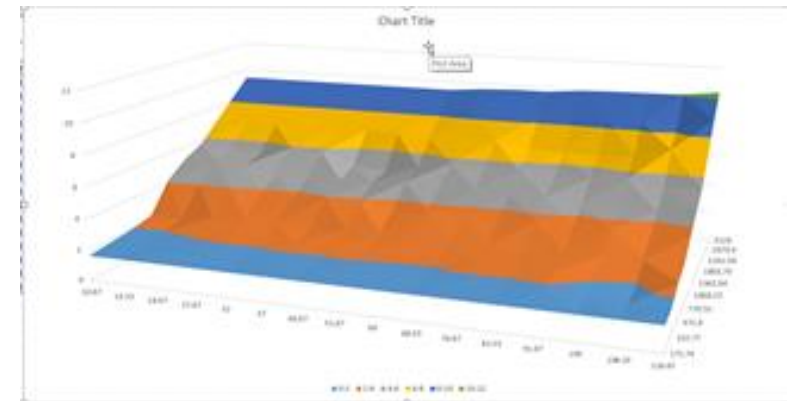
Scope of the project

- Is it possible to get an exploit against Office Online?
- Where would an attacker go?
- Do we need insider knowledge?
- How much time would it take?
- What would it look like?
- What can be done better?



Hacking Excel Online

- Xlsrv.dll on the server, ~40mb, using Excel's core functionalities
 - A bug affecting Desktop Excel will likely affect Excel Online
- How to start? Fuzzing?
 - In 2019 the MSRC received 50+ cases affecting Excel
 - Excel has been fuzzed for 20 years
 - Can we try fuzzing for a limited period of time and hope to find a cool bug?
 - Running a smart fuzzer on the cloud?
- Also what does a “cool bug” look like?
 - What are we looking for exactly?



No scripting but... Formulas!

- Exploiting without interaction?
 - Uncommon but happens
 - <https://scarybeastsecurity.blogspot.com/2016/11/0day-exploit-advancing-exploitation.html>
- Formulas!
 - Easy to manipulate/craft a file (XLSX, XLSB, XLS)
 - Provide interaction with the server
 - Lots of features (Math, Text, Finance)

```
137 <row r="16" spans="1:4" x14ac:dyDescent="0.3">
138   <c r="A16" t="s">
139     <v>390</v>
140   </c>
141   <c r="B16">
142     <f>GETPIVOTDATA("Sum of Qux",B32)</f>
143     <v>57</v>
144   </c>
145 </row>
```

No scripting but... Formulas!

- How does the average exploit behave?
 - Set/Get variables => **INDIRECT** formula for getter, **cannot set**
 - Heap spray, allocate strings quickly => **REPT** formula
 - If / Switch case statements => **IF/IFS/SWITCH** formulas
 - Iterating over arrays => **(V/H/X)LOOKUP** formulas
 - Use string routines => **MID, SEARCH, REPLACE** formulas
 - Eval() => **Unlikely, macros are unsupported, Evaluate() is an embedded macro**
 - Free / allocate objects => **???**
 - Automatic / manual recalc
- For example:

```
✕ ✓ fx =IF(K1="NOPE","",CONCAT(UNICHAR(HEX2DEC(MID(K6,5,4))),UNICHAR(HEX2DEC(MID(K6,1,4))),base_high,null_wchar) ▼
```

Looking at Excel formulas

- Back in 2008, [CVE-2008-4019](#) – Integer Overflow in REPT formula
- The vulnerability: REPT(“AAAA”, 1073741825)
 - $4 * 1073741825 = 4 * 0x40000001 = \dots = 4$ on 32 bits!
 - Was leading to an exploitable stack overflow
- 10 years later? What happened to that bug?

```
case FUNC_REPT:
{
    WCHAR *pch;
    int ichTotal;
    BOOL fOverflow = fFalse;

    ichTotal = CbAllocSafe(ich, cch, 0, &fOverflow);
    if (fOverflow)
        goto LRetErrOom;

    /* ich is actually count */
    if (ichTotal > pevalglob->m_cchMaxStCell || ichTotal < 0 || ichTotal > cchMaxSt
        goto LRetErrOom;
```


Looking at Excel formulas

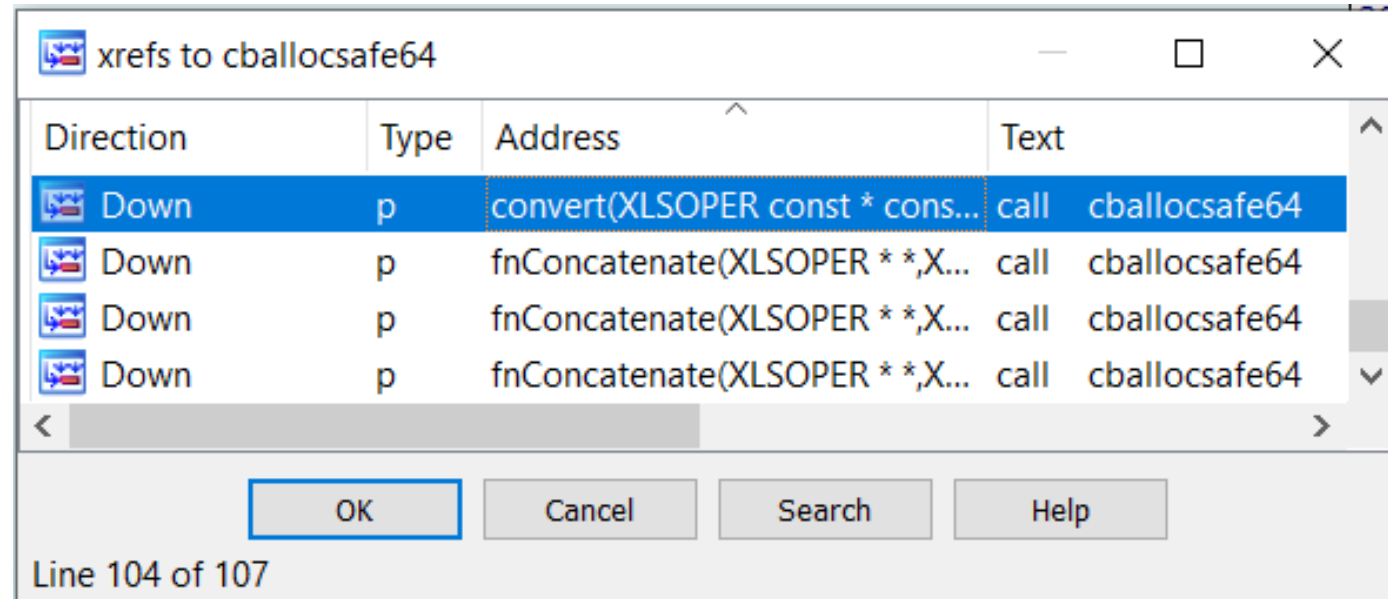
- CbAllocSafe now checks the parameters

```
DECL_CSYM UINT32 __fastcall CbAllocSafe(UINT32 cRec, UINT32 cbRec, UINT32 cbExtra, BOOL *pfOverflow)
{
    SAFEINT si;

    si.Init(cRec);
    si.Mult(cbRec);
    si.Add(cbExtra);
    *pfOverflow = si.FOverflow();

    return(si.Acc());
}
```

- Can we find anything similar?
- 3 refs in fnConcatenate?



Direction	Type	Address	Text
Down	p	convert(XLSOPER const * cons...	call cballocafe64
Down	p	fnConcatenate(XLSOPER **,X...	call cballocafe64
Down	p	fnConcatenate(XLSOPER **,X...	call cballocafe64
Down	p	fnConcatenate(XLSOPER **,X...	call cballocafe64

Line 104 of 107

Looking at Excel formulas

- Look at that!

```
.text:000000018012DBDF  
.text:000000018012DBE2  
.text:000000018012DBE5  
.text:000000018012DBE8  
.text:000000018012DBEB  
.text:000000018012DBEF  
.text:000000018012DBF2  
.text:000000018012DBF4  
.text:000000018012DBF7  
.text:000000018012DBFA  
.text:000000018012DBFD
```

```
add    ecx, r13d  
sub    eax, r14d  
add    eax, r13d  
imul   ecx, eax  
lea    edx, [r8+8]  
mov    eax, [rsi+4]  
sub    eax, [rsi]  
add    eax, r13d  
imul   ecx, eax  
mov    [rbp+3Ch], ecx  
call   cballocsafe64
```

- Quick X-Ref on fnConcatenate, what is “TEXTJOIN”?

```
dq offset aTextjoin      ; "TEXTJOIN"  
dq offset ?fnConcatenate@@YAXPEAPEAVXLSOPER@@PEAV1@HPEBUFunc@@PEAVEvalGlobals@@@Z
```

Looking at Excel formulas: TEXTJOIN

- Syntax:

TEXTJOIN(delimiter, ignore_empty, text1, [text2], ...)

argument	Description
delimiter (required)	A text string, either empty, or one or more characters enclosed by double quotes, or a reference to a valid text string. If a number is supplied, it will be treated as text.

- Example:

The screenshot shows an Excel spreadsheet with the following data:

	A	B	C	D	E	F	G
1	a	b	c	d			
2	=TEXTJOIN("AAAA", TRUE, A1:D1)						

The formula bar shows: `=TEXTJOIN("AAAA", TRUE, "AAAA", "BBBB", "CCCC")`

Looking at Excel formulas: TEXTJOIN

- This formula was extended in 2015 to support 3D references
- That's the code in question:

```
cDelimiter = pcalcrefData->GetHeight() * pcalcrefData->GetWidth() * (isheetLast - isheet + 1);  
cbrgDelim = CbAllocSafe(cDelimiter, sizeof(XCHAR*), 0, &fOverflow);
```

```
if (fOverflow)  
    goto LRetErr;
```

```
if (!SUCCEEDED(pevalglob->PmemheapRecalcBuffer()->HrAllocPv(cbrgDelim, (void**) &rgstDelim)))  
    goto LRetErr;
```

- And to trigger:
TEXTJOIN (Sheet2 : Sheet10 !A1:KZB529328 ,TRUE, "AAAA", "BBBB", "CCCC")
 - A1:KZB529328 is an array of... 0x100000060 cells
- CVE-2018-8574

Exploitation, straightforward?

- Three loops to follow, to iterate over sheets, rows and columns:

```
while (true)
{
    for (rw = pcalcrefData->RwFirst(); rw <= pcalcrefData->
    {
        for (col = pcalcrefData->ColFirst(); col <= pcalcre
        {
            xlsoper.FastInit();

            rgstDelim[iIndexDelimiter] = stDelimItem;
```

- We're writing pointers to Strings
- No re-entrancy
- But the good news is...
 - We can exit safely!
 - => controlled overflow

```
if (xlsoper.FIsErr())
{
    hr = xlsoper.HrFinalizeAndTransferErrorResult(pxlsoperRes);
    if (FAILED(hr))
    {
        fNeedDoImp = true;
    }
    xlsoper.FastFree();
    goto LDoneConcat;
}
```

Exploitation, straightforward?

- Excel only supports up to 1048576 rows and 16384 columns:
 - $r < 0x100000$, $c < 0x4000$, s (sheets) and $c*r*s > 0x100000000$
 - A1:KZB529328 fits perfectly in there
- Since we're causing an exception, everything is free()'d before fnConcatenate returns:

```
LDoneConcat:  
    pevalglob->Pgcd()->SetPenvMem(penvSav);  
  
    for (iIndexDelimiter = 0; iIndexDelimiter < cDelimiterAllocated; iIndexDelimiter++)  
    {  
        PchBufReleaseXls(pevalglob->Pgcd()->PmemheapRecalcBuffer(), const_cast <XCHAR*>(rgstDelim[iIndexDelimiter]));  
    }  
  
    pevalglob->Pgcd()->PmemheapRecalcBuffer()->FreePv(rgstDelim);
```

- Integer overflow => heap overflow => use-after-free!



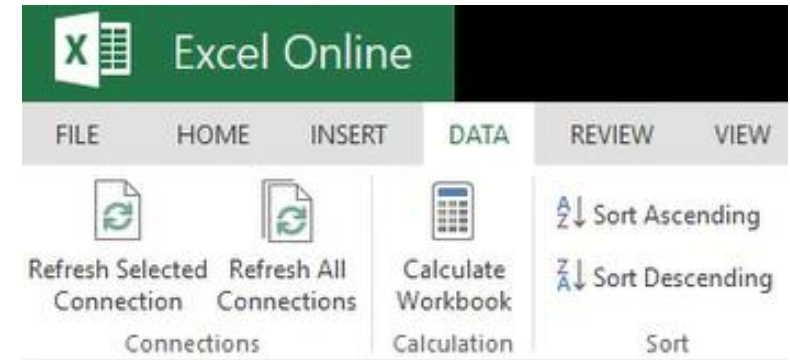
Exploitation, straightforward?

- Strings make a great primitive
 - Excel stores those as SIZE (two bytes) + String
 - Overwriting the size of a string with a pointer gives read access on the heap
- Here's the plan for an infoleak:
 - Spray the heap with strings with REPT
 - Free some strings by using formulas to change a few cells
 - Allocate our vulnerable buffer in between
 - Overwrite a string length with a pointer
 - Read stuff, find some vtable and enjoy!
- Here's why it fails:
 - CTRL-Z or why UNDO makes things unfriendly!



Exploitation, straightforward?

- Making holes in the heap is not trivial
 - Create lots of actions to fill up the Undo stack?
- A possible solution: recalc the workbook
 - Flush the cache and free everything
 - Undo not possible afterwards
 - Complicate the exploit and require user interaction (or script)
 - Save the file and create additional overhead
- Overwriting a length by a pointer can cause read AV
- But when it works...



Exploitation, straightforward?

- Making



base_boom

- Creating



base_boom

FILE HOME INSERT DATA REVIEW VIEW

Refresh Selected Connection Connections Refresh All Connections Calculate Workbook Calculation Sort Ascending Sort Sort Descending Sort Flash Fill Data Tools

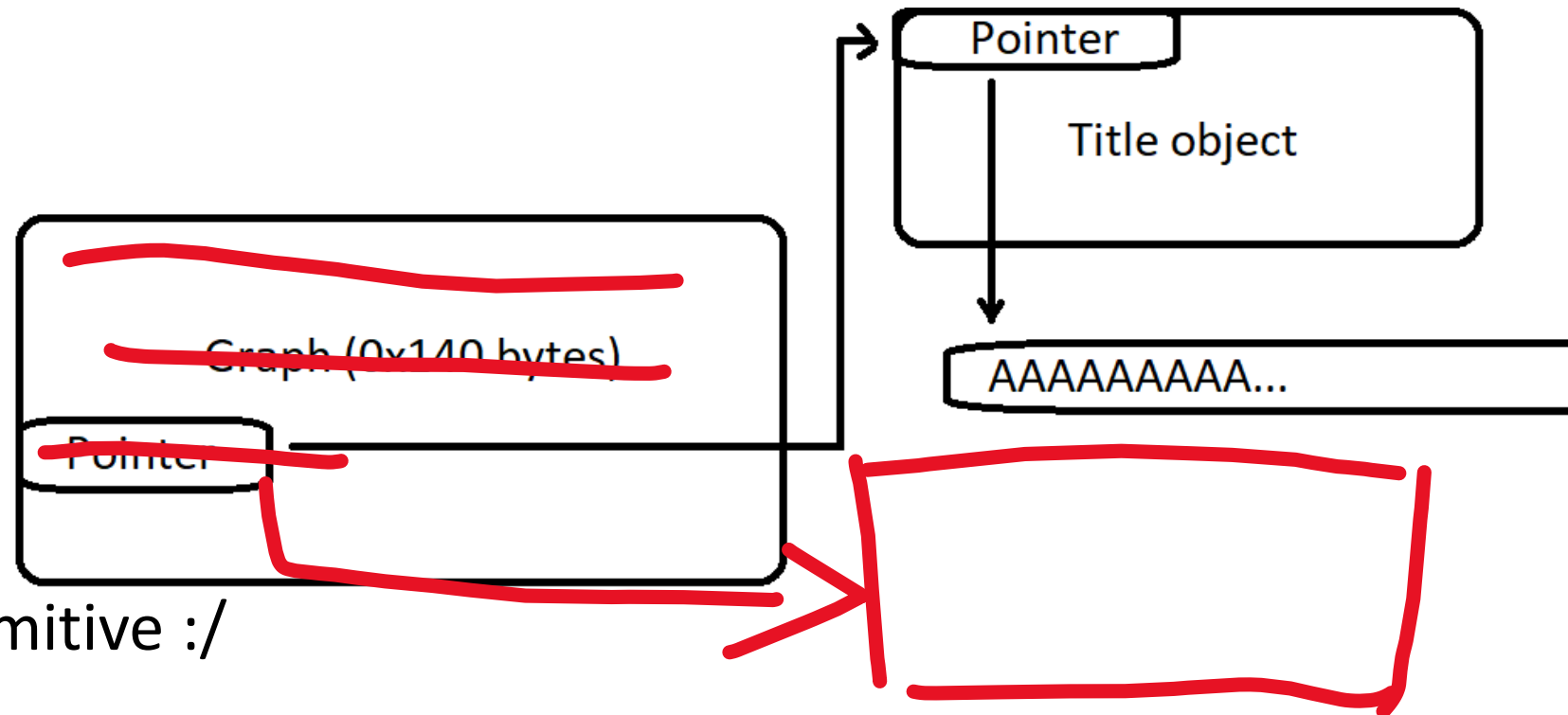
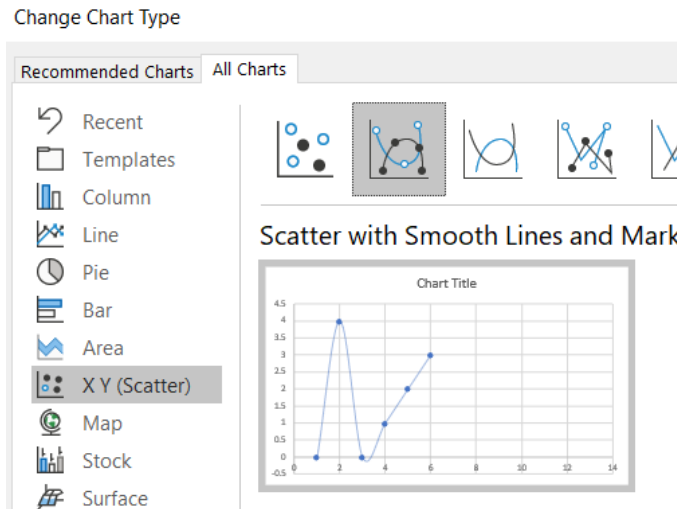
f_x =CONCAT("len: 0x", DEC2HEX(LEN(B1)))

	A	B	C	D	E	F	G	H	I	J	K	L
1	target_cell:	獸&免獸&葦泮&蒂泮	char at offset:	1	獸	char value in hex:	6B5E	offset in hex:	2	pointer?	#N/A	#N/A
2	len_cell:	len: 0xFA1A	char at offset:	2	;	char value in hex:	BF	offset in hex:	4	pointer:	#N/A	
3			char at offset:	3		char value in hex:	0	offset in hex:	6	null_wchar:	#N/A	
4			char at offset:	4	免	char value in hex:	FA32	offset in hex:	8	base_low:	#N/A	
5			char at offset:	5	獸	char value in hex:	6B5E	offset in hex:	A	base_high:	#N/A	#N/A
6			char at offset:	6	;	char value in hex:	BF	offset in hex:	C	gadget1_low (multiple calls):	#N/A	#N/A
7			char at offset:	7		char value in hex:	0	offset in hex:	E	gadget2_low (set @rdx):	#N/A	#N/A
8			char at offset:	8	葦	char value in hex:	8466	offset in hex:	10	gadget3_low (set [rdx]=f()):	#N/A	#N/A
9			char at offset:	9	泮	char value in hex:	6C7C	offset in hex:	12	gadget4_low (reset @rdx):	#N/A	#N/A

105	獸&免獸&葦泮&蒂泮&爺泮&爾泮&詳泮&嚮泮&昭泮&惣泮&牒泮&牒泮&掌泮&物泮&狛泮&具泮&狎泮&㊦波&狒泮&狒泮&狒泮&狒泮&
106	////////////////////
107	////////////////////

Exploitation, straightforward?

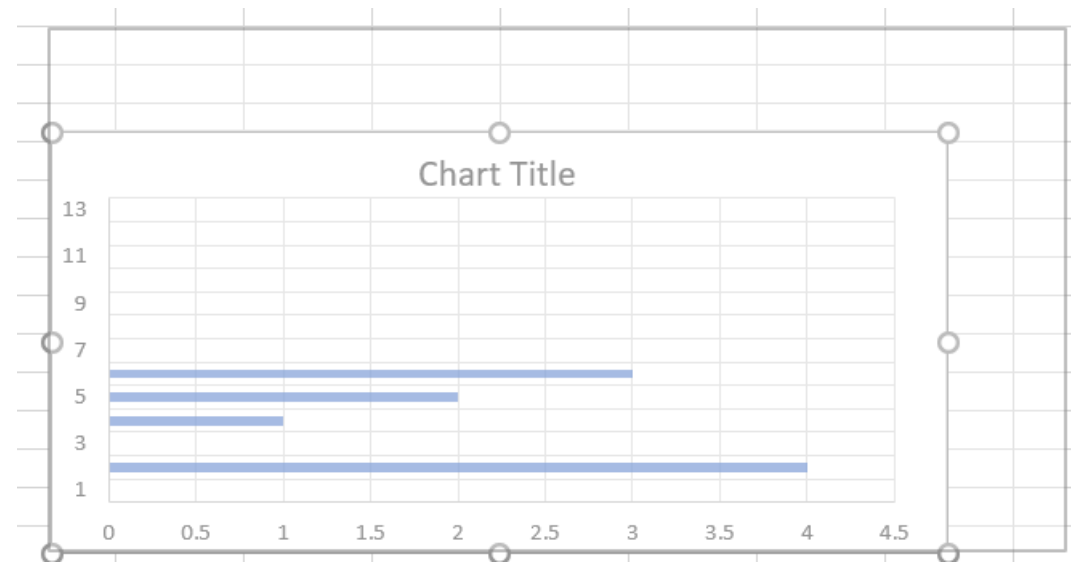
- Leaking was the easy part, but leaking what?
- Looked first at all the formulas
 - Saw nothing using C++ objects or vtables :/
- Looked at Charts



- Failed to get a RW primitive :/

Exploitation, straightforward?

- Eventually went for the easy way
 - Leaked a Graph object vtable
 - Built a ROP to load a library
 - Major issue: doesn't scale if we don't know xlsrv.dll
- To trigger, add a Graph, overwrite its vtable and just resize it
 - Will trigger a vtable call
- Didn't work?
 - Just retry



Demo

Wrapping up

- A cool exploit written for Excel Online
 - Shows exploits are possible and feasible for Office Online
 - Two exploitable CVEs uncovered CVE-2018-8331 and CVE-2018-8574
 - Would we see the same exploit in the cloud?
 - Unlikely, holes in the heap are difficult to secure
- Raise more questions
 - Can we do the same on Office Desktop?
 - What about the other Office applications?
 - Once on the server, what can we do?

THANK YOU



References

- [Mateusz “j00ru” Jurczyk - Windows Metafiles – PacSec 2016](#)
- <https://scarybeastsecurity.blogspot.com/2016/11/0day-exploit-advancing-exploitation.html>
- [CVE-2008-4019](#) – Integer Overflow in REPT formula
- [TEXTJOIN Formula](#)