Hacking Excel Online

How to exploit Calc

Nicolas Joly - @n_joly
MSRC Vulnerabilities and Mitigations Team

SSTIC – June 5th, 2020

This presentation is for informational purposes only. MICROSOFT MAKES NO WARRANTIES, EXPRESS OR IMPLIED, IN THIS SUMMARY.
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
• 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)
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](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)
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:
  ```excel
  =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)
    • $4 \times 1073741825 = 4 \times 0x40000001 = \ldots = 4$ on 32 bits!
    • Was leading to an exploitable stack overflow
• 10 years later? What happened to that bug?

```c
    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

```c
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.F0verflow();

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

• Can we find anything similar?
• 3 refs in fnConcatenate?
Looking at Excel formulas

• Look at that!

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

  dq offset aTextjoin     ; "TEXTJOIN"
  dq offset ?fnConcatenate@@YAXPEAPEAVXLS0PER@@PEAV1@HPEBUFunct@@PEAVEvalGlobalst@@OZ
Looking at Excel formulas: TEXTJOIN

• Syntax:

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

<table>
<thead>
<tr>
<th>argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>delimiter</td>
<td>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.</td>
</tr>
</tbody>
</table>

• Example:

```
=TEXTJOIN(A1:D1,TRUE,"AAAA","BBBB","CCCC")
```

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>a</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>AAAAaBBBBbCCCC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Looking at Excel formulas: TEXTJOIN

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

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

    if (fOverflow)
        goto LRetErr;

    if (!SUCCEEDED(pevalglob->PmemheapRecalcBuffer()->HrAllocPv(cbrefDelim, (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 <= pcalcref
        {
            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
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:

```c
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 holes in the heap is not trivial
  • Create lots of actions to fill up the Undo stack?
• A possible solution:
  • Recalculate 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 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?

- 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 function