Microsoft Office Protected-View Out-Of-Bound Array Access

2017-11-23

<table>
<thead>
<tr>
<th>Software</th>
<th>Microsoft Office</th>
</tr>
</thead>
<tbody>
<tr>
<td>Affected Versions</td>
<td>Microsoft Excel 2010, 2013, 2016 (x86 and x64)</td>
</tr>
<tr>
<td>CVE Reference</td>
<td>CVE-2017-8502 (MS Office Memory Corruption Vulnerability)</td>
</tr>
<tr>
<td>Author</td>
<td>Yong Chuan Koh (@yongchuank)</td>
</tr>
<tr>
<td>Severity</td>
<td>Important</td>
</tr>
<tr>
<td>Vendor</td>
<td>Microsoft Corporation</td>
</tr>
<tr>
<td>Vendor Response</td>
<td>Fixed on 11 July 2017</td>
</tr>
</tbody>
</table>

Description:

Microsoft Office is a suite of desktop applications consisting of Word, Excel, Powerpoint, Outlook and various other productivity applications. Among these, Word, Excel and Powerpoint implemented the Protected-View sandbox technology as a defence-in-depth exploit mitigation.

An out-of-bound array access was discovered while the Excel broker parsed an attacker controlled Protected-View Inter-Process Communication (IPC) message from the sandbox process.

Impact:

Successful exploitation would allow an attacker to elevate his privileges from AppContainer to Medium, thereby breaking out of the Protected-View sandbox.

Cause:

The vulnerability existed because the IPC message execution were influenced by a global flag which was set by a preceding IPC message. Subsequently the broker process made an incorrect assumption on the array size and dereference an out-of-bound object at a hardcoded offset.
Interim Workaround:

Avoid opening Microsoft Office Excel files from untrusted sources.

Solution:

Users should apply the July security updates from Microsoft.

Technical details

The following analysis is based on EXCEL 16.0.4266.1001.

In Protected-View mode, the Excel broker receives and services IPC messages from the Excel sandbox, differentiated by a message-tag. In order to trigger this vulnerability two 0x0E1200 messages need to be sent. The first message sets the global variable “dword_1461918” to 2Fh as shown in the orange code-block below, in EXCEL.EXE.

Figure 1: In the main function to service 0x0E1200 message, sub_5195E() is called to allocate and parse the objects array

In the second message, executes the code-block highlighted in blue in the above image. This then calls EXCEL.sub_5195E0. Within this function the broker allocates memory for an array of objects that are each 7Ch-bytes long:
Figure 2: Allocated array for "eax" number of 7Ch-bytes objects

Next, a pointer to this array is passed to MSO.MSO_23080, and eventually to MSO.sub_1203040 within MSO.DLL where it is stored at offset DCh within the global variable 'dword_C0637C':

Figure 3: Array pointer is saved to poi(poi(poi(poi(poi( poi(poi(dword_C0637C)+8))+8))+8)+8)+8)+8)+8)+8) in MSO.sub_1203040
The exact relation between ‘dword_C0637C’ and the buffer is as below:

\[
\begin{align*}
\text{poi(poi(poi(poi(poi(MSO+C0637C))+8))+8)} + \text{DC} &= \\
\text{poi(poi(poi(poi(MSO+C068C8))+8))+DC} &= \\
\text{poi(poi(poi(MSO+C06390)+8))} + \text{DC} &= \\
\end{align*}
\]

It was observed that there were either 5 or 6 objects within this buffer. The vulnerability existed because Excel assumes that if the global variable “dword_1461918” is set to 2Fh then there will exactly 6 such objects. This assumption however is incorrect because if two 0x0E1200 messages are sent in a row then there is only 5 such objects in the array leading to out-of-bound access beyond this array.

This starts at EXCEL.0099FF26, where MSO.MSO_97() is called if ‘dword_1461918’ is 2Fh.

Figure 4: Looping the array to initialize each 7Ch-bytes object in MSO.sub_120304()
Figure 5: In EXCEL.sub_99FEEE(), MSO.MSO_97() is called if global variable dword_1461918 is 2Fh

In MSO.MSO_97(), Excel dereferences the 6th object directly by the offset (15h–10h)*7Ch = 26Ch. The following 2 screenshots shows the calculation of this offset:
Figure 6: Retrieving array pointer from `poi(poi(poi(poi(poi(dword_C0637C))+8))+DC)`

Figure 7: Add offset 26Ch bytes from array pointer for the 6th 7Ch-byte object
The next 2 screenshots show that ESI, which is pointing to the 6th object in the array (which may not exist), is being dereferenced at offset 34h:

Figure 8: esi is pointing to the 6th object in the array

Figure 9: Object is dereferenced at offset 34h
The following output from windbg show the result of sending two 0x0E1200 messages to the Excel broker.

Figure 10: Windbg trace of OOB access due to 0x0E1200 IPC message

In summary, on the first 0x0E1200 message, Excel broker sets dword_1461918 to 2Fh, and allocates an array for 6 7C-byte objects long. On the second 0x0E1200 message, an array for 5 such objects is allocated at address 0x38002D90. However, because the global variable dword_1461918 is 2Fh, MSO.MSO_97() attempts to access a 6th object in an array of 5 objects. It then attempts to deference a pointer at offset 34h within the nonexistent object leading to a crash due to out-of-bound access of the array. The "heap -p -a" command as shown above shows that ECX belongs to 0x38002D90, which verifies the buffer allocation.

labs.mwrlabs.com // @mwrlabs
A few basic-blocks down the crash address, offset 3Ch of this object is used as an index into a function-pointers table at ‘dword_A64010’ which does not perform any bounds checking. Therefore, if an attacker is able to manipulate the heap and control the fields of this object, it is possible to set an arbitrary index out of this function-pointers table to execute code in context of Excel broker.

Figure 11: Offset 3Ch of out-of-bound object is used as an index into function-pointers table dword_A64010

Figure 12: Function-pointers table at dword_A64010
## Detailed Timeline

<table>
<thead>
<tr>
<th>Date</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>2017-03-30</td>
<td>MWR Labs reported vulnerability and POC to MSRC</td>
</tr>
<tr>
<td>2017-03-30</td>
<td>MSRC acknowledged and open MSRC case 38000</td>
</tr>
<tr>
<td>2017-04-05</td>
<td>MSRC confirmed they have reproduced the vulnerability</td>
</tr>
<tr>
<td>2017-06-17</td>
<td>MSRC responded that this will be patched in July 2017</td>
</tr>
<tr>
<td>2017-07-11</td>
<td>MSRC assigned CVE-2017-8502 and released patch for this vulnerability</td>
</tr>
<tr>
<td>2017-11-23</td>
<td>MWR Labs released advisory</td>
</tr>
</tbody>
</table>