OffVis 1.0 Beta: Office visualization tool

By Russ McRee – ISSA member, Puget Sound (Seattle), USA Chapter

Prerequisites

Windows XP or later
.NET 3.5

Similar Projects

OfficeMalScanner[1]

Malware in its various forms has been blessing computing platforms with its malevolent presence for many years now. Virus history includes 1986's Brain, which infected the boot sector of floppy disks; 1995 brought us the macro virus; email distribution of Melissa came our way in 1999; and Love Letter caused major mayhem in 2000. Famous worms including Morris, Code Red, Nimda, Blaster, and others have cost industry and consumers billions of dollars.[2]

Fairly recent history includes targeted Microsoft Office-specific attacks. Although Office documents have suffered abuse since 1999 (think Melissa and Visual Basic macros), it wasn't until late 2005 that we saw attackers stepping up to exploit parsing flaws in the Office software itself. As attackers and researchers continued to "move up the stack" when looking for security vulnerabilities, they focused on applications that parse binary file formats, facilitated by advanced fuzzers that allow easier bug hunting in such applications.[3] If you review Microsoft bulletins released from 2003 to 2005, you won't find any specific to Microsoft Office 2003 SP2 and lower; then in 2006 there were 12 bulletins, followed by 13 in 2007, and so on.

Our focus rests squarely on targeted Office attacks and the associated malicious methodology as we discuss OffVis 1.0 Beta,[4] a tool created by Microsoft Security Response Center (MSRC[5]) Engineering. As part of the Microsoft Active Protections Program (MAPP[6]), the MSRC Engineering team spends a great deal of time looking for ways to detect exploitation of given vulnerabilities, in particular those that are Office-related. These efforts led to the creation of OffVis, starting in November 2008. First released in beta to MAPP participants, it has matured into a UI-based tool that analyzes a very specific set of vulnerabilities in order to better help defenders. MSRC Engineering's work allows them to build detection logic, and then reuse it as part of ongoing analysis efforts.

OffVis is not intended as a supported, enterprise tool, but is available as a no-charge download and is simple to use. Truly understanding the nuances of OffVis output may be a bit challenging for those who don't spend a great deal of time parsing Office documents (me included), but I'll make every effort to share with you what I've learned while researching OffVis for this month's column. At its simplest OffVis will tell you that a file is either *Definitely Malicious* or *Possibly Malicious*, and that should be sufficient to serve you in defense against the dark arts. If your CEO asks you to take a look at an Office document that she received from a stranger with a *.cn or *.ru email domain, this is the tool to use.

Understanding the Office file format

First, I must clarify an important distinction. OffVis only concerns itself with the binary Office (Office 2003 and lower) file format rather than the OpenXML format. Office documents based on the binary format are considered by many to be convoluted. These files are based on the Compound Document Format (OLE2 Structured Storage (OLESS)), a format that is basically a block-based file system with specific files and directories for each type and version of Office document. The actual "file" entries within these documents are also proprietary and change based on the version and features of the Office software used to create them. Consider it a file system within a file complete with a FAT table, sectors, and streams. In order to detect a file format exploit, most parsing software needs to understand OLESS, locate the correct entry containing the document contents, and parse through that content to locate the specific content that triggers the exploit. This process can be CPU intensive and requires the parsing software to have a deep understanding of the version-specific Office document data inside of the OLESS container. OffVis parses each unique format specifically for certain binary format detection logic specific to eight different vulnerabilities. SRD chose these CVEs based on their prevalence in the wild (see Figure 1).

1 http://www.reconstructer.org/code/OfficeMalScanner.zip.

2 http://www.mcafee.com/us/local_content/white_papers/partners/ds_wp_telconote.pdf.

3 R. Hensing, http://cansecwest.com/csw08/csw08-hensing.pptx, pg. 4.

4 http://blogs.technet.com/srd/archive/2009/07/31/announcing-offvis.aspx.

5 http://blogs.technet.com/msrc/default.aspx.

6 http://www.microsoft.com/security/msrc/collaboration/mapp.aspx.

CVE	Product	Bulletin
CVE-2006-0009	PowerPoint	MS06-012 (March 2006)
CVE-2006-0022	PowerPoint	MS06-028 (June 2006)
CVE-2006-2492	Word	MS06-027 (June 2006)
CVE-2006-3434	Word	MS06-062 (October 2006)
CVE-2007-0671	Excel	MS07-015 (February 2007)
CVE-2008-0081	Excel	MS08-014 (March 2008)
CVE-2009-0238	Excel	MS09-009 (April 2009)
CVE-2009-0556	PowerPoint	MS09-017 (May 2009)

Figure 1 – CVEs detected

As are most block-based file systems, the OLESS format becomes easily fragmented. When Office software writes data, it seeks out any available free blocks before allocating new ones. The OLESS format utilizes two different block tables: one for small entries (normally set to be less than 4096 bytes), and another for larger contiguous segments. Fragmentation can occur during normal editing of an Office document, yet it is rare for documents to be heavily fragmented.[7] Nonetheless, OffVis offers a defragment tool and will warn you when you should use it to ensure better parsing accuracy.

Excel documents are the most popular to exploit, followed by PowerPoint, then Word documents. Therefore we'll focus specifically on the Excel binary file format as we use OffVis, and I'll provide a bit more specific detail. Excel documents are written to disk in the binary interchange file format (BIFF) and all data is stored inside the "workbook" stream. Data inside said stream is organized BOF <data> EOF BOF <data> EOF BOF <data> EOF, and so on. BIFF records follow the TLV (Type, Length, Value) format, a very convenient, efficient method of organizing data, especially variable length strings. Record data has an upper bound of ~ 2000–8000 bytes (BIFF version dependent) as mentioned above, but if record data is longer, a CONTINUE record is utilized.[8]

Anatomy of an Excel exploit

A typical targeted attack often includes an email sent to an intended victim with a malicious Excel document attached. When the victim opens the Excel document the following sequence might occur. First, it exploits a vulnerability to force Excel to run embedded shellcode. The shellcode then extracts an XOR'd, well-formed XLS file, and an EXE. The XLS opens in Excel, and the extracted EXE is executed which installs a backdoor as a service.[9] This actual limited targeted attack resulted in Microsoft releasing KB 947563[10] on January 15, 2008. The OffVis Excel parser includes detection logic for CVE-2008-0081,[11] the National Vulnerability Database CVE released in accordance with KB 947563. We'll look at a specific sample exploiting CVE-2008-0081 in Using OffVis.

Stepping through the exploit more specifically might appear as seen in Figure 2.

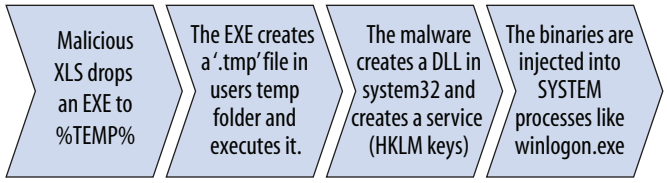

Figure 2 – Exploit walkthrough

Typical exploit structure (Figure 3) ensures that everything is included in the document; please note that there can be variations including multiple shellcode stages, multiple Trojans, and obfuscation of both Trojan and the document.[12]

For a much deeper dive into exploit structure, as well as disassembly and debugging techniques, see Bruce Dang's topical Black Hat Japan 2008 presentation.

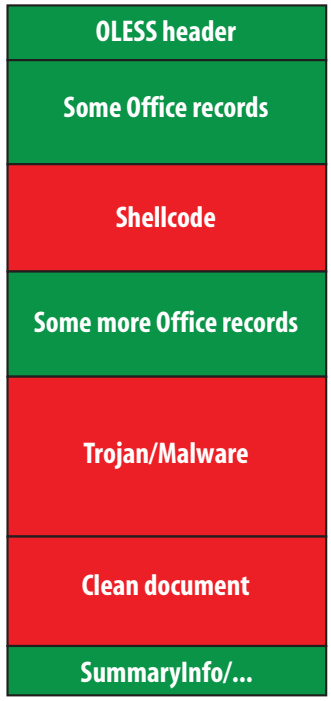

Figure 3 – Exploit structure details

Using OffVis

Using OffVis is quite simple. After unpacking, execute `OffVis.exe`. Note the *Parser* drop down menu; this is the most important selection in the UI. Select the parser appropriate to the file type you're analyzing.

WARNING: The following usability is so straightforward that brain-freeze may ensue from lack of needed cycles.

If you're studying an Excel file, choose Cases.dll : ExcelBIFF8BinaryFormatDetectionLogic(CVE-2008-081, CVE-2007-0671, CVE-2009-0238).

If you wish to analyze a PowerPoint document, select Cases.dll : PowerPoint97, etc.

Still there? ;-)

Click *File*, then *Open Data File*, select the file you wish to analyze, then click *Parse*.

As you can see in Figure 4, OffVis displays OLESS-based binary files in a hex view of the raw file contents in the left pane of the window, and the tree of objects built up from parsing the raw file contents in the right pane.

7 http://www.breakingpointsystems.com/community/blog/evasion-with-ole2-fragmentation.

8 B. Dang, http://www.blackhat.com/presentations/bh-jp-08/bh-jp-08-Dang/BlackHat-Japan-08-Dang-Office-Attacks.pdf, pg. 13.

9 R. Hensing, http://cansecwest.com/csw08/csw08-hensing.pptx, pg. 10.

10 http://support.microsoft.com/kb/947563.

11 http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2008-0081.

12 B. Dang, pg. 21.

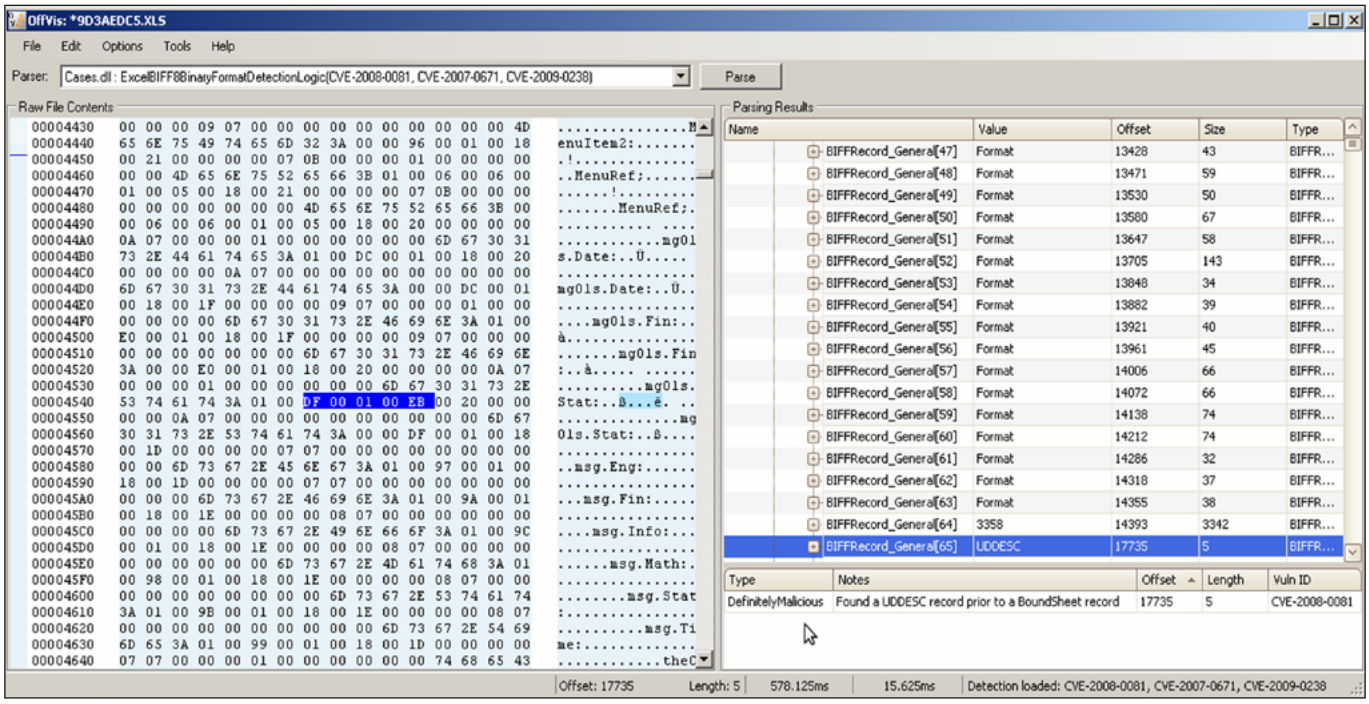

Figure 4 – OffVis detecting an exploit targeting CVE-2008-0018

On occasion, as you examine files, you may receive an error message stating that *"This OLESS file has not been de-fragmented – this file may fail to parse as expected or parsing will fail."* If so, click *Tools*, then *Defragment* followed by Parse again.

You can see that OffVis has identified the sample file as *Definitely Malicious,* having found a UDDESC record prior to a *BoundSheet* record at offset 17735 as pertinent to CVE-2008-0081 discussed above. I double-clicked the *Definitely Malicious* entry and the hex view highlighted `DF 00 01 00 EB`. A *UDDESC* record is a description string for chart autoformat and is identified in hex as `0xDF`. The BoundSheet record contains sheet information and is identified in hex as `0x85`.[13] Simply, if the parser spots `0xDF` before `0x85`, it considers the parsed Excel document as malicious. When I clicked *Edit*, then *Find*, and searched for `85` with the hex radio button selected, the UI hex view reset to address `0000FAC0` with a highlighted hex result of `85 00 00 00`, after the BoundSheet recorded (DFh) found in address `00004540`. Long live the parser.

You can dive into the complexities and nuances of the file format to satisfy your inner geek, or you can simply accept the parser's results: *Definitely Malicious.*

To your CEO, who asked you to check out the file for her (think royal food taster), you can say "Ma'am, I wouldn't open that spreadsheet if I were you."

In conclusion

Download OffVis and make good use of it – another fine tool for the defender's arsenal. The future of the tool is unclear; perhaps a command line version is in the making, but no guarantees. Should you find malicious samples exploiting vulnerabilities that are not detected by OffVis, please let MSRC Engineering know via their contact[14] page so that they can consider adding detection too for additional vulnerabilities. Their hope is to keep the correct balance between giving defenders more information to help them detect attacks, and keeping vulnerabilities away from attackers. Keep an eye on the SRD blog for updates and feel free to provide additional feedback via the blog as well.

Cheers…until next month.

Acknowledgments

—Dan Beenfeldt, MSRC Engineering

—Kevin Brown, MSRC Engineering

—Bruce Dang, MSRC Engineering

—Robert Hensing, MSRC Engineering

—Jonathan Ness, MSRC Engineering

About the Author

Russ McRee, GCIH, GPEN, GCFA, CISSP, is a security analyst on the Security Incident Management team for Microsoft's Online Services. As an advocate of a holistic approach to information security, Russ' website is holisticinfosec.org. Contact him at russ@holisticinfosec.org..

13 MICROSOFT OFFICE EXCEL 97-2007 BINARY FILE FORMAT SPECIFICATION, http://download.microsoft.com/download/0/B/E/0BE8BDD7-E5E8-422A-ABFD-4342ED7AD886/Excel97-2007BinaryFileFormat(xls)Specification.pdf.

14 http://blogs.technet.com/srd/contact.aspx.