OFFICE BUGS ON THE RISE
Gabor Szappanos
Sophos, Hungary

gabor.szappanos@sophos.com

ABSTRACT
It has never been easier to attack Office vulnerabilities than it is nowadays. Office exploits have always been high-value assets for criminal groups because Microsoft Office documents are very efficient in delivering their malicious content – users tend to open them without a second thought. This paper will look more deeply into the dramatic changes that have happened in the past 12 months in the Office exploit scene – a scene that has appeared stale in the past couple of years, with only one or two new vulnerabilities appearing each year that made their way to the commercial exploit builders. There has always been a hunger for new exploitable Office vulnerabilities in cybercrime, but the most important builders supported exploits that had been fixed for a couple of years already – which hurt the efficiency of the malware delivery process. 2017 brought a drastic change in many respects. The number of widely used exploits multiplied compared to the previous five years. More importantly, the new exploits turned out to be much simpler. The previous major vulnerabilities were complex memory corruption vulnerabilities, and working with them required a deep knowledge of document file formats and an advanced understanding of the concepts of exploitation. Last year’s new vulnerabilities, on the other hand, were much simpler logic bugs (CVE-2017-0199, CVE-2017-8759) or very simple classic stack overflows (CVE-2017-11882, CVE-2018-0802) – easier to understand and more robust to detection evasion tweaking.

Creating builders for these exploits is no longer the privilege of skilled hackers – average programming skills are now sufficient. As a result, we have seen a lot of these builders showing up on GitHub, free for the taking. This triggered a decline in the usage of commercial exploit builders: their usual customers switched to the free offerings. In this paper we will look at this transition, and at the efforts of the commercial exploit builder developers to keep up with the changing trends. The easy availability of these builders enabled many cybercrime actors to use the exploits with little to no investment, resulting in the large number of Office exploit-related attacks seen in the past 12 months.

The life cycle of an Office exploit starts with initial zero-day targeted attacks, then at some point a few well-resourced cybercrime groups start using it. Later, the exploit ends up in builders, which leads to an explosion of its use by many groups, hitting the general user population.

This cycle usually takes a few months, as we have observed with many exploits in the past few years. However, last year, driven by the great demand for fresh Office exploits, the cycle was cut down to just weeks.

This paper will reconstruct the timeline of one of the hottest Office exploits (CVE-2017-0199) that featured the following typical scenarios in its life cycle:

- Zero-day APT activities.
- Enthusiastic security researchers playing with the exploit.
- APT groups experimenting with bypassing virus scanners.
- The appearance of exploit builders (both commercial and free).
- The explosion of the usage of the exploit in cybercrime.

INTRODUCTION
2018 brought a dramatic change in the usage of document exploits. The old legacy exploits that had been so popular in the previous couple of years became obsolete and were replaced with the emerging exploits of 2017 and 2018. In our research we investigated the malware attacks that used Microsoft Office exploits in the first quarter (Q1) of 2018.

The key findings are the following:

- New vulnerabilities from 2017/2018 completely replaced the old ones: 96% of the attacks were carried out using vulnerabilities that were no more than a year old.
- This was powered by the emergence of a new generation of exploit builders: three new exploit builders were responsible for 75% of the attacks, while the older tools were completely abandoned.
- Over 90% of the attacks used Rich Text Format documents because of the powerful obfuscation methods it enables.
- Criminal groups who previously had no interest in Office exploits started to use them in their distribution campaigns, adding previously unseen malware families (most notably Trickbot) to this specific threat scene.
- New exploits were utilized with a shorter turnover time, usually within weeks of discovery.

DOCUMENT EXPLOIT STATS
Figure 1 shows the overall distribution of vulnerabilities in the 2018 Q1 malware campaigns.

In a number of cases, the criminals used samples with multiple exploits within the same file; in these cases, each of the vulnerabilities was accounted for in the final stats.
The most prominent exploit in use was one targeting the recent Equation Editor vulnerability (CVE-2017-11882) – this alone was responsible for over half of the attacks. Combined with the newer CVE-2018-0802 variation of the same kind, attacks on the Equation Editor component were responsible for 80% of all Office exploitation attempts.

The even newer Flash vulnerability (CVE-2018-4878) also made a significant impact, landing in fourth position in the chart, indicating that fresh vulnerabilities quickly make their way through the ecosystem.

It is worth mentioning that the high prevalence of the two previously mentioned 2018 vulnerabilities is mainly attributable to the Threadkit builder, which uses multiple new vulnerabilities.

However, analysing the latest attacks indicates that some exploits don’t stick around: shortly after the Flash bug was added to the kit, we saw it used in many attacks, but as time went on, the exploit was dropped from the malicious samples.

A similar thing seemed to happen with the CVE-2018-8174 Internet Explorer exploit: it was added to Threadkit, used in a few instances, but not utilized thereafter.

It was interesting to observe that, after dominating the second half of 2017, the CVE-2017-0199 vulnerability fell off the charts. This was the vulnerability that last year broke the four-year dominance of the infamous CVE-2012-0158 vulnerability that just didn’t seem to want to go away [1]. Not more than six months later, it had joined the other obsolete old bugs down the plughole.

This proves that criminals are quick to turn their attention to newer vulnerabilities if they are given a chance. By ‘chance’, we mean a new exploit builder.

We have observed that a complete shift can happen in the usage of vulnerabilities, and within a very short time frame – only a couple of months are needed.

**EXPLOIT BUILDERS**

It is no secret that cybercrime groups prefer to use exploit builders rather than creating the malicious files themselves. The impact of the builders in 2018 was clear. We have seen the offspring of at least four exploit builders, with the documents generated by them responsible for 75% of all the attacks.

The most active of the builders was Threadkit, which alone was responsible for one third of all incidents.

It is important to note that the old exploit builders – like Microsoft Word Intruder and AKBuilder, using the older Microsoft Office vulnerabilities – were completely absent from the attacks.

In the following sections we summarize the characteristics of the samples and campaigns related to the exploit builders featured in our stats. In most cases we have not (yet) been able to identify the builder itself, we have merely observed the effects of its utilization.

**Threadkit**

Threadkit is a commercial product implemented in Python, sold on Russian-speaking underground forums. The typical price is 800 USD for the licence; 400 USD for an update [2].

Threadkit supports all of the recent vulnerabilities:

- CVE-2017-0199
- CVE-2017-8750
- CVE-2017-11882
- CVE-2018-0802
- CVE-2018-4878
- CVE-2018-8174

The generated samples are RTF documents that contain multiple exploit blocks, each exploiting one of the above-listed vulnerabilities.

A typical exploited document has multiple consecutive exploit blocks that independently trigger the batch installer that finally executes the Win32 payload, which is also embedded in the malicious document.
In the incidents observed in 2018 Q1, all of the samples were droppers, as described above. Recently, we have started to see downloaders, where the embedded executable is missing, and a short PowerShell script is triggered that downloads the Win32 payload from an external website.

The malware families distributed in the Threadkit-related incidents (weighted by the number of reports) are shown in Figure 4.

![Figure 4: Payload delivered in Threadkit-powered attacks.](image)

The distributed payload had a couple of newcomers. Threadkit was the only builder to deliver Trickbot (in a couple of very intensive campaigns), Ursnif and Quant loader – malware families that we had not seen associated with Office exploits before. Apparently, cybercrime groups that had previously used other methods to distribute their payload were now purchasing this exploit builder and starting to use it for their campaigns.

Other than that, the usual low-end cybercrime gangs utilized Threadkit to deliver their usual malware payloads (e.g. Lokibot, Betabot).

**EQN_kit1**

This builder generates RTF documents to exploit the CVE-2017-11882 vulnerability. Junk keywords are inserted at certain positions in the RTF file.

In the example shown in Figure 5, tags like this are the junk content:

```
\PTBWFFWPJMMFZERXAKDDUXJVEFWSAJRXMBQIPJ\NRPFZSTTVLG
CVXYM0QFOQBEJAMA\RJKBMD0FL0J2CMAXK9I\ADEDFXQRTJS9P
DZENFNYL021ALHQHQLDQML8JADYHGAK3YSBDOHQX68RT
KRAUYVCCZYPXSLMVQPVZM1P5AIH0GF9FH0Q2FCUWPFDCS
```

These tags have no role in the exploitation process; in fact, the RTF parser in *Word* ignores them when opening the files.

The payloads delivered by EQN_kit1 are shown in Figure 6.

![Figure 6: Payloads delivered by EQN_kit1.](image)

Only four malware families were observed as being delivered by this kit, and those are the usual low-end cybercrime trojans. It looks like this is not a commercial tool but a custom solution (developed by modifying one of the many free builders) used by only one or a few closely connected criminal groups.

**EQN_kit2**

This kit generates samples that exploit the CVE-2017-11882 vulnerability. The generated samples are usually *Word* RTF documents or *Excel* XLSX workbooks, but we have seen PDF files with embedded RTF as well. It is much more sophisticated than the previous builder, and very likely a commercial tool. A recent case involving this builder was documented in [3].

```
\%generator Riched2 6.3.9600\%v4.49.66.1000\%v4.49.66.1000\%v4.49.66.1000\%v4.49.66.1000\%v4.49.66.1000\%v4.49.66.1000\%v4.49.66.1000\%v4.49.66.1000\%v4.49.66.1000\%v4.49.66.1000\%v4.49.66.1000\%v4.49.66.1000\%v4.49.66.1000\%v4.49.66.1000\%v4.49.66.1000\%v4.49.66.1000\%v4.49.66.1000\%v4.49.66.1000\%v4.49.66.1000\%v4.49.66.1000\%v4.49.66.1000\%v4.49.66.1000\%v4.49.66.1000\%v4.49.66.1000\%v4.49.66.1000\%v4.49.66.1000\%v4.49.66.1000\%v4.49.66.1000\%v4.49.66.1000\%v4.49.66.1000\%v4.49.66.1000\%v4.49.66.1000\%v4.49.66.1000\%v4.49.66.1000\%v4.49.66.1000\%v4.49.66.1000\%v4.49.66.1000\%v4.49.66.1000\%v4.49.66.1000\%v4.49.66.1000\%v4.49.66.1000\%v4.49.66.1000\%v4.49.66.1000\%v4.49.66.1000\%v4.49.66.1000\%v4.49.66.1000\%v4.49.66.1000\%v4.49.66.1000\%v4.49.66.1000\%v4.49.66.1000\%v4.49.66.1000\%v4.49.66.1000\%v4.49.66.1000\%v4.49.66.1000\%v4.49.66.1000\%v4.49.66.1000\%v4.49.66.1000\%v4.49.66.1000\%v4.49.66.1000\%v4.49.66.1000\%v4.49.66.1000\%v4.49.66.1000\%v4.49.66.1000\%v4.49.66.1000\%v4.49.66.1000\%v4.49.66.1000\%v4.49.66.1000\%v4.49.66.1000\%v4.49.66.1000\%v4.49.66.1000\%v4.49.66.1000\%v4.49.66.1000\%v4.49.66.1000\%v4.49.66.1000\%v4.49.66.1000\%v4.49.66.1000\%v4.49.66.1000\%v4.49.66.1000\%v4.49.66.1000\%v4.49.66.1000\%v4.49.66.1000\%v4.49.66.1000\%v4.49.66.1000\%v4.49.66.1000\%v4.49.66.1000\%v4.49.66.1000\%v4.49.66.1000\%v4.49.66.1000\%v4.49.66.1000\%v4.49.66.1000\%v4.49.66.1000\%v4.49.66.1000
```

Figure 5: EQN_kit1-generated document.
The samples exploit the vulnerability in a very peculiar way. Usually, the malicious samples targeting this vulnerability have an embedded Equation Editor object, which is stored as an embedded Equation Editor stream. The samples generated by EQN_kit2 are different: they contain only an Ole10Native stream and the CLSID for the Equation Editor object.

This alone is enough for Microsoft Word to handle the embedded object and trigger the vulnerability. The stream contains the exploit trigger, followed by a very short redirector code (which points to the second-stage shellcode), and finally an address to a location in EQNEDT32.EXE (ROP address) that contains a RET instruction. This RET instruction is the first to execute after the exploit is triggered and continues the execution on the first-stage redirector code.

The polymorphic redirector code calculates the memory address of the second stage in one of the registers and jumps there. But the calculation of the memory address varies from sample to sample. In one of the samples the values might be set by a combination of MOV and ADD, as shown in Figure 8.

In another sample it is achieved by a combination of MOV and XOR, as shown in Figure 9.

In other samples OR and SUB instructions were also used to perform the same task. Additionally, the address of the RET instruction varies from sample to sample – after all, EQNEDT32.EXE contains a lot of RET instructions to choose from.

The second-stage shellcode is protected by a highly polymorphic decryptor layer, which performs a four-byte XOR decryption. There are a lot of junk redirections to make the code analysis difficult.

The decrypted final code is a downloader that gets the Win32 payload from an external website and executes it.

The malware families distributed by the samples generated with this kit are shown in Figure 11.
The families are the typical tools used by the Nigerian BEC scammers, who are the typical customers of this kit. Agent Tesla, Lokibot and Fareit were long-time favourites of these groups, while Formbook has recently been added to their toolkit.

**EQN_kit3**

Only a handful of malicious documents were seen belonging to this group, which is responsible for only 1% of the attacks. The small number of samples and incidents does not give us sufficient data to produce reliable stats.

The samples use the same exploit implementation as Metasploit, but the embedded object is obfuscated by embedding the data bytes in do-nothing *par* tags, as shown in Figure 12.

For example, the nibble 0 is represented as \par574 0. The RTF parser in Word ignores everything but the 0 value. Thus, the following RTF fragment

```
{par574 0}{par603 1}{par736 0}{par943 5}{par778 0}
```

will be simplified to the three-byte sequence 010500 (which denotes the header of the embedded OLE object).

**Other builders**

There are many other exploit builders available for the new Office exploits. This section describes a handful of them. Some of them may be connected to the builders listed in the previous sections, but there is no conclusive proof of that.

**Embedi**

The mother of all CVE-2017-11882 builders was the builder published by Embedi on GitHub [4] just a week after the initial Microsoft Security Bulletin [5]. This security company was the first to report the vulnerability and publish detailed information about it, along with a proof-of-concept builder (see Figure 13).

(On a totally unrelated note, in an interesting twist, the US Department of Treasury blocked the properties of Embedi for having provided material and technical support to Russia’s Federal Security Service (FSB) [6].)

The builder is a Python script that assembles the exploited documents from the hard-coded header, trailer and exploit segments:

```
RTF_HEADER = R'''\rtf1\ansi\ansicpg1252\def0\nouicompat\deflang1033\{\fonttbl\{f0\fnil\fcharset0Calibri;\\}\}\generator Riched20 6.3.9600\viewkind4\uc1\pard\sa200\sl276\slmult1\f0\fs22\lang9'''

RTF_TRAILER = R'''\par}'''

OBJECT_HEADER = R'''\object\objemb\objupdate\{\*\objclass Equation.3}\objw380\objh260\{\*\objdata}

OBJECT_TRAILER = R'''\result\{\picprop\wmetafile8\picw380\pich260\picwgoal380\pichgoal260

0100090000039e0000000200
1c000000000000000000000000000000000000000000000000000000000000

00000102fffffff000000002e0118000000050000000000000b0200
0000000000000000000000000000000000000000000000000000000000000

1200000260602001a00ffffff000010000000000c0fffffff6
ffffffff202000066010000000000

```

Figure 13: Proof-of-concept builder by Embedi.
The builder itself was republished several times, and subsequent builders followed the same logic and even borrowed large chunks of code from it. This proof-of-concept code inspired many of the later released builders.

**NebulaOne**

This builder was promoted and distributed (for free) on hacking forums. Figure 14 shows an advertisement.

The Nebula builder is a .Net application, but it only serves as a user interface. The core of the builder is the exploit module, which targets CVE-2017-11882.

![Figure 14: NebulaOne advertisement.](image)

The exploit module itself is a standalone Python script, stored as a separate file in the /bin directory.

This Python script is very similar to the original proof-of-concept code released by Embedi. It uses an earlier implementation of the exploit that was limited to an at most 43-character-long command line. The other builders discussed here overcome this limitation with an improved implementation.

**Omree**

This is a Python script compiled into a standalone executable for easier portability.

![Figure 16: Omree usage.](image)

The malicious documents generated by this kit match the characteristics of the EQN_kit1 samples except for the junk comments.

However, the object reference is slightly different, using

```latex
\{\text{object}\text{	extbackslash objemb}\text{	extbackslash objupdate}\{\textbackslash\textbackslash\textbackslash Equation.3}\}
```

instead of

```latex
\{\text{object}\text{	extbackslash objemb}\text{	extbackslash objupdate}\{\textbackslash\textbackslash objectclass Equation.3}\}
```

**Anony_sec**

This builder was published on GitHub and described in a Chinese forum [7]. We found several thousand malicious documents generated by this builder – it is very actively used (see Figure 17).
This builder matches the EQN_kit1 samples most closely, but there are no random comments inserted. Still, EQN_kit1 is the most likely origin, with someone adding the random junk comment feature to the Python script.

Elm0d

A typical example of the current ‘commercial’ exploit builders available on the scene is the Elm0d (a.k.a Elmod) builder mentioned in [8].

Its pricing structure places it in the high-end market, with a yearly subscription rate of 450 USD [9].

The builder itself support multiple exploits, including most of the recent Office vulnerabilities. Unlike Threadkit, the documents generated by this builder will only contain a single vulnerability, selected during generation. Figure 19 shows the vulnerability selection process.

The higher price tag and the multiple selection of fresh vulnerabilities would indicate that there is some serious development effort behind the builder.

However, on looking behind the scenes (see Figure 20) we can see that this assumption is not correct. The modules that implement the individual exploits are stored as resources inside the executable. Taking a closer look reveals that the exploit modules for the Office vulnerabilities are nothing other than the freely available builders taken from GitHub.

Despite its fancy user interface, this builder is merely a pricey front end built around the free solutions.

TIMELINE OF AN EXPLOIT

We mentioned earlier in this paper that the new exploits follow an accelerated timeline compared to the vulnerabilities we had become accustomed to seeing in previous years. In this section we explain this observation in detail.

Microsoft Office exploits usually follow the same path – they go through a couple of stages in their life cycle, as illustrated in Figure 21.

The following stages are usual for exploits that end up being used in the wild:

- The exploit is used in limited-distribution early APT attacks.
- At some point the vulnerability is discovered and a patch is released.
- The exploit slowly exfiltrates into further targeted attacks.
At some point a sample becomes available to the security and criminal community.
Within a few weeks, a Metasploit module is released.
Within a couple of months, commercial exploit builders release support for the exploit.
At this point the exploit is available for the cybercrime groups who start massive infection campaigns.

In this classical scheme there is an approximate one-to-two-month window between a patch for the vulnerability being made available and the mass-distribution of the exploit by cybercrime groups. This allows enough time for defences to be prepared and for fixes to be deployed throughout organizations.

However, with the recent Office exploits we have observed an accelerated timeline that changes the nature of the game.

TIMELINE OF CVE-2017-0199
As an example, we take the most popular vulnerability of 2017, CVE-2017-0199. In this case we were able to reconstruct all stages of the life cycle. The other vulnerabilities should follow the same path.

The main events related to the exploit are summarized in Table 1.

<table>
<thead>
<tr>
<th>Date</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>23/11/2016</td>
<td>First known sample of the exploit</td>
</tr>
<tr>
<td>07/04/2017</td>
<td>McAfee releases report about zero-day samples [10]</td>
</tr>
<tr>
<td>08/04/2017</td>
<td>FireEye first blogs about the exploit [11]</td>
</tr>
<tr>
<td>10/04/2017</td>
<td>Massive Dridex distribution</td>
</tr>
<tr>
<td>10/04/2017</td>
<td>Proofpoint releases report with first hashes [12]</td>
</tr>
<tr>
<td>11/04/2017</td>
<td>Microsoft releases the patch [13]</td>
</tr>
<tr>
<td>11/04/2017</td>
<td>FireEye releases full report [14]</td>
</tr>
<tr>
<td>12/04/2017</td>
<td>AV evasion experiments start</td>
</tr>
<tr>
<td>14/04/2017</td>
<td>Metasploit module released</td>
</tr>
<tr>
<td>18/04/2017</td>
<td>Builder 1 released (based on Metasploit)</td>
</tr>
<tr>
<td>24/04/2017</td>
<td>Builder 2 released (based on Dridex)</td>
</tr>
<tr>
<td>08/05/2017</td>
<td>MWI support released [15]</td>
</tr>
<tr>
<td>19/06/2017</td>
<td>Builder 3 first known sample (based on Builder 1)</td>
</tr>
</tbody>
</table>

This vulnerability has been used for months in targeted attacks. Most of the activity went on in March and April 2017, but the earliest sample that we could locate dated back to November 2016.

The vulnerability was first mentioned in a McAfee blog post talking about a recently analysed sample exploiting an unidentified zero-day Office vulnerability [10]. This forced FireEye researchers to come out with a follow-up post, revealing the fact that they had been working with Microsoft on this vulnerability [11] for some time. These two reports triggered wide media coverage and boosted general interest in the exploit.
At this point, most security researchers and virus labs had no reliable information about the exploit, let alone any samples. Yet somehow, the criminals behind the Dridex distribution campaigns found a working sample of the exploit and started using it for malware distribution, all within a day. They were able to react quickly because they were reusing existing distribution mechanisms, replacing only the first-stage downloader with the new exploit.

The large volume of exploited Dridex loader samples made it possible for security researchers to obtain samples, analyse them and publish reports. The first one was by Proofpoint researchers [12], who were the first to publish sample hashes. This amount of exposure forced Microsoft to release a patch earlier than planned [13], after which FireEye published a report [14] containing full details of the exploit. At this point, information about the exploit was available in the public domain, and not surprisingly, experiments soon began.

Within a week a Metasploit module had been released, after which a series of free and commercial builders surfaced.

The timeline features a couple of unusual events, which are highlighted in Table 1.

First, massive cybercrime campaigns started while the exploit was still in zero-day stage. Second, the exploit builders appeared within a couple of weeks of the release of the patch. As a result of the accelerated timeline, this exploit was already dominating the scene just two weeks after its initial public appearance, with over three quarters of all document exploit attacks using this new vulnerability.

After completing its downloads, it displays a decoy that looks like it comes from a textbook for the military forces in Donetsk People’s Republic.

The payload was the commercial spyware program FinSpy [14].

Cybercrime

Soon after the initial exposure, an explosion of samples turned up, all related to cybercrime activities. It took a very short time for cybercriminals to jump on the opportunity and integrate the exploit into their malware distributions [5].

It is extremely rare for cybercriminals to manage to integrate an exploit while the vulnerability is still unpatched, but it happened in this case, with a handful of samples that were distributing the Dridex banking trojan.

Dridex

The first cybercrime campaigns started in the zero-day stage, on day before the Microsoft patch was released. Distributed in...
email messages, the exploited documents delivered the Dridex banking trojan.

Hash: 3770051d8cb7df081b5409f2be3b86dc916a2755
Original name: Scan_45807.pdf
First seen: 10/04/2017
Downloads hxxp://rottastics36w[.]net/template.doc

This sample was distributed in an unsophisticated form in email messages with hardly any content, as shown in Figure 24.

Hash: c10b1c9a34d3d9a720aacecd55f704fc42e1267
Original name: uk_confirmation_ph88704796.pdf
First seen: 11/04/2017
Activity:
Downloads hxxp://hyoeyeep[.]ws/template.doc; probably downloads hxxp://hyoeyeep[.]ws/sp.exe

This sample was distributed in large volumes in email messages, mostly in Australia. The messages were disguised as scanned images, and in some cases even the message date was faked in the header to date back to 2014, as can be seen in Figure 25.

---

The AV evasion game

As information about the exploit become widespread, and the related samples became widely available (the latter mostly due to the massive Dridex distributions), security researchers and criminals started to experiment with it in an attempt to understand the exploit and find out how to evade detection by anti-virus programs. This generated a lot of test files from different sources. The following sections detail two typical examples.

**Player 1: White hat researcher (?)**

These samples were submitted to VirusTotal from China by the same submitter. The samples were derived from 04a2977b0307834806214df219636711352b67c7 (Dridex downloader) by manually editing the RTF file in multiple points and eventually breaking the download URL. The original URL was hxxp://hyoeyeep[.]ws/template.doc, the changes are highlighted in the following list. All of the samples were submitted on 13 April, two days after the availability of the original sample:

Hash: 2c9f7fcf7765890d324eb373d601167cfa0b09be
Downloads hxxp://hyoeyeep[.]ws/template.doc

Hash: 064709d96ab4139e0c255ed06af147d8835637abd
Downloads hxxp://hyoeyeep[.]ws/template.doc

Hash: 0c20ffcc43b9b3896d7eaa0099ce5442af1aa177f8
Downloads hxxp://hyoeyeep.ws/template.doc

**Player 2: Chinese APT(?)**

These samples were submitted to VirusTotal from Vietnam by the same submitter.

The samples were derived from the Dridex downloade (as one of the used file names suggests from the one with SHA256 value ae48d23e39b4619881b5c4dd7212b8f48b6bebeoc1e6747995ba68100e), but with more modifications than Player 1, who only changed a couple of bytes in the embedded object. In this case larger (though insignificant) portions of the RTF file were modified.

Hash: 660f52c8d1db7d70a0b2baacb77f84da693b09
Original name: simpleize.rtf
First seen: 12/04/2017

This is the same as the original Dridex sample, with some of the decoy content removed.

Hash: 20978bco3f08c3bb7850e8ec6c520449ad96db28
Original name: goc2.rtf
First seen: 13/04/2017

Downloads hxxp://hyoeyeep.ws/template.doc

Then there were a series of samples from the same submitter that all had the download URL set to hxxp://127.0.0.1/s/template.doc, a clear indication of being a test sample:

Hash: 5ad786f8835bc5e29339e12fb0a69ff589e85e1
Original name: ae48d23e39b4619881b5c4dd7212b8f48b6bebeoc1e6747995ba68100e_mod.doc
First seen: 13/04/2017

Hash: 7916bb2eaf2a2fcb900dbdd9336df72913ad7b1da4
Original name: mod2.rtf
First seen: 13/04/2017

Original name: mod2.doc
First seen: 13/04/2017

Hash: 93ab042b1e1b2ea3b40e88ca182c02f940c804ce
Original name: mod2z.rtf
First seen: 13/04/2017

Hash: c578edec7d2dd0a1a3837dacc66d04792fdca
Original name: mod2.rtf
First seen: 13/04/2017

Hash: eef36fcd60e072997c5a5b6402007f8e2ab45
Original name: mod3.doc
First seen: 13/04/2017

**Figure 25: Dridex delivery message.**
The experiments culminated in the final sample, which was used in real-world attacks, mostly against Vietnamese targets:

**Hash:** c281898ca141104ba791dc146a4402f53814d00d
**Original name:** g-mirror.rtf
**First seen:** 17/04/2017

**Activity:**
Downloads hxxps://g-mirror.appspot[.]com/report.rtf which downloads hxxps://g-mirror.appspot[.]com/favicon.ico;

It drops two components:
- `%PROFILE%\AppData\Roaming\Microsoft\Display Control Panel\DpiScaling.exe` (installer)
- `%PROFILE%\AppData\Roaming\Microsoft\Dynamic COM+\comuid.dll` (main backdoor)

It registers the latter for autostart in HKCU\Software\Microsoft\Windows\CurrentVersion\Run → DpiScaling.

A backup copy of the original dropped component is created in an alternate data stream (ADS) – a rarely used trick that works only on NTFS file systems.

**Figure 26:** Dimoc backup copy stored in ADS.

It also displays a simple decoy document in Vietnamese.

**Figure 27:** Simple Vietnamese decoy content.

The decoy is stored as a resource within the executable file, with the bytes stored in reverse order, as shown in Figure 28.

The installer contains the payload in a similar way, stored with the bytes in reverse order, as shown in Figure 29.

**Figure 28:** Decoy document stored in the resources.

**Figure 29:** Payload ‘encrypted’ by reversing byte order.
The final payload is the Dimoc backdoor that connects to the C&C server at fillin.michellegipps[.]com.

THE EXPLOIT BUILDERS

The next logical step was the appearance of the underground exploit builders, which ignited an explosion of the use of this exploit.

Metasploit

Metasploit Framework support was added for the exploit on 14 April, only four days after the availability of the first sample (Figure 30).

Metasploit is not an underground tool; it is a legitimate commercial product with a free community edition, frequently used by security researchers. However, the disclosure of this module led to the development and release of a builder that was later heavily used by criminal groups.

Builder 1

This builder is a Python script, developed using a Metasploit-generated document as a skeleton template.

The code of this builder was first published on GitHub on 18 April 2017 [16], just four days after the Metasploit module, and is clearly based on a document generated by it.

In fact, the only difference between the two is that the Metasploit-generated document has author info in the header (Microsoft), while Builder 1 has this information removed.

The original Metasploit-generated file looked like this:

```plaintext
{\object\objautlink\objupdate\rsltpict\objw291\objh230\objscalex99\objscaley101
\info
{\author Microsoft}
{\operator Microsoft}
}
{\*\xmlnstbl \{\xmlns1 http://schemas.microsoft.com/office/word/2003/wordml\}}
{\object\objautlink\objupdate\rsltpict\objw291\objh230\objscalex99\objscaley101
\info
{\author }
{\operator }
}
{\*\xmlnstbl \{\xmlns1 http://schemas.microsoft.com/office/word/2003/wordml\}}

Later versions of the builder introduced another feature. The –x option will add obfuscation to the RTF output – random keywords are inserted at several locations, as shown in Figure 31.

Here, the random \*NZOWDLYSVM blocks are inserted into the embedded object, and the download URL is inserted with the \*92a79a58c2a9bae81c59a37d171a0 elements.

There were hundreds of documents generated by this builder within a couple of weeks – we can only provide a couple of examples. The distributed payload is a wide variety of malware, including Dofoil, Remote Utilities and Sennoma.

The following file was probably the first file generated by the builder, surfacing one day after the release of the builder. The sample was generated without obfuscation:

Hash: e310acf0a13351268df24721d1366f696bb4f0ed
Original name: coolxm.rtf
First seen: 19/04/2017
Downloads hxxp://135.84.177.155/svchost.exe.

There were also samples with obfuscation.

Obfuscation was added to the builder on 24 April 2017 (at least that is when the update was uploaded to GitHub), and we started to see these samples immediately after the release.

Hash: a194b24f7017301c4f4d8ab60de0b9d915cdf0
Original name: 2.rtf
First seen: 23/04/2017

The downloaded file is most likely a test document created during the development of the builder to test the new obfuscation feature, because the upload time predates the official release by a day.

Hash: aa194b247017301c4f4d8ab60ede09d915cd0
Original name: First seen: 24/04/2017
Downloads hxxp://5.79.98.106/logo.doc.

The downloaded file is the first document that we could find that used the obfuscation feature of this builder.

**Builder 2**

This builder represents a different development branch. It started with an earlier exploited document (the infamous Dridex downloader, that was already used by Player 2 in the evasion games), as clearly stated in the script itself:

```
#CVE-2017-0199
#create from: https://www.hybrid-analysis.com/sample/
ae48d23e3abf461988b54dd2712b8fadb4d8b6eb0ae167647
995ba68100e?environmentId=100
```

It adds a bit of randomness to the generated documents by inserting random meta info into the RTF header. This results in samples like this one:

```
\rtf1\ansi\ansicpg1252\uc1\adeff31507\def0\stshfdbch31505\stshfich31506\stshfbi31507\def0\stshfich31506\stshfdbch31505\def0\stshfbi31507\deff0\stshfich31506\stshfdbch31505\def0\stshfbi31507\deff0\stshfich31506\stshfdbch31505\def0\stshfbi31507\deff0\stshfich31506\stshfdbch31505\def0\stshfbi31507\deff0\stshfich31506\stshfdbch31505\def0\stshfbi31507\deff0\stshfich31506\stshfdbch31505\def0\stshfbi31507\deff0\stshfich31506\stshfdbch31505\def0\stshfbi31507\deff0\stshfich31506
\info{\author B9bW7MOjGnwWJUJ4}\{\creatim\yr2009\mo10\dy13\hr12\min18}\{\revtim\yr2009\mo10\dy13\hr12\min21}\{\version1}\{\edmins3}\{\nofpages1}\{\nofwords36}\{\nofchars1585}\{\*\company CmpeJRNeQID\r\x\ken\n\nofcharsw1585}\{\vern27079}\}
```

It may also insert a random tag in the middle of the download URL:

```
48007400540070003\{\deftab\defa3b0483281122134805066348220
978309385915184940421561347753045227470181672525
099943469a\}032f02c03f103d0230377002e0300000e0300000e0
031002f0740060703074002e0864806f06f06f000000000000
0000000000000000000000000000000000000000000000000000
```

The first samples generated by this builder started to appear around 24 April 2017. After that we observed hundreds of malicious documents generated by it within a few weeks. The most notable payload distributed by this builder was the Cerber ransomware.

A full distribution site was found when following the download link of the sample with SHA1: ee19337c75a4afdc6b46f1a311a0f923815bf83. This downloaded the second stage from 5.101.5.24/0199/tasks/lxE5Hb/hta.php. The site was open for browsing and a large set of prebuilt documents were found there, as shown in Figure 32.

The site also conveniently stored the original builder and a slightly modified version of it (for no obvious reason), as shown in Figure 33.

The payload in this case was Kasidet (Neutrino bot).

---

Samples:

- Hash: d0756e4b252521bafeab10f4db15505727efd75b
- Original name: Порядок определения размера пени .doc
- First seen: 24/04/2017
- Downloads hxxp://87.120.254.189/BFbGXDVNjwJaGfFg.txt.

This is probably the first sample generated by this builder.

- Hash: 7a4ae8b7fa54d1685c9baf0fac04153a0f873a03
- Original name: coolxm.rtf
- First seen: 27/04/2017
- Downloads hxxp://wowaskopoq.top/1.xls

The downloaded file is not an Excel workbook, as the extension would suggest, but a Windows executable that drops the Cerber ransomware.

Interestingly, this builder was used by groups distributing Cerber and Kasidet. In the past, these groups had showed no interest in using Office exploits for malware distribution. But as opportunity presented itself in the form of a fresh exploit, they did not hesitate to use it.
Builder 3

This builder was found in the open directory on subaat.com, along with a lot of other tools:

<table>
<thead>
<tr>
<th>Index of /files</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parent Directory</td>
</tr>
<tr>
<td>(1) Facebook_3.MP4</td>
</tr>
<tr>
<td>(16)2269_131909991459758_83914995453942002_p.exe</td>
</tr>
<tr>
<td>193d966e6172442d73293395_6790983679419603674_p.exe</td>
</tr>
<tr>
<td>2012.doc</td>
</tr>
<tr>
<td>2013.doc</td>
</tr>
<tr>
<td>2016.doc</td>
</tr>
<tr>
<td>2018hta.htm</td>
</tr>
<tr>
<td>2017.doc</td>
</tr>
<tr>
<td>2pal.exe</td>
</tr>
<tr>
<td>2pal.exe</td>
</tr>
<tr>
<td>3pal.exe</td>
</tr>
<tr>
<td>555.exe</td>
</tr>
<tr>
<td>714.exe</td>
</tr>
<tr>
<td>Action Screen Recorder.rar</td>
</tr>
<tr>
<td>App.APK</td>
</tr>
<tr>
<td>Application.apk</td>
</tr>
<tr>
<td>Army Full Book.pdf</td>
</tr>
<tr>
<td>BSO</td>
</tr>
<tr>
<td>Backdoor.exe</td>
</tr>
<tr>
<td>Client.exe</td>
</tr>
<tr>
<td>CodecxCryptoV2.6.1.rar</td>
</tr>
<tr>
<td>Cry.EXE</td>
</tr>
<tr>
<td>DarkComet_v5.3_special edition.rar</td>
</tr>
<tr>
<td>DarkShadeBat.exe</td>
</tr>
<tr>
<td>Detail.xls</td>
</tr>
<tr>
<td>GhostCrypt.exe</td>
</tr>
<tr>
<td>FOREX.rar</td>
</tr>
<tr>
<td>File.exe</td>
</tr>
<tr>
<td>HP</td>
</tr>
<tr>
<td>IDM Universal Crack.rar</td>
</tr>
<tr>
<td>IDM Universal Web Crack.rar</td>
</tr>
<tr>
<td>KHADDIM-doc</td>
</tr>
<tr>
<td>KMSAuto Net.exe</td>
</tr>
<tr>
<td>Lost@Door_E;ite_v9.1.zip</td>
</tr>
</tbody>
</table>

Figure 34: Repository containing the builder.

This builder appears to have been released by a well-known player, known by the handle kareem.alex1, who was also very active with AKBuilder [17].

Figure 35: Kareem.alex1 is a well-known figure.

Just as in the case of AKBuilder released by the same author, this is a wrapper, Builder 1 is repackaged and protected with the MPress runtime cryptor. The Python script is dropped into the %TEMP% directory and executed with a simple batch file:

cmd /c C:\Python27\python.exe dle.py -M gen -w usx.doc -u 127.0.0.1/test.hta -x 1

CONCLUSIONS

We have seen that the new Office exploits completely replaced the old ones. This is a result of the appearance of a new generation of exploit builders, which are usually available for free in the public domain. Criminal groups simply switched to the new builders.

The easy availability of fresh Office exploits is a great temptation that pushed a handful of high-end cybercrime groups (those behind Trickbot, Kasidet, etc.) to use them in their distribution campaigns, even though in the past they had showed no interest in Office exploits.

We have observed an accelerated timeline for the new Office vulnerabilities. Previously, it took a couple of months for the appearance of the exploit builders and the escalation to cybercrime campaigns. Nowadays it takes only a couple of weeks to reach the same threat level. This forces defenders into shorter reaction times in patch deployment and protection development.

REFERENCES
