SOPHOS

Security made simple.

Exploit This: Evaluating the Exploit Skills of Malware Groups

By **Gabor Szappanos**, Principal Researcher, SophosLabs Hungary

A SophosLabs technical paper - January 2015

Contents

Introduction 2

CVE-2014-1761 exploitation process 4

Analysis of the samples 5

- Cycoomer (HombresMercurio) 5
- Metasploit sample 7
- Inception (Rodagose, Tionas) 7
- Goldsun (MM RAT) 10
- Rerol 12
- Fonten (BlackEnergy2, PhDet, Lancafdo) 13
- MiniDuke 14
- Ixeshe (Etumbot) 14
- Swrort (Zbot) 15
- Plugx 17
- Appat 17
- Farfli 17
- Dyzap 17
- Corkow 18
- Cromptui 18
- Tinba 19

Relations between families 20

Evaluation of families 21

Conclusions 24

References 25

Introduction

It is common belief that APT groups are masters of exploitation. If anyone, they should know everything about the art of exploitation, right? Our research into the real world uses of the CVE-2014-1761 vulnerability shows that this is far from being true.

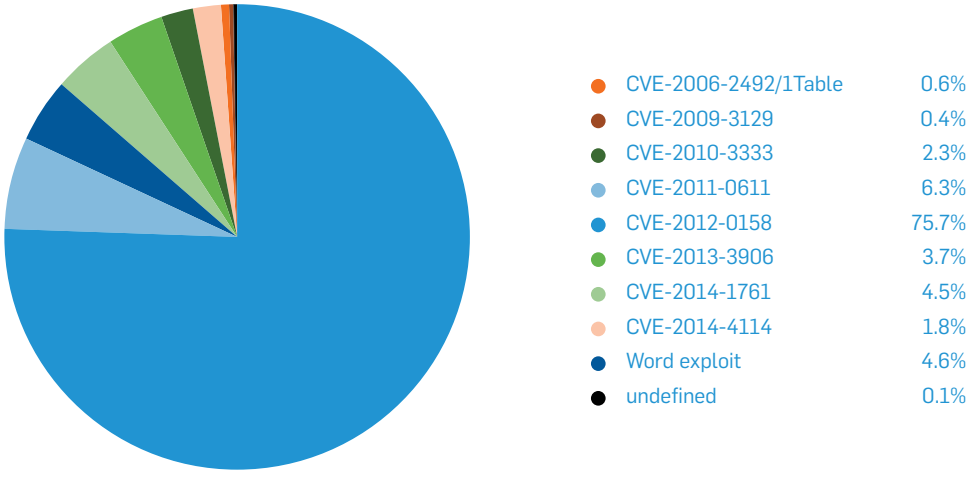
It is a common practice in the anti-malware world that security products are compared to each other in comparative tests. Even the tests themselves can be evaluated in relation to the criteria of the Anti-Malware Testing Standards Organization. The only players who are not rated are the malware authors. This is for a good reason: their activities cover a wide range of operations that don't fully match and can't be exactly measured.

There don't exist general common criteria for rating them, although this information could be useful for the defenders: it is always good to know the strength of the enemy at the other side of the gate.

Our deep analysis of malware samples using the CVE-2014-1761 vulnerability gave us a rare opportunity to compare the skill of a few different malware author groups. This is not a full and comprehensive test; we could estimate the skills only by a single criterion: the attackers' understanding of the exploit. But the situation is the same as with any other test: if you know exactly what you are measuring, you can make valid conclusions. This is what we attempt in this paper.

The vulnerability, reported by Microsoft in April 2014, soon became a standard and popular choice for cyber criminals. Our statistics, gathered during the last three months of 2014 indicates that this is the third most popular document-based exploit.

Exploit Usage



According to the Microsoft security bulletin [4], a wide range of Office versions were affected by this critical memory corruption vulnerability:

- Microsoft Office 2003 Service Pack 3
- Microsoft Office 2007 Service Pack 3
- Microsoft Office 2010 Service Pack 1 (32-bit editions)
- Microsoft Office 2010 Service Pack 2 (32-bit editions)
- Microsoft Office 2010 Service Pack 1 (64-bit editions)
- Microsoft Office 2010 Service Pack 2 (64-bit editions)
- Microsoft Office 2013 (32-bit editions)
- Microsoft Office 2013 (64-bit editions)
- Microsoft Office 2013 RT
- Microsoft Office for Mac 2011
- Microsoft Word Viewer
- Microsoft Office Compatibility Pack Service Pack 3
- Microsoft SharePoint Server 2010 Service Pack 1
- Microsoft SharePoint Server 2010 Service Pack 2
- Microsoft SharePoint Server 2013
- Microsoft Office Web Apps 2010 Service Pack 1
- Microsoft Office Web Apps 2010 Service Pack 2
- Microsoft Office Web Apps 2013

But there is a difference between being vulnerable and being attacked. In practice, despite the popularity of this vulnerability, only one application version of the above list was ever attacked: Microsoft Office 2010 Service Pack 2 (32 bit). All of the malware samples targeted this version only.

In fact, we found that the malware groups have limited understanding of, or ability to modify with success, the initial exploit. Surprisingly, known APT groups showed less sophistication than more mainstream criminal groups. Even so, these groups are able to work with what they have to infect their targets.

CVE-2014-1761 exploitation process

CVE-2014-1761 is a file format vulnerability in the Rich Text Format (RTF) document parsing library of the Microsoft Office suite. The details of it were exhaustively explained in [1] and [2], so we will not discuss it here; only an overview of the exploitation process is provided.

The vulnerability happens when a specially crafted RTF file contains more listoverride structures than Word expects to see and overwrites a memory pointer as a result of the confusion.

At this point the execution takes a detour from the normal path of document opening and the attacker takes control. In order to make the exploit work on newer operating system versions, where Microsoft's Data Execution Prevention is active, the initial stage of the exploitation runs a sequence of code snippets (called a ROP chain) taken from one of the Windows system libraries used by Word, mscomctl.ocx. The vulnerable library is not subject to this restriction, and even more conveniently, also not subject to Address Space Layout Randomization (ASLR). Therefore, the attackers can be sure that it will always be loaded to the same address, and the absolute memory locations of the code snippets will be the same on all systems.

The addresses of the code snippets are combined from several control tags within the listoverride structures. This structure can't hold much data; therefore the attack is multi-staged; only a shorter, bootstrap code chain is stored here.

The initial ROP chain transfers execution to a longer ROP chain and the first stage shellcode, both stored in a leveltext array within the listoverride structure. This array can hold large enough data to perform the necessary steps for the infection process. The first stage ROP chain executes further code snippets from mscomctl.ocx that allocate a new memory block, which already has executable permission (thus getting away from DEP), then copies the first stage shellcode there and executes it.

The first stage shellcode is necessarily short (it has to fit into the constraints of the leveltext array); the entire shellcode functionality can't be stored there. Its purpose is to find and execute the main shellcode that finally decrypts and executes the payload of the exploit; the payload is usually some sort of a Windows Trojan program.

Analysis of the samples

The characteristics of the early samples, using the then zero-day vulnerability have already been detailed in [5].

We will not get into that much detail; instead we focus on categorizing the differences by functionality, and assigning complexity rating to the modifications. This will make it possible to compare the skill level of the malware authors – how much they understand of the vulnerability, and how comfortable they are in making modifications.

We will investigate all samples that surfaced since the announcement of the vulnerability, in order to observe how malware authors were adopting this exploit to their repertoire.

Cycoomer (HombresMercurio)

SHA1:
200f7930de8d44fc2b00516f79033408ca39d610

Documented in [5], this was the first known sample that exploited the vulnerability — the author is unknown. It seems to be the source of all further documents; all of them can be derived from this. There is no evidence that would indicate that, independently from this one, a different root document was known to the malware authors.

This sample serves as a baseline for comparison; it will be referred to as *core sample* throughout the rest of the document.

The document starts with a large chunk, about 6000 bytes, of unused metadata and RTF junk content that play no role in the exploitation:

```
{\rt{{{\{\info{\author ismail - [2010{\n{\info{\
author ismail - [2010]}ofcharsws69}{\operator ismail
- [2010]}{\*
sidtbl
sid8596814
sid8926214
sid10110685}{\leveltext\leveltemplateid67698693'01
\u-3929 ?;}]}info{\revtim\yr{\creatim\yr2014\{\info{\
author ismail - [2010]}mo3\dy8\hr3\min9}2014\m{\
revt{\*\company home}im\yr2014\mo3\dy8\hr3\min9}{\
info{{\revtim\yr2014\mo3\dy8\hr3\min9}\author ismail
- [201{\crea{{\revtim\yr2014\mo3\dy8\hr3\min9}\info{\
author ismail - [2010]}tim\yr2014\mo3\dy8\hr3\min9}0]}
o3\dy8\hr3\min9}{\aut{\nofcha{\info{\author ismail -
[2010]}rsws69}{\operator ismail - [2010]}{{\revtim\
yr2014\mo3{\creatim\yr2014\mo3\dy8\hr3\min9}\dy8\hr3\
min9}\*
…
```

The first stage shellcode locates the host RTF document by enumerating all open handles, and if any of them belongs to an open file handle, checks for two ID strings. The 4 bytes from offset 0 have to be "{\rt" and the 4 bytes at offset 0xf000 have to be "p!11". If both conditions match then it jumps to the second stage code at offset 0xf004. The second stage shellcode decrypts and executes the embedded Win32 payload program and displays a decoy document.

The decoy document is a male dating advertisement, while the payload is a destructive Trojan written in Visual Basic, which will delete files from local and mapped remote drives.

Decoy document displayed by Cycooner

Because of the destructive payload, and the highly unusual decoy, it is most likely that this sample was not used in actual infection campaigns. Perhaps it was deliberately uploaded to VirusTotal, to circulate the exploit.

Metasploit sample

A week after the core document appeared on VirusTotal, a new exploit module was added to the Metasploit Framework [5] that generated RTF files carrying the exploit.

The generated sample is an exact copy of the core document up to the first stage shellcode and ROP chain. So the heading junk, the exploit trigger and the initial ROP chain are exactly the same. The shellcode is variable; whatever is selected in Metasploit is used as a payload.

The ROP chain in the first stage shellcode is slightly modified. Some of the bytes in the ROP chain are not used; they only serve for filling in, in case some of the code fragments have code that pops value (never to be used later) to a register.

The filler dwords are replaced with a random value, as can be observed from the Metasploit module itself:

```
 def exploit
    junk = rand(0xffffff)
    rop_chain = [
      0x275de6ae, # ADD ESP,0C # RETN [MSCOMCTL.ocx]
      junk,
      junk,
      0x27594a2c, # PUSH ECX # POP ESP # AND DWORD
PTR [ESI+64],0FFFFFFFB # POP ESI # POP ECX # RETN
[MSCOMCTL.ocx]
      0x2758b042, # RETN [MSCOMCTL.ocx]
      0x2761bdea, # POP EAX # RETN [MSCOMCTL.ocx]
```

Inception (Rodagose, Tionas)

Non-working sample SHA1:

6c55ebe34222a2f04a8a2a8f354fb5e65aebbc34
d66eb4b6f7037b0a40b4e3c030d8f9bd4f425f28
9358bea376a9733fa763518a09362c891a00a777
e64bb744981e07a4bdd7e22d468be9191eb6f4bb
b70b2d9e4184ccaefe3600738c227a2a984c34a0
073e3789386f99c43711052e22470f60334d41bf
0f302d6a731cbcdcd3ef50ad6718c9afa0fe8991
307b99b88245f1ad5b2bed0711887e95ea3f37b9
910ed6e372e503d5157500029eba960ccd85881f
f769fdae782fe1f96b1194545803fa77ab94ad1d
f83541fbe7a002060cfcb71c59eb70f7c6118632

Working sample SHA1:

2d430f11f9c9c0dc19c0d03fc7a713cd3422a33c
2eaed93b012b266d80460fca4bea917adbeb810e

This campaign was documented in [8].

The initial infection vectors of this campaign were RTF files. Thirteen of them used both CVE-2012-0158 and CVE-2014-1761 vulnerabilities within the same RTF carrier; however in only two of them was the second exploit actually functional. In the rest of them the prepended CVE-2012-0158 block broke the RTF format, and the subsequent CVE-2014-1761 component was not parsed properly — which in the case of a file format vulnerability effectively disables the exploit. Without the prepended first exploit, the CVE-2014-1761 block in most of the cases would be working.

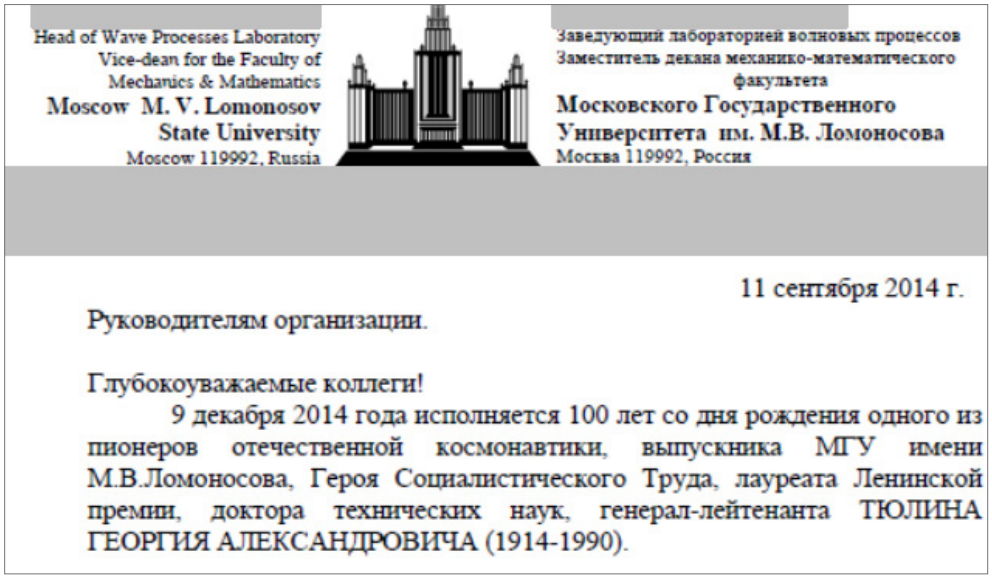
The junk RTF content at the beginning of the block was not modified, only compacted by removing line breaks and unnecessary white space characters.

The RTF components playing a role in the exploitation were not modified, except for the *listoverridecount* control word, which was padded with zeros. Although it looks like a minor change, even this modification broke the exploit in a couple of documents; in these files the CVE-2014-1761 block by itself was non-working.

The ROP chain in the first stage shellcode was slightly modified. The filler dwords were replaced with a random value (which was different in each sample), just like in the case of the Metasploit generated samples. This is a trivial change in the ROP chain.

The first stage shellcode locates the host RTF document by enumerating all open handles, and if any of them belongs to an open file handle, checks for two ID strings. The 4 bytes from offset 0 have to be "[\rt" and the 4 bytes at a second offset variable (the exact location varied in the samples, depending on the preceding CVE-2012-0158 block length) have to be "PT@T". If both conditions match, then it jumps to the second stage code located right after the second ID. The second stage shellcode decrypts and executes the embedded Win32 PE payload program, and drops and opens a decoy document.

The decoy documents usually have themes related to Russia, often some official-looking document:



Head of Wave Processes Laboratory
Vice-dean for the Faculty of
Mechanics & Mathematics
Moscow M. V. Lomonosov
State University
Moscow 119992, Russia

Заведующий лабораторией волновых процессов
Заместитель декана механико-математического
факультета
Московского Государственного
Университета им. М.В. Ломоносова
Москва 119992, Россия

11 сентября 2014 г.

Руководителям организации.

Глубокоуважаемые коллеги!

9 декабря 2014 года исполняется 100 лет со дня рождения одного из пионеров отечественной космонавтики, выпускника МГУ имени М.В.Ломоносова, Героя Социалистического Труда, лауреата Ленинской премии, доктора технических наук, генерал-лейтенанта ТЮЛИНА ГЕОРГИЯ АЛЕКСАНДРОВИЧА (1914-1990).

Or, in a couple of cases, a car advertisement:

The Windows payload is a backdoor DLL, dropped into the Windows system directory, and registered for startup in *HKCU\Software\Microsoft\Windows\CurrentVersion\Run*. The backdoor uses a cloud C&C infrastructure and connects to webdav.cloudme.com [8].

Goldsun (MM RAT)

Sample hashes:
3a6ecfc82ec7bc9917cc4dec94b4673c3dc1444f
317704aa46443573e53509aa6858e62ed262db0c
e2474cc0da5a79af876771217eb81974e73c39e5
7d8a3cf12e8bf272a37ab0f350fec3d5d1090ae8
f3221eb7a1e0c4a8eaa540a53156fdaa35711289
71c17b1983791dd0b9d151e5f369d4cf1a1131ad

Goldsun Trojan variants were reportedly used by the Pitty Tiger group [7]. It is very likely that these samples can also be attributed to the same group.

The samples are exact copies of the core document, including the heading junk, the exploit trigger, the initial ROP chain, the first stage shellcode and ROP chain.

There is one minor difference, in the RTF header: a line break is added and in the metadata the author name is changed from the original *ismail*:

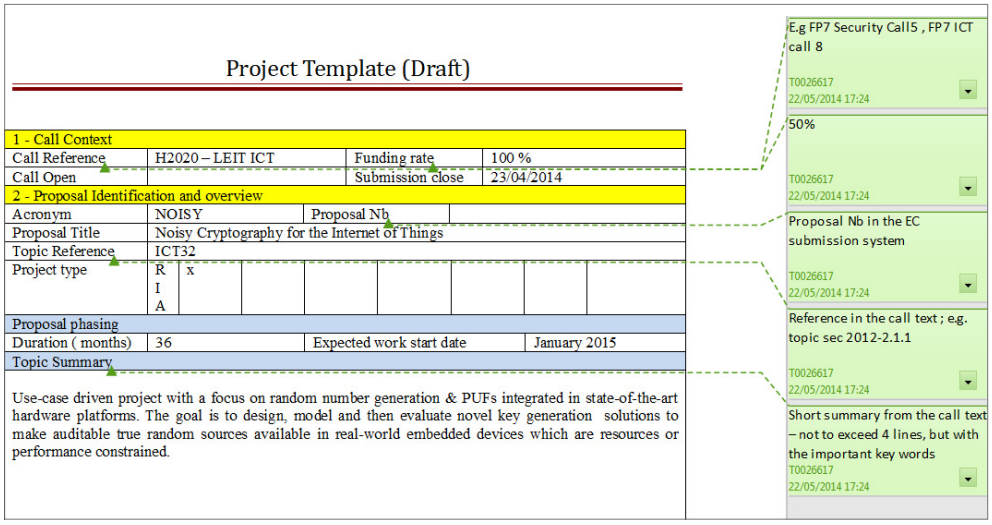
```
{\rt{{{\{\info{\author ismail - [2010{\n{\info{\
author ismail - [2010]}ofcharsws69}{\operator ismail
- [2010]}{\*
sidtbl
sid8596814
sid8926214
sid10110685}{\leveltext\leveltemplateid67698693'01
\u-3929 ?;}]}info{\revtim\yr{\creatim\yr2014\{\info{\
author ismail - [2010]}mo3\dy8\hr3\min9}2014\m{\
revt{\*\company
```

to ismaim (or ismahm or ismaiÿ):

```
{\rt
{{\{\info{\Author ismail - [2010{\n{\info{\author
ismaim - [2010]}ofcharsws69}{\operator ismail -
[2010]}{\*
sidtbl
sid8596814
sid8926214
sid10110685}{\leveltext\leveltemplateid67698693'01
\u-3929 ?;}]}info{\revtim\yr{\creatim\yr2014\{\info{\
author ismaim - [2010]}mo3\dy8\hr3\min9}2014\m{\
revt{\*\company
```

The only significant difference is the appended encrypted payload, which is replaced with the Goldsun Trojan. So this sample is a simple copycat of the core document.

The decoy document is a project template, sometimes blank, sometimes filled with some data; in one case even revision tracking was added to look more realistic:



Project Template (Draft)

1 - Call Context			
Call Reference	H2020 – LEIT ICT	Funding rate	100 %
Call Open		Submission close	23/04/2014
2 - Proposal Identification and overview			
Acronym	NOISY	Proposal Nb	
Proposal Title	Noisy Cryptography for the Internet of Things		
Topic Reference	ICT32		
Project type	R x I A		
Proposal phasing			
Duration (months)	36	Expected work start date	January 2015
Topic Summary			

Use-case driven project with a focus on random number generation & PUFs integrated in state-of-the-art hardware platforms. The goal is to design, model and then evaluate novel key generation solutions to make auditable true random sources available in real-world embedded devices which are resources or performance constrained.

The final payload is a backdoor, dropped into the "Application Data" directory in the user home directory as verclsid.dll, then registered for startup in the registry under *HKCU\Software\Microsoft\Windows NT\CurrentVersion\Winlogon\Shell*.

The list of the C&C servers connected by the different backdoor variants is the following:

bz.kimoo.com.tv
mca.avstore.com.tw
star.yamn.net
182.33.250.60
gmail.pcchipsmarketing.com

Rerol

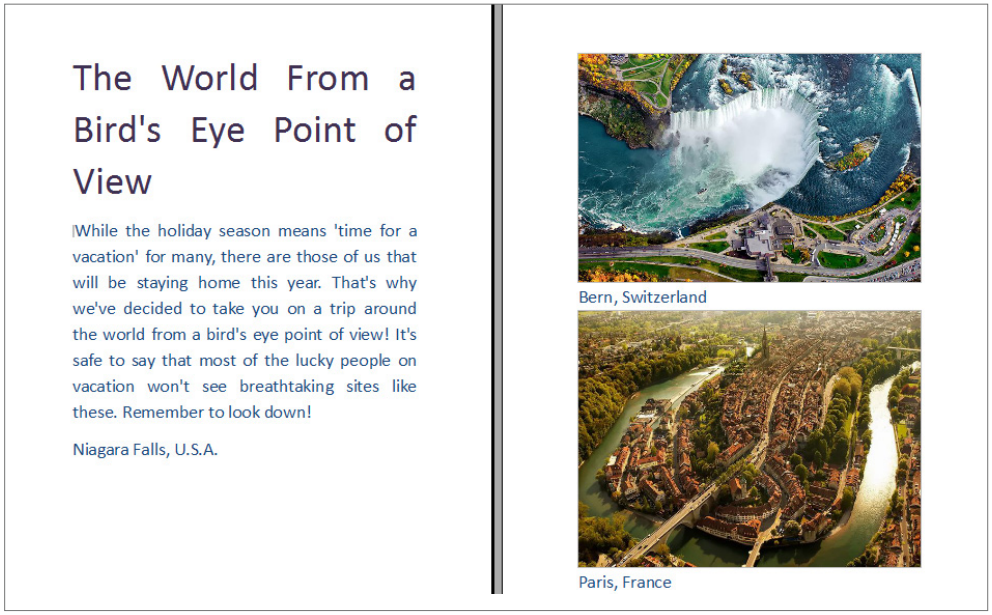
Sample hashes:
58e856675c1e8713486ee1533d1606f238e80ca8

d778d3a7d66f0a80bd198c4808e47b42e6f1e601

Rerol variants were also reported to be used by the Pitty Tiger group [7].

These samples are essentially the same as the Goldsun samples (as allegedly coming from the same group, it is not a surprise). The exploit trigger, initial and first stage ROP chain, first and second stage shellcode are the same (in fact, the RTF documents match byte-to-byte until the second stage shellcode). The appended encrypted payload executable was changed; this required minor modifications in the second stage shellcode: the modified total file length and the appended payload length were modified in the code. Other than that, this is the same as Goldsun.

The decoy document in these cases is a travel advisory:



The final payload is a backdoor, dropped into the "Application Data" directory in the user home directory as svchost.exe, then registered for startup in the registry under HKCU\Software\Microsoft\Windows NT\CurrentVersion\Winlogon\Shell.

The list of the C&C servers connected by the different backdoor variants is the following:

sophos.skypetm.com.tw
mac.avstore.com.tw

Fonten (BlackEnergy2, PhDet, Lancafdo)

SHA1:
befb4ec7992be8cd4c3065f77f88b1c83a4b6a9f
3c734737e0af6f163c7d721ef81920e958fef3d7

These samples were documented in [5].

In these samples the heading RTF junk and metadata is completely removed, the RTF structure is stripped to the minimum, and only the parts essential for the exploit remained.

But nothing else was changed in the *listoverrode* structure definitions. Although it would not even qualify as a trivial modification of the exploit condition (the parts modifying the exploit behavior were not even touched), the authors clearly understood some of the exploitation — they at least realized what the necessary components are.

Also the *\listoverridecount* was changed from 25 to 26, which is halfway between a non-trivial and trivial change in exploit condition; it effectively modifies one of the parameters essential for the exploit, within its region of indifference. Given this level of understanding, and the early appearance of the sample, one can speculate that the authors of this malware could be close to the authors of the core documents. But this is just speculation, without hard evidence.

The first stage shellcode locates the host RTF document by enumerating all open handles, and if any of them belongs to an open file handle, checks for two ID strings. The 4 bytes from offset 0 have to be *"{\rt"* and the 4 bytes at offset 0x4000 have to be "OO*" plus a terminating 0 byte . If both conditions are match then jumps to the second stage code at offset 0x4004. The second stage shellcode decrypts and executes the embedded Win32 PE payload program and displays a decoy document.

The decoy documents are diplomatic related documents such as the following:

Russian ambassadors: 'next we'll take Catalonia, Venice, Scotland and Alaska'

Unauthenticated, expletive-laden recording of pair joking about which countries to annex after Crimea is leaked online

- Shaun Walker in Moscow
- theguardian.com, Friday 4 April 2014 07.13 EDT

Link to video: 'Russian ambassadors' plan world domination: audio recording of leaked phonecall

A recording has surfaced online purporting to be a leaked conversation between two Russian ambassadors discussing which parts of the world they would like to annex after Crimea.

The five-minute recording, laden with expletives, has been posted on YouTube and claims to be a telephone call between Igor Chubarov, Russia's

The payload is dropped to *"Local Settings\Application Data\FONTCACHE.DAT"* in the user home directory, and started by a shortcut file created in the startup directory: *%STARTMENU%\Programs\Startup\{EA894B35-29CB-48DE-B50F-70680C963CBF}.lnk*

MiniDuke

SHA1:
58be4918df7fbf1e12de1a31d4f622e570a81b93

This sample was documented in [5].

The exploit part of the sample is almost exactly the same as the Fonten samples; the only difference is that *\listoverridecount* has the original value of 25.

The initial and first stage ROP chains are the same as in the core sample. One minor change was committed in the first stage ROP chain: the absolute address of the allocated memory block is changed from *0x40000000* to *0x50000000*.

The first stage shellcode locates the host RTF document by enumerating all open handles, and if any of them belongs to an open file handle, checks for two ID strings. The 4 bytes from offset 0 have to be *"{\rt"* and the 4 bytes at offset 0xf000 have to be *'Q"33'*. If both conditions match then jumps to the second stage code at offset 0xf004. The second stage shellcode decrypts and executes the embedded Win32 PE payload program. This time no decoy document is dropped.

The payload is dropped to the user home directory as *ntuser.dat:init* - an Alternate Data Stream (available only on NTFS file systems) attached to the existing and innocent ntuser.dat file, then registered for startup in the registry under *HKCU\Software\Classes\Drive\shell\Open\command*.

Ixeshe (Etumbot)

SHA1:
df6ce42457a7ffa2a08048190c600a51e5a682b4

This sample was documented in [5].

The sample is exactly the same as the core sample up till the first stage ROP and shellcode. The heading RTF junk, the exploit condition and the initial ROP chain are not modified.

The first stage ROP chain was trivially changed; the filler dwords were changed from *0x41414141* to *0x3d27d2d2*.

The big change comes afterwards. The second stage shellcode is not stored as appended binary data after the RTF structure; rather it is in the *leveltext* data of the subsequent *listoverride* structure. The first and second stage shellcode buffers are thus stored in the memory in subsequent memory regions.

This is a major change in the shellcode and also shows some understanding of the memory layout during the exploitation.

There is one problem though. When the first stage shellcode tries to locate the second stage, it starts scanning for an ID dword starting from offset 0x1000 in the allocated memory. However, the two shellcode buffers are only 0x200 bytes apart in all of the replication environments we tested; the scanning will start well after the end of the second stage shellcode — it will never be found, and the scanning will eventually reach an inaccessible memory address and crash.

While this is a unique approach, showing some skills, the fact that it is not working casts shadow on the accomplishment.

I am still hesitant to claim that this is a total failure. The alignment of the *leveltext* buffers in the memory is external to the shellcode and the exploit process. In theory there could be a Windows/Office combination, where these buffers are at least 0x1000 bytes apart. In practice, we couldn't find this combination, but the possibility still remains.

Swrort (Zbot)

SHA1:
a44308788bbd189e532745a79d126feaf708c3cd
d05e586251b3a965b9c9af76568eff912e16432f
fa616b8e2f91810a8d036ba0adca6df50da2ad22

These samples were documented in [6] in detail.

These are dual-exploit samples that have a CVE-2012-0158 exploit block and a CVE-2014-1761 exploit block, both functional. Interestingly, supposedly from an earlier template, there is another instance of an encrypted Zbot executable at the beginning of the RTF file, but that is dormant, not used by either of the two exploits.

The CVE-2014-1761 exploit block is exactly the same as the core sample; only the heading RTF junk is removed.

The *listoverridecount* control word is modified a little bit, with some junk data added to it.

Other than that, the exploit condition, the initial ROP chain and the first stage ROP chain are the same as in the core sample. The first stage shellcode is only different because the prepended CVE-2012-0158 block shifted the file offsets, and the second stage offset had to be modified in the code to reflect this shift. Essentially, the authors of these samples only prepended a CVE-2012-0158 block to the core documents and replaced the appended encrypted payload.

The first stage shellcode locates the host RTF document by enumerating all open handles, and if any of them belongs to an open file handle, checks for two ID strings. The 4 bytes from offset 0 have to be *"{\rt"* and the 4 bytes at offset 0x2xd67 have to be "!p11". If both conditions match, then it jumps to the second stage code starting right after the second ID. The second stage shellcode decrypts and executes the embedded Win32 PE payload program and displays a decoy document.

The decoy has an Arabic theme:

The payload is dropped to the user "Application Data" directory as an executable with a random name, and registered for startup under a random key in HKCU\Software\Microsoft\Windows\CurrentVersion\Run.

The list of the C&C server connected by the different backdoor variants is the following:

www.starorder.ezua.com
pop3.sec-homeland.com

Plugx

SHA1:
21b3e540746816c85e5270a1b8bb58bf713ff5f5
80f965432ce872fc3592d9f907d5a4f66ab07f9c
176273806e6fe338123ff660e70145935bac77c3
4ad76ce333b38c5bdd558e3d76640fa322e3cca6

Appat

SHA1:
0dfd883c1f205f0740d50688683f1869bcc0e9d7
9bc128f120996677d3c4f7c1d7506315b232e49e

Farfli

SHA1:
960ac7329a6e80682959d6da0469921f8167e79a

These samples were documented in [6] in detail; here we will not go into the details.

These samples are essentially the same as the Swrort samples; only the appended payload is swapped with the appropriate malware.

However, the offset of the second stage shellcode was not fixed in these samples. Thus the CVE-2014-1761 component in these samples is not working (the CVE-2012-0158 is operational). Word hangs upon opening them.

Dyzap

SHA1:
7bb6ea632a9944ee90fc2714a362c19451ea0e36
b08f351c228ff246d29f548ef910fe6e58c83caf

These samples are successfully exploiting three vulnerabilities within the same RTF carrier: CVE 2012-0158, CVE-2013-3906 and CVE-2014-1761, with separate blocks for all of them.

There are several modifications in the CVE-2014-1761 *listoverride* blocks, all of them minor.

The *listoverridecount* control word is modified to have the value of:

```
00000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000
00000000000000000000002611111
```

The *leveldata* structures directly preceding the ones that take part in the exploitation have an additional *levelnumbers* data added, but these do not play a role in the exploitation process.

The *levelnumbers* data in the relevant blocks were also slightly modified; the bytes that do not have a role in the exploitation were changed.

Additionally, in these samples the end of the first stage ROP chain is different from the original in the code sample: the finishing piece that copies the shellcode to the allocated memory is taken from a different location in mscomctl.ocx.

The first stage shellcode is an in-memory egg-hunter frequently used in Zbot droppers [9] that looks for the encrypted executable in readable memory pages. This makes use of the fact that the parsed content of the RTF file, including the embedded payload executable, should be somewhere in the memory space of the Word application, accessible to the shellcode.

It looks for 3 consecutive dwords, 0x8F8F4242, 0x8F8F4242 and 0xDDDD9292 in the memory. If found, it calculates a checksum of the following region of 0x7fc31 bytes. If it matches a precalculated value, the payload executable is decrypted and executed (the ID dwords and the length of the checksum region is different in the samples, the above values apply for the sample with SHA1 7bb6ea632a9944ee90fc2714a362c19451ea0e36). These samples do not drop a decoy document.

The payload executable is dropped into the Windows directory, with a random filename.

Overall, the authors of these samples made the most significant changes in the initial exploit condition, even though it was only the modification of unused bytes. But at least they understood some of the *leveldata* element and its role in the exploitation process.

Corkow

SHA1:
238ea4c615a7d924fda45fe0c1a99087a5f55113

Cromptui

SHA1:
08b98d597329ad7500b5e0fc2aff9088a1476040

These samples are the same as the Dyzap sample, only the payload is swapped. The exploit trigger, the initial and first stage ROP chains are exactly the same. The first stage shellcode is only modified to account for the modified length of the payload Trojan, and the modified checksum.

In case of Corkow, the final payload is dropped to *%PROGRAM FILES%\Windows NT\Microsoft\SP2oa\loaupapi.32s*. It is registered for autostart by modifying *HKLM\SYSTEM\CurrentControlSet\Services\lanmanserver\parameters\ServiceDll* to point to this value.

Cromptui is dropped to Local Settings\Temp\adobeupd.exe in the user home directory, to ensure autostart two shortcuts are created one in the current user's Startup directory, the other in the startup directory for all users. Both shortcuts are named *"Adobe Center.lnk"*.

The C&C server contacted by Cromptui is business.onmypc.org.

Tinba

SHA1:
a4adf95a2f0cf230a388fb653d371e4ab3e0388f

This sample is a CVE-2012-0158 + CVE-2013-1761 + CVE-2014-1761 combo like the Dyzap sample, but the *leveldata* structures are more similar to the original core sample. The preceding structures do not have additional *levelnumbers* tags.

The *levelnumbers* tags were modified in all three structures; the bytes that do not have a role in the exploitation were changed.

The *listoverridecount* tag was moved to a different position within the RTF file, and its value was modified to:

```
00000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000
00000000000000000000026112847940857370449405885673400
95036021117842807947778329342903490238432329098963758
22393900277583421388878778333911111111200000000285293
48348870543087442507875703487580875436929534057480454
38452240857806833475034932 00
```

This last modification broke the exploit; in this sample the CVE-2014-1761 exploit is not working. The problem was not the value, but the modified location, that broke the parsing.

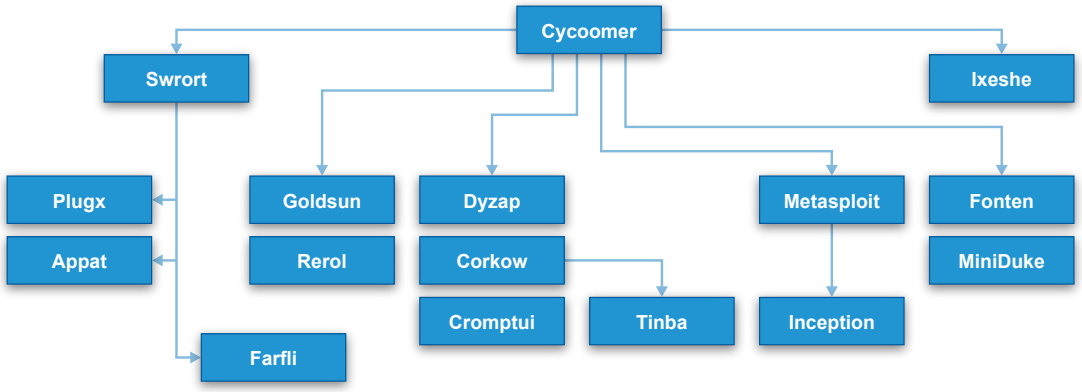
Additionally, in this sample the end of the first stage ROP chain is changed from the original in the code sample in a similar way to the Dyzap sample: the finishing part that copies the shellcode to the allocated memory is taken from a different location in *mscomctl.ocx*.

The first stage shellcode is an in-memory egg-hunter frequently used in Zbot droppers [9] that looks for the encrypted executable in readable memory pages. The parsed content of the RTF file, including the embedded payload executable, should be somewhere in the memory space of the Word application, accessible to the shellcode.

It looks for 3 consecutive dwords, 0x75755555, 0x75755555 and 0xC4C49898 in the memory. If found, it calculates a checksum of the following region of 0xa27 bytes. If it matches a precalculated value, the second stage is decrypted. This is not an embedded executable in this case, but a downloader shellcode that downloads and executes a Tinba variant.

Relations between families

The characteristics, levels of modification and similarities between the samples make it possible to create a relationship chart between the malware families.



The similarity in the RTF is conclusive enough to connect some of the families. Fonten and MiniDuke are similar enough to each other and different enough from any other samples to group them together.

For the same reason Goldsun and Rerol are in the same group (also these two families are both attributed to the Pitty Tiger group, and there is an overlap in the C&C server lists).

Similarly, Dyzap, Corkow and Cromptui belong to the same group.

Tinba is similar to the Dyzap group in using three exploits, but the CVE-2014-1761 block shows a significant enough difference to separate it from the group.

Plugx and Appat are connected by using the same C&C infrastructure [8].

Inception seems to be closer to the Metasploit sample than the core sample, because of the similar use of random filler dwords in the first stage ROP chain.

The malware families grouped together are likely to be created by the same group, or by groups in close relationship with each other.

Evaluation of families

While some of the moving parts were removed or changed, the majority of them were never touched. Malware authors are usually very keen to tweak their samples in order to avoid detection by antivirus programs. The fact that they missed these opportunities indicates that they never completely understood the nature of this exploit.

The skill levels of the APT groups will be placed on a zero-to-Neo scale [3], with basic "exploit kiddie" skills on one end, and professional exploitation at the opposite end.

	Zero	Basic	Intermediate	Skilled	Advanced	Pro	Neo
Generate sample with Metasploit	✔	✔	✔	✔	✔	✔	✔
Replace payload in existing sample	-	✔	✔	✔	✔	✔	✔
Modify shellcode	-	-	✔	✔	✔	✔	✔
Trivial modification in ROP chain	-	-	-	✔	✔	✔	✔
Significant modification in ROP chain	-	-	-	-	✔	✔	✔
Trivial modification in exploit trigger	-	-	-	-	-	✔	✔
Significant modification in exploit trigger	-	-	-	-	-	-	✔

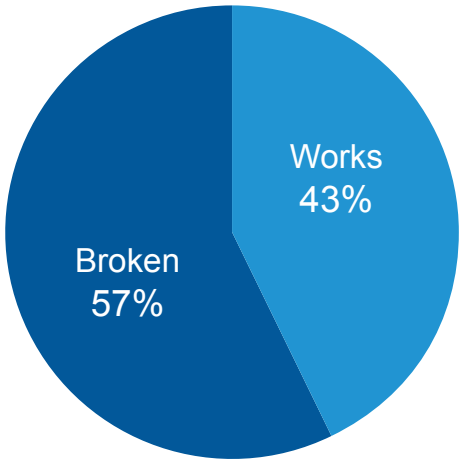
During the research, 70 samples using the exploit were analyzed in detail. These samples cover a wide range of different malware families and malware authors; it can be considered a representative set.

Taking this into account, this is how the examined malware families roughly fit into these categories:

Zero	Basic	Intermediate	Skilled	Advanced	Pro	Neo
	Goldsun		Metasploit	MiniDuke	Fonten	Cycoomer
	Swrort		Inception		Dyzap	
	Plugx				Tinba	
	Appat				Corkow	
	Farfli				Cromptui	
	Rerol				Ixeshe	

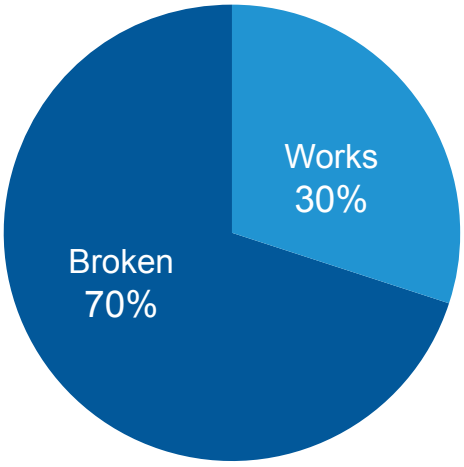
As an overall evaluation, we can say that the most infamous APT groups (Plugx, Pitty Tiger, Inception) are at the lower end of the table, while the top is dominated by the authors of common commercial families.

There is one additional factor worthy of consideration: whether the exploited RTF samples work or not. The result was more than surprising: in more than half of the samples, the CVE-2014-1761 exploit didn't work.



This doesn't mean that the "broken" samples were not able to infect their targets: those were dominantly multi-exploit samples, and the other exploits (usually the good old CVE-2012-0158) were still successful in infection.

Considering only the RTF samples that use multiple exploits, and removing the ones that are seemingly test files (e.g., drop and execute calc.exe), the real-life multi-exploit combos have a disappointingly low 30% success rate:



It is not trivial to test the operability of this exploit, especially if multiple exploits are present. Multiple replication environments need to be in place for the proper solution. But this 30% is still a very low number.

Taking into account whether the samples are actually working, the above table is a bit modified: green colour represents the samples where the exploit works, red denotes the ones that failed.

Zero	Basic	Intermediate	Skilled	Advanced	Pro	Neo
	Goldsun		Metasploit	MiniDuke	Fonten	Cycoomer
	Swrort		Inception		Dyzap	
	Plugx				Tinba	
	Appat				Corkow	
	Farfli				Cromptui	
	Rerol				Ixeshe	

Conclusions

Using this comparative approach to rate malware authors is not completely accurate. This methodology clearly underestimates their skills, because the malware authors will not show the full scope of their knowledge — only as much as is minimally necessary to accomplish the task of infecting the target. However, it is still a good estimation of how comfortable these authors are in the exploitation stage of malware creation.

In addition, in the cases when the created samples turned out to be non-working, that clearly indicates a point at which the malware authors reached their upper limit in understanding this exploit.

Conclusion 1:
The APT groups are clearly lacking in release process QA. A large percentage of their creations are not tested thoroughly, and they fail to recognize when some functionality is not working properly.

Conclusion 2:
The common malware authors show more skills in understanding exploits than the well-known APT groups. This is bad news, because the malware created by these authors reaches a general audience in much higher numbers than the targeted attacks. Therefore, the criminals with larger outreach appear to also be more skilled.

Conclusion 3:
Neither the APT authors, nor the commercial malware authors, showed enough knowledge to significantly modify the exploit trigger and the initial ROP chain. Therefore, it is not expected that (apart from the original Office 2010 SP2) any other Office version will be targeted by this exploit.

That would require modifying all the offsets in the initial (and for that matter, the first stage) ROP chain, because the absolute offsets in mscomctl.ocx are different in the other Office installations. New ROP gadgets should be collected, and the *leveldata* fields should be modified carefully.

The malware authors of the samples examined in this research proved to be far from this level of competence. The author of the initial exploit could fit it to a different version, but so far there is no sign of this development. And since the original bug in wwlib.dll is long fixed, the author is not likely to put effort into exploiting a fixed bug.

Conclusion 4:
The APT players lack deep skills of exploitation. They are quick to adopt new exploits, as samples or Metasploit modules become available, but they don't (usually) develop the exploit themselves and don't make significant modifications to them.

If security researchers and system administrators follow and act upon vulnerability announcements, they are likely to be prepared for these groups.

Despite all this, one should never underestimate the malware authors mentioned in this report. They develop sophisticated Trojan families, and they manage to deploy them successfully to high profile organizations. The fact that they are not the masters of exploitation doesn't mean that they are any less dangerous.

But they are not omnipotent either. Understanding their limitations helps us to prepare our defenses.

References

1] http://h30499.www3.hp.com/t5/HP-Security-Research-Blog/Technical-Analysis-of-CVE-2014-1761-RTF-Vulnerability/ba-p/6440048#.VIbZQNfI8UP

2] http://blogs.mcafee.com/mcafee-labs/close-look-rtf-zero-day-attack-cve-2014-1761-shows-sophistication-attackers

3] https://drive.google.com/a/psiborg.net/file/d/0B33j7-lL-ZKfQmFCWTJCOE9qZFE

4] https://technet.microsoft.com/library/security/ms14-017

5] Sebastien Duquette: *Exploitation of CVE-2014-1761 in targeted attack campaigns*, AVAR 2014, Sydney

6] http://blogs.sophos.com/2014/10/30/the-rotten-tomato-campaign-new-sophoslabs-research-on-apts/

7] http://blog.cassidiancybersecurity.com/post/2014/07/The-Eye-of-the-Tiger2

8] https://www.bluecoat.com/security-blog/2014-12-09/blue-coat-exposes-%E2%80%9C-inception-framework%E2%80%9D-very-sophisticated-layered-malware

9] http://securelist.com/analysis/publications/37158/the-curious-case-of-a-cve-2012-0158-exploit/

United Kingdom and Worldwide Sales
Tel: +44 (0)8447 671131
Email: sales@sophos.com

North American Sales
Toll Free: 1-866-866-2802
Email: nasales@sophos.com

Australia and New Zealand Sales
Tel: +61 2 9409 9100
Email: sales@sophos.com.au

Asia Sales
Tel: +65 62244168
Email: salesasia@sophos.com

Oxford, UK | Boston, USA

01.15.RP.tpna.simple