Rampant Kitten – An Iranian Espionage Campaign

By lotemf

Published: 2020-09-18 · Archived: 2026-04-11 02:22:09 UTC

Introduction

Check Point Research unraveled an ongoing surveillance operation by Iranian entities that has been targeting Iranian expats and dissidents for years. While some individual sightings of this attack were previously reported by other researchers and journalists, our investigation allowed us to connect the different campaigns and attribute them to the same attackers.

Among the different attack vectors we found were:

- Four variants of Windows infostealers intended to steal the victim's personal documents as well as access to their Telegram Desktop and KeePass account information
- Android backdoor that extracts two-factor authentication codes from SMS messages, records the phone's voice surroundings and more
- Telegram phishing pages, distributed using fake Telegram service accounts

The above tools and methods appear to be mainly used against Iranian minorities, anti-regime organizations and resistance movements such as:

- Association of Families of Camp Ashraf and Liberty Residents (AFALR)
- Azerbaijan National Resistance Organization
- Balochistan people

Table of Contents

Initial Infection

We first encountered a document with the name `وحشت_رژیم_از_گسترش_کانونهای_شورشی.docx`, which roughly translates to `The Regime Fears the Spread of the Revolutionary Cannons.docx`. The title of the document was in fact referring to the ongoing struggle between the Iranian regime and the Revolutionary Cannons, a Mujahedin-e Khalq movement.

Mujahedin-e Khalq, or The People's Mujahedin of Iran, is an anti-regime organization whose aim is to free Iran from its current leadership. In 1986, Mujahedin-e Khalq (MEK) started building their new headquarters, which later became known as Camp Ashraf, near the Iraqi town of Khalis. However, years of political tension in Iraq eventually led to the transfer of the camp's residents to a new, remote, and unlikely destination: Albania.

The above document leverages the external template technique, allowing it to load a document template from a remote server, which in this case was `afalr-sharepoint[.]com`.

Figure 1: Remote template

Curious by this website, we set out to discover more about it. At first, we found a handful of tweets from accounts opposing the Iranian regime mentioning a very similar SharePoint site, which the website in the document was likely impersonating:

Figure 2: Tweets mentioning similar website

Figure 3: AFALR’s official website

Infection Chain

After the victim opens the document and the remote template is downloaded, the malicious macro code in the template executes a batch script which tries to download and execute the next stage payload from `afalr-sharepoint[.]com`:

Figure 4: Infection chain

The payload then checks if Telegram is installed on the infected machine, and if so it proceeds to extract three additional executables from its resources:

- **BOBC3953C59DA7870** – Loader, executed by `RunDLL`, injects the main payload into `explorer.exe`
- **CO9D5A739B85C37C1** – Infostealer payload
- **Updater.exe** – Modified Telegram updater

Payload Analysis

The main features of the malware include:

- **Information Stealer**
  - Uploads relevant Telegram files from victim's computer. These files allow the attackers to make full usage of the victim's Telegram account
  - Steals information from KeePass application
  - Uploads any file it could find which ends with pre-defined extensions
  - Logs clipboard data and takes desktop screenshots
- **Module Downloader**
  - Downloads and installs several additional modules.
- **Unique Persistence**
  - Implements a persistence mechanism based on Telegram's internal update procedure

Telegram structure basics

First, let us review how Telegram Desktop organizes its files. The following is an ordinary Telegram file structure which can normally be found at `%APPDATA%\Roaming\Telegram Desktop`.

Figure 5: Telegram Desktop directory structure

As explained above, several files are dropped to the Telegram working directory during the infection chain. The dropped files are in a directory named `03A4B68E98C17164s`, which looks like a file at first glance because of a custom `Desktop.ini` file, but it is actually a directory.

Figure 6: Infected Telegram Desktop directory

Configuration

One of payload's resources contains encoded configuration data.

The encoding scheme uses the regular Base64 algorithm but with a custom index table: `eBaEFGHOQRS789TUYZdCfPbDIJ+/KLMNwxyzquv0op123VWXghijmnkl45rst6Ac`.

Decoding that resource gives us the following configuration:

Key	**Value**
AES encryption key	ssher54276@@5!!

File suffixes	.txt;.csv;.kdbx;.xls;.xlsx;.ppt;.pptx;
SOAP username	9BEF4B32-0D40-4A92-9E35-6094B8AA8B58
SOAP password	D5F69342-A3CC-438F-B3B6-5E7BF6B6E327
Main C&C	hxxps://www.afalr-sharepoint[.]com/B6D9E741-DFE3-4470-9174-C95FB2B958AD/TelBService.asmx
Backup C&C	hxxps://www.afalr-onedrive[.]com/B6D9E741-DFE3-4470-9174-C95FB2B958AD/TelBService.asmx

C&C Communication

The malware uses SOAP for its communication purposes. SOAP is a simple XML-based data structure for web-service communication. The API in SOAP web-services is public and can be observed by accessing the website from a browser:

Figure 7: SOAP API in C&C website

The messages (commands) can be divided into the following categories:

- **Authentication**:
  - `HelloWorld` – Authentication message

- **Module Downloader:**
  - `DownloadFileSize` – Checks whether a module should be downloaded
  - *`DownloadFile`* – Downloads a module from the remote server
- **Data Exfiltration:**
  - *`UploadFileExist`* – Checks whether a specific victim file has been uploaded
  - *`UploadFile`* – Uploads a specific victim file

Authentication

The first message for a valid communication tunnel should be `HelloWorld`, which implements a simple Username/Password authentication. The credentials are hard-coded in the sample, and the SOAP response for that message contains a session ID which must be used for the remainder of the session.

Module Downloader

The program tries fetching updates for its current modules and also download several additional modules.

Some of the additional missing modules that could not be fetched:

- `D07C9D5A79B85C331.dll`
- `EO333A57C7C97CDF1`
- `EO3A7C3397CDF57C1`

Data Exfiltration

The core functionality of the malware is to steal as much information as it can from the target device. The payload targets two main applications**: Telegram Desktop** and **KeePass**, the famous password manager.

Once the relevant Telegram Desktop and KeePass files have been uploaded, the malware enumerates any relevant file it can find on the victim's computer which has one of the following extensions: `.txt;.csv;.kdbx;.xls;.xlsx;.ppt;.pptx;`. For each such file, the malware then uploads it after encoding the file to base64.

Persistence

The malware uses a unique persistence method which is tied to the **Telegram update procedure**.

Periodically, it copies the Telegram main executable into `Telegram Desktop\tupdates`, which triggers an update procedure for the Telegram application once it starts. The hidden trick of the malware's persistence method is changing the default Telegram updater file – `Telegram Desktop\Updater.exe`, with one of its dropped payloads (more specifically – `CO79B3A985C5C7D30`). The most notable changed feature of that updater is running the payload again:

Figure 8: Telegram updater runs the main payload

Infrastructure and Connections

After analyzing the payload we were able to find multiple variants that date back to 2014, indicating that this attack has been in the making for years.

Malware variants developed by the same attackers often have minor differences between them, especially if they are used around the same time frame. In this case however, we noticed that while some of the variants were used simultaneously, they were written in different programming languages, utilized multiple communication protocols and were not always stealing the same kind of information.

In the table below, we list the variants we identified and highlight their unique characteristics. Please refer to the Technical Appendix below, for a deep dive information regarding each variant.

Name	Artifacts	Dates	Malicious Activity	Properties
TelB Variant	KeePassOnlineCreator.exe BOBC3953C59DA7870 CO9D5A739B85C37C1 CO79B3A985C5C7D30 D07C9D5A79B85C331.dll EO333A57C7C97CDF1 EO3A7C3397CDF57C1	June 2020 – July 2020	Telegram-focused infostealer	SOAP. Delphi 64bit payload. Persistence through Telegram updater.
TelAndExt Variant	TelegramUpdater.exe TelegramUpdater2.exe TelegramUpdater3.exe TelegramUpdater.dll Updater.exe	May 2019 – February 2020	Telegram-focused infostealer	FTP . Delphi 32bit payload. Persistence through Telegram updater.

Python Info Stealer	keyboard-EN.exe speaker-audio.exe audio-driver.exe	February 2018 – January 2020	Focused on – Telegram, Chrome, Firefox, Edge, Paltalk NG Data Exfiltration via FTP	FTP. Python (Pyinstaller)
HookInjEx Variant	DrvUpdt.exe / ehtmlh.exe DrvUpdtd.dll / dhtmlh.dll CapDev.exe / rregg.exe uflScan.exe	December 2014 – May 2020	Infostealer (Browsers, audio, keylogging and clipboard)	FTP. C++

The related samples also revealed more C&C servers, and looking up their passive DNS information and additional metadata led us to similar domains that were operated by the same attackers. As it turns out, some of the domains appeared in malicious Android applications and phishing pages, exposing more layers of this operation:

Figure 9: Maltego graph of the malicious infrastructure

Android Backdoor

During our investigation we also uncovered a malicious Android application tied to the same threat actors. The application masquerades as a service to help Persian speakers in Sweden get their driver’s license.

We have located two different variants of the same application, one which appears to be compiled for testing purposes, and the other is the release version, to be deployed on a target's device.

Figure 10: Android application's main interface

This Android backdoor contains the following features:

- Steal existing SMS messages
- Forward two-factor authentication SMS messages to a phone number provided by the attacker-controlled C&C server
- Retrieve personal information like contacts and accounts details
- Initiate a voice recording of the phone's surroundings
- Perform Google account phishing
- Retrieve device information such as installed applications and running processes

For a deep dive information regarding this application, please refer to the Technical Appendix below.

Telegram Phishing

The backdoors were not the only way in which the attackers tried to steal information about Telegram accounts. Some of the websites that were related to this malicious activity also hosted phishing pages impersonating Telegram:

Figure 11: Telegram phishing page

What was surprising is that several Iranian Telegram channels have actually sent out warnings against those phishing websites, and claimed that the Iranian regime is behind them.

Figure 12: Translated message warning against phishing attempts

According to the channels, the phishing messages were sent by a Telegram bot. The messages warned their recipient that they were making an improper use of Telegram's services, and that their account will be blocked if they do not enter the phishing link.

Figure 13: Phishing message content

Another Telegram channel provided screenshots of the phishing attempt showing that the attackers set up an account impersonating the official Telegram one. At first, the attackers sent a message about the features in a new Telegram update to appear legitimate. The phishing message was sent only five days later, and pointed to `http://telegramreport[.]me/active` (same domain as in figure 11 above).

Figure 14: Phishing message sent from fake Telegram account

Payload Delivery

Although in some cases we were unable to determine how the malicious files reached the victims, we gathered some potential clues about the ways the attackers distributed their malware. For example, accessing `mailgoogle[.]info` shows that it impersonates `ozvdarozv[.]com` and promotes a software to increase the number of members in Telegram channels.

Figure 15: `mailgoogle[.]info` download page

But after clicking on “Download”, a password-protected archive called `Ozvdarozv-Windows.rar` is downloaded, containing one of the malware variants.

Possible Additional Delivery Vectors

A removed blog entry from 2018 accused a cyber-security expert of plagiarism when he was interviewed by **AlArabiya** news channel to discuss Iranian cyber-attacks.

We believe this page was created as part of a targeted attack against this person or his associates.

The blog included a link to download a password-protected archive containing evidence of the plagiarism from `enduploa[.]com`.

`enduploa[.]com` is connected to both the PC and the Android operations mentioned above via several passive DNS hops, including a direct connection via historic DNS server information to the domain `mailgoogle[.]info` we described above. Not only did we not find any instance of it being used in a legitimate context, we also found evidence of the domain being registered by a Persian speaking hacker. (See “attribution” section below)

Figure 16: Removed blog post with link to `endupload[.]com`

A different blog entry from 2012 discusses a human rights violations report by HRANA, a news agency affiliated with the Iranian Association of Human Rights Activists. Once again, this blog refers to a document that can be downloaded from `endupload[.]com` :

Figure 17: Blog post with link to enduploa[.]com

Unfortunately, we were unable to get our hands on the files both blog entries were referring to, and could not confirm that they were indeed malicious. However, it appears that `endupload[.]com` has been controlled by the attackers for years, as some of the malicious samples related to this attack and dating back to 2014 communicated with this website.

Attribution

Although we found many files and websites that were used over the years in this attack, they were not attributed to a specific threat group or entity. Nevertheless, some of the fingerprints that the threat actors left in the malicious artifacts allowed us to gain a better understanding of where the attack might be coming from.

Attack Origin

To begin with, the WHOIS information of some of the malicious websites revealed that they were supposedly registered by Iranian individuals:

Figure 18: WHOIS information of `endupload[.]com` and `picfile[.]net`

The WHOIS records for `endupload[.]com` also mentioned an e-mail address, `nobody.gu3st@gmail[.]com`. Apparently, the website's registrant was very active online, because looking up the username "***nobody.gu3st***" led us to many posts in Iranian hacking forums:

Figure 19: Translated post by nobody.gu3st

Political Targeting

The list of targets we observed reflects some of the internal struggles in Iran and the motives behind this attack. The handpicked targets included supporters of Mujahedin-e Khalq and the Azerbaijan National Resistance Organization, two prominent resistance movements that advocate the liberation of Iranian people and minorities within Iran.

Figure 20: Mujahedin-e Khalq and Azerbaijan National Resistance Organization logos

The conflict of ideologies between those movements and the Iranian authorities makes them a natural target for such an attack, as they align with the political targeting of the regime.

In addition, the backdoor's functionality and the emphasis on stealing sensitive documents and accessing **KeePass** and **Telegram** accounts shows that the attackers were interested in collecting intelligence about those victims, and learning more about their activities.

Conclusion

Following the tracks of this attack revealed a large-scale operation that has largely managed to remain under the radar for at least six years. According to the evidence we gathered, the threat actors, who appear to be operating from Iran, take advantage of multiple attack vectors to spy on their victims, attacking victims' personal computers and mobile devices.

Since most of the targets we identified are Iranians, it appears that similarly to other attacks attributed to the Islamic Republic, this might be yet another case in which Iranian threat actors are collecting intelligence on potential opponents to the regiment.

SandBlast Mobile provides real-time threat intelligence and visibility into mobile threats, protecting from malware, phishing, Man-in-the-Middle attacks, OS exploits, and more.

Check Point's anti-phishing solutions include products that address all of the attack vectors from which phishing attacks come – email, mobile, endpoint and network.

Technical Appendix

PC Backdoor Variants Analysis

TelB Variant

"TelB" is the latest variant we encountered, and its analysis shown above. We named it as such because of the next debug string: `D:\Aslan\Delphi\TelB\BMainWork\SynCryptoEN.pas`.

TelAndExt Variant

This variant is probably the older version of "TelB", and has been active mostly during 2019 and 2020. It shares the following properties and techniques with the newer versions:

- Developed in Delphi
- Shares a great amount of code with the "TelB" variant
- Focuses on the Telegram Desktop application
- Similar persistence and update methods
- Uses FTP instead of SOAP for data exfiltration

We named this variant "TelAndExt" because of the next debug string:
`D:\Aslan\Delphi\TelAndExt\TelegramUpdater\SynCryptoEN.pas`

Python Info-stealer Variant Analysis

We discovered several samples which use the following methods:

- Two-layer SFX (self-extracted archive) which extract several .bat/.vbs/.nfs/.conf files.
- Persistence method by copying the executable (ends with `.nfs`) to `%appdata%\Microsoft\Windows\Start Menu\Programs\Startup\audio-driver.exe`
- Executable name is `speaker-audio.exe` or `keyboard-EN.exe`, depending on the sample.
- The executable was created with Pyinstaller.
- Downloads a second-stage payload under the name of `net-update.exe`
- Before being uploaded, the data is encrypted using library `pyAesCrypt` with a hard-coded password.

Info stealing

According to our analysis, the script communicates with an FTP server using hard-coded credentials, and steals the following data:

- Telegram Desktop application related files.
- Paltalk NG application related files.
- Chrome, Firefox and Edge related data.
- Any file which ends with extensions listed in a given configuration. If no configuration is given, it searches for files with the following extensions: `.txt;.docx;.doc;.exe;.jpg;.html;.zip;.pdf`

Operation

During our investigation, we saw several Python info-stealers that communicate with the same FTP server, but store the stolen information in different pages under different aliases.

We suspect this is how the malware authors operate:

- Choose a target, and create a designated folder in the FTP server for them.
- Build a sample customized for the target, with a unique AES key and FTP credentials for information uploading.
- Deliver the weaponized executable via one of the infection chain vectors.

Second-stage Payload – HookInjEx

One of its core functionalities is fetching a second-stage payload. If the designated FTP folder contains a file named `net-update.exe`, then it downloads and executes it.

We analyzed few of those `net-update.exe` samples and found a complete overlap with the "HookInjEx" variant below, making it a targeted advanced payload.

HookInjEx Variant

This variant has been in use since 2014, and has 32-bit and 64-bit versions. Over time, the variant evolved and added some features while also changed the names of the different components in its infection chain.

Infection Chain

We found two main types of infection chains:

1. **SFX** (self-extracting archive) which contains all the components. It drops all of them into a folder and then executes the main loader – `DrvUpdt.exe` (`ehtmlh.exe` in older versions).
2. Fake **SCR** file that is functioning as an executable. In order to look like a legitimate SCR file, the loader contains a **decoy** – a JPEG/PPTX/DOC file as a resource (`Resource_1`), which is loaded upon running.

The SCR file also drops other payloads as its resources, and runs the main loader with the command line:

```
cmd.exe /C choice /C Y /N /D Y /T 3 & "%APPDATA%\\Microsoft\\Windows\\Device\\DrvUpdt.exe" -pSDF32fsj8979_)$
```

Figure 21: Malicious SCR opens decoy JPG resource

Hooking and Injection

The main loader uses the hooking and injection method called “HookInjEx”. That method maps a DLL into `explorer.exe`, where it subclasses the `Start` button. In our case, the loaded DLL is `DrvUpdtd.dll` (`dhtmlh.dll` in older versions).

In newer versions, the malware also hooks the `Start` button in other languages as well. The existence of different languages probably shows that it has victims from countries all around the world. The different translations are:

Plain text

Copy to clipboard

Open code in new window

EnlighterJS 3 Syntax Highlighter

Start - English

başlat - Turkish

開始 - Chinese

Sākt - Latvian

ابدأ - Arabic

Iniciar - Spanish

Käynnistä - Finish

התחל - Hebrew

スタート - Japanese

Štart - Slovakian

Pradėti - Lithuanian

Pokreni - Bosnian

เริ่ม - Thai

Έναρξη - Greece

Démarrer - French

Старт - Bulgarian

Запустити - Ukrainian

시작 - Korean

Start - English başlat - Turkish 開始 - Chinese Sākt - Latvian ابدأ - Arabic Iniciar - Spanish Käynnistä - Finish התחל - Hebrew スタート - Japanese Štart - Slovakian Pradėti - Lithuanian Pokreni - Bosnian เริ่ม - Thai Έναρξη - Greece Démarrer - French Старт - Bulgarian Запустити - Ukrainian 시작 - Korean

```
Start - English
başlat - Turkish
開始 - Chinese
Sākt - Latvian
ابدأ - Arabic
Iniciar - Spanish
Käynnistä - Finish
התחל - Hebrew
スタート - Japanese
Štart - Slovakian
Pradėti - Lithuanian
Pokreni - Bosnian
เริ่ม - Thai
Έναρξη - Greece
Démarrer - French
Старт - Bulgarian
```

```
Запустити - Ukrainian
시작 - Korean
```

Configuration

The malware receives its configuration from a file named `Devinf.asd` (in older versions it was named `file2.asd`). The configuration is decrypted and written into a new file named `Drvcnf.asd` (in older version it named `file3.asd`). The encryption method is:

Plain text

Copy to clipboard

Open code in new window

EnlighterJS 3 Syntax Highlighter

def decode(content):

dec_array = [0, 1, 2, 3, 4, 5, 6, 7, 8, 0xe, 0xf, 0x10, 0x11, 0x12, 0x13, 0x14, 0x15, 0x16, 0x17, 0x18, 0x19, 0x1a,0x1b, 0x1c, 0x1d, 0x1e, 0x21, 0x22, 0x23, 0x24, 0x25, 0x26, 0x27, 0x28, 0x29, 0x2a, 0x2b, 0x2c, 0x2d, 0x2e, 0x2f, 0x30, 0x31, 0x32, 0x33, 0x34, 0x35, 0x36, 0x37, 0x38, 0x39,0x3a, 0x3b, 0x3c, 0x3d, 0x3e, 0x3f, 0x40, 0x41, 0x42, 0x43, 0x44]

output_str = ''

known_values = [9, 10, 13, 32]

for j in range(len(content)):

int_cur_content = ord(content[j])

cur_byte = 0

for i in range(62):

if int_cur_content == dec_array[i]:

if i < 26:

cur_byte = i + 0x61

elif 26 <= i < 52:

cur_byte = i + 0x27

elif 52 <= i < 62:

cur_byte = i - 0x4

output_str += (chr(cur_byte))

break

if cur_byte == 0:

if int_cur_content in known_values:

cur_byte = int_cur_content

elif int_cur_content - 0x61 <= 0xe:

cur_byte = int_cur_content - 0x40

elif int_cur_content - 0x70 <= 0x6:

cur_byte = int_cur_content - 0x36

elif int_cur_content - 0x77 <= 0x5:

cur_byte = int_cur_content - 0x1c

elif int_cur_content - 0x53 <= 0x3:

cur_byte = int_cur_content + 0x28

output_str += (chr(cur_byte))

return output_str

def decode(content): dec_array = [0, 1, 2, 3, 4, 5, 6, 7, 8, 0xe, 0xf, 0x10, 0x11, 0x12, 0x13, 0x14, 0x15, 0x16, 0x17, 0x18, 0x19, 0x1a,0x1b, 0x1c, 0x1d, 0x1e, 0x21, 0x22, 0x23, 0x24, 0x25, 0x26, 0x27, 0x28, 0x29, 0x2a, 0x2b, 0x2c, 0x2d, 0x2e, 0x2f, 0x30, 0x31, 0x32, 0x33, 0x34, 0x35, 0x36, 0x37, 0x38, 0x39,0x3a, 0x3b, 0x3c, 0x3d, 0x3e, 0x3f, 0x40, 0x41, 0x42, 0x43, 0x44] output_str = '' known_values = [9, 10, 13, 32] for j in range(len(content)): int_cur_content = ord(content[j]) cur_byte = 0 for i in range(62): if int_cur_content == dec_array[i]: if i < 26: cur_byte = i + 0x61 elif 26 <= i < 52: cur_byte = i + 0x27 elif 52 <= i < 62: cur_byte = i - 0x4 output_str += (chr(cur_byte)) break if cur_byte == 0: if int_cur_content in known_values: cur_byte = int_cur_content elif int_cur_content - 0x61 <= 0xe: cur_byte = int_cur_content - 0x40 elif int_cur_content - 0x70 <= 0x6: cur_byte = int_cur_content - 0x36 elif int_cur_content - 0x77 <= 0x5: cur_byte = int_cur_content - 0x1c elif int_cur_content - 0x53 <= 0x3: cur_byte = int_cur_content + 0x28 output_str += (chr(cur_byte)) return output_str

```
def decode(content):

    dec_array = [0, 1, 2, 3, 4, 5, 6, 7, 8, 0xe, 0xf, 0x10, 0x11, 0x12, 0x13, 0x14, 0x15, 0x16, 0x17
    output_str = ''
    known_values = [9, 10, 13, 32]
    for j in range(len(content)):
```

```
        int_cur_content = ord(content[j])
        cur_byte = 0
        for i in range(62):
            if int_cur_content == dec_array[i]:
                if i < 26:
                    cur_byte = i + 0x61
                elif 26 <= i < 52:
                    cur_byte = i + 0x27
                elif 52 <= i < 62:
                    cur_byte = i - 0x4
                output_str += (chr(cur_byte))
                break
        if cur_byte == 0:
            if int_cur_content in known_values:
                cur_byte = int_cur_content
            elif int_cur_content - 0x61 <= 0xe:
                cur_byte = int_cur_content - 0x40
            elif int_cur_content - 0x70 <= 0x6:
                cur_byte = int_cur_content - 0x36
            elif int_cur_content - 0x77 <= 0x5:
                cur_byte = int_cur_content - 0x1c
            elif int_cur_content - 0x53 <= 0x3:
                cur_byte = int_cur_content + 0x28
            output_str += (chr(cur_byte))
    return output_str
```

After the configuration is decrypted, the malware parses the values and puts them in global variables.

Plain text

Copy to clipboard

Open code in new window

EnlighterJS 3 Syntax Highlighter

<Reg></Reg> - Registry key for persistence

<Pre></Pre> - pre value for info files to send

<Pas></Pas> - extensions for info files to send

<Path_Log></Path_Log> - log path direcory

<L_s></L_s> - minimum size for file to send

<S_n></S_n> - FTP domain

<F_k></F_k> - FTP User value

<F_R></F_R> - FTP Password value

<Ver></Ver> - version

<U_u2></U_u2> - Downloads updates URL

<U_u1></U_u1> - Downloads updates URL

<F_f></F_f> - Directory in ftp connection.

<U_t></U_t> - timer_1

<S_t></S_t> - timer_2

<S_q></S_q> - timer_3

<U_u3></U_u3> - Downloads updates URL

<El></El> - value for encryption method to files

<Ez></Ez> - value for encryption method to files

<F_n></F_n> - Fake name

<E_dt></E_dt>

<E_dy></E_dy>

<Snd_P></Snd_P> - Value to choose name template for info files to send.

<Mlt></Mlt> - flag for exeucting again from different place.

<Ws1></Ws1> - WHOIS first URL

<Ws2></Ws2> - WHOIS second URL

<Ws3></Ws3> - WHOIS third URL

<S_li></S_li>

<RTL></RTL>

<SHttpR></SHttpR> - value to download using computer-name and username

<OPP></OPP> - Opera gather data flag

<FIP></FIP> - Firefox gather data flag

<CHP></CHP> - Chrome gather data flag

<WHP></WHP> - WHOIS flag

<TRP></TRP> - tracert flag

<FRC></FRC> - number of tries

<Clipfp></Clipfp> - clipboard data flag of CF_HDROP (CLB-f.jpg)

<Cliptp></Cliptp> - clipboard data flag of text, unicode, oemtext and locale (clb-t.jpg)

<Clipip></Clipip> - clipboard data flag of bitmap and dib (clb-p.jpg)

<Reg></Reg> - Registry key for persistence <Pre></Pre> - pre value for info files to send <Pas></Pas> - extensions for info files to send <Path_Log></Path_Log> - log path direcory <L_s></L_s> - minimum size for file to send <S_n></S_n> - FTP domain <F_k></F_k> - FTP User value <F_R></F_R> - FTP Password value <Ver></Ver> - version <U_u2></U_u2> - Downloads updates URL <U_u1></U_u1> - Downloads updates URL <F_f></F_f> - Directory in ftp connection. <U_t></U_t> - timer_1 <S_t></S_t> - timer_2 <S_q></S_q> - timer_3 <U_u3></U_u3> - Downloads updates URL <El></El> - value for encryption method to files <Ez></Ez> - value for encryption method to files <F_n></F_n> - Fake name <E_dt></E_dt> <E_dy></E_dy> <Snd_P> </Snd_P> - Value to choose name template for info files to send. <Mlt></Mlt> - flag for exeucting again from different place. <Ws1></Ws1> - WHOIS first URL <Ws2></Ws2> - WHOIS second URL <Ws3></Ws3> - WHOIS third URL <S_li></S_li> <RTL></RTL> <SHttpR></SHttpR> - value to download using computer-name and username <OPP></OPP> - Opera gather data flag <FIP></FIP> - Firefox gather data flag <CHP> </CHP> - Chrome gather data flag <WHP></WHP> - WHOIS flag <TRP></TRP> - tracert flag <FRC></FRC> - number of tries <Clipfp></Clipfp> - clipboard data flag of CF_HDROP (CLB-f.jpg) <Cliptp></Cliptp> - clipboard data flag of text, unicode, oemtext and locale (clb-t.jpg) <Clipip></Clipip> - clipboard data flag of bitmap and dib (clb-p.jpg)

```
<Reg></Reg> - Registry key for persistence
<Pre></Pre> - pre value for info files to send
<Pas></Pas> - extensions for info files to send
<Path_Log></Path_Log> - log path direcory
<L_s></L_s> - minimum size for file to send
<S_n></S_n> - FTP domain
<F_k></F_k> - FTP User value
<F_R></F_R> - FTP Password value
<Ver></Ver> - version
<U_u2></U_u2> - Downloads updates URL
<U_u1></U_u1> -  Downloads updates URL
<F_f></F_f> - Directory in ftp connection.
<U_t></U_t> - timer_1
<S_t></S_t> - timer_2
<S_q></S_q> - timer_3
<U_u3></U_u3> - Downloads updates URL
<El></El> - value for encryption method to files
<Ez></Ez> - value for encryption method to files
<F_n></F_n> - Fake name
<E_dt></E_dt>
```

```
<E_dy></E_dy>
<Snd_P></Snd_P> - Value to choose name template for info files to send.
<Mlt></Mlt> - flag for exeucting again from different place.
<Ws1></Ws1> - WHOIS first URL
<Ws2></Ws2> - WHOIS second URL
<Ws3></Ws3> - WHOIS third URL
<S_li></S_li>
<RTL></RTL>
<SHttpR></SHttpR> - value to download using computer-name and username
<OPP></OPP> - Opera gather data flag
<FIP></FIP> - Firefox gather data flag
<CHP></CHP> - Chrome gather data flag
<WHP></WHP> - WHOIS flag
<TRP></TRP> - tracert flag
<FRC></FRC> - number of tries
<Clipfp></Clipfp> - clipboard data flag of CF_HDROP (CLB-f.jpg)
<Cliptp></Cliptp> - clipboard data flag of text, unicode, oemtext and locale (clb-t.jpg)
<Clipip></Clipip> - clipboard data flag of bitmap and dib (clb-p.jpg)
```

Persistence

The malware sets the registry key which is in the `<Reg>` value of the configuration file (which is almost always the key **RunOnce**) to the following values – it sets the name to `SunJavaHtml` or `DevNicJava` and the value is `DrvUpdt.exe 11`. That way the malware knows it was already executed.

In older versions, the malware used to drop a file named either `rreegg.exe` or `Capdev.exe`, which was added to **RunOnce**, and in turn executed `DrvUpdt.exe 11`

Info stealing

Main feature of the malware is stealing information from the victim's computer and send it to the C2 using FTP.

The malware steals different types of data:

- Opera/Chrome/Firefox login data.
- Firefox information: profiles, keys and db files.
- The output of `tracert www.google.com`
- WHOIS information (based on the `<Ws1>` value).
- Screenshots and title of the foreground window.
- Waveform-audio recording for a minute.
- Files from removable drivers. The types of files are based on the `<cpy>` tags in the file `Devufl1.tmp` (`winufl1.tmp` in older versions). In some versions, that logic was implemented in a file named `uflscan.exe`.
  Interestingly, if driver's name is one of the following: `A65RT52WE3F`, `09353536557` or `transcend20276`, the malware ends the thread. We believe it to be a debug code (Fig. X) that stayed and its purpose is to prevent the malware from gathering files from the author's computers.

- Files from other drives based on the `<cpy>` tags in the file `Devufl2.tmp` (`winufl2.tmp` in older versions).
- Keylogging and clipboard data from various formats – drag and drop/ `CF_HDROP` , `CF_TEXT` , `CF_TEXT` , `CF_UNICODETEXT` , virtual key codes, `CF_OEMTEXT` , `CF_LOCALE` , `CF_BITMAP` and `CF_DIB` .
- Capture using webcam (`tcwin.exe` in older versions).
- Since 2018 – Telegram Desktop data.

Figure 22: Debug code with hardcoded removable drivers

C2 communication

This variant uploads files to its C2 domain using the FTP protocol. The FTP domain is placed in the configuration file inside the `<S_n>` tag.

The connection starts with authentication using the password and username from the configuration file.

The malware then creates a directory according to the `<F_f>` tag and a subdirectory inside it with the user id it generated before. The user id is generated from the network adapter's info that was written into the file `Mcdata.dat` (`PAdata.dat` In older versions).

After that, the connection continues with `TYPE I` and `PASV` commands before storing the files with the `STOR` command.

The variant also contacts other domains to update its different components. The domains are placed in the configuration file inside `<U_U1>` , `<U_U2>` and `<U_U3>` tags. The files are downloaded using `URLDownloadToFileW` from the given URLs, the `user_id` is included in the URLs.

String Obfuscation

In newer versions (since 2018), the strings are encrypted with the following script:

Plain text

Copy to clipboard

Open code in new window

EnlighterJS 3 Syntax Highlighter

buf_1 = 'qweyuip[];lkjhgdszcm,.><MNBVCXZ|ASDFGHJK:}{POIUYTREWQ123456789-=+_)(*&^%$#@!'

buf_2 = '!#$%&()*+,-.123456789:;<=>@ABCDEFGHIJKMNOPQRSTUVWXYZ[]^_cdeghijklmpqsuwyz{|}'

input_str = "" # The encrypted string

output_str = ''

for i in range(len(input_str)):

cur_byte = input_str[i]

place = buf_2.find(cur_byte)

if place == -1:

output_str += cur_byte

continue

new_byte = buf_1[place]

output_str += new_byte

print(output_str)

buf_1 = 'qweyuip[];lkjhgdszcm,.><MNBVCXZ|ASDFGHJK:}{POIUYTREWQ123456789-=+_)(*&^%$#@!'
buf_2 = '!#$%&()*+,-.123456789:;<=>@ABCDEFGHIJKMNOPQRSTUVWXYZ[]^_cdeghijklmpqsuwyz{|}'
input_str = "" # The encrypted string output_str = '' for i in range(len(input_str)): cur_byte = input_str[i] place = buf_2.find(cur_byte) if place == -1: output_str += cur_byte continue new_byte = buf_1[place] output_str += new_byte print(output_str)

```
buf_1 = 'qweyuip[];lkjhgdszcm,.><MNBVCXZ|ASDFGHJK:}{POIUYTREWQ123456789-=+_)(*&^%$#@!'
buf_2 = '!#$%&()*+,-.123456789:;<=>@ABCDEFGHIJKMNOPQRSTUVWXYZ[]^_cdeghijklmpqsuwyz{|}'
input_str = "" # The encrypted string
output_str = ''

for i in range(len(input_str)):
      cur_byte = input_str[i]
      place = buf_2.find(cur_byte)
      if place == -1:
            output_str += cur_byte
            continue
      new_byte = buf_1[place]
```

```
      output_str += new_byte
print(output_str)
```

```

```

Android Backdoor Analysis

The first activity is `MainActivity`, which is responsible for presenting the user with the decoy content and requesting permissions to perform privileged activity. It also starts a background service called `MainService`, and launches the second `MainActivityFake` (`GmailActivity`) when the server sends a command to do so.

Figure 23: User is requested to allow a set of permissions

Data Collection

Once the `GmailActivity` launches the `MainService`, it in turn is responsible for the following tasks: Timer registration, configuration monitoring, showing fake notification (described below) and **sensitive data collection**.

During this initial data collection process, the following information is read and prepared:

- Installed applications list
- Accounts information
- SMS messages
- Contacts information

The rest of the information is collected on demand, once a command is received from a C&C server:

- Voice recording – A 30 seconds recording by default.
- Google credentials – The server triggers an authentication phishing attempt.

Google Credentials Theft

Upon receiving the proper command from the C&C server, a Google login page will be displayed to the victim, by activating the `MainActivityFake` (`GmailActivity`).

Figure 24: Google login page

At this point the user is presented with a legitimate `accounts.google.com` login page, inside Android's WebView. In order to steal the typed-in credentials, Android's JavascriptInterface is used, alongside a timer which periodically retrieves the information from the username and password input fields.

Figure 25: Periodic retrieval of Google account credentials

“**Google protect is enabled”**

As we previously mentioned, one of its core functionalities is to turn on the microphone and record the surroundings. In order to achieve this goal in a real-time manner, the application needs to have its service running in the background.

Any Android application that wants to perform such action, is required to post an ongoing notification to the user, which will alert the user of the uninitiated activity on the device. In order to circumvent this issue, the malware developers chose to display the user with a fake notification of “**Google protect is enabled**“.

Figure 26: Applications displays a fake notification

The result is an always-on decoy notification masquerading as “Google protect”.

Figure 27: Fake Google Protect notification

C&C Communication

The malware uses regular HTTP to communicate with the C&C servers. It sends the initial request to `alarabiye[.]net`, and proceeds to communicate with `gradleservice[.]info` in order to get configuration, commands and status updates.

In order to upload all the sensitive information, the malware uses FTPS with hard-coded credentials.

Figure 28: FTPS connection routine

In addition, the sensitive files are encrypted using the AES algorithm, with a pre-configured passphrase before being uploaded to the FTP server,

Figure 29: AES encryption routine

Two Factor Exfiltration by SMS

One of the unique functionalities in this malicious application is forwarding any SMS starting with the prefix `G-` (The prefix of Google two-factor authentication codes), to a phone number that it receives from the C&C server.

Furthermore, all incoming SMS messages from Telegram, and other social network apps, are also automatically sent to the attackers phone number.

Work in Progress

During our analysis, it was often obvious that this malicious application was still being actively developed, with various assets and functions which were either leftovers of previous operations, or not yet utilized.

One of the unused phishing assets even contains a pre-entered username, possibly a target in a previous operation conducted by the attackers.

Figure 30: Unused phishing HTML assets

Figure 31: Unused location tracking code

Indicators of Compromise

Phishing

`telegramreport[.]me`
`telegramco[.]org`
`telegrambots[.]me`
`mailgoogle.info`

Android

C&C servers

`gradleservice[.]info`
`alarabiye[.]net`

Files

MD5
4ae3654b7ed172b0273e7c7448b0c23c
ca154dfd01b578b84c0ec59af059fb62
SHA-1
f3a4feedd4f62702c65b037a91bd8332d9518c08
735f761462443deff23dde5b76746b7ab0ceaf71
SHA-256
24e5b2967437dbc1866df3ac1bf776a4960a5a56676b48bb9a143e62849a43d2
881ab44385541ac7cd0f3279ba4fb8519df07d529456c9e34074787ebb33f658

PC variants

TelB Variant

C&C servers

`afalr-sharepoint[.]com`
`afalr-onedrive[.]com`

Backdoor

MD5
315e6338bf9c9bcbe3d5af0482f51dfd
SHA-1
8b00d62a5c03efa76dfca8bd8c95c969167f83ee
SHA-256
a713a2749e9791243a89471a2603bf1f32ec11c9179771ca46fb5583b8412cb0

Artifacts

MD5
ecb8c2cc5efe580d4ea8f212e39eb9b5
c9a28ae2b52d13cc98cdaeaff6d72332
77d9ebb41bf12a96284747cbeeeed889

a7675a6eee18746705c90a9290168b60
01e4c30e374bd26a2e5e5cb8ef27b255
5844fe7ffb3333c23d201d70c7419a6d
1bd82146445e2dcb3cafacefe2e913ed
975b81ecf54f67e8d091be053ae7fa99
SHA-1
e642c9898b8d18238ca525e74db22e6dfe431e2f
ee96340d3b0845fcaad0ee328c49095302cee6e9
817835661f1e3be4ff13ed1762054475cc8e1223
A6e1f60d5e3651d1e029293fba7da72749282ca1
A778f565bbf851efe50a46476fe0e9f8b0e1c830
5d09311a4b0c18572dede3bbf5620268baf39318
0dc484e36b62cf4f2512e1b634dbfe60260c8447
a3b8eb53d595e3a272942e98eac24f3c38cfb2e4
SHA-256

b743c9b4968b65577d60d0f3a3c4ae6dd6beedf08a02625836d598f8600a1321
409da7a4f191e37d3d3aa8f36e8c3789fc998b63241a5f05c6816e54ed7dcd3a
41629c54b2f3dd68897c04a8ed10f7c78534ba67a048da75885a857f68b37624
F9f4aaba897b15f8c77c46f2efb0672b044b7cb79dfd84eac4a41e2f1cee1344
Fdfcf1790faf4dc97ea7c5d84c76b7abbdb080ab931777a6259b09ae0166fcae
233ee2ea02322d3da68217ab4b51722a4a3aa833667a45377dfd4742d5979c4c
512e28afe8d32008cd8a9e95c938d2551689098ea93f75ba2a23c246248d7124
4c0c33fff8d4929f7a0d742f1d251b61794b185538b8ceb4939283d1b3d73795

TelAndExt Variant

C&C servers

`exemplifiable-taps.000webhostapp[.]com`
`telegramup[.]com`
`148.251.97[.]102`

Backdoors

MD5
281908f5afa399f725a06df767486837
5b813b679779a60947d4ed6e671394b0
A763350f2a5b2fdde3216cd1ea2bec5d
2e4e20bb01c9ca4ef5df2a75473c1aee
0aa07a6bf12a2a87a66202e768146e49
5666585faaf4fe77c8354ff76881f29b
SHA-1
67a328fc2362253fd7cc9163d7da6d8688d76d1f
E541372d93e4e26fe75fb44eb8aa009e1fc48b38
3275c02dbcb2b3467b55bb6927e2d80aeab43357
107b5afd843a53715ca89dc9b180a0f761a87f90
ab6ccfe1c9a27c1225dbe94a85246656837ce38
b7397af85faee45c3d9e0f2e7c0e1b248f064317
SHA-256

cec533ecd881f014efa7416867d6e3c6b4362741e97c1609860c6223935dec8d
21118e91cc1537c849a382d87cb113568c5e6d6ce204e8f4592c26f74f713f79
b65676321e2138affd5c38a1f2b882f19ac1ca9bf414b6f3d44e35c43c36ae78
65a3dec040bddf615bd2ce8c9f08ff074442fb521ac97b869e51d35a417719e9
2d161588e7314ed268144b14bf00ff02b4b875f140d5ff8ba51ed50318e4b603
dc627b6419366cdf50eccfa3d1995c111b71112e5abb725b6096b9e0026af395

Artifacts

MD5
a330253626349a1f0a6f16255f05b5f7
a871124091acc7c865f34e9d4cc6b6ad
72eb19c60056174b7d5722cabed90ef8
99dab6b39475e1088a4dd33d4cad9896
f6a1a831d77cc6f2a2c636f7c17fd499
ad33e3d934fef9ed58b1f1c8b0fa0091
SHA-1

a208ecaa6ef313abedb3d07d168655af0de0287f
1c1d7bc97c49b046c5040c9a74aa803111b8b487
6a37014e9ff0df749f58c74f787608d66b039a43
b1755edc051acc27c04ae9f05a07db47cb816f57
86d5a8450b80627ddc900bc13d970a9917cf1586
5a60267edb2021e30cbf3540226562701232e512
SHA-256
e9bf479de992e8a7cfff4d5d528ec85614e9ad0892feb5f588047dd78decf069
4ea4671ef8678197dbc82a584832d0dd23d67b0427873ac610bb266d0678f305
baad0de1026a3a807c4e4170b9291548afa900614a1dfdc00cf4f63d1946d555
1ffd162d377b84ddb91766f43c0a7a0ba92f358fa2146a33726ed1e08529a691
509ab695001be527b6c32f2d200067f2d433169e86724336579e08ea44799dd6
baf779a4a3c9d901eff32a46a004bbb258551cac57d63f0a878d882d2ebbdcf3

Python Variant

C&C servers

`tbackup.000webhostapp[.]com`
`vareangold[.]de`
`telegrambackups[.]com`
`telegramdesktop[.]com`
`picfile[.]net`

Backdoors

MD5
aac5bc1f94f32a69d7dcea33f305e6fc
9238f7a1ec7cbeb3dbb9370f02fde040
30973d4a637354cad945ab94205b0323
SHA-1
16335373c2b9438002fbe3a648a0709d8c111a6b
a9480fa19e90e46f9fd4a3c96e5ad08c11ef3822

94f9ee0dbd13014b19f42c2fa125f3f9e73b98a3
SHA-256
3310c0b2fd8a8d96288eb241f6948cfa0f15b39d2e6ca6687aab45dc6fccf9fc
13e924700a346234eaf2376c61ef0a36c86d94847b232a4ad772e35e0b9a6e87
d52a5ece34828b4201df630a7bc07449289f0c15833ee13f93f105c510a8282e

Artifacts

MD5
67f523757199203a5e4eae3e17ab00a4
2f1120f5089af58315891fd316333161
b9a888a23af000c6d1c846b9d0fd853c
c7041a9de03af5c2c85ec70c3e8daefb
fb063ebd13296eef1fd556ebb4d843a3
b44428524ea196992358148ee3eeddb0
b99abe396772819815eac7728580f41e
5fcefebf48018774f278f5fa83c664b3

cc95e164fc390fa3b75a2c49518edbb7
854418d163b0e1269970338916ff6374
ca1e45cd176751931c87edbf25aa4469
SHA-1
4807035760bd758cfe05adf81da2618914928a62
ba4b04a8b20cff6ba27ccf7e79f4bcc8134e1c2c
6537f6ea9f0a3edb5469c7235d70571e5a46c3e1
1e6a569979dd3cac95d9d1c481ebf9bb1e0b1f12
ff4af69cdc3c24a7f10efa23c9b1431751c1f0f0
767c02bbaf80745dfb0a6438c21927beb2123962
b14804d46febfe811cf5634c8059666bf5c6fc55
9ae405cbb9c6e959e4f680e2a73952e89c81ab4a
b110923d4ec5cf737bfff3904b1527b041ecfe58
295f01317d14e1548ecdfad1342cfae844f5dd8d
5b15fb002162591bab0067a5c15c7e5c1726dc24

SHA-256
c004fa7111c5ea0d902a7f9863b525fb26a3be086926f39246f0dbcb7804f2b5
30c71764ff80f82a190fc7d2212f0b7eebde4de46327f34e3326acbfd87f268d
0da88a1645f39b41e8cdfe14eaee40b8845bf92b446ddc646fddc85389b78495
4ffbf798a68aa5bcc5a52efd64456483172be892125085d2c82e2f351a48342a
d8395183c234836b9138d0ade196b8ab60aae6add8c84e004df049a27afe5ffa
fe15c79508885b5288c5cf93708d5b40eab05877cb9b1d954ab7e814a20c7978
ee295bd3669ddaebcd9be020debd1853c6eb7029c8017734e44c8cdce5e15241
815a89091ed15779071bbd6d7ad207a0041a199a562f105595278258880f1e03
3010d9eddb0b97b7f61025d05b543f572c7900170240b56bd9568efb79799f11
b5e571eb492eaee853abdf8b6202f7e543f09d8343a85f467cd4806f8e19a14f
e444a49b260e815c7d2f3e309f7c7b62226d4f0658fc756ec0aed5effb5226a8

HookInjEx Variant

C&C servers

```
tbackup.000webhostapp[.]com
```

developerchrome[.]com
firefox-addons[.]com
picfile[.]net
cpuconfig[.]com
update-help[.]com
winchecking[.]com
endupload[.]com
176.31.4[.]14
148.251.224[.]29
144.76.177[.]244
137.74.153[.]98

Backdoor

MD5
ca554a866389796b65f0b5eb1576e691
3bcddbfd757de15ec350f1b4c9e92926
74c3049ae9229675ccce544f0491e2f9
a05b6a10d7643a2ef059d7e296cb87a6
c887fc425351a824a143a015d51ad0a7
1d6a516c77aaf1bbab1ac4051f86475c
f9b9b9e2c87f9f4b5fbb89e5a1ac05eb
48873bf5f51ed996b237ce3495bf6219
f4e7111f9a5cd4451d422bc009844ec5
SHA-1
7ba64923c79cb2742393ff1ad9cb9fd3f6660024
b136739bf5c161684433a94a80ccaa9db029bac8
142f7bd57d3623fd44f5d7406bc9dc8b0fba0bd8
8a7f8d1dcbb9c5d4766f49e41ad17c00776bdb50

4e4a8dce1192769ac447ffc41a39df543420e1dd
96af18c2f4afbcde3854a53c1a3bbb964296b241
31922929229c7b49c626ecdaf2e3927683fbe0cc
eea85b2b8fbb5724f58424c1878ac10fd343a155
abb636cbab0bc591ba94203f41635fff009304b6
SHA-256
55c5a17976d253c7c4df1b59973c610336c5482d2063d511d54d512fe04ca
a4fcc308e9a364d29057cc76dbe6a8c32ce24a1dbae5c0b6306471f61cbefb29
79baf679e84b02a660e03602ff7aa4c9c86a92e0885b1a298c672db842be258d
58018aac8beb89271ef88d0fd4ada64079e1af09fad441e7b39a2463f95602f4
3cedd91bb4c7a5874a3ad286addb0860c33931ceb09d2c18385b7d6cab6953e0
a60f5b41251d0bf126fc3c2b836de7d59aa608fd6d37726d71960dd408575512
5a8f53f7c65af0cb3f269f8653405cd7bd98fae5c256e6264e5ebc5f75ea6c08
08b61faed24b35224a505dd9cbf39cd59776627de7991161d376134a854c3227
3ff1864e5fe1ebcce0a60c9594c9ac9f2eedd94367680dc3d77ca39a0b0e3d06

Artifacts

MD5
cf4ed89d96dab84a455a4f52400388cd
16706dff8db6fcc1fbd6f80cfb2baeb1
d26a8b8d1c6f77dd9ecc02e0edeecbaa
092b436347f80cffd74f4caffa75f4d5
663f9b47a983c2ebe9f70df74956dcc9
3ef7daf8cbce7a9aa68ee5c0baef8b28
f78855f488ce965a6a4c60820df2e696
62d8c20b64281b0d934358bf8d0fd2cf
80c9fc38e7f6d96a09feaa99b7777e7d
6f5a36fb82de3aecd847978846be312e
1ac4f4f7c5217adb16d83f902e51624a
661dee790ea438b14553e622052909a5
a68fcf5b97265d97c6bc5613ae82c093

dd07291265098edef72d39b11c8a1e37
87866ae8936ec3fc04af3e0783ec36bf
db4c95ea37fed6403546eadd9e691a1e
ccb6f24ff38770ab2efeb8f51de2a123
e7c0e92855a1b7d9b81eeb06cce5ce60
326843b42fca324e9fd023058a6c6b7a
68b84d8057f6e6def9e191ed218da0ee
87878f5404083c4c0ebf7a78e386a487
f1598a2901388dc5244931226d300633
2d64174dc0bed8222eea4494a49744a5
20691b32c1839cb1e106f937dd101e4d
86320eb8adb48106b899e21be5d5387d
e20f58c1afb7d9262e5a15620b172bd4
cb93aad2354aea2623a70abdd9ecc87b
91be9e93c7602202963650103ff8ee50

e130f1305948f0f7bd25f9d7101bd98a
2c8a7d32667b7b7c410f3b3347087996
f55277807457e2a3e9ad4b6de64b549f
64bd09506365a0cf351a56edb2bb2bdc
5c4b2cf2bed7db57b7335ec426fe776b
470175c447f025f4057b4dbacb931e42
f499cabf7c2ddacad965ed2a086b481b
83cba14904fdbf0e21d251fc5ab00666
a314ff2714660be06f9eb49e6024c8c5
bb186b0f3f2a1e0ef51d86d3494fd3be
2e8e25f179172778f8efefac33f2dcb7
b547b27751022900d9126a82d82a411f
a0c46b3f8370f2a2a6486d0ac686363c
SHA-1
7bbfb347a762da6be65484a2d721669269099af1

a9de74562a373fae1e02b6f290c3c4189f9f52c5
4e4570200d81ab296f29dcbc56c8371484114077
2c55956f5422a5fd08e11042d49e6fa478b9cc2e
9d694ca2a311eafc409f128e1044162ddea5687c
cb765cc4028a1c2e6930aca826567bc8253d8479
4511d3627b2432e18c02271ac9ef67a373d2dc4a
a1c8b69ca2f6f8763e65bdb148c9f9422130fced
7fae11c9f144912eef2557b21f44b112857f2bee
a0bffcd0d9ab5651476375b1e0edef18b81c2d90
ddb494f286c36c4216b3e325b8e8e4e61f1c7906
8963a67f5001a3ee5459a8ebe1e8fa3059df786b
c66b1ac78b55661cfbf14c330c2b9615d6c15125
f42195c131e1bab859aa61f52edf37c587288eaf
2747b43c07845feb832115f992c3ff08f2ac220b
93b3a4d118131981fff5f65da2f8642947f2e43a

87376e4522a673d5326a456dc6cc11e5c8349dbf
1af2cb91b45f684d5fd30187643d9d2d51474be7
8c59a117faed95777e15fefe0a2ed34d492e3205
ad0a4e312d21e513a3fec0bc7bf27afdde4bcde6
694ddbd3d19ac153f29d52f350faccb257fec841
31c85366409d5b5ae5f87da2b60f8f116b4bec99
c8d2b9ff069c7e3977e988cebba273bd320abfcc
2c13c0ef28485320634c235a097d62017bed36d6
a8548208fd950a8215e8ae0fac0d00db2592ccf0
6e97334921c15cc27ccfb1e147a74d69f873ff64
2d69897eccff1efe908c69c2d0af81f9fc7a57aa
dbd60bf24dc0099a4b45b2610be91b5cd75be31a
f5b3dc229f00e4c726a9e6e990ad4983ede0f073
cc8f8ae46807c1b6b56a1877628d2140f0158b85
3da61604ca8c6da190906ff122d56e1cb9836f4c

b8738bbe9c35181ea4261b81b6c9fc58d8bb593d
cbdf5c9ada304b73cebda7753bd14bcb5cedab2e
db88c16dc592d5b11445fa0f437016651706bfaf
02fe03f6f2914551e7096b7938ff1b6d7dce17e1
bbb38fa43bbb8c984cb70b155c230539f6ff6e51
749a8aaf2f9f96a914e3dfff76ff9c9fa43c5bf2
311a4fdb018baf924ff1301dd489b822b40f6c51
22a3855c9c9c05e1789d45d40ba325d9406a1f3b
e036dbf4b0e6b36526f8b4f180ac624cfdc8f756
d7d58a818649eae5116ab26351993436fc1255ee
SHA-256
4415e6240b037f4ac693c7e4a88f5ab2567b68dddbaa8fbfb0b40d37748fa8ba
2c4156bb1d1e3f0abafd5d03fad277f6aab705cb917bc07e05de3170fd80854f
69cbda8c2ea92eace49d678cc660432d0ad0c44bd79c3a02dd841066f80bc51b
525e99feb0a32a96aaf6e34be899e6a68c7abb6a8542f30e3822d07fe4e8d278

54a20f35d302499c925e5855f782bacb6bdd0a345f57c9e80772ef29fb81f465
083fe2c0feca89a6011ea2749123e216e0a53b573ebef2f25d856412cee7f99c
51a9a7e764a509b979dd438719840369718a320acbba32abbf51d4926e7d3486
b7730f9a05be8a0f25a3979b2f8d2fed791340a32385a9fd37d0e8b81119627d
63a655fde88ea26c73cea1e1764305e44203db771f64155b3b3e3d805203f65a
5eb4c94c9927e90426b6227754ae97fca06d468d5512d15773c48817ea082dbf
dff78dc100c1efd116de1a1d9e0b9169380801a1e7e864d63dc81a263f8929e8
845a0e5720a6288794a6452adb8d3e7c22f5e6e6b9d4f7481fbd30e3efba4f28
b778ab921e7268334efdc8aa371909c4bbd0f1621e39ab9d7e37167fe448581e
0e4a8eb2fe861c45071626da24147e922b167efb543e37ace7466c74c1e98be6
0f7082926241659fbebd229cdc41abe358be49110a80729b9ee891f2f7dcdf16
71085b661fea6cf040586b462b07ce8e0471fb9208c4f69cfd168e168beab6fe
37f40214d2f150597c52cb868c1e2f723d9c2d3155ab18ab2f1279eaf09bdf71
f211a92c2e215c2691006407bc919a892dd998120d83d333f2295059cd3c1c60
1b8cd7c93dce63878dadae0cf77482ae367477841a4604c6a842158466790737

d148562a49a09333b2b02d13e12b183d4c3fcf23fbb024d4e0b440631a3a3663
e4e210aedf8120a4c765bd340bd78b4a84f7ee486314132a8364fd417f4fa128
41d3378e99a410756170056e4941e86325826c45389ae18172114be535a73355
e7782cedc67fe36d2fb9005c5bb165c75db9587f3de57b408acb20f6757c7f56
09f953c4abfa799e2137887db5e90ddb993f76d20ce22a5ca290e43ae07074b7
023151cf0fb47d758946fa85a952a2b6758fbbfb762083a01bb70c5a6d96c781
2e656ea0b05ffa6cd945848176d1c9fb6174a6253b2a42891487d120358f0bec
07247bb81cca445e0df110d73ea6bf7eb327cc99b614b99dfbcb5632025c99a0
085a42cf3705bade9cd970f003f82158563aba06e9152e00928778bc0bd9585e
b26b024fa7be56d2b2e3815d8e97434f95b30bf25cda4259d3e20c14a92bd8ec
d3bb736d8a8b500c75ad853392afac37fd8cd519b274db4cba9451d2f1899059
986a9bd00d5b22431ab949916828aa25542afae4875b5cee00f703424b5ffb34
de339d3fe5acf83a0df5991bcce02574e1f2c4749b6d0e8f9edc563ef4f91d79
0af51a0ffb5798fb90a14070809fa9909195068ad1b91c1cadf5633b521e5132
75972d15f3b2e97d52b9f8a6f42ea85976ed5bb9d609c3bf93ee98d6f4f4a648

35e3f08ae93a7b4cd3e77a6438e318cd3c3b41efa5def52e5ebd182347e94fd9
af31cc534aa49f02e6c18a8cf3fd4c9cf366d462ee7caaf8c2a461405382073f
013edd19a9e796d54b82dc34a400a0981c5e17fd65a235dd45231e7ef06ee53b
e8f785efb62fbdf31a12012d38798301329e5262090991152e94342ef6dfa276
e7eeb7781f521ddc5481626a2410ed8cc871809c36d8d8f74af9dd3f8c42505d
9c75a6957a0294d929787b6e8217e4127b77cc2702c19ddb8e0b6319dc3b5127
bfb2a7f8e7396f8edee131eca9715ab8b2fc957478b7cf0d58840a707b718e09

Source: https://research.checkpoint.com/2020/rampant-kitten-an-iranian-espionage-campaign/