Malware Headliners: LokiBot

atomicmatryoshka.com/post/malware-headliners-lokibot

z3r0day_504

January 28, 2022

<pre>remnux@remnux:~\$ capa lot</pre>	kidump.exe
loading : 100%	579/579 [00:00<00:00, 1371.18 rules/s
matching: 100%	203/203 [00:07<00:00, 26.64 functions/s
md5	e9d4075a81abce614c259908323438a9
sha1	5d6a92cccb58163bee4355c0f2844d9b96ca2548
sha256	71e155ee000c0d1cbba18b92f0d512217afe195ba40f9326c60523cdfd3fa742
path	lokidump.exe
+	+
ATT&CK Tactic	I ATT&CK Technique
DEFENSE EVASION EXECUTION PRIVILEGE ESCALATION	Obfuscated Files or Information::Indicator Removal from Tools [T1027.005] Obfuscated Files or Information [T1027] Shared Modules [T1129] Access Token Manipulation [T1134]
↓	↓
MBC Objective	MBC Behavior
ANTI-STATIC ANALYSIS	Disassembler Evasion::Argument Obfuscation [B0012.001]
DATA	Encode Data::XOR [C0026.002]
DEFENSE EVASION	Obfuscated Files or Information::Encoding-Standard Algorithm [E1027.m02]

LokiBot, or Loki, is a password stealing malware and was considered the 8th most prevalent malware family in 2021 according to MalwareBazaar. Available for sale on underground forums, its within reach of anyone willing to pay the right price. As far as its use by organized actors, <u>MITRE</u> has linked its usage to the SilverTerrier threat group, known for having financial cybercrime motives.

In this blog post we'll conduct some static and dynamic analysis on a LokiBot sample to extract IOCs and characterize its behavior.

STATIC ANALYSIS WITH PESTUDIO 9.27

Using the latest version of PeStudio, we start to build a picture of what the specimen is capable of. Taking a look at the imports/functions category, we see the following:

□···	functions (155)	blacklist (29)	anonymous (1)	library (8)
indicators (45)	SearchPathA	x	-	kernel32.dll
····> virustotal (error)	MoveFileA	x	-	kernel32.dll
dos-header (64 bytes)	SetCurrentDirectoryA	x	-	kernel32.dll
dos-stub (144 bytes)	SetFileAttributesA	x	-	kernel32.dll
File header (Visual Studio)	CreateProcessA	x	-	kernel32.dll
ontional-header (GUI)	RemoveDirectoryA	x	-	kernel32.dll
directories (3)	GetTempFileNameA	x	-	kernel32.dll
> sections (virtualized)	GetExitCodeProcess	x	-	kernel32.dll
	WritePrivateProfileStringA	x	-	kernel32.dll
	WriteFile	x	-	kernel32.dll
	<u>FindNextFileA</u>	x	-	kernel32.dll
····⊶• tls-callbacks (n/a)	<u>FindFirstFileA</u>	×	-	kernel32.dll
	<u>DeleteFileA</u>	×	-	kernel32.dll
····· 🔂 resources (6) *	CloseClipboard	×	-	user32.dll
abc strings (3464)	<u>SetClipboardData</u>	×	-	user32.dll
	EmptyClipboard	×	-	user32.dll
manifest (aslnvoker)	<u>SystemParametersInfoA</u>	x	-	user32.dll
	OpenClipboard	x	-	user32.dll
imperiation (Nullsoft)	<u>ExitWindowsEx</u>	x	-	user32.dll
	<u>SHGetPathFromIDListA</u>	x	-	shell32.dll
	SHBrowseForFolderA	x	-	shell32.dll
	<u>SHGetFileInfoA</u>	x	-	shell32.dll
	ShellExecuteA	x	-	shell32.dll

Based on the imports, this sample shows potentially:

- Anti-debugging capabilities (EmptyClipboard, GetTickCount)
- Parsing through files and folders (FindFirstFileA, FindNextFileA, SearchPath, CreateFileA)
- Evasive behaviors/artifact destruction (DeleteFile, RemoveDirectory)
- File writing (CreateFileA, WriteFile, MoveFile)
- Registry interactions (RegCreateKey, RegDeleteKey, RegEnumKey, RegOpenKey, RegSetValue, RegQueryValue)

Looking at the strings tab, we see a lot of the same references to the API calls, especially if sorting for blacklist items to show first:

file settings about						
iii 🗑 📈 🗎 💡						
:\users\user\desktop\c27e339893d3e5fc1e61e73	encoding (2)	size (bytes)	file-offset	blacklist (33)	hint (99)	value (3464)
— Jul indicators (45)	ascii	9	0x0000654A	×	function	WriteFile
	ascii	18	0×000065EC	×	function	GetExitCodeProcess
— ▷ dos-header (64 bytes)	ascii	13	0x00006A58	×	function	ExitWindowsEx
dos-stub (144 bytes)	ascii	14	0x00006BE2	×	function	CloseClipboard
— ▷ rich-header (Visual Studio)	ascii	16	0x00006BF4	×	function	SetClipboardData
— > file-header (Oct.2008)	ascii	14	0x00006C08	×	function	EmptyClipboard
- > optional-neader (GUI)	ascii	13	0x00006C1A	×	function	OpenClipboard
- m directories (3)	ascii	26	0x00006DDA	×	function	SHGetSpecialFolderLocation
 V sections (virtualized) Sections (2) * 	ascii	10	0x000064F0	×	-	DeleteFile
functions (155)	ascii	13	0x000064FE	~		FindFirstFile
- vinctions (155)	ascii	12	0×00006510	2		FindNevtFile
tls-callbacks (n/a)	ascii	25	0×00006572	~		WritePrivateProfileString
NFT (n/a)	accii	10	0×00006595		-	SearchPath
resources (6) *	accii	9	0×000066500	÷		MoveFile
abc strings (3464)	accii	10	0-000066D8	÷		SetCurrentDirectory
	ascii	17	0,00005720			SetEile Attributes
manifest (aslnvoker)	ascii	12	0x0000072A	×		CreateBrocess
- 1.0 version (n/a)	ascii	15	0x00000800	×		CreateProcess
overlay (Nullsoft)	ascii	15	0x00006872	×		CetTerreFileNerre
	ascii	15	0x00000586	×	-	Gettemprileivame
	ascii	20	0x00006AA8	×		SystemParametersInfo
	ascii	15	0x00006D76	×	-	SHFileOperation
	ascii	12	0×00006D8A	×	•	ShellExecute
	ascii	13	0x00006D9A	×		SHGetFileInfo

Outside of those, nothing is proving to be too conclusive here.

I pushed the file over to REMnux to give a stab at it with capa. Capa gave the following output:



In order to analyze further with capa, I'll need to dump the actual malware executable once it starts running during the dynamic analysis stage and rebuild it. I'll have those details after the dynamic analysis section.

DYNAMIC ANALYSIS

Prior to detonating the sample, I had Process Hacker, Process Monitor, and WireShark running to capture any events. I was able to capture the following data:

	Results - c27e	339893d3e5fc1e61	e73ffafac8a7bcf76813a92f — 🛛	× "			
de free reg 1	,268 results.					Strings	Refrest
e address	Address	Length	Result		Total WS	Private WS	Shareable
0x10000	0x9e160	34			68 kB		68
0x30000	0x9ebb0	32	YMACHINE/Softwa		64 kB		64
0x40000	0x9ec20	32			124 kB		124
0x60000	0x9ec60	52	ies/Microsoft/Cryptography		32 kB	32 kB	
0xa0000	0x9ee90	58	C:\Windows\SysWOW64\ntdll.dll		20 kB	20 kB	
0x1a0000	0x19c022	20	rofani.dl		16 kB		1
0x1b0000	0x19c038	22	sechost.dll		4 kB		
0x1c0000	0x19c12a	20	rofapi.dll		8 kB	8 kB	
0x1d0000	0x19c620	28	C:\ProgramData		68 kB		6
0x1f0000	0x19c660	20	C:\Windows		12 kB		1
0x200000	0x19c684	32	gram Files (x86)		56 kB	56 kB	
0x400000	0x19d4f8	38	C:\Windows\SvsWOW64		648 kB	648 kB	
)x4b0000	0x19d730	54	C:\Windows\System32\NSI.dll		68 kB		68
0x4d0000	0x19e2ec	16	s442136.smrtp.ru		60 kB	60 kB	
)x4e0000	0x19e6f4	16	s442136.smrtp.ru		4 kB	4 kB	
0x4f0000	0x19ee9c	32	s442136.smrtp.ru		68 kB		68
0x510000	0x19eebe	32	s442136.smrtp.ru		12 kB		12
Jx520000	0x19ff12	26	;OLEAUT32.dll		8 kB	8 kB	
0x530000	0x19ff30	20	ws2_32.dll		12 kB		1
0x540000	0x1c03e0	12	AppheloDebug		824 kB		824

The memory strings in Process Hacker offered some IOCs. We see a domain (s442136.smrtp[.]ru) as well as some registry interactions.

Process Monitor offered the following:

2:18:5	c27e339893d3	10436 🖺 CreateFile	C:\Users\User\AppData\Local\Nichrome\User Data\Default\Login Data	PATH NOT FOUND Desired Access: R
2:18:5	ac27e339893d3	10436 🖹 CreateFile	C:\Users\User\AppData\Local\Nichrome\User Data\Default\Web Data	PATH NOT FOUND Desired Access: R
2:18:5	ac27e339893d3	10436 🖹 CreateFile	C:\Users\User\AppData\LocalNichrome\Login Data	PATH NOT FOUND Desired Access: R
2:18:5	ac27e339893d3	10436 🖹 CreateFile	C:\Users\User\AppData\LocalNichrome\Default\Login Data	PATH NOT FOUND Desired Access: R
2:18:5	c27e339893d3	10436 🕑 CreateFile	C:\Users\User\AppData\Local\RockMelt\User Data\Default\Login Data	PATH NOT FOUND Desired Access: R
2:18:5	c27e339893d3	10436 🖹 CreateFile	C:\Users\User\AppData\Local\RockMelt\User Data\Default\Web Data	PATH NOT FOUND Desired Access: R
2:18:5	c27e339893d3	10436 🖹 CreateFile	C:\Users\User\AppData\LocalRockMelt\Login Data	PATH NOT FOUND Desired Access: R
2:18:5	c27e339893d3	10436 🖹 CreateFile	C:\Users\User\AppData\LocalRockMelt\Default\Login Data	PATH NOT FOUND Desired Access: R
2:18:5	ac27e339893d3	10436 🖹 CreateFile	C:\Users\User\AppData\Local\Spark\User Data\Default\Login Data	PATH NOT FOUND Desired Access: R
2:18:5	ac27e339893d3	10436 🖹 CreateFile	C:\Users\User\AppData\Local\Spark\User Data\Default\Web Data	PATH NOT FOUND Desired Access: R
2:18:5	ac27e339893d3	10436 🖹 CreateFile	C:\Users\User\AppData\LocalSpark\Login Data	PATH NOT FOUND Desired Access: R
2:18:5	ac27e339893d3	10436 🖹 CreateFile	C:\Users\User\AppData\LocalSpark\Default\Login Data	PATH NOT FOUND Desired Access: R
2:18:5	ac27e339893d3	10436 🖹 CreateFile	C:\Users\User\AppData\Local\Chromium\User Data\Default\Login Data	PATH NOT FOUND Desired Access: R
2:18:5	ac27e339893d3	10436 🕑 CreateFile	C:\Users\User\AppData\Local\Chromium\User Data\Default\Web Data	PATH NOT FOUND Desired Access: R
2:18:5	ac27e339893d3	10436 💽 CreateFile	C:\Users\User\AppData\LocalChromium\Login Data	PATH NOT FOUND Desired Access: R
2:18:5	ac27e339893d3	10436 💽 CreateFile	C:\Users\User\AppData\LocalChromium\Default\Login Data	PATH NOT FOUND Desired Access: R
2:18:5	ac27e339893d3	10436 🖹 CreateFile	C:\Users\User\AppData\Local\Titan Browser\User Data\Default\Login Data	PATH NOT FOUND Desired Access: R
2:18:5	ac27e339893d3	10436 💽 CreateFile	C:\Users\User\AppData\Local\Titan Browser\User Data\Default\Web Data	PATH NOT FOUND Desired Access: R
2:18:5	ac27e339893d3	10436 💽 CreateFile	C:\Users\User\AppData\LocalTitan Browser\Login Data	PATH NOT FOUND Desired Access: R
2:18:5	ac27e339893d3	10436 🖺 CreateFile	C:\Users\User\AppData\LocalTitan Browser\Default\Login Data	PATH NOT FOUND Desired Access: R
2:18:5	ac27e339893d3	10436 🖺 CreateFile	C:\Users\User\AppData\Local\Torch\User Data\Default\Login Data	PATH NOT FOUND Desired Access: R
2:18:5	ac27e339893d3	10436 🖺 CreateFile	C:\Users\User\AppData\Local\Torch\UserData\Default\WebData	PATH NOT FOUND Desired Access: R
2:18:5	ac27e339893d3	10436 🕒 CreateFile	C:\Users\User\AppData\LocalTorch\Login Data	PATH NOT FOUND Desired Access: R

We see a ton of "CreateFile" operations with browser file paths. It would be easy to be misled by the fact that the operation title is "CreateFile" and believe that the executable is attempting to generate files on the victim system. Reading Microsoft documentation offers some more context:

CreateFileA function (fileapi.h)

Article • 10/13/2021 • 29 minutes to read

Is this page helpful? \bigtriangleup \checkmark

Creates or opens a file or I/O device. The most commonly used I/O devices are as follows: file, file stream, directory, physical disk, volume, console buffer, tape drive, communications resource, mailslot, and pipe. The function returns a handle that can be used to access the file or device for various types of I/O depending on the file or device and the flags and attributes specified.

To perform this operation as a transacted operation, which results in a handle that can be used for transacted I/O, use the CreateFileTransacted function.

Not only does this function allow for the creation of files, but also opening them. On the right side of the Process Monitor screenshot we see the value "path not found," meaning that the malware tried to open or access the browser file paths and they did not exist. Being that the malware is a password stealer, it is likely checking these file paths for saved credentials.

Seeing the WireShark output, we see information that corroborates earlier findings:

	*Etherr	net0				- 0
File	Edit	View Go Ca	apture Analyze Statistic	s Telephony Wireless Tools He	lp	
		💿 🚞 🛅 🛛	🕻 🖻 । ९ 🗢 🗢 鼞 👔	🕹 📃 🔳 @, Q, @, II		
	dns					X 🗆
No.		Time	Source	Destination	Protocol	col Length Info
	1089	84.348713	255.22.168.192.in-a.	192.168.22.133	DNS	105 Standard query response 0xcda9 A au.download.windowsupdate.com A 19_
+	1153	95.320881	192.168.22.133	255.22.168.192.in-addr.arpa	DNS	76 Standard query 0x5068 A s442136.smrtp.ru
-	1154	95.322228	255.22.168.192.in-a.	192.168.22.133	DNS	92 Standard query response 0x5068 A s442136.smrtp.ru A 192.168.22.128

So like I mentioned early on, the initial file is an installer and not the actual child process that we've analyzed in the dynamic stage. I'll demonstrate how one could actually get a "tangible" version of the malware that's executing to then analyze it with tools like capa. The tool we'll be using is <u>Scylla</u>, and imports reconstructor developed by NtQuery and available on GitHub.

First, make sure the malware is already running. Next open Scylla x86 and attach it to the active process. Click "IAT autosearch," and then "get imports," followed by "dump." Name it something intuitive, and voila. This will reconstruct the executable.



	×
--	---

File Imports Trace Misc Help

4860 - c27e339893d3e5fc1	e61e73ffafac8a7bcf76813a9	2f91ecfa38535210d6c7	7a7.exe - C:\Us 🗸 Pick DLL
	Impo	orts	
	unk: 00015000		
	unk: 00015018		
	<: 0001504C		
🗄 🧏 ? (2038) FThunk: 0	001506C		
🗄 🛷 vaultcli.dll (7) FThu	nk: 0001A0E4		
🗄 🛛 🎇 ? (1) FThunk: 0001	A110		
Show Invalid Show S	Suspect		Clear
IAT II	ıfo	Actions	Dump
DEP 00403225	TAT Automorph	Autotrace	
A 00415000		Autorace	Dump PE Rebuild
A 00413000	Get Imports		Fix Dump
ize 0008696C			
	L.	0	
		9	
getApiByVirtualAddress :: N aetApiByVirtualAddress :: N	o Api found 756AA39C		
IAT parsing finished, found	26 valid APIs, missed 2039 A	PIs	
DIRECT IMPORTS - Found 0	possible direct imports with	0 unique APIs!	
	the PE image, requires rebas	sing!	
WARNING! IAT is not inside			
WARNING! IAT is not inside Dump success C: \Users \Use	r pesktop vokidump.exe		
WARNING! IAT is not inside Dump success C:\Users\Use	r vpesktop vokidump.exe		

I pushed this version back over to the REMnux box for analysis, and it worked fine with capa:

1	<pre>remnux@remnux:~\$ capa lok: Loading : 100% matching: 100% </pre>	idump	.exe 579/579 [00:00<00:00, 1371.18 203/203 [00:07<00:00, 26.64 fur	rules/s] nctions/s]
	md5 sha1 sha256 path	e9d4 5d6a 71e1 loki	4075a81abce614c259908323438a9 a92cccb58163bee4355c0f2844d9b96ca2548 155ee000c0d1cbba18b92f0d512217afe195ba40f9326c60523cdfd3fa742 idump.exe	
	ATT&CK Tactic	+	SCK Technique	
	DEFENSE EVASION EXECUTION PRIVILEGE ESCALATION	Obfi Obfi Shai Acce	uscated Files or Information::Indicator Removal from Tools [T1027.005] uscated Files or Information [T1027] red Modules [T1129] ess Token Manipulation [T1134]	
	MRC Objective		L MPC Rehavior	
	ANTI-STATIC ANALYSIS DATA DEFENSE EVASION		Disassembler Evasion::Argument Obfuscation [B0012.001] Encode Data::XOR [C0026.002] Obfuscated Files or Information::Encoding-Standard Algorithm [E1027.m02]	

We can also feed this new executable back into PEStudio for its new assessment:

<u>ivp=kv</u>
kvp?kv
<u>\$@0123456789ABCDEF</u>
UNIQUE
SQLite format 3
DIRycq1tP2vSeaogj5bEUFzQiHT9dmKCn6uf7xsOY0hpwr43VINX8JGBAkLMZ
Lu., //

An interesting catch that I didn't catch before is the reference in strings to SQLite. A lot of browsers saved passwords in SQLite databases which, if we didn't know what this was ahead of time, we could safely lean towards it being a browser password stealer.

IOCs

File Hashes:

Installer:

c27e339893d3e5fc1e61e73ffafac8a7bcf76813a92f91ecfa38535210d6c7a7

Dropped executable: 71e155ee000c0d1cbba18b92f0d512217afe195ba40f9326c60523cdfd3fa742

Domains:

s442136.smrtp[.]ru

REFERENCES:

LokiBot Malware (CISA)

New Campaign Sees LokiBot Delivered Via Multiple Methods (TrendMicro)

Scylla