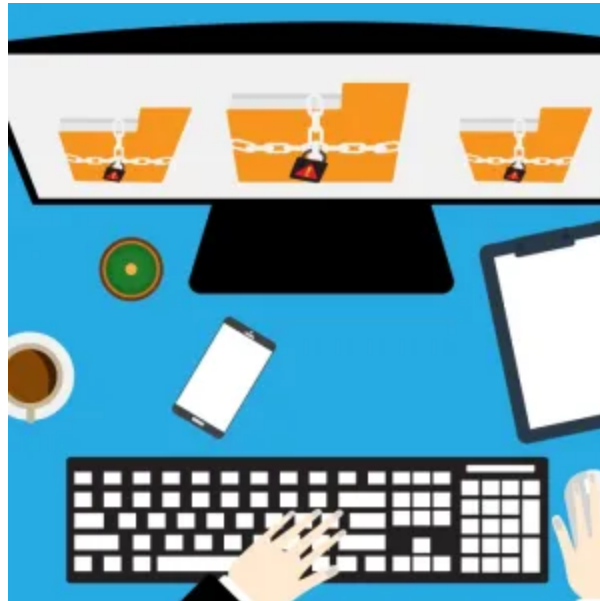


New Locky variant – Zepto Ransomware Appears On The Scene

securityaffairs.co/wordpress/49094/malware/zepto-ransomware.html

July 7, 2016



July 7, 2016 By [Pierluigi Paganini](#)

New threat dubbed Zepto Ransomware is spreading out with a new email spam campaign. It is a variant of the recent Locky Ransomware.

The news was recently reported in a [blog post](#) by the Cisco Talos team:

“We are watching Zepto very carefully. It’s closely tied to Locky, sharing many of the same attributes,” said Craig Williams, senior technical leader and global outreach manager at Cisco Talos.

“There is still a lot to learn about Zepto. As far as we can tell, it’s either a new variant of Locky or an entirely new ransomware with many copycat Locky features,” he said.

In the last week, experts observed more than 140,000 emails using a particular naming convention to deliver a malicious attachment.

That email is generated by a template body text, where it fetches the header greeting randomly from an array followed by the [NAME] of the receiver.

As previous variants of the same malware family, the text of the email attempts to trick the victim to open the attachment.

The attachment is a .zip archive containing the hard-coded js downloader.

The naming conventions used to rename the js downloader have the following format “swift [XXX|XXXX].js” where X are some combination of letters (a-f) and numbers (0-9).

Once the js file is executed through wscript it downloads the main payload binary from the C2 Servers.

Many of them have a list of hardcoded domains for download the binary, other variations use just a few domains.

That is done through HTTP GET requests to define C2 domains and the server functionalities are implemented in PHP.

We observed through dynamic analysis that it uses the same technique of Locky ransomware to decode the main payload, spawning the process through wScript with the argument ‘321’, otherwise, the decryption routine will produce junk code and the execution flow will jump into that junk code and crash the process.

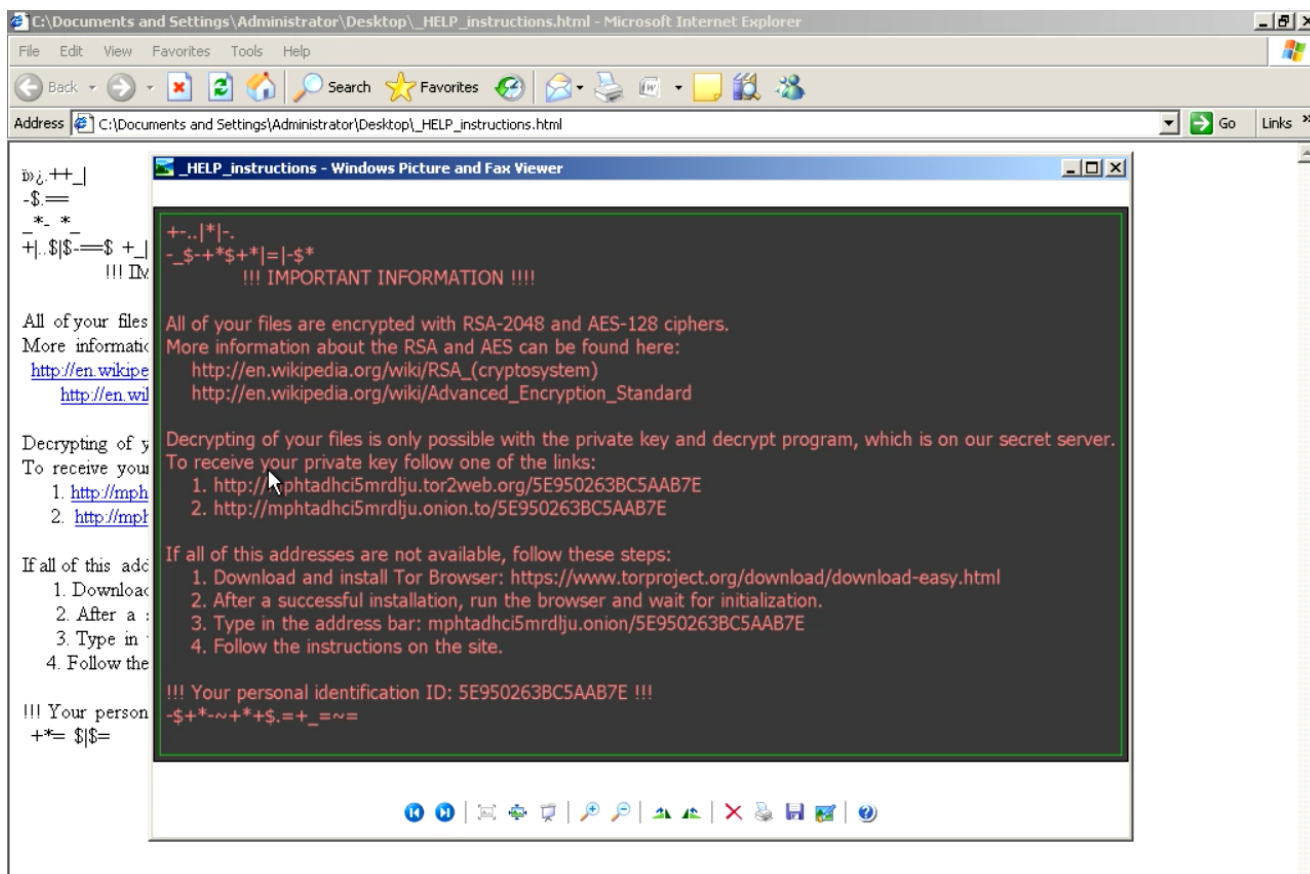
The encrypted files have the “.zepto” extensions and it targets the same extensions files of Locky taking care of the system files, it uses a lot of code of Locky ransomware to implement its malicious behaviors.

One of the smartest features of the ransomware is the fact they do not encrypt all the files needed for the correct functioning of the OS, otherwise, how can the victims pay?

Once the encryption routine of all the files is over, it shows the instructions on how to get the files back:

one picture (_HELP_instructions.jpg) and one html page (_HELP_instructions.html) are prompted to the victim for the explanation on how to unlock the files.

Following an image of a machine infected by the Zepto Ransomware:



Cisco Talos researchers tracked all the attachment they found and on 137,731 spam emails and discovered that there were 3,305 unique samples. They collected them [here](#).

OUR ANALYSIS

Our main contribution will be to find the actual code that differs from the previous version of the Locky Ransomware. We hope to help in detecting variation on some core features (as encryption routine, files enumeration, drive enumeration...), and to allow experts to distinguish the Locky ransomware family from the Zepto ransomware family.

We will do this through bindiff software that let us to compare two binary files and calculate the differences, we will use a Locky Ransomware sample with the following hash
 SHA256:e5a6828f732bea6b66c4f6d850b235f6c1f139b10f8d9f2c3760298cfd88c163.

So Cisco Talos researchers give us a good advice on where to start for this new variant, unfortunately, they didn't publish some samples to use in our analysis so we found some way to get them.

Search results for swift

Timestamp	Input	Threat level	Analysis Summary	Countries	Environment
July 4 2016, 14:23 (CEST)	b8385356c0ac7c3c9f67510be217921d04b483f599e875c55783d4915770ab_1467634230781_swift_010716_copy.jar Java jar file data (zip)	malicious	Threat Score: 93/100 AV Multiscan: 14% Matched 27 Signatures Classified as Agent.PA	FR	Windows 7 32 bit
July 4 2016, 3:38 (CEST)	swift copy.zip Java jar file data (zip)	malicious	Threat Score: 65/100 AV Multiscan: 11% Matched 22 Signatures Classified as Agent.PA	FR	Windows 7 32 bit
July 2 2016, 9:03 (CEST)	swift ca6.js ASCII text, with very long lines, with CRLF line terminators	malicious	Threat Score: 100/100 Matched 31 Signatures	FR	Windows 7 32 bit
July 1 2016, 19:35 (CEST)	swift transfer.doc data	malicious	Threat Score: 100/100 AV Multiscan: 15% Matched 24 Signatures Classified as Exploit.Rtf.Heuristic	FR	Windows 7 32 bit
July 1 2016, 18:14 (CEST)	Swift Copy.pub Composite Document File V2 Document, Little Endian, Os: Windows, Version 6.2, Co...	malicious	Threat Score: 45/100 AV Multiscan: 15% Matched 15 Signatures Classified as Downloader.act	-	Windows 7 32 bit
July 1 2016, 18:09 (CEST)	Inv_payment_swift.exe PE32 executable (GUI) Intel 80386 Mono/Net assembly, for MS Windows	suspicious	Threat Score: 43/100 Matched 19 Signatures	-	Windows 7 32 bit
July 1 2016, 14:50 (CEST)	pdf_copy-manuel-ortiz_802890.zip ASCII text, with CRLF, LF line terminators	malicious	Threat Score: 100/100 Matched 36 Signatures	FR	Windows 7 32 bit
July 1 2016, 14:13 (CEST)	swift 2c1.js ASCII text, with very long lines, with CRLF, LF line terminators	malicious	Threat Score: 100/100 AV Multiscan: 42% Matched 25 Signatures Classified as Trojan.Genecid	FR	Windows 7 32 bit
July 1 2016, 3:00 (CEST)	swift 06ae.js ASCII text, with CRLF, LF line terminators	malicious	Threat Score: 100/100 AV Multiscan: 11% Matched 31 Signatures Classified as Nemucod.F	FR	Windows 7 32 bit

We grabbed the most recent one in order to study the most recent variant.

The file name is “swift ca6.js”

SHA256:068e08f01e117f66f607a27492a500cc7c3ffa91cac76dceb97667394a9cde.

As we can see, the file has the same name pattern discovered by Cisco Talos researchers.

Now we will need to extract the main payload from the execution of the JavaScript file.

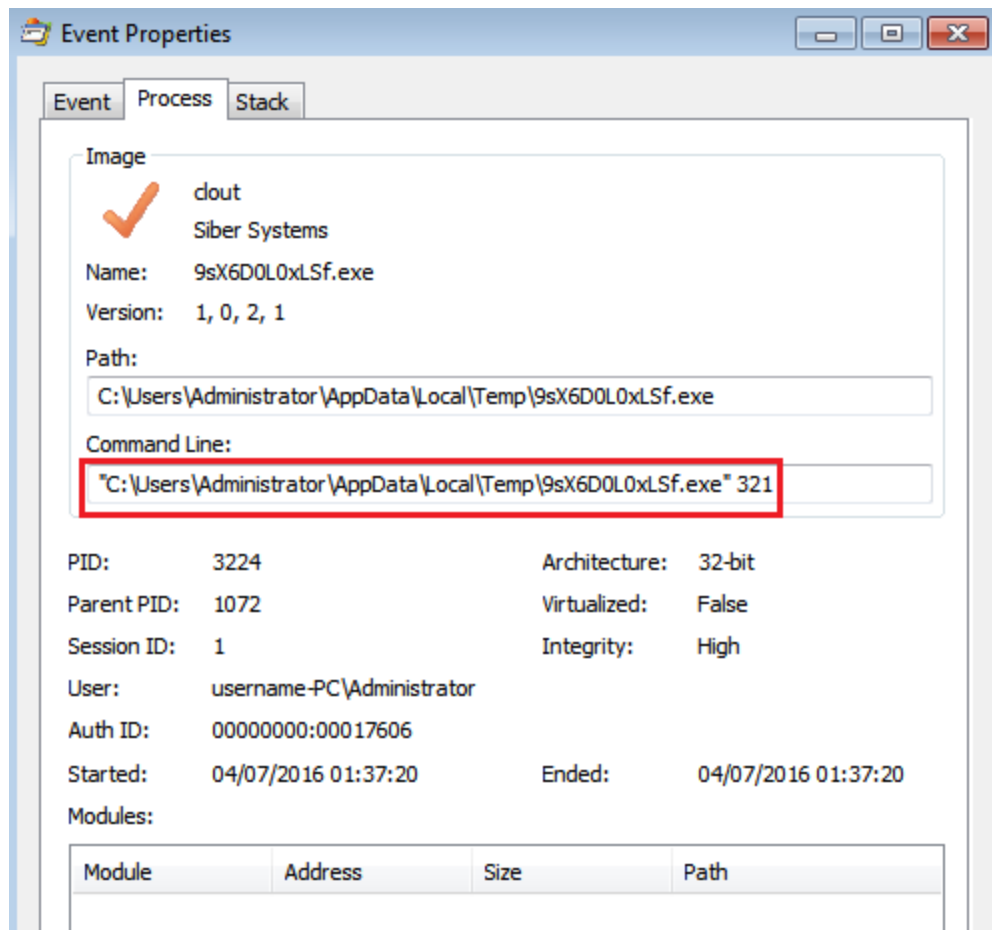
We will monitor our file system activities with procMon tools and we will take care on the dropped files of the malicious js.

Time	Process Name	PID	Operation	Path	Result	Detail
01:37:...	WScript.exe	1072	RegCloseKey	HKCU\Software\Microsoft\Windows\CurrentVersion\Policies\Explorer	SUCCESS	
01:37:...	WScript.exe	1072	RegOpenKey	HKLM\Software\Microsoft\Windows\CurrentVersion\Policies\Explorer	SUCCESS	NAME NOT FOUND Desired Access: Q...
01:37:...	WScript.exe	1072	RegOpenKey	HKCU\Software\Microsoft\Windows\CurrentVersion\Policies\Explorer	SUCCESS	Desired Access: Q...
01:37:...	WScript.exe	1072	RegQueryValue	HKCU\Software\Microsoft\Windows\CurrentVersion\Policies\Explorer\DisallowRun	SUCCESS	NAME NOT FOUND Length: 144
01:37:...	WScript.exe	1072	RegCloseKey	HKCU\Software\Microsoft\Windows\CurrentVersion\Policies\Explorer	SUCCESS	
01:37:...	WScript.exe	1072	RegOpenKey	HKCU\Software\Microsoft\Windows\CurrentVersion\App Paths\%X6D0L0xLSf.exe	SUCCESS	NAME NOT FOUND Desired Access: R...
01:37:...	WScript.exe	1072	RegOpenKey	HKLM\Software\Microsoft\Windows\CurrentVersion\App Paths\%X6D0L0xLSf.exe	SUCCESS	NAME NOT FOUND Desired Access: R...
01:37:...	WScript.exe	1072	RegQueryKey	HKCR\exefile\shell\open	SUCCESS	Query: Name
01:37:...	WScript.exe	1072	RegOpenKey	HKCU\Software\Classes\exefile\shell\open	SUCCESS	NAME NOT FOUND Desired Access: M...
01:37:...	WScript.exe	1072	RegQueryValue	HKCR\exefile\shell\open\SetWorkingDirectoryFromTarget	SUCCESS	NAME NOT FOUND Length: 144
01:37:...	WScript.exe	1072	RegQueryKey	HKCR\exefile\shell\open	SUCCESS	Query: Name
01:37:...	WScript.exe	1072	RegOpenKey	HKCU\Software\Classes\exefile\shell\open	SUCCESS	NAME NOT FOUND Desired Access: M...
01:37:...	WScript.exe	1072	RegQueryValue	HKCR\exefile\shell\open\NoWorkingDirectory	SUCCESS	NAME NOT FOUND Length: 144
01:37:...	WScript.exe	1072	RegOpenKey	HKCU\Software\Microsoft\Windows\Explorer\MountPoints2\CPC\Volume	SUCCESS	Desired Access: R...
01:37:...	WScript.exe	1072	RegOpenKey	HKCU\Software\Microsoft\Windows\Explorer\MountPoints2\CPC\Volume\{5d9a644-941e-11e5-a771-806e6f6e6963}	SUCCESS	Desired Access: R...
01:37:...	WScript.exe	1072	RegOpenKey	HKCU\Software\Microsoft\Windows\Explorer\MountPoints2\CPC\Volume\{5d9a644-941e-11e5-a771-806e6f6e6963}	SUCCESS	Type: REG_DWO...
01:37:...	WScript.exe	1072	RegQueryValue	HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\MountPoints2\CPC\Volume\{5d9a644-941e-11e5-a771-806e6f6e6963}\Generation	SUCCESS	
01:37:...	WScript.exe	1072	RegOpenKey	HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\MountPoints2\CPC\Volume\{5d9a644-941e-11e5-a771-806e6f6e6963}	SUCCESS	
01:37:...	WScript.exe	1072	CreateFile	C:\Users\Administrator\Desktop	SUCCESS	Desired Access: R...
01:37:...	WScript.exe	1072	QueryBasicInfor...	C:\Users\Administrator\Desktop	SUCCESS	CreationTime: 26/1...
01:37:...	WScript.exe	1072	CloseFile	C:\Users\Administrator\Desktop	SUCCESS	
01:37:...	WScript.exe	1072	CreateFile	C:\Users\Administrator\AppData\Local\Temp\%X6D0L0xLSf.exe	SUCCESS	Desired Access: R...
01:37:...	WScript.exe	1072	WriteFile	C:\Users\Administrator\AppData\Local\Temp\%X6D0L0xLSf.exe	SUCCESS	Offset: 0, Length: 1...
01:37:...	WScript.exe	1072	SetEndOfFileInF...	C:\Users\Administrator\AppData\Local\Temp\%X6D0L0xLSf.exe	SUCCESS	EndOfFile: 165 888
01:37:...	WScript.exe	1072	CreateFileInApp...	C:\Users\Administrator\AppData\Local\Temp\%X6D0L0xLSf.exe	SUCCESS	SyncType: SyncTy...
01:37:...	WScript.exe	1072	CreateFileInApp...	C:\Users\Administrator\AppData\Local\Temp\%X6D0L0xLSf.exe	SUCCESS	FILE LOCKED WI... SyncType: SyncTy...
01:37:...	WScript.exe	1072	QueryStandar...	C:\Users\Administrator\AppData\Local\Temp\%X6D0L0xLSf.exe	SUCCESS	AllocationSize: 167...
01:37:...	WScript.exe	1072	CreateFileMap...	C:\Users\Administrator\AppData\Local\Temp\%X6D0L0xLSf.exe	SUCCESS	SyncType: SyncTy...
01:37:...	WScript.exe	1072	RegOpenKey	HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\%X6D0L0xLSf.exe	SUCCESS	NAME NOT FOUND Desired Access: Q...
01:37:...	WScript.exe	1072	RegOpenKey	HKCU\Software\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\%X6D0L0xLSf.exe	SUCCESS	Information: Label
01:37:...	WScript.exe	1072	QuerySecurityF...	C:\Users\Administrator\AppData\Local\Temp\%X6D0L0xLSf.exe	SUCCESS	Name: \Users\Ad...
01:37:...	WScript.exe	1072	QueryNameInfo...	C:\Users\Administrator\AppData\Local\Temp\%X6D0L0xLSf.exe	SUCCESS	Parent PID: 1072...
01:37:...	WScript.exe	1072	Process Create	C:\Users\Administrator\AppData\Local\Temp\%X6D0L0xLSf.exe	SUCCESS	PID: 3224, Comma...
01:37:...	%X6D0L0xLSf.exe	3224	Process Start		SUCCESS	Parent PID: 1072...
01:37:...	%X6D0L0xLSf.exe	3224	Thread Create		SUCCESS	Thread ID: 2100...
01:37:...	WScript.exe	1072	RegOpenKey	HKLM\System\CurrentControlSet\Control\Session Manager\AppDataLis	REPARSE	Desired Access: Q...
01:37:...	WScript.exe	1072	RegOpenKey	HKLM\System\CurrentControlSet\Control\Session Manager\AppDataCertDlIs	REPARSE	NAME NOT FOUND Desired Access: Q...

In the above image, we can see in (1) that the script creates the binary and create a process launching it (2).

We found interesting that the js downloader calls the binary with an argument needed for

decoding the main payload like the Locky ransomware and most weird is that it uses the same argument for the decryption routine:



Once identified the dir location of the ransomware payload we could identify him:
SHA256:5bbc9afa3128956b3f6116037cc97d0ea1c79d8bb5d3e15473d1e9c5c8eecfdf

The only problem we face executing it is that the ransomware does not execute itself but it changes its behavior killing its main thread and it auto-delete itself maybe because it detects the virtual environment.

So start to patch that binary in order to study our sample:

We open the executable with the Olly debugger with the argument 321 and starting analyzing the code searching for some tricks used for vm detection.

Looking at the list of the intermodular calls we investigated on the **GetProcAddress** syscall and we found something interesting:

```

[*G.P.U* - main thread, module 9sX6D0L0]
File View Debug Plugins Options Window Help Tools BreakPoint->
Ln E Me Th Wv Ha Cp Pa St Br Re Tr Sr
013A2C94 . CC INT3
013A2C95 . 55 PUSH EBP
013A2C96 . 8BEC MOV EBP,ESP
013A2C98 . 83EC 24 SUB ESP,24
013A2C9B . 56 PUSH ESI
013A2C9C . 57 PUSH EDI
013A2C9D . 68 E8943B01 PUSH 9sX6D0L0.013B94E8
013A2CA2 . 68 F4943B01 PUSH 9sX6D0L0.013B94F4
013A2CA7 . 33FF XOR EDI,EDI
013A2CA9 . FF15 EC913B01 CALL DWORD PTR [&KERNEL32.LoadLibraryW]
013A2CAF . 50 PUSH EAX
013A2CB0 . FF15 E8913B01 CALL DWORD PTR [&KERNEL32.GetProcAddress]
013A2CB6 . A3 8C0E3C01 MOV DWORD PTR [13C0E8C],EAX
013A2CBB . B8 79B5E236 MOV EAX,36E2B579
013A2CC0 . 8945 EC MOV DWORD PTR [EBP-14],EAX
013A2CC3 . 8945 F0 MOV DWORD PTR [EBP-10],EAX
013A2CC6 . C745 E0 F87A MOV DWORD PTR [EBP-20],6F7AF8
013A2CCD . C745 E4 9155 MOV DWORD PTR [EBP-1C],9A485591
013A2CD4 > 8D2424 LEA ESP,DWORD PTR [ESP]
013A2CD7 . 0F31 RDTSC
013A2CD9 . 55 PUSH EBP
013A2CDA . 5D POP EBP
013A2CDB . 8945 F4 MOV DWORD PTR [EBP-C],EAX
013A2CDE . FF15 F0913B01 CALL DWORD PTR [&KERNEL32.GetProcessHeap]
013A2CE4 . 8BE4 MOV ESP,ESP
013A2CE6 . 0F31 RDTSC
013A2CE8 . FF35 68DE3B01 PUSH DWORD PTR [13BDE68]
013A2CEE . 8F45 E8 POP DWORD PTR [EBP-18]
013A2CF1 . 3345 EC XOR EAX,DWORD PTR [EBP-14]
013A2CF4 . 8945 F8 MOV DWORD PTR [EBP-8],EAX
013A2CF7 . 3345 F0 XOR EAX,DWORD PTR [EBP-10]
013A2CFA . 8175 F8 79B5 XOR DWORD PTR [EBP-8],36E2B579
013A2D01 . 6A 00 PUSH 0
013A2D03 . FF15 8C0E3C01 CALL DWORD PTR [13C0E8C]
013A2D09 . FF75 E0 PUSH DWORD PTR [EBP-20]
013A2D0C . 8F45 DC POP DWORD PTR [EBP-24]
013A2D0F . 0F31 RDTSC
013A2D11 . 8D36 LEA ESI,DWORD PTR [ESI]
013A2D13 . 3345 E4 XOR EAX,DWORD PTR [EBP-1C]
ProcNameOrOrdinal = "CloseHandle"
FileName = "kernel32.dll"
LoadLibraryW
hModule
GetProcAddress

```

This ransomware uses the RDTSC anti-vm technique:

“The Time Stamp Counter (TSC) is a 64-bit register present on all x86 processors since the Pentium. It counts the number of cycles since reset”. (Wikipedia)

If the code is being emulated then, there will be a change in the time stamp between. The Result in stored in EDX:EAX format.

Now the time difference in a real host machine would be usually less than 100, but if the code is emulated the difference will be huge.

Filling those instructions with NOP and patching the executable let us successfully launch the ransomware.

Now we wait until it decodes itself and, when it will contact the domains to take the RSA key (it means it decoded itself and loaded in memory), we will suspend the process in order to dump it from the memory for our further analysis.

We will use a useful tool for dump a process loaded into the memory: [Process Dump](#).


```

C:\dump>pd32.exe -pid 1748
Process Dump v1.5
Copyright © 2015, Geoff McDonald
http://www.split-code.com/
https://github.com/glmcdona/Process-Dump

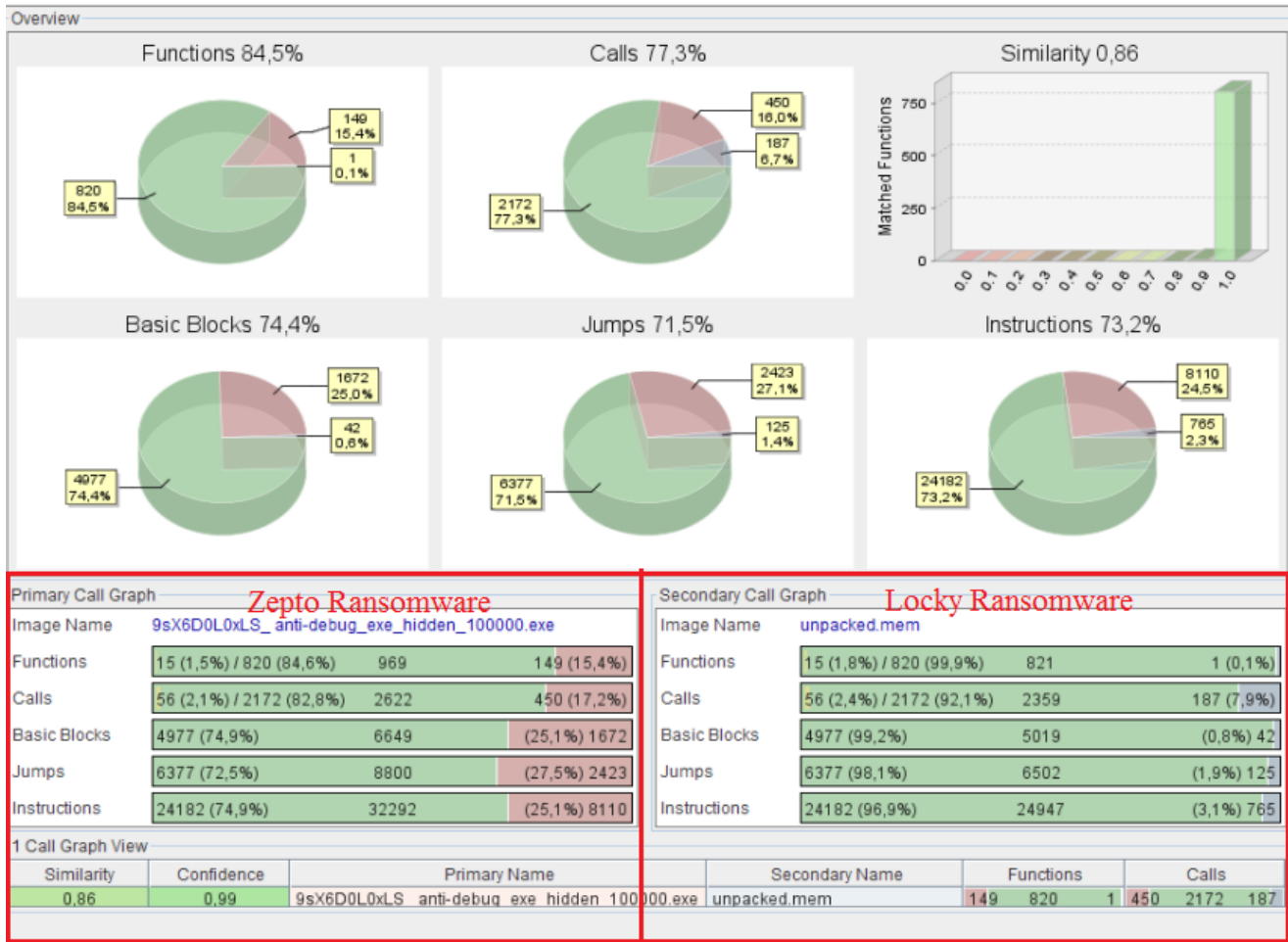
Dumping process 9sX6D0L0xLS_anti-debug1_exe with pid 0x6d4...
building import reconstruction table
dumping 'exe' at 5F0000 to file '9sX6D0L0xLS_anti-debug1_exe_hidden_5F0000.exe'
dumping 'exe' at 1010000 to file '9sX6D0L0xLS_anti-debug1_exe_1010000.exe'
dumping 'dll' at 6F260000 to file '9sX6D0L0xLS_anti-debug1_exe_sensapi.dll_6F260000.dll'
dumping 'dll' at 709C0000 to file '9sX6D0L0xLS_anti-debug1_exe_MPR.dll_709C0000.dll'
dumping 'dll' at 70A20000 to file '9sX6D0L0xLS_anti-debug1_exe_RASAPI32.dll_70A20000.dll'
dumping 'dll' at 70DD0000 to file '9sX6D0L0xLS_anti-debug1_exe_rasadhlp.dll_70DD0000.dll'
dumping 'dll' at 71D20000 to file '9sX6D0L0xLS_anti-debug1_exe_wshqos.dll_71D20000.dll'
dumping 'dll' at 71E90000 to file '9sX6D0L0xLS_anti-debug1_exe_rasman.dll_71E90000.dll'
dumping 'dll' at 72220000 to file '9sX6D0L0xLS_anti-debug1_exe_oledbg.dll_72220000.dll'
dumping 'dll' at 737C0000 to file '9sX6D0L0xLS_anti-debug1_exe_WINNS1_DLL_737C0000.dll'
dumping 'dll' at 737D0000 to file '9sX6D0L0xLS_anti-debug1_exe_iphlpapi.DLL_737D0000.dll'
dumping 'dll' at 73970000 to file '9sX6D0L0xLS_anti-debug1_exe_DSROLE.DLL_73970000.dll'
dumping 'dll' at 73C20000 to file '9sX6D0L0xLS_anti-debug1_exe_rtutils.dll_73C20000.dll'
dumping 'dll' at 73C50000 to file '9sX6D0L0xLS_anti-debug1_exe_NLAapi.dll_73C50000.dll'
dumping 'dll' at 73E90000 to file '9sX6D0L0xLS_anti-debug1_exe_ntmarta.dll_73E90000.dll'
dumping 'dll' at 73ED0000 to file '9sX6D0L0xLS_anti-debug1_exe_wkscli.dll_73ED0000.dll'
dumping 'dll' at 73EE0000 to file '9sX6D0L0xLS_anti-debug1_exe_netutils.dll_73EE0000.dll'
dumping 'dll' at 73EF0000 to file '9sX6D0L0xLS_anti-debug1_exe_NETAPI32.dll_73EF0000.dll'
dumping 'dll' at 74440000 to file '9sX6D0L0xLS_anti-debug1_exe_uxtheme.dll_74440000.dll'
dumping 'dll' at 74670000 to file '9sX6D0L0xLS_anti-debug1_exe_comctl32.dll_74670000.dll'
dumping 'dll' at 74CD0000 to file '9sX6D0L0xLS_anti-debug1_exe_wshtcpip.DLL_74CD0000.dll'
dumping 'dll' at 74F60000 to file '9sX6D0L0xLS_anti-debug1_exe_rsaenh.dll_74F60000.dll'
dumping 'dll' at 75040000 to file '9sX6D0L0xLS_anti-debug1_exe_dnsapi.DLL_75040000.dll'
dumping 'dll' at 75170000 to file '9sX6D0L0xLS_anti-debug1_exe_wship6.dll_75170000.dll'
dumping 'dll' at 75180000 to file '9sX6D0L0xLS_anti-debug1_exe_mssock.dll_75180000.dll'
dumping 'dll' at 751C0000 to file '9sX6D0L0xLS_anti-debug1_exe_CRYPTSP.dll_751C0000.dll'
dumping 'dll' at 75360000 to file '9sX6D0L0xLS_anti-debug1_exe_srucli.dll_75360000.dll'
dumping 'dll' at 75620000 to file '9sX6D0L0xLS_anti-debug1_exe_SspiCli.dll_75620000.dll'
dumping 'dll' at 75690000 to file '9sX6D0L0xLS_anti-debug1_exe_CRYPTBASE.dll_75690000.dll'
dumping 'dll' at 75740000 to file '9sX6D0L0xLS_anti-debug1_exe_profapi.dll_75740000.dll'
dumping 'dll' at 757B0000 to file '9sX6D0L0xLS_anti-debug1_exe_MSASN1.dll_757B0000.dll'
dumping 'dll' at 75810000 to file '9sX6D0L0xLS_anti-debug1_exe_KERNELBASE.dll_75810000.dll'
dumping 'dll' at 75890000 to file '9sX6D0L0xLS_anti-debug1_exe_CRYPT32.dll_75890000.dll'
dumping 'dll' at 75A40000 to file '9sX6D0L0xLS_anti-debug1_exe_SHLWAPI.dll_75A40000.dll'
dumping 'dll' at 75AA0000 to file '9sX6D0L0xLS_anti-debug1_exe_LPK.dll_75AA0000.dll'
dumping 'dll' at 75AB0000 to file '9sX6D0L0xLS_anti-debug1_exe_kerne132.dll_75AB0000.dll'
dumping 'dll' at 75B90000 to file '9sX6D0L0xLS_anti-debug1_exe_ADUAPI32.dll_75B90000.dll'
dumping 'dll' at 75C30000 to file '9sX6D0L0xLS_anti-debug1_exe_MSCTF.dll_75C30000.dll'
dumping 'dll' at 75D00000 to file '9sX6D0L0xLS_anti-debug1_exe_OLEAUT32.dll_75D00000.dll'
dumping 'dll' at 75D90000 to file '9sX6D0L0xLS_anti-debug1_exe_RPCRT4.dll_75D90000.dll'
dumping 'dll' at 75E40000 to file '9sX6D0L0xLS_anti-debug1_exe_GDI32.dll_75E40000.dll'
dumping 'dll' at 75E90000 to file '9sX6D0L0xLS_anti-debug1_exe_USP10.dll_75E90000.dll'
dumping 'dll' at 75F30000 to file '9sX6D0L0xLS_anti-debug1_exe_USER32.dll_75F30000.dll'
dumping 'dll' at 76110000 to file '9sX6D0L0xLS_anti-debug1_exe_SHELL32.dll_76110000.dll'
dumping 'dll' at 76D60000 to file '9sX6D0L0xLS_anti-debug1_exe_ur1mon.dll_76D60000.dll'
dumping 'dll' at 76EB0000 to file '9sX6D0L0xLS_anti-debug1_exe_msvcrt.dll_76EB0000.dll'
dumping 'dll' at 76F60000 to file '9sX6D0L0xLS_anti-debug1_exe_WLDAP32.dll_76F60000.dll'
dumping 'dll' at 76FB0000 to file '9sX6D0L0xLS_anti-debug1_exe_ws2_32.DLL_76FB0000.dll'
dumping 'dll' at 76FF0000 to file '9sX6D0L0xLS_anti-debug1_exe_WININET.dll_76FF0000.dll'
dumping 'dll' at 770F0000 to file '9sX6D0L0xLS_anti-debug1_exe_iertutil.dll_770F0000.dll'
dumping 'dll' at 77490000 to file '9sX6D0L0xLS_anti-debug1_exe_ole32.dll_77490000.dll'
dumping 'dll' at 775F0000 to file '9sX6D0L0xLS_anti-debug1_exe_ntdll.dll_775F0000.dll'
dumping 'dll' at 77740000 to file '9sX6D0L0xLS_anti-debug1_exe_NSI.dll_77740000.dll'
dumping 'dll' at 77750000 to file '9sX6D0L0xLS_anti-debug1_exe_IMM32.DLL_77750000.dll'
dumping 'dll' at 77800000 to file '9sX6D0L0xLS_anti-debug1_exe_sechost.dll_77800000.dll'

```

The highlighted file will be our unpacked sample of Zepto Ransomware.

Now we have our fresh and unpacked sample of Zepto Ransomware and we need to produce the .idb files of the two ransomware used for the comparison in the bindiff software, ida Pro will do this easily.

So let's compare our two ransomware and look at the results:



The first result returned by the tool are pretty pie graphs where we have the numbers of Functions, Calls, Basic Blocks and Jumps.

In green there are the matched elements (included also the changed), in red we have the new Zepto ransomware elements that aren't present in the Locky ransomware and in gray we have the Locky elements that aren't present in Zepto Ransomware.

Overall, the two binary have a similarity coefficient of **0.86** that is high for two different families of a ransomware.

As we can see from the lower part of the image there is a table representing our results, the Zepto ransomware has more functions, calls, basic blocks, jumps and instructions than locky ransomware.

And interesting enough are the results shown in Secondary Call Graph window saying that the **99.9%** on 821 functions of locky ransomware are matched with the Zepto ransomware and 15 functions changed (**1.8%**), impressive is that just 1 (**0.1%**) function unmatched.

On the left window, we can see that 149 functions are unmatched (**15.4%**), it means there are added functions to the new version of that ransomware.

In the overall instructions of Locky ransomware (24,947) we have the **96.9%** of identical codes and just **3.1%** of different instructions.

How much changed Zepto Ransomware and how many new features it has?

Well, answering exactly can't be that easy, but we can give you some good statistical numbers.

We can tell you that the Zepto ransomware has, of course, more overall instructions than locky ransomware, it has 32,292 and over that there are 8,110 new instructions, so **25.1%** new code.

It means that for sure that Zepto ransomware will have some new behavior than locky ransomware, but in most aspects, it will act as locky ransomware, but also with little improvements it will still avoid the av engine.

It looks like the author of the ransomware take the previous code of the locky ransomware and added new features and changed some code to evade signature-based detection.

Let's investigate on some changed functions and try to extract some big difference.

Looking at the list of the matched functions, we can easily identify the functions that changed for this new version of the ransomware thank at the **similarity coefficient** computed by bindiff tool:

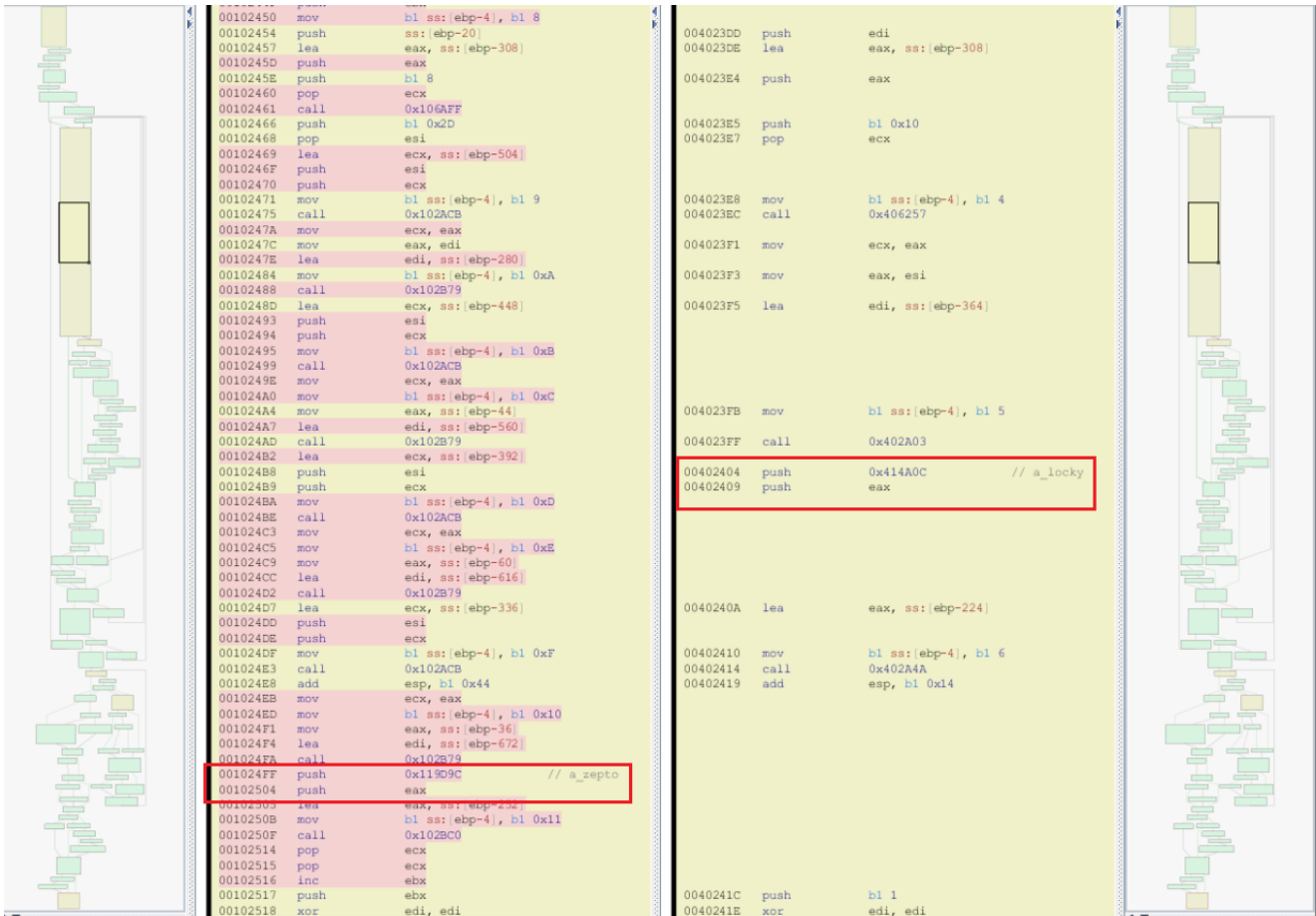
Similarity	Confidence	Address	Primary Name	Type	Address	Secondary Name	Type	Basic Blocks	Jumps
0.38	0.54	0010203E	sub_10203E	Normal	00403670	sub_403670	Normal	11	19
0.42	0.62	00111A5C	sub_111A5C	Normal	0040364D	sub_40364D	Normal	0	1
0.42	0.70	001097FC	sub_1097FC	Normal	00402045	sub_402045	Normal	23	23
0.50	0.99	001171C0	RtlUnwind	Imported	0040FF92	RtlUnwind	Thunk	1	1
0.51	0.91	00103804	sub_103804	Normal	00407F65	sub_407F65	Normal	21	21
0.89	0.99	00104884	sub_104884	Normal	00404641	sub_404641	Normal	11	71
0.93	0.94	0010689F	sub_10689F	Normal	0040E277	sub_40E277	Normal	0	3
0.94	0.95	0010904E	sub_10904E	Normal	004034C1	sub_4034C1	Normal	0	3
0.95	0.95	00115CD7	sub_115CD7	Normal	0041121B	sub_41121B	Normal	0	4
0.95	0.95	00115CF0	sub_115CF0	Normal	00411250	sub_411250	Normal	0	4
0.95	0.96	00103637	sub_103637	Normal	004087B7	sub_4087B7	Normal	0	3
0.97	0.97	001113B8	sub_1113B8	Normal	00402020	sub_402020	Normal	0	3
0.97	0.97	00102045	sub_102045	Normal	00402045	sub_402045	Normal	0	3
0.98	0.99	00102281	sub_102281	Normal	00402288	sub_402288	Normal	0	78
0.99	0.99	00104789	sub_104789	Normal	00404387	sub_404387	Normal	0	95
1.00	0.99	001170CC	EnterCriticalSection	Imported	004120A8	EnterCriticalSection	Imported	0	2
1.00	0.95	001161B0	sub_1161B0	Normal	004114FE	sub_4114FE	Normal	0	0
1.00	0.98	00103238	sub_103238	Normal	004030C2	sub_4030C2	Normal	0	16
1.00	0.99	0010AF32	sub_10AF32	Normal	0040A694	sub_40A694	Normal	0	4
1.00	0.98	00106592	sub_106592	Normal	00405950	sub_405950	Normal	0	17
1.00	0.99	00103199	sub_103199	Normal	00403023	sub_403023	Normal	0	3
1.00	0.95	00115EB7	sub_115EB7	Normal	00411357	sub_411357	Normal	0	2
1.00	0.99	00111845	sub_111845	Normal	00410165	sub_410165	Normal	0	5
1.00	0.99	00101A7E	sub_101A7E	Normal	00401A85	sub_401A85	Normal	0	3
1.00	0.99	0010CED2	sub_10CED2	Normal	0040BCD2	sub_40BCD2	Normal	0	23
1.00	0.99	0011724C	HeapSetInformation	Imported	004121EC	HeapSetInformation	Imported	0	0
1.00	0.99	001121C0	sub_1121C0	Normal	00410A86	sub_410A86	Normal	0	0
1.00	0.99	0010BD83	sub_10BD83	Normal	0040AB83	sub_40AB83	Normal	0	1
1.00	0.99	0011718C	SetFilePointer	Imported	00412148	SetFilePointer	Imported	0	0
1.00	0.96	0010BE96	sub_10BE96	Normal	0040AC96	sub_40AC96	Normal	0	1
1.00	0.96	00108C65	sub_108C65	Normal	004083CE	sub_4083CE	Normal	0	1
1.00	0.99	00116081	sub_116081	Normal	004116D3	sub_4116D3	Normal	0	4

We can realize from the above image that on the 820 matched functions, just 15 functions are changed, and 805 functions are identical.

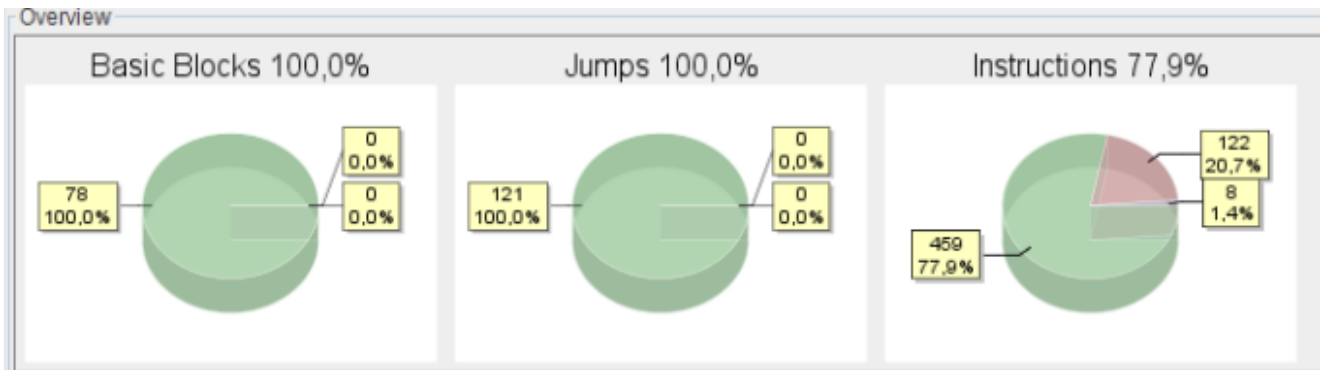
It means that **98.1%** of the Locky ransomware functions are **identical** to the Zepto ransomware.

For that, we can confirm that the Zepto ransomware is just an **extension** of the Locky ransomware **adding it new features**.

Analyzing the changed functions the most notable discover was on the encryption routine function used to encrypt the files because it has the same CFG and changes are made just in adding the final extension of the files:



The Encryption routine implemented in the Zepto ransomware is similar to the Locky one. On the left and right sides we can realize that the CFG graphs are identical if we look just at branching instructions and calls, it changed just the instructions in the yellow basic blocks. In fact, the report for that function say us exactly that:



CONCLUSION

“If Zepto sticks with this attack vector it may never become a serious threat. However, it’s very likely Zepto moves into exploit kits as time goes on,” Williams said. “A move by Zepto to malvertising, for example, could get bad very fast,” he said.

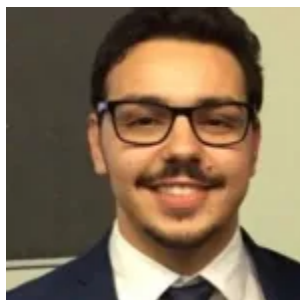
What we can say is that Zepto Ransomware isn't a new variant of the Locky Ransomware that uses some copycat, because there are too much identical code.

If an av engine tracked the main behaviors of Locky, ransomware as drive enumeration or encryption routine will still spot this threat as a Locky ransomware because, as we saw, this new version of Locky doesn't change the inner logic of the most crucial behaviors.

We can define Zepto ransomware as Locky Ransomware 2.0 and with a lot of probability, the authors of that new variant are the same behind Locky Ransomware.

REFERENCES

- <http://securityaffairs.co/wordpress/48725/malware/locky-ransomware-back.html>
- http://www.iswatlab.eu/wp-content/uploads/2015/09/Technical_Report_Ransomware.pdf
- <http://blog.talosintel.com/2016/06/gotta-be-swift-for-this-spam-campaign.html>
- <https://threatpost.com/locky-variant-zepto-debuts-with-big-spam-push/119017/>
- <http://resources.infosecinstitute.com/anti-debugging-and-anti-vm-techniques-and-anti-emulation/>



Written by the IT Security Expert Antonio Cocomazzi

Antonio Cocomazzi is an IT Security Expert specialized in the malware analysis field. Young and recently graduated, he conducts a 6 months research focused on Ransomware giving a full characterization of the recent families defining a new methodology for dissecting this kind of malware.

Edited by Pierluigi Paganini

(Security Affairs – Locky Ransomware, Zepto ransomware)

CybercrimeLockymalwareransomwareZepto

Share On



You might also like

There you can buy or download for free private and compromising data of your competitors. We public schemes, drawings, technologies, political and military secrets, accounting reports and clients databases. All this things were gathered from the largest worldwide companies, conglomerates and concerns with every activity. We gather data using vulnerability in their IT infrastructure. in their IT infrastructure.

Industrial spy team processes huge massives every day to devide you results. You can fid it in their portal:

http://[REDACTED]

(Tor browser required)

We can save your time gaining your own goals or goals of your company. With our information you could refuse partnership with unscrupulous partner, reveal dirty secrets of your competitors and enemies and earn millions dollars using insider information.

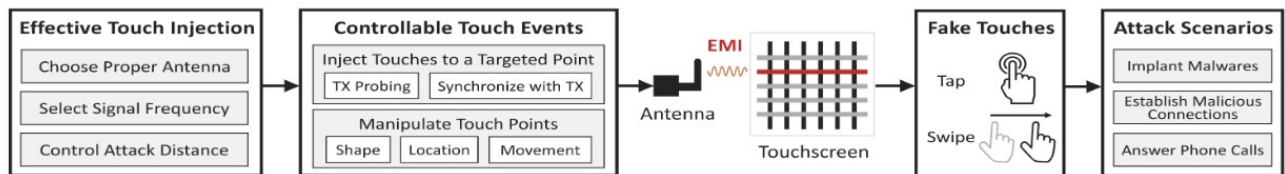
"He who owns the information, owns the world"

Nathan Mayer Rothschild

The strange link between Industrial Spy and the Cuba ransomware operation

May 28, 2022 By [Pierluigi Paganini](#)

How does GhostTouchAttack work?



GhostTouch: how to remotely control touchscreens with EMI

May 27, 2022 By [Pierluigi Paganini](#)

Copyright 2021 Security Affairs by Pierluigi Paganini All Right Reserved.

[Back to top](#)

- [Home](#)
- [Cyber Crime](#)
- [Cyber warfare](#)
- [APT](#)
- [Data Breach](#)
- [Deep Web](#)
- [Digital ID](#)
- [Hacking](#)
- [Hacktivism](#)
- [Intelligence](#)
- [Internet of Things](#)

- [Laws and regulations](#)
- [Malware](#)
- [Mobile](#)
- [Reports](#)
- [Security](#)
- [Social Networks](#)
- [Terrorism](#)
- [ICS-SCADA](#)
- [EXTENDED COOKIE POLICY](#)
- [Contact me](#)