Hunting Cyber Evil Ratels: From the targeted attacks to the widespread usage of Brute Ratel - Yoroi

Published: 2023-02-15 · Archived: 2026-04-05 15:30:26 UTC

02/15/2023

Introduction

Red team operations are fundamental for achieving an adequate cybersecurity maturity level. So, many different C2 commercial frameworks were born to provide help in managing security tests. However, these technologies can be used at the same time even by attackers to make cyber intrusions.

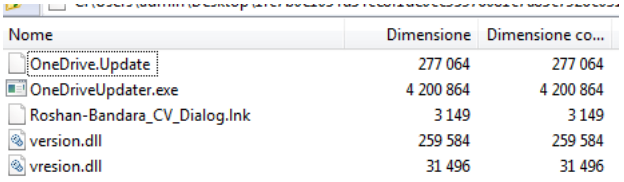
One of the most emblematic examples of this phenomenon is “Brute Ratel”, a commercial Red Team Operations framework developed by Chetan Nayak, an expert red teamer, formerly both in Mandiant and Crowdstrike, which, starting from the past year, has been used by attackers both in cybercrime and APT operations.

This Red Team framework is designed to be capable of being highly evasive and undetectable by security products, as demonstrated also by many shellcodes we intercepted through hunting activities with zero detection rate on VirusTotal platform.

For these reasons, the reconstruction of the abuse of that tool is a necessary step in order to provide technical insights for the cybersecurity community.

Reconstructing the abuse of Brute Ratel

The first time when the Cybersecurity Community started to face-off the abuse of Brute Ratel was between May and June 2022. The observed attack involved a ISO file, used to compress the files and reduce the detection rate, because this type of file isn’t easily detected by security protections. The content of the archive is a LNK file used to perform DLL sideloading of the malicious “version.dll” library, a necessary dependency of the legit Microsoft’s executable “OneDriveUpdtater.exe”.



Nome	Dimensione	Dimensione co...
OneDrive.Update	277 064	277 064
OneDriveUpdater.exe	4 200 864	4 200 864
Roshan-Bandara_CV_Dialog.lnk	3 149	3 149
version.dll	259 584	259 584
vresion.dll	31 496	31 496

Figure 1: Content of the ISO image

This campaign has been deeply analyzed by Unit42 of PaloAlto, and attributing the threat to Nobellium, AKA APT29. However, this campaign was important because it was the first documented one maliciously leveraging Brute Ratel Framework.

On 13 September an archive containing a leaked version of Bruteratel 1.2.2 named “**BruteRatel_1.2.2.Scandinavian_Defense.tar.gz”** was uploaded on VirusTotal and in the next days shared among the cyber criminals on underground forums and popular Telegram channels. This was a very good starting point to take a closer look, but the leak didn’t have a valid license, making the application not working.

```
remnux@remnux:~/Downloads/591c2cd3a9b902a182fbf05bf5423cae17e3e6874c0d2e09107e914d86f39780/bruteratel$
./brute-ratel-linx64
[!] Enter activation key: test
[!] Enter registered email ID: test
License is invalid or expired
```

Figure 2: Testing the Brute Ratel Leak

Nevertheless, this hasn’t stopped the criminals, who we believe reversed the license related routines to develop a crack aimed at bypassing the usage restrictions of the product. In fact, the following days a manumitted version of the framework appeared, with the addition of a license file named “.brauth”, which made the tool work.

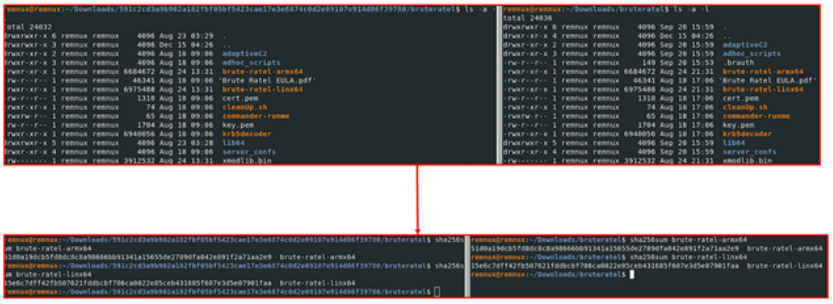


Figure 3: Evidence of the cracked version of brute ratel

Thus, we decided to test even this version, and it worked.

Figure 4: Cracked Version of Brute Ratel Working

After that, this cracked version started to be spread on many different forums and private hacking groups. So, threat actors started to adopt this framework during their human operated cyber intrusions, replacing or alongside CobaltStrike, which nowadays remains the landmark for this type of malicious operations.

Technical Insights

The malicious code is contained inside the exported function **main**, the sample iterates the PEB to find ntdll.dll and dynamically resolve the APIs. The API Hashing is one of the most trending techniques adopted by malware writers: in this way, they can obfuscate the calling of the most critical API calls to perform, making the analyses harder. In this case (4766553ce5ff67a2e28b1ee1b5322e005b85b26e21230ffba9622e7c83ed0917), the algorithm used to perform the hashing operation is **ROR13**.

```
v3 = (__int64)sub_295092450(0x3CFA685D);      // ntdll.dll
v4 = v3;
if ( v3 )
{
  v20 = 0i64;
  v22 = 0i64;
  v21 = unk_2950E7C20;
  v5 = mw_resolve_api(2352565641i64, v3);     // NtProtectVirtualMemory
  v6 = mw_resolve_api(3543911101i64, v4);     // NtAllocateVirtualMemory
  v7 = mw_resolve_api(2919678386i64, v4);     // NtWaitForSingleObject
  mw_resolve_api(NtCreateThreadEx_0, v4);     // NtCreateThreadEx
```

Figure 5: Resolving the hashed APIs

After resolving the functions, the badger starts an anti-analysis phase, where it tries to evade when an analyst reverses the payload.

The first technique is an anti-debug trick to evade when the analyst tries to insert some breakpoints on the API function to be resolved. In this case, the control is performed by retrieving the address of the API call to resolve, and check whether it is overwritten with the 0xCC opcode, which is the code of the software breakpoint in assembly language.

A second technique is an anti-hooking trick aimed at checking if the call is long 20bytes and the last 3 bytes are 0x0F, 0x05 and 0xC3, which are the machine code for the operations “syscall” and “ret”. In this way, the control allows the malware to identify any tampering inside the API calls to be used: in fact, many EDRs and security appliance adopt hooking techniques to intercept the code and identify malicious behaviour.

Figure 6: Anti-Analysis and Anti-hooking techniques

A detail of the called syscall not tampered by any security control is the following:

Figure 7: Evidence of a not tampered syscall

After those controls, the badger can extract its shellcode contained inside the ".data" section in a new memory section and it is executed with a syscall to **NtCreateThreadEx**.

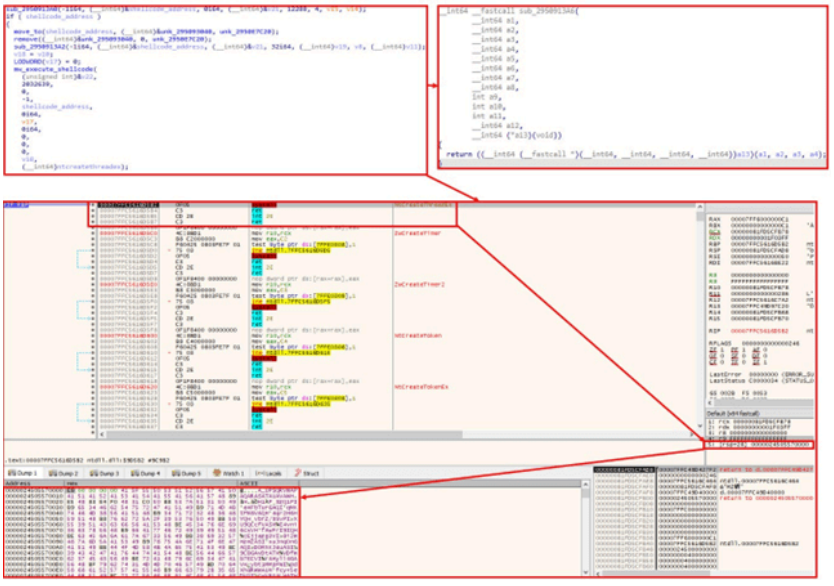


Figure 8: Shellcode execution through syscall

At this point, the shellcode is able to allocate in memory a PE file, but with the header missing, which is the final payload.

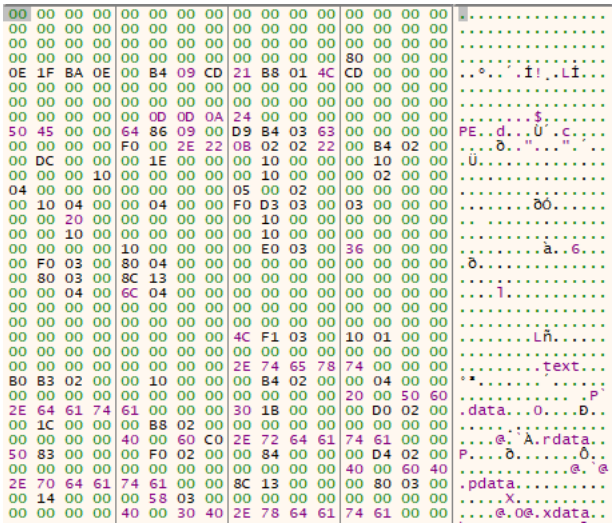


Figure 9: Loaded badger shellcode without MZ header

A view of the badger strings inside the just loaded code is the following:

Offset	Strings recognized UNICODE	Offset	Strings recognized UNICODE
0002F000	UNKNOWN	0002F410	%02x
0002F010	DISABLED	0002F41A	%02x
0002F022	QUEUED	0002F426	0x%02x,
0002F030	READY	0002F438	\x%02x
0002F03C	RUNNING	0002F448	BYTE data[] = {
0002F04C	Create	0002F47E	[+] %s Password History:
0002F05A	Win32_Process	0002F4B2	-%2u:
0002F07C	%ls\%ls	0002F4C4	[+] Active %s:
0002F08C	[+] Connected [%ls]	0002F4FC	%s%.*s%s
0002F0B6	CommandLine	0002F510	[+] Object RDN:
0002F0CE	[+] Process Created	0002F548	[+] SAM Username:
0002F10A	[+] Connected [%ls]	0002F586	[+] User Principal Name:
0002F13E	- %-45ls : %d	0002F5E4	8[+] UAC: %08x [
0002F160	- %-45ls : %ls	0002F626	6[+] Password last change:
0002F184	- %-45ls : %llu	0002F68A	6[+] SID history:
0002F1AA	- %-45ls : %u	0002F6B6	[+] Object SID:
0002F1CC	- %-45ls : %f	0002F6D8	[+] Object RID: %u
0002F1EE	- %-45ls : %S	0002F714	0NTLM
0002F270	True	0002F76E	[-] E: 0x%08x (%u) - %s
0002F27A	False	0002F7A0	[-] E: no item!
0002F286	- Path: %ls	0002F7C2	[-] E: bad version (%u)
0002F2A4	- Enabled: %ls	0002F7F4	[-] E: 0x%08x (%u)
0002F2C8	- Last Run: %ls	0002F81C	[-] E: (%08x)
0002F2EE	- Next Run: %ls	0002F83A	[-] E: DRS Extension Size (%u)
0002F314	- Current State: %ls	0002F87A	[-] E: No DRS Extension
0002F344	- XML Output:	0002F8AC	[-] E: DRSBind (%u)
0002F36E	- Error fetching xml	0002F8D6	[-] E: DC '%s' not found
0002F3A2	[+] Name: %ls	0002F90A	[-] E: Version (%u)
0002F3C0	[+] Task: %ld	0002F934	[-] E: 0x%08x
0002F3DC	- Name: %ls	0002F952	[-] E: DC not found
0002F410	%02x	0002F97C	[-] E: Binding DC!
0002F41A	%02x	0002F9A4	[-] E: %u
0002F426	0x%02x,	0002F9BA	[-] E: Domain not found
0002F438	\x%02x	0002F9EC	[+] Syncing DC: %ls
0002F448	BYTE data[] = {	0002FA14	==========

Figure 10: Strings of the badger

The last decoding routine of the payload is to retrieve the configuration of the badger. It uses the RC4 algorithm to encrypt this information and it decodes that at runtime; in this case the key is the hardcoded string "**mnan#:<(**"

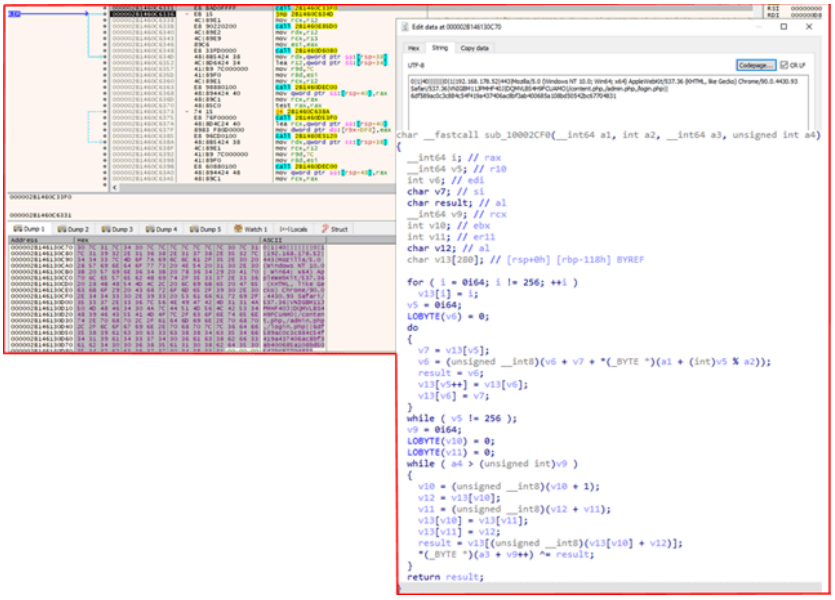


Figure 11: RC4 algorithm for the configuration decryption

An example of the configuration of the decrypted configuration of a badger of Brute Ratel is the following:

> 0|1|40||||||0|1|[IP (Redacted)]|443|Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/90.0.4430.93 Safari/537.36|VNIGBM11JPMHF40J|DQMVLBS4H9FCUAMO|/content.php,/admin.php,/login.php||6df589ac0c3c884c54f419a437406ac8bf3ab4

Threat Hunting

Upon the release of the leaked version, we decided to track the malicious campaigns leveraging this pentesting framework in order to identify how many actors are using it to do malicious cyber-attacks. It was surprising to observe that some of these samples we are able to track with the Yara rule provided at the end of the blogpost are fully undetected by the antimalware and VirusTotal give dame also zero detection rate. So, it indicates that the effort to make this tool as optimal to simulate an adversary.

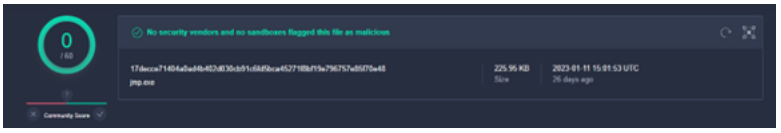


Figure 12: Detection rate of a file uploaded on VT which is a Brute Ratel

So, it was interesting to notice that, at time of writing, there are three low detection rate sample we found through our signatures.

0 detection:

- 17decce71404a0ad4b402d030cb91c6fd5bca45271f8bf19e796757e85f70e48
- Fdeb6a6aaee94fe204fb986f6d78e64a9086c5f64e315d8c5e90b590f0007af8

3 detections:

- 5f4782a34368bb661f413f33e2d1fb9f237b7f9637f2c0c21dc752316b02350c

Conclusion

Threat hunting is a proactive approach to identifying and tracking security threats. It involves actively searching for and analyzing potential security breaches or anomalies in an organization's systems and networks. The job of proactively hunting for threats helps organizations to minimize the risk of cyber intrusion and minimize the damage caused by such an attack. In addition, threat hunting enables organizations to continuously monitor and improve their security posture, helping to stay ahead of the evolving threat landscape.

The example of the abuse of Brute Ratel framework is emblematic to understand that the continuous improvement of Threat Intelligence and Hunting capabilities are fundamental for the growth of the cyber security measures of the company, guaranteeing a best practice to the early detection of the cyber threats.

Indicators of Compromise

- Hashes retrieved from threat hunting operations:
  - 025ef5e92fecf3fa118bd96ad3aff3f88e2629594c6a7a274b703009619245b6
  - 086dc27a896e154adf94e8c04b538fc146623b224d62bf019224830e39f4d51d
  - 17decce71404a0ad4b402d030cb91c6fd5bca45271f8bf19e796757e85f70e48
  - 17e4989ff7585915ec4342cbaf2c8a06f5518d7ba0022fd1d97b971c511f9bde
  - 200955354545ef1309eb6d9ec65a917b08479f28362e7c42a718ebe8431bb15d
  - 221e81540e290017c45414a728783cb62f79d9f63f2547490ec2792381600232
  - 25e7a8da631f3a5dfeec99ca038b3b480658add98719ee853633422a3a40247d
  - 28a4e9f569fd5223bffe355e685ee137281e0e86cae3cc1e3267db4c7b2f3bcd
  - 2ddc77de26637a6d759e5b080864851b731fdb11075485980ece20d8f197104c
  - 31fe821e4fac6380701428e01f5c39c6f316b6b58faff239d8432e821a79d151
  - 331952c93954bd263747243a0395441d0fae2b6d5b8ceb19f3ddb786b83f0731
  - 34c1d162bf17cdb41c2c5d220b66202a85f5338b15019e26dcab1a81f12fc451
  - 38b3b10f2ddeecda0db029dacc6363275c4cdf18cc62be3cc57b79647d517a44
  - 3a946cba2ba38a2c6158fa50beee20d2d75d595acc27ea51a39a37c121082596
  - 3baace2a575083a7031af7e9e13ff8ed46659f0b25ce54abe73db844acfad11a
  - 3f63fbc43fc44e6bf9c363e8c17164aeb05a515229e2111a2371d4321dcde787
  - 4766553ce5ff67a2e28b1ee1b5322e005b85b26e21230ffba9622e7c83ed0917
  - 4e5d89844135dca1d9899a8eedfbabc09bcb0fb5c5c14c29f7df5a58d7cf16d4
  - 4f88738e04447344100bb9532c239032b86e71d8037ccb121e2959f37fff53cf
  - 54e844b5ae4a056ca8df4ca7299249c4910374d64261c83ac55e5fdf1b59f01d
  - 56ced937d0b868a2005692850cea467375778a147288ac404748c2dea9c17277
  - 5f4782a34368bb661f413f33e2d1fb9f237b7f9637f2c0c21dc752316b02350c
  - 6021d5500fdea0664a91bdd85b98657817083ece6e2975362791c603d7a197c7
  - 62cb24967c6ce18d35d2a23ebed4217889d796cf7799d9075c1aa7752b8d3967
  - 62e88163b51387b160e9c7ea1d74f0f80c52fc32c997aa595d53cbc2c3b6caf4
  - 64a95de2783a97160bac6914ee07a42cdd154a0e33abc3b1b62c7bafdce24c0c
  - 6a85451644a2c6510d23a1ab5610c85a38107b3b3a00238f7b93e2ce6d1ba549
  - 6ade03a82d8bb884cae26c6db31cf539bec66861fc689cf1c752073fb79740c5
  - 6fdd81e31f2bec2bdda594974068a69e911219d811c8de4466d7a059dd3183a3
  - 74c00f303b87b23dffb59718187ff95c9d4d8497c61a64501166ac5dbed84b9f
  - 7757a76ca945f33f3220ad2b2aa897f3e63c47f08e1b7d62d502937ba90360a7
  - 7824197ad3b9c0981a1cdabf82940ac7733d232442bd31d195783a4e731845d2
  - 79e232b2a08a2960a493e74ab7cba3e82c8167acc030a5ca8d080d0027a587fe
  - 7fe1ff03e8f5678d280f7fd459a36444b6d816b2031e37867e4e36b689eccd33
  - 83b336deca35441fa745cd80a7df7448ce24c09dd2a36569332ae0e4771f36a6
  - 88249de22cefaf15f7c45b155703980fb09eb8e06b852f9d4a7c82126776ee7e
  - 8b8f7e8030e2ba234a33bc8a2fa3ccb5912029d660e03ed40413d949142b98fe
  - 8d979a1627dea58e9b86f393338df6aabfd762937e25e39f1d325fce06cf5338
  - 8dd3faf0248890e8c3efb40b800f892989204ba3125986690669f0a914f26c5d
  - 9521f51e42b8e31d82b06de6e15dbf9a1fa1bbff62cf6bc68c0b9e8fd1f8b2c5
  - 97a00056c459a7ce38ad8029413bf8f1691d4ae81e90f0d346d54c91dd02a511
  - 991f883556357a3b961c31e2b72f6246b52b27a5c45b72914abc61c5b5960cc3
  - 9f06583bd4b8c4aefc470ef582ff685cd3d03b404e67ce8bf9dbbd5828c90c43
  - a0c3da2ebf94f6671537a80d26b3288f8fcdf845fe2780ef81fd9da48c0162bf
  - a8759ef55fed4a9410cc152df9ef330a95f776619901054715ed4721a414d15c
  - a8cc14bd56aa4a2da40717cb3f11ecb6aff4e0797a9cebcff51461db19eaf580
  - ae38ec0ddc58424bf6de8858c82c4c6902fc947604943d58d8cbca00991c7f7d
  - aeb82788aad8bdee4c905559c4636536fb54c40fdc77b27ba4308b6a0f24bedf
  - bdd028922220ff92acb8530c894e2705743a968a8159fe955c1057736c7e1ebd
  - c3cc43492d005b25fc2cc66f82a550420bb4c48b5aae0a77f1ccef0603a3e47c
  - c4f40e2eb029ef11be4ac43ccc6895af6fb6dabd3a5bcc02f29afb9553da625c
  - c6aa2c54eee52f99a911dadfbf155372bd9f43fb9f923500b0b374799204d7a3
  - c6e2562a2ae399a851b0e5bfb92011e9f97ab45fa536a61eb89b3aee062461f7
  - ca2b9a0fe3992477d4c87a6e2a75faaac9ea0f3828d054cb44371b3068b76ba5
  - cdc5e05843cf1904e145dad3ae6c058b92b1bc3cbffffc217884b7cc382172a1
  - cee890a9e7ab521125372c13b71fc154ef5332d333fe43798303b198e9314dcd
  - d90beab9a3986c26922e4107dccb0b725b8b0eea398f2aeb8848cbe25c3becee
  - db987749ef4a58c6a592a33221770d23adcb2efce4a5504aabc73d61cd356616
  - dc9757c9aa3aff76d86f9f23a3d20a817e48ca3d7294307cc67477177af5c0d4
  - dcb986e45f1cf38794acec5e7f576a8dff6fbec66e6a09e3cc92596c796ad0d3
  - e400a196e7128a3cf40085629db8f26b73b6980be7df3da60928a4a062bc85cb
  - e491d06e3a556c79e922274af04c1786a957775ba2d5d0b02d13bdee91bf5ce4
  - ea6d9ff8f768fc0132f9f543d9546744d04f9f83e2241950f63f60b520b9ece0

- ead189bb18ee839db3d221701e208c4d2845c232cec66764bb3ea6c688ca18e8
- ee035537c3b8fc54ca2e1fa98c18e2fb0e203d863005c878bc8ceaa690a6689f
- ee53521e7d8b2b05fef77877440738ee169f3b75228931f9aaf96621a2f64c25
- eef36bc6f208abd46541bac1b1de18bb3a69057b1a54e67d71d259cc0f1bef5b
- f59fe0945f97df4e3d2efc9b31d00602fc5a16e05453e0d853e275cadb63a057
- f875e68899afe172394176fa9cabededeaa19ad6816a90746bb630c064c69e6a
- fdeb6a6aaee94fe204fb986f6d78e64a9086c5f64e315d8c5e90b590f0007af8

Yara Rules

```
rule brute_ratel
{
    meta:
        author = "Yoroi Malware ZLab"
        description = "Rule for BruteRatel Badger"
        last_updated = "2023-02-15"
        tlp = "WHITE"
        category = "informational"

    strings:
        $1 = {8079ffcc74584585c075044883e920448a094180f9e9740a448a41034180f8e97507ffc24531c0ebd731c04180f94c752f8079018b75
        $2 = {565389d34883ec2885db74644889cee8????????31c9ba????????4989c0e8????????448d430165488b142530000000488b5260488b
    condition:
        (uint16(0) == 0x5A4D or uint16(0) == 0x00E8 or uint16(0) == 0x8348) and ($1 or $2)
}
```

This blog post was authored by Luigi Martire, Carmelo Ragusa of Yoroi Malware ZLAB
