Botconf
April 2022

Evolution of the Sysrv mining-botnet

Reversing Golang Binaries with Ghidra

Dorka Palotay
Senior Threat Researcher, CUJO AI

Gyorgy Luptak
Junior Threat Researcher, CUJO AI

Who are we

Background

Gyorgy Luptak (@gyluptak):

- Junior Threat Researcher at CUJO AI
- BSc in Computer Science
- Currently pursuing an MSc in Computer Science, IT Security

Dorka Palotay (@pad0rka):

- Senior Threat Researcher at CUJO AI
- BSc in Applied Mathematics
- MSc in Security and Privacy – Advanced Cryptography
- Worked at financial and security companies as well
- Malware researcher and reverse engineer

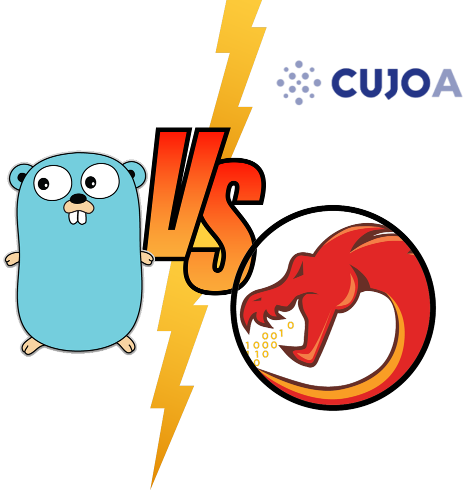
Special thanks to Albert Zsigovits (@albertzsigovits) for his contribution to this research.

Why we did all this

The quest

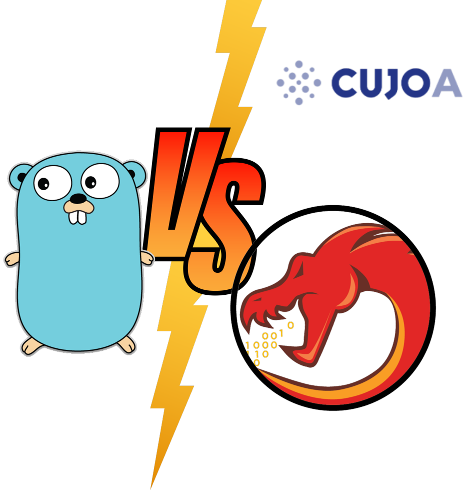
Background:

- IoT/Linux malware research -> more and more malware families are written in Go
- Sysrv is a good example of this



Why we did all this

The quest



Background:

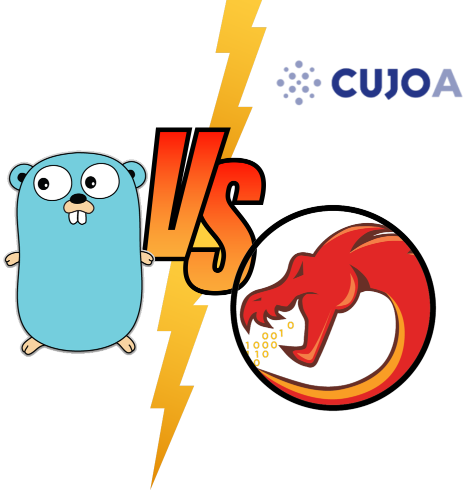
- IoT/Linux malware research -> more and more malware families are written in Go
- Sysrv is a good example of this

Issue:

- Reverse engineering Go binaries is challenging
  - Huge file size
  - Unusual string handling
  - No symbol names due to stripping
- Ghidra open-source development is in early stage compared to other tools
  - Only a few open-source scripts are available, solving only parts of the problem

Why we did all this

The quest



Background:

- IoT/Linux malware research -> more and more malware families are written in Go
- Sysrv is a good example of this

Issue:

- Reverse engineering Go binaries is challenging
  - Huge file size
  - Unusual string handling
  - No symbol names due to stripping
- Ghidra open-source development is in early stage compared to other tools
  - Only a few open-source scripts are available, solving only parts of the problem

Goal:

- Understanding Sysrv botnet evolution
- Making reverse engineering Go binaries with Ghidra easier

Result:

- Create our own scripts: https://github.com/getCUJO/ThreatIntel

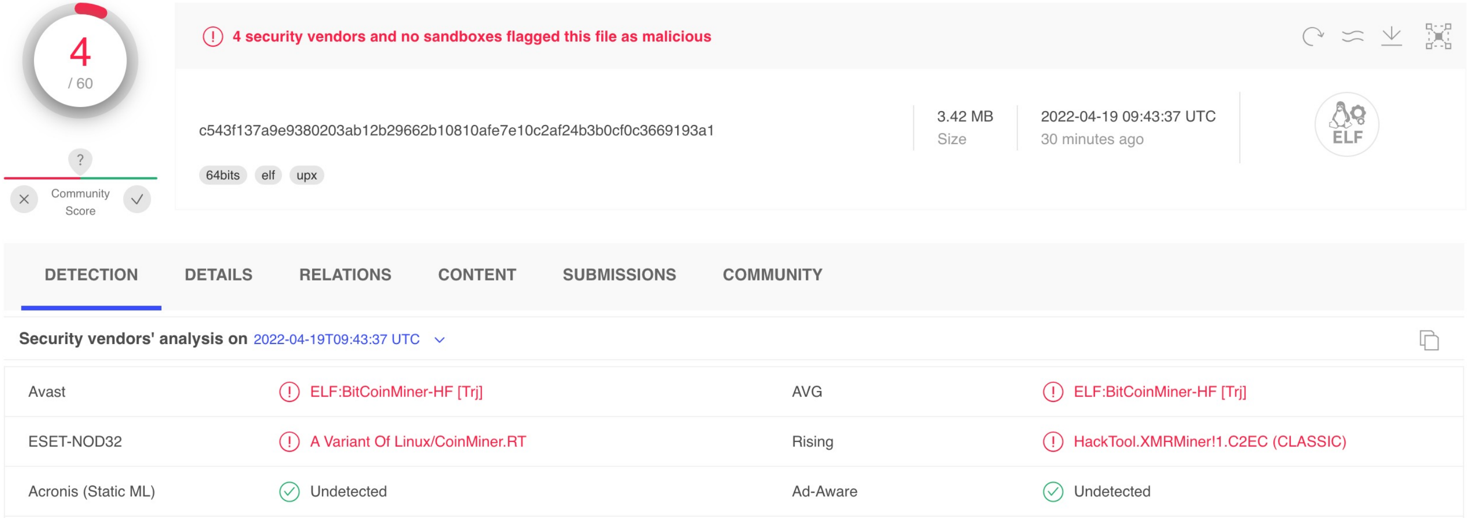
Agenda

- The Sysrv botnet - https://cujo.com/the-sysrv-botnet-and-how-it-evolved/
    - General introduction
    - Downloader script
    - Malicious binary and used exploits
    - Mining and monetization
- Go binary analysis with Ghidra - https://cujo.com/reverse-engineering-go-binaries-with-ghidra/
    - Lost function names
    - String recovery
    - Data type recovery

The Sysrv botnet

Introduction

- First mentioned in December 2020 by multiple sources
- It is a worm and a cryptocurrency miner
- It stood out due to its use of Golang
- The botnet is distributed for both Linux and Windows environments
- Still active today
- In our analysis we were focusing on variants attacking Linux
- Name coming from the used filenames: sysrv, sysrvv, sys



4 security vendors and no sandboxes flagged this file as malicious

c543f137a9e9380203ab12b29662b10810afe7e10c2af24b3b0cf0c3669193a1

64bits elf upx

3.42 MB
Size

2022-04-19 09:43:37 UTC
30 minutes ago

Security vendors' analysis on 2022-04-19T09:43:37 UTC

Avast	ELF:BitCoinMiner-HF [Trj]	AVG	ELF:BitCoinMiner-HF [Trj]
ESET-NOD32	A Variant Of Linux/CoinMiner.RT	Rising	HackTool.XMRMiner!1.C2EC (CLASSIC)
Acronis (Static ML)	✓ Undetected	Ad-Aware	✓ Undetected

The Sysrv botnet

The downloader script (Linux version)

- Linux: ldr.sh, Windows: ldr.ps1
- First part of development from December 2020 to the end of February 2021
  - First version: hardcoded C2 and sysrv version, curl and wget to download the binary (different one for 32-, and 64-bit systems)
  - Quick expanding: kill other miners and processes with high CPU usage, removing/disabling system security, cron-based persistence

The Sysrv botnet

The downloader script (Linux version)

- Linux: ldr.sh, Windows: ldr.ps1
- First part of development from December 2020 to the end of February 2021
  - First version: hardcoded C2 and sysrv version, curl and wget to download the binary (different one for 32-, and 64-bit systems)
  - Quick expanding: kill other miners and processes with high CPU usage, removing/disabling system security, cron-based persistence, remove ld.so.preload
- Second part from the end of February 2021 to December 2021
  - At the start: removed almost every functionality besides downloading the binary
  - Slow expansion from here: reintroduce some of the lost parts of the script
  - At first, it kills 'kthreaddi' process, then uses it as cryptominer, later replaced by 'kthreaddk'
  - New methods introduced: randomized sysrv version, install cron if not existing, spread via SSH, kill process listening on specific ports

The Sysrv botnet

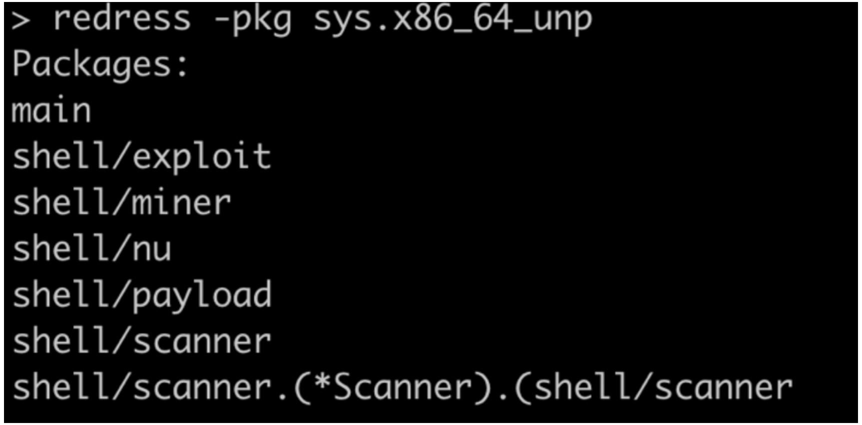
The downloader script (Linux version)

- Linux: ldr.sh, Windows: ldr.ps1
- First part of development from December 2020 to the end of February 2021
  - First version: hardcoded C2 and sysrv version, curl and wget to download the binary (different one for 32-, and 64-bit systems)
  - Quick expanding: kill other miners and processes with high CPU usage, removing/disabling system security, cron-based persistence, remove ld.so.preload
- Second part from the end of February 2021 to December 2021
  - At the start: removed almost every functionality besides downloading the binary
  - Slow expansion from here: reintroduce some of the lost parts of the script
  - At first, it kills 'kthreaddi' process, then uses it as cryptominer, later replaced by 'kthreaddk'
  - New methods introduced: randomized sysrv version, install cron if not existing, spread via SSH, kill process listening on specific ports
- Third part from 2022
  - Builds onto the previous version, but with lot of modifications
  - Low-level custom curl, wget-like code, replaces 'kthreaddk' by 'hezb', also downloads kthmimu.sh

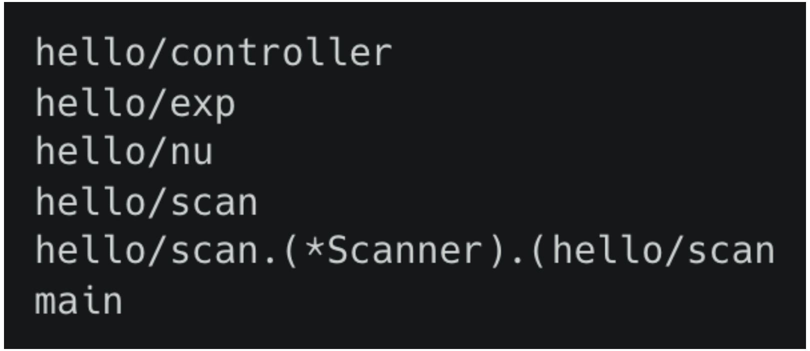
The Sysrv botnet

The binaries

- 32- and 64-bit binaries
- We analyzed more than 100 ELF binaries
- Grouped them based on their package structures – 9 different groups
- *Go programs are organized into packages. A package is a collection of source files in the same directory that are compiled together. Functions, types, variables, and constants defined in one source file are visible to all other source files within the same package.*



```
> redress -pkg sys.x86_64_unp
Packages:
main
shell/exploit
shell/miner
shell/nu
shell/payload
shell/scanner
shell/scanner.(*Scanner).(shell/scanner
```

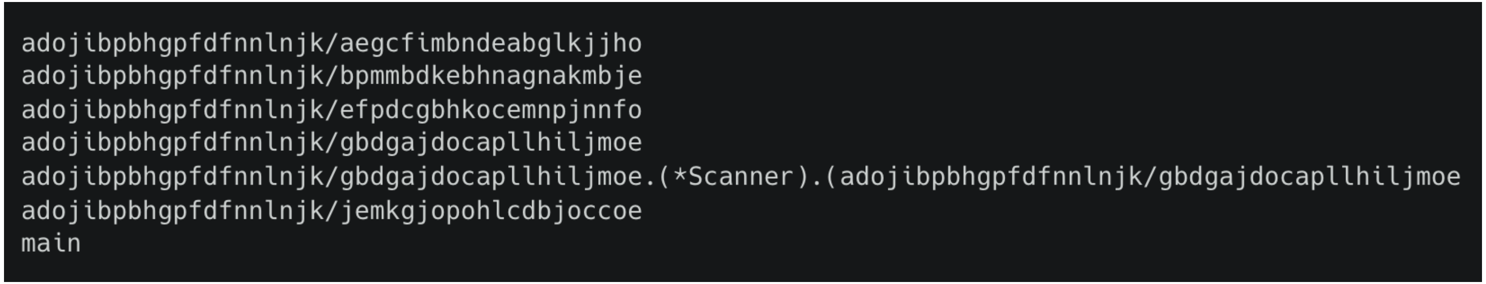


```
hello/controller
hello/exp
hello/nu
hello/scan
hello/scan.(*Scanner).(hello/scan
main
```

The Sysrv botnet

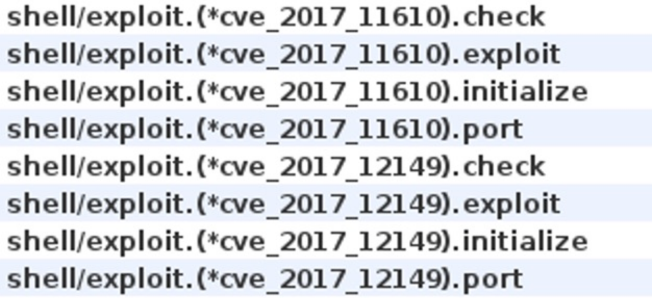
The binaries

- Packed with UPX
- The first obfuscated sample appeared at the end of March 2021
  - Used gobfuscate - https://github.com/unixpickle/gobfuscate
  - Package names were obfuscated

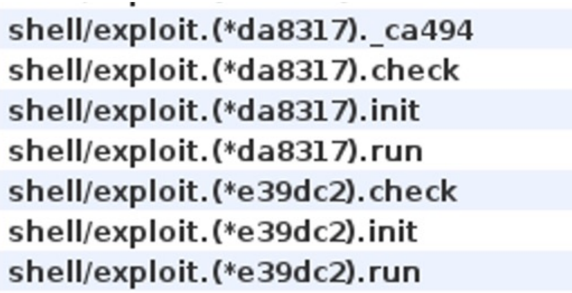


```
adojibpbhgpfdfnnlnjk/aegcfimbndeabglkjjho
adojibpbhgpfdfnnlnjk/bpmmbdkebhnagnakmbje
adojibpbhgpfdfnnlnjk/efpdcgbhkocemnpjnnfo
adojibpbhgpfdfnnlnjk/gbdgajdocapllhiljmoe
adojibpbhgpfdfnnlnjk/gbdgajdocapllhiljmoe.(*Scanner).(adojibpbhgpfdfnnlnjk/gbdgajdocapllhiljmoe
adojibpbhgpfdfnnlnjk/jemkgjopohlcdbjoccoe
main
```

- For later samples some of the function names were slightly obfuscated



```
shell/exploit.(*cve_2017_11610).check
shell/exploit.(*cve_2017_11610).exploit
shell/exploit.(*cve_2017_11610).initialize
shell/exploit.(*cve_2017_11610).port
shell/exploit.(*cve_2017_12149).check
shell/exploit.(*cve_2017_12149).exploit
shell/exploit.(*cve_2017_12149).initialize
shell/exploit.(*cve_2017_12149).port
```



```
shell/exploit.(*da8317)._ca494
shell/exploit.(*da8317).check
shell/exploit.(*da8317).init
shell/exploit.(*da8317).run
shell/exploit.(*e39dc2).check
shell/exploit.(*e39dc2).init
shell/exploit.(*e39dc2).run
```

The Sysrv botnet

The exploits

- Primarily targeting Linux and Windows servers, not IoT devices
- Initial campaigns – small set of exploits
    - Apache Tomcat RCE – used by every sample
    - CVE-2020-14882 – Oracle WebLogic RCE – used by almost every sample
    - MySQL RCE – only used by the early samples
    - CVE-2018-1000861 – Jenkins RCE – used by almost every sample
- Latest exploits
    - CVE-2021-22204 – ExifTool RCE – published in January 2021, used by samples from November 2021
    - CVE-2021-3129 – Ignition RCE – published in January 2021, used by samples in March 2021
    - CVE-2022-22947 – Spring Cloud Gateway RCE – published in January 2022, used by samples from March 2022

The Sysrv botnet

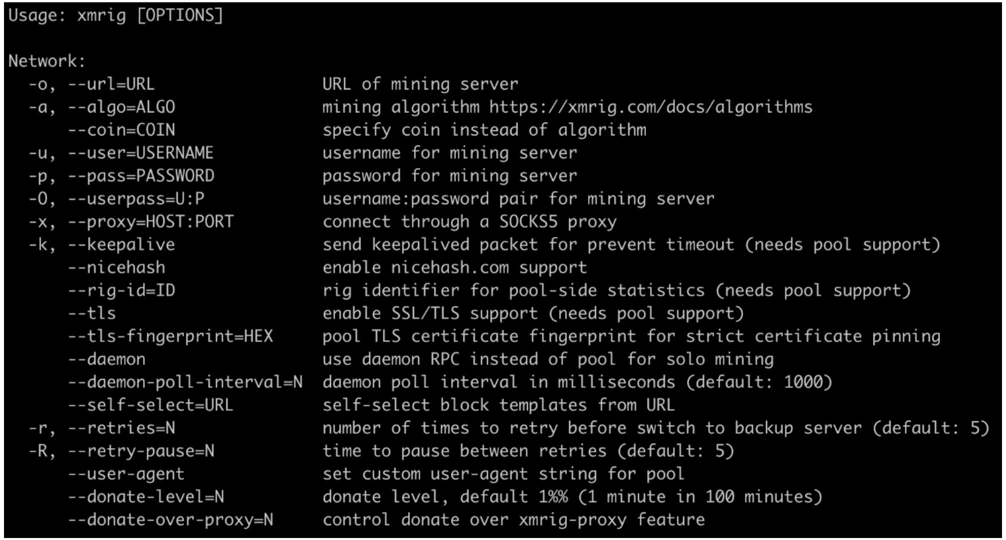
The vulnerabilities exploited

Exploits with the corresponding CVE number:	Exploits without a CVE number:
CVE-2015-8562 – Joomla! RCE	Apache Flink RCE
CVE-2017-11610 – Supervisor XML-RPC server RCE	Apache Hadoop YARN ResourceManager Unauthenticated RCE
CVE-2017-12149 – Jboss RCE	Apache NiFi Api RCE
CVE-2017-3066 – Adobe ColdFusion RCE	Apache Tomcat RCE
CVE-2017-5638 – Apache Struts RCE	Jupyter Notebook RCE
CVE-2017-9841 – PHPUnit RCE	MySQL RCE
CVE-2018-1000861 – Jenkins RCE	Redis RCE
CVE-2018-7600 – Drupal RCE	SSH brute-force
CVE-2019-0193 – Apache Solr RCE	ThinkPHP RCE
CVE-2019-10758 – Mongo Express RCE	WordPress brute-force
CVE-2019-11581 – Atlassian Jira RCE	XXL-JOB Unauth RCE
CVE-2019-15107 – Webmin RCE	
CVE-2019-3396 – Atlassian Confluence RCE	
CVE-2019-7238 – Nexus Repository Manager RCE	
CVE-2019-9193 – PostgreSQL RCE	
CVE-2020-13942 – Apache Unomi RCE	
CVE-2020-14882 – Oracle WebLogic RCE	
CVE-2020-16846 – Saltstack RCE	
CVE-2020-9496 – Apache OFBiz RCE	
CVE-2021-22204 – ExifTool RCE	
CVE-2021-3129 – Ignition RCE	
CVE-2022-22947 – Spring Cloud Gateway RCE	

The Sysrv botnet

The miner

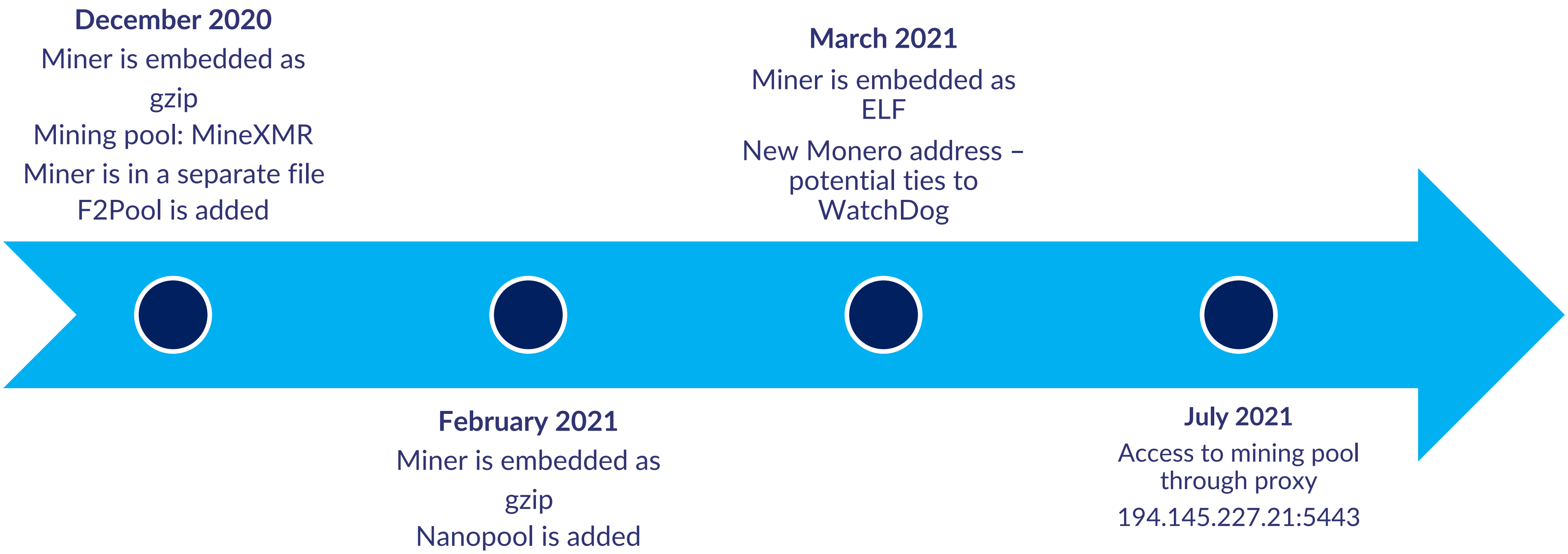
- Monero cryptocurrency mining
- Uses the open-source XMRig project to mine Monero
- Details extracted from config files
- Mining address: 49dnvYkWkZNPrDj3KF8fR1BHLBfiVArU6Hu61N9gtrZWgbRptntwht5JUrXX1ZeofwPwC6fXNxPZfGjNEChXttwWE3WGURa
- Mining pools:
    - pool.minexmr.com:5555
    - xmr.f2pool.com:13531
    - xmr-eu1.nanopool.org:14444
    - xmr-eu2.nanopool.org:14444
    - xmr-asia1.nanopool.org:14444
    - 194.145.227.21:5443



```
Usage: xmrig [OPTIONS]

Network:
  -o, --url=URL                 URL of mining server
  -a, --algo=ALGO               mining algorithm https://xmrig.com/docs/algorithms
      --coin=COIN               specify coin instead of algorithm
  -u, --user=USERNAME           username for mining server
  -p, --pass=PASSWORD           password for mining server
  -O, --userpass=U:P            username:password pair for mining server
  -x, --proxy=HOST:PORT         connect through a SOCKS5 proxy
  -k, --keepalive               send keepalived packet for prevent timeout (needs pool support)
      --nicehash                enable nicehash.com support
      --rig-id=ID               rig identifier for pool-side statistics (needs pool support)
      --tls                     enable SSL/TLS support (needs pool support)
      --tls-fingerprint=HEX     pool TLS certificate fingerprint for strict certificate pinning
      --daemon                  use daemon RPC instead of pool for solo mining
      --daemon-poll-interval=N  daemon poll interval in milliseconds (default: 1000)
      --self-select=URL         self-select block templates from URL
  -r, --retries=N               number of times to retry before switch to backup server (default: 5)
  -R, --retry-pause=N           time to pause between retries (default: 5)
      --user-agent              set custom user-agent string for pool
      --donate-level=N          donate level, default 1%% (1 minute in 100 minutes)
      --donate-over-proxy=N     control donate over xmrig-proxy feature
```

The Sysrv botnet

The miner



December 2020

Miner is embedded as gzip

Mining pool: MineXMR

Miner is in a separate file

F2Pool is added

February 2021

Miner is embedded as gzip

Nanopool is added

March 2021

Miner is embedded as ELF

New Monero address – potential ties to WatchDog

July 2021

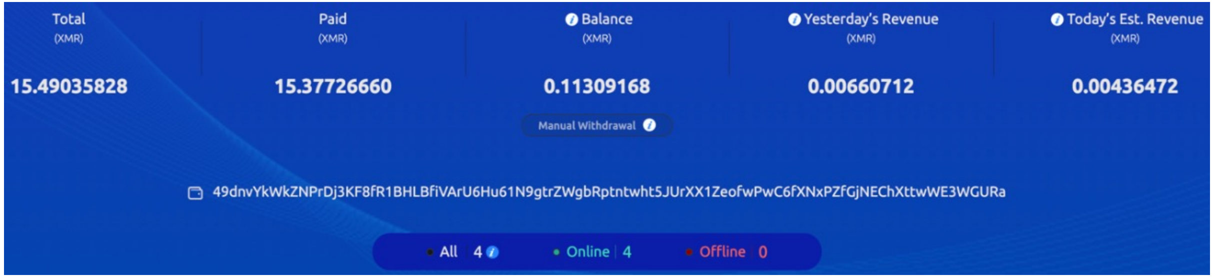
Access to mining pool through proxy

194.145.227.21:5443

The Sysrv botnet

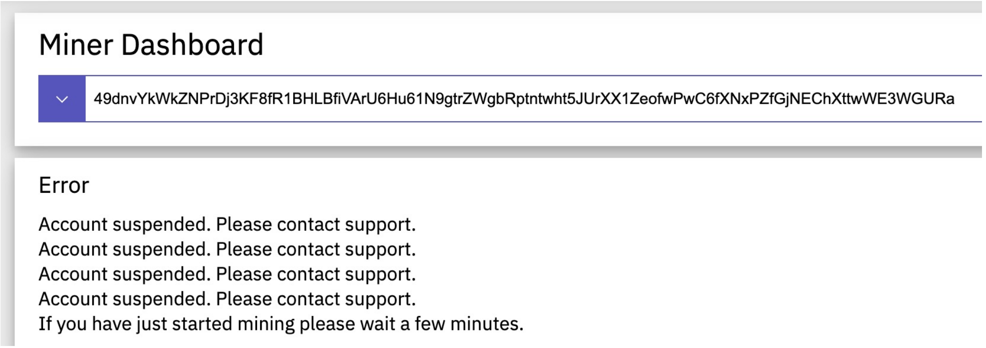
Monetization

- f2pool
  - Started in November 2020
  - 15 XMR (3900 USD)
  - Closed XMR mining pool November 2021
  - Details from September 2021
- MINEXMR
  - Suspended account
- Nanopool
  - 76 XMR (20000 USD)
  - First payment: 28 February 2021
  - Last payment: 2 July 2021



Total (XMR)	Paid (XMR)	Balance (XMR)	Yesterday's Revenue (XMR)	Today's Est. Revenue (XMR)
15.49035828	15.37726660	0.11309168	0.00660712	0.00436472

49dnvYkWkZNPrDj3KF8fR1BHLBfiVArU6Hu61N9gtrZWgbRptntwht5JUrXX1ZeofwPwC6fXNxPZfGjNEChXttwWE3WGURa

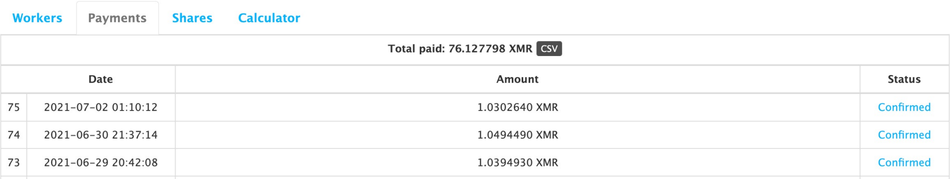


Miner Dashboard

49dnvYkWkZNPrDj3KF8fR1BHLBfiVArU6Hu61N9gtrZWgbRptntwht5JUrXX1ZeofwPwC6fXNxPZfGjNEChXttwWE3WGURa

Error

Account suspended. Please contact support.
Account suspended. Please contact support.
Account suspended. Please contact support.
Account suspended. Please contact support.
If you have just started mining please wait a few minutes.



Total paid: 76.127798 XMR CSV

	Date	Amount	Status
75	2021-07-02 01:10:12	1.0302640 XMR	Confirmed
74	2021-06-30 21:37:14	1.0494490 XMR	Confirmed
73	2021-06-29 20:42:08	1.0394930 XMR	Confirmed

Golang

Introduction

- Go (also called Golang) is an open source programming language
- Designed by Google in 2007
- Made available to the public in 2012
- Current version is Go 1.18
- https://golang.org/

- Go comes out top of the languages most developers want to learn[1]
- Advantages:
  - Simple and clear documentation
  - Easy to learn, ease of coding
  - Compiled language (faster than Python)
  - Cross compiling (Windows, Linux, macOS)
  - Scalability and concurrency
  - Garbage collection – automatic memory management

[1]: https://www.zdnet.com/article/developers-say-googles-go-is-most-sought-after-programming-language-of-2020/

Static linking

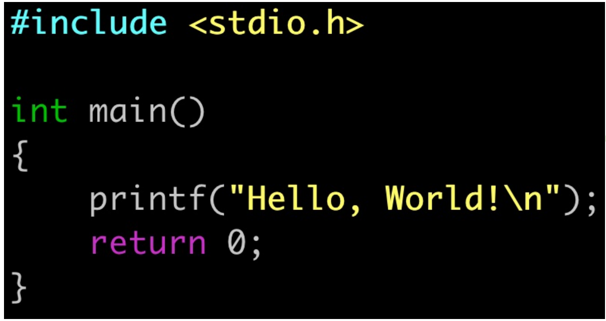
Big Bad Binaries

- Go binaries are statically linked by default
- All the necessary libraries are included in the executable image
- No dependency issues
- Large size
    - Difficult malware distribution
    - Anti – virus products have difficulty to detect
    - Reverse engineering can be more time consuming

Hello World - Unstripped

C vs Go

- C



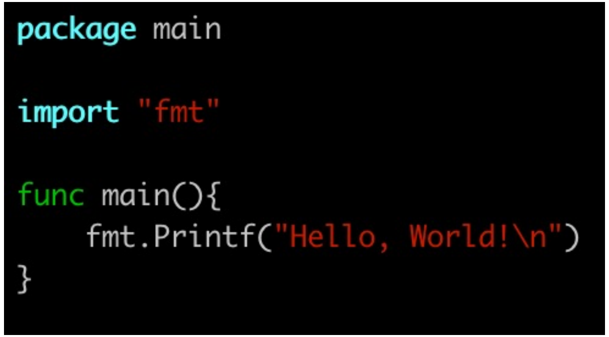
```c
#include <stdio.h>

int main()
{
    printf("Hello, World!\n");
    return 0;
}
```

gcc -o world_c world.c

ELF 64-bit LSB shared object,
x86-64, version 1 (SYSV),
dynamically linked,
not stripped

size: 16,3 kB

- Go



```go
package main

import "fmt"

func main(){
    fmt.Printf("Hello, World!\n")
}
```

go build -o world_go world.go

ELF 64-bit LSB executable,
x86-64, version 1 (SYSV),
statically linked,
not stripped

size: 2,0 MB

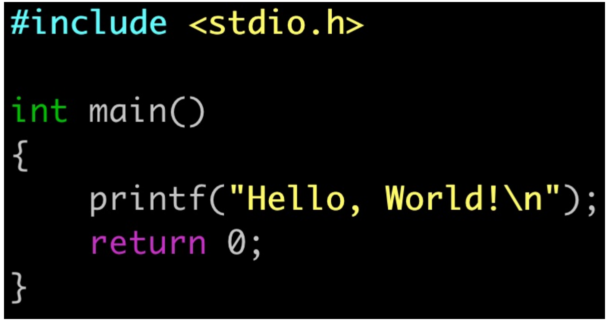
Stripped Binaries

- Discard debugging symbols
- Reduced size
- No names for routines and variables
- More difficult debugging and reverse engineering
- Malware files are usually stripped

Hello World - Stripped

C vs Go

- C



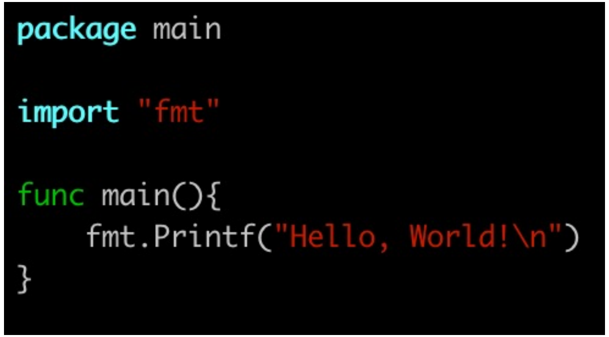
```c
#include <stdio.h>

int main()
{
    printf("Hello, World!\n");
    return 0;
}
```

gcc -o world_c_strip **-s** world.c

ELF 64-bit LSB shared object,
x86-64, version 1 (SYSV),
dynamically linked,
stripped

size: 14,1 kB

- Go



```go
package main

import "fmt"

func main(){
    fmt.Printf("Hello, World!\n")
}
```

go build -o world_go_strip –
ldflags “-s” world.go

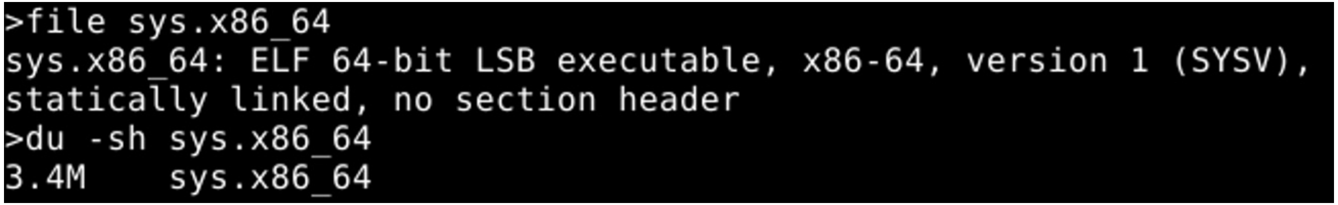
ELF 64-bit LSB executable,
x86-64, version 1 (SYSV),
statically linked,
stripped

size: 1,3 MB

Sysrv

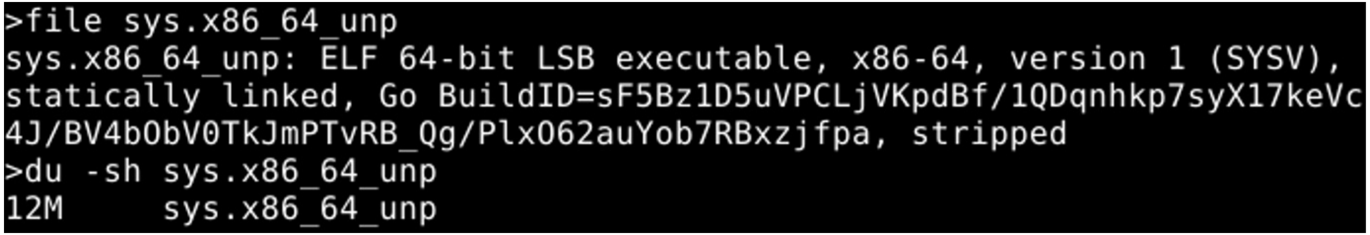
Example files

- sys.x86_64
    - UPX packed
    - SHA256 = f719736bb794d9a2a4fc3574391f34920130709b659231003a6fdcf34ecf68ec



```
>file sys.x86_64
sys.x86_64: ELF 64-bit LSB executable, x86-64, version 1 (SYSV),
statically linked, no section header
>du -sh sys.x86_64
3.4M    sys.x86_64
```

- sys.x86_64_unp
    - Unpacked
    - SHA256 = 5190dda119756910f41646609def181b7549fbf14cd761f3053721500af0ead3

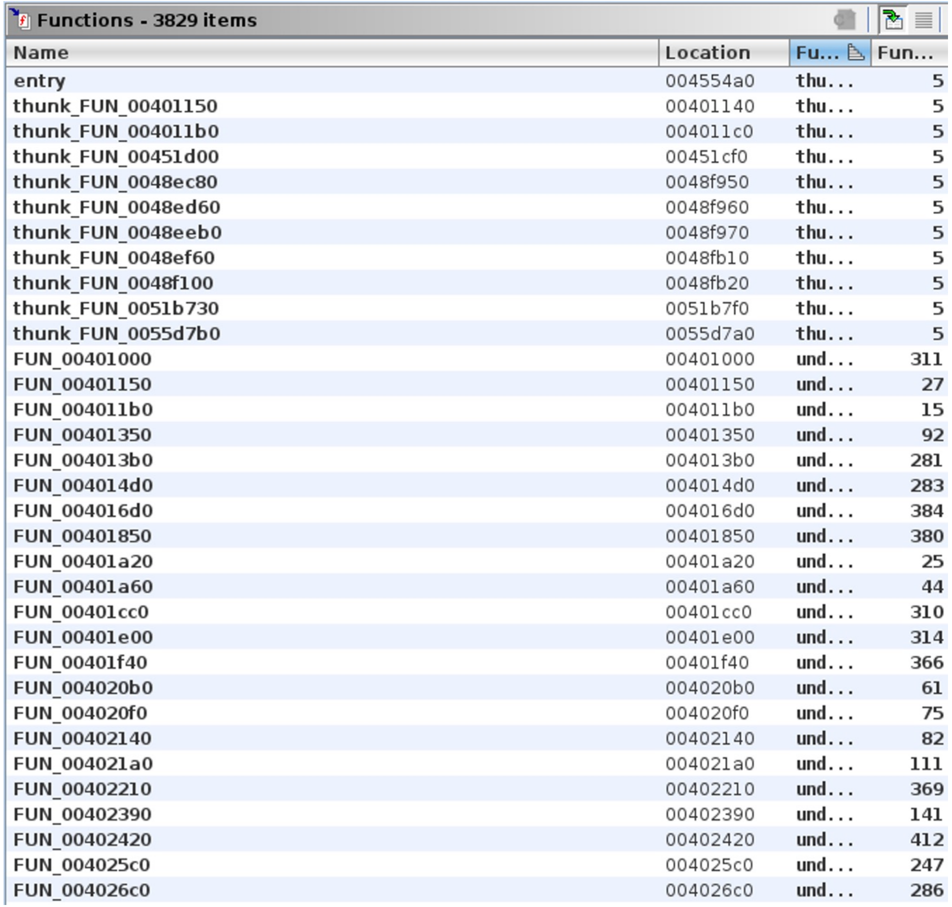


```
>file sys.x86_64_unp
sys.x86_64_unp: ELF 64-bit LSB executable, x86-64, version 1 (SYSV),
statically linked, Go BuildID=sF5Bz1D5uVPCLjVKpdBf/1QDqnhkp7syX17keVc
4J/BV4bObV0TkJmPTvRB_Qg/PlxO62auYob7RBxzjfpa, stripped
>du -sh sys.x86_64_unp
12M     sys.x86_64_unp
```

Recover function names

Function list

- 3829 function recognized by Ghidra
- No proper function names
- Not helpful in reverse engineering



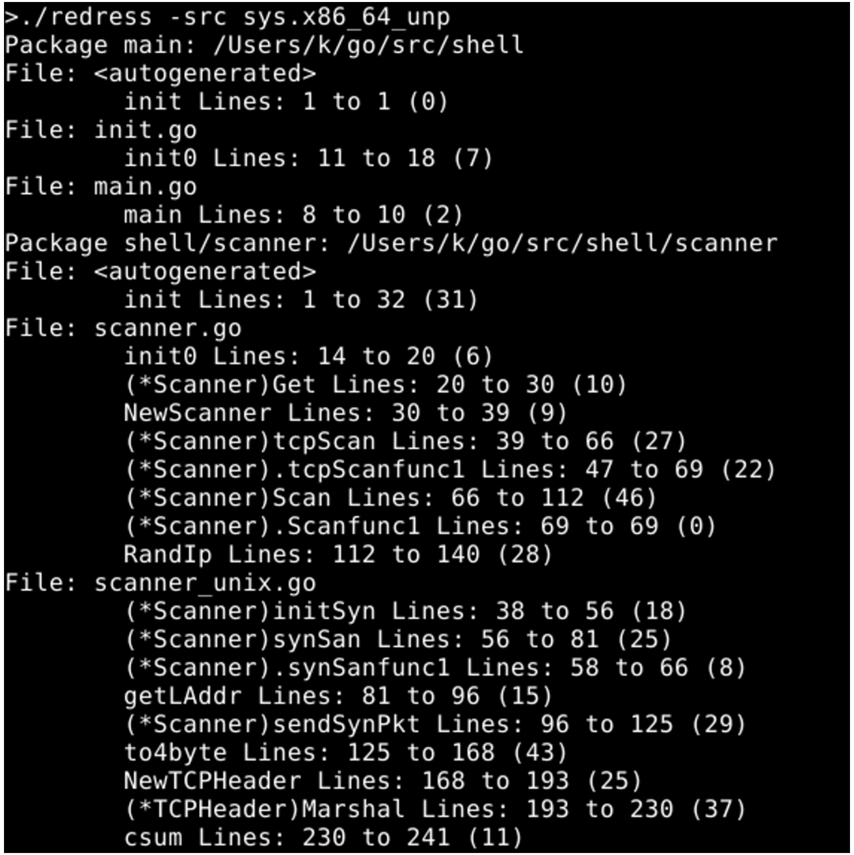
Functions - 3829 items

Name	Location	Fu...	Fun...
entry	004554a0	thu...	5
thunk_FUN_00401150	00401140	thu...	5
thunk_FUN_004011b0	004011c0	thu...	5
thunk_FUN_00451d00	00451cf0	thu...	5
thunk_FUN_0048ec80	0048f950	thu...	5
thunk_FUN_0048ed60	0048f960	thu...	5
thunk_FUN_0048eeb0	0048f970	thu...	5
thunk_FUN_0048ef60	0048fb10	thu...	5
thunk_FUN_0048f100	0048fb20	thu...	5
thunk_FUN_0051b730	0051b7f0	thu...	5
thunk_FUN_0055d7b0	0055d7a0	thu...	5
FUN_00401000	00401000	und...	311
FUN_00401150	00401150	und...	27
FUN_004011b0	004011b0	und...	15
FUN_00401350	00401350	und...	92
FUN_004013b0	004013b0	und...	281
FUN_004014d0	004014d0	und...	283
FUN_004016d0	004016d0	und...	384
FUN_00401850	00401850	und...	380
FUN_00401a20	00401a20	und...	25
FUN_00401a60	00401a60	und...	44
FUN_00401cc0	00401cc0	und...	310
FUN_00401e00	00401e00	und...	314
FUN_00401f40	00401f40	und...	366
FUN_004020b0	004020b0	und...	61
FUN_004020f0	004020f0	und...	75
FUN_00402140	00402140	und...	82
FUN_004021a0	004021a0	und...	111
FUN_00402210	00402210	und...	369
FUN_00402390	00402390	und...	141
FUN_00402420	00402420	und...	412
FUN_004025c0	004025c0	und...	247
FUN_004026c0	004026c0	und...	286

Recover function names

Find the strings

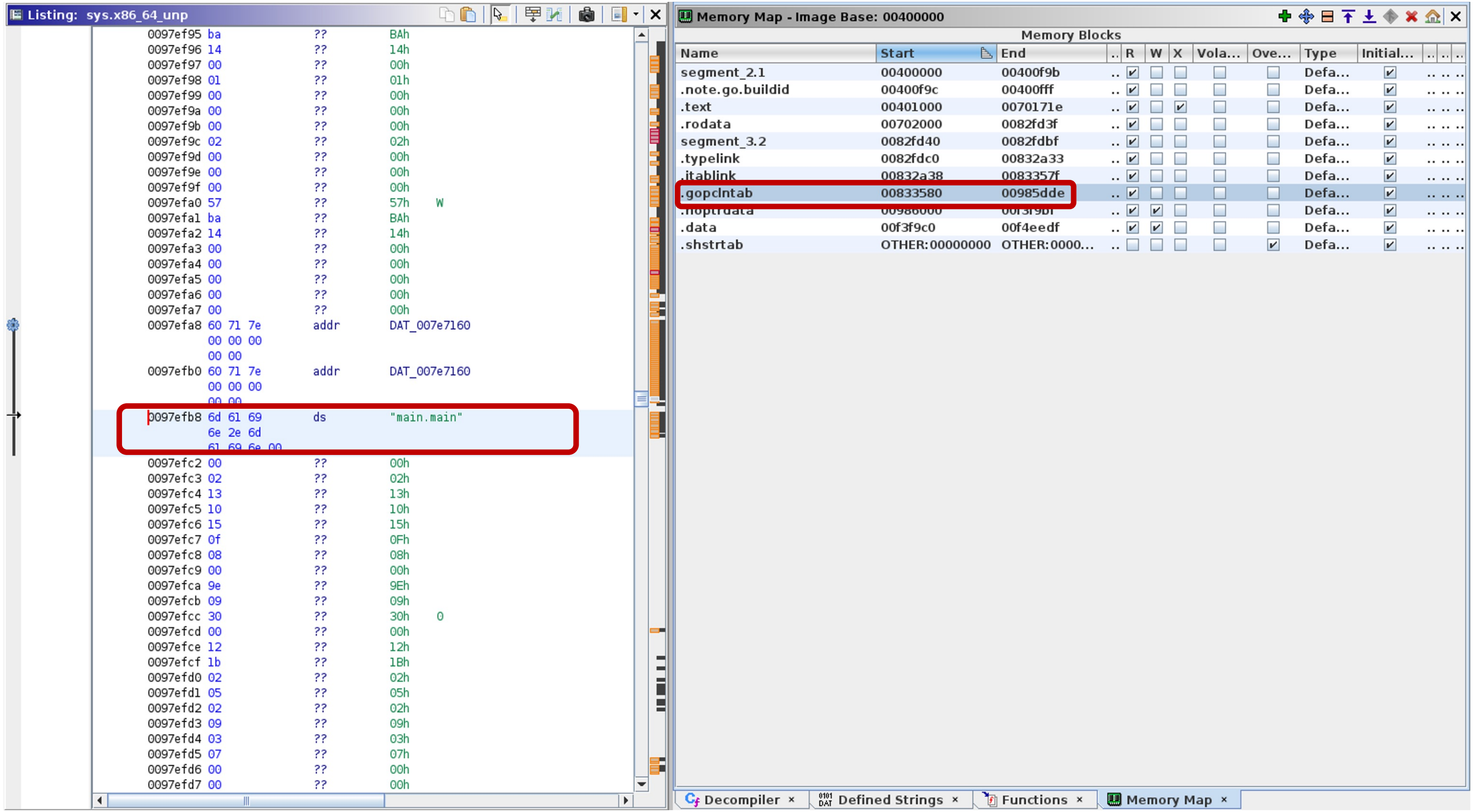
- Function name strings are present in the binary
- Redress – tool for analyzing stripped Go binaries https://github.com/goretk/redress



```
>./redress -src sys.x86_64_unp
Package main: /Users/k/go/src/shell
File: <autogenerated>
        init Lines: 1 to 1 (0)
File: init.go
        init0 Lines: 11 to 18 (7)
File: main.go
        main Lines: 8 to 10 (2)
Package shell/scanner: /Users/k/go/src/shell/scanner
File: <autogenerated>
        init Lines: 1 to 32 (31)
File: scanner.go
        init0 Lines: 14 to 20 (6)
        (*Scanner)Get Lines: 20 to 30 (10)
        NewScanner Lines: 30 to 39 (9)
        (*Scanner)tcpScan Lines: 39 to 66 (27)
        (*Scanner).tcpScanfunc1 Lines: 47 to 69 (22)
        (*Scanner)Scan Lines: 66 to 112 (46)
        (*Scanner).Scanfunc1 Lines: 69 to 69 (0)
        RandIp Lines: 112 to 140 (28)
File: scanner_unix.go
        (*Scanner)initSyn Lines: 38 to 56 (18)
        (*Scanner)synSan Lines: 56 to 81 (25)
        (*Scanner).synSanfunc1 Lines: 58 to 66 (8)
        getLAddr Lines: 81 to 96 (15)
        (*Scanner)sendSynPkt Lines: 96 to 125 (29)
        to4byte Lines: 125 to 168 (43)
        NewTCPHeader Lines: 168 to 193 (25)
        (*TCPHeader)Marshal Lines: 193 to 230 (37)
        csum Lines: 230 to 241 (11)
```

Recover function names

pclntab



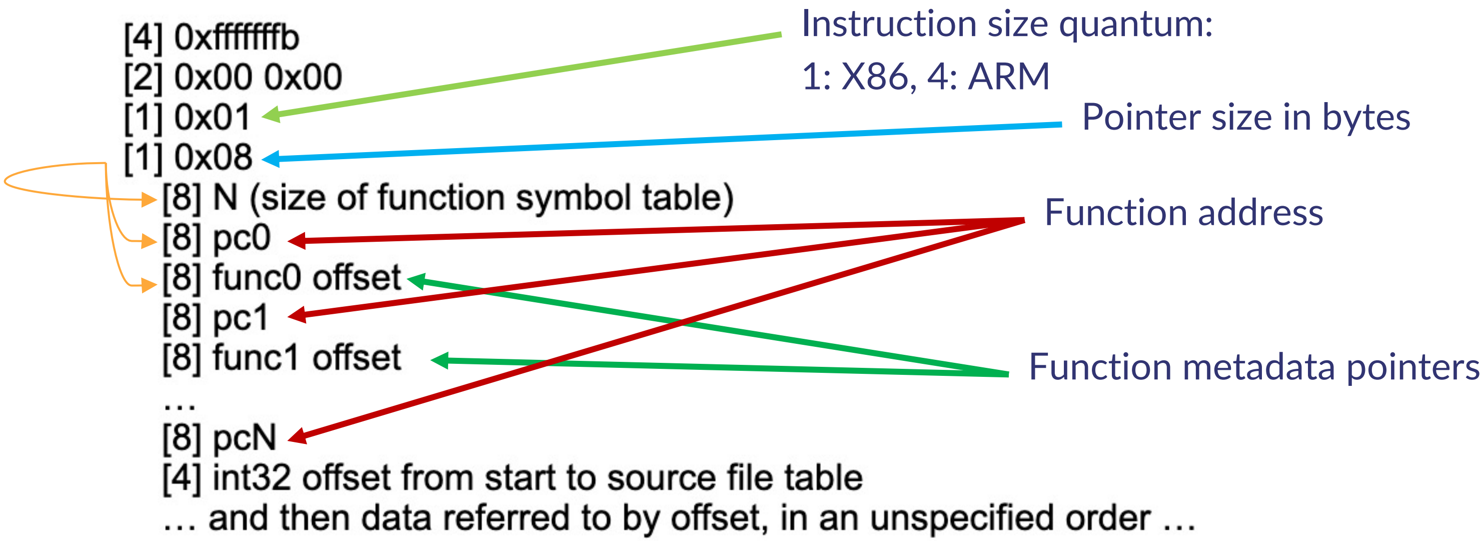
```
0097ef95 ba        ??      BAh
0097ef96 14        ??      14h
0097ef97 00        ??      00h
0097ef98 01        ??      01h
0097ef99 00        ??      00h
0097ef9a 00        ??      00h
0097ef9b 00        ??      00h
0097ef9c 02        ??      02h
0097ef9d 00        ??      00h
0097ef9e 00        ??      00h
0097ef9f 00        ??      00h
0097efa0 57        ??      57h    W
0097efa1 ba        ??      BAh
0097efa2 14        ??      14h
0097efa3 00        ??      00h
0097efa4 00        ??      00h
0097efa5 00        ??      00h
0097efa6 00        ??      00h
0097efa7 00        ??      00h
0097efa8 60 71 7e  addr    DAT_007e7160
         00 00 00
         00 00
0097efb0 60 71 7e  addr    DAT_007e7160
         00 00 00
         00 00
0097efb8 6d 61 69  ds      "main.main"
         6e 2e 6d
         61 69 6e 00
0097efc2 00        ??      00h
0097efc3 02        ??      02h
0097efc4 13        ??      13h
0097efc5 10        ??      10h
0097efc6 15        ??      15h
0097efc7 0f        ??      0Fh
0097efc8 08        ??      08h
0097efc9 00        ??      00h
0097efca 9e        ??      9Eh
0097efcb 09        ??      09h
0097efcc 30        ??      30h    0
0097efcd 00        ??      00h
0097efce 12        ??      12h
0097efcf 1b        ??      1Bh
0097efd0 02        ??      02h
0097efd1 05        ??      05h
0097efd2 02        ??      02h
0097efd3 09        ??      09h
0097efd4 03        ??      03h
0097efd5 07        ??      07h
0097efd6 00        ??      00h
0097efd7 00        ??      00h
```

Name	Start	End	..	R	W	X	Vola...	Ove...	Type	Initial...	..	..	..
segment_2.1	00400000	00400f9b	..	☑	□	□	□	□	Defa...	☑	..	..	..
.note.go.buildid	00400f9c	00400fff	..	☑	□	□	□	□	Defa...	☑	..	..	..
.text	00401000	0070171e	..	☑	□	☑	□	□	Defa...	☑	..	..	..
.rodata	00702000	0082fd3f	..	☑	□	□	□	□	Defa...	☑	..	..	..
segment_3.2	0082fd40	0082fdbf	..	☑	□	□	□	□	Defa...	☑	..	..	..
.typelink	0082fdc0	00832a33	..	☑	□	□	□	□	Defa...	☑	..	..	..
.itablink	00832a38	0083357f	..	☑	□	□	□	□	Defa...	☑	..	..	..
.gopclntab	00833580	00985dde	..	☑	□	□	□	□	Defa...	☑	..	..	..
.noptrdata	00986000	00f3f9bf	..	☑	☑	□	□	□	Defa...	☑	..	..	..
.data	00f3f9c0	00f4eedf	..	☑	☑	□	□	□	Defa...	☑	..	..	..
.shstrtab	OTHER:00000000	OTHER:0000...	..	□	□	□	□	☑	Defa...	☑	..	..	..

Recover function names

pclntab

- Detailed documentation of pclntab[1] is available



```
[4] 0xfffffffb
[2] 0x00 0x00
[1] 0x01
[1] 0x08
    [8] N (size of function symbol table)
    [8] pc0
    [8] func0 offset
    [8] pc1
    [8] func1 offset
    …
    [8] pcN
    [4] int32 offset from start to source file table
    … and then data referred to by offset, in an unspecified order …
```

Instruction size quantum:

1: X86, 4: ARM

Pointer size in bytes

Function address

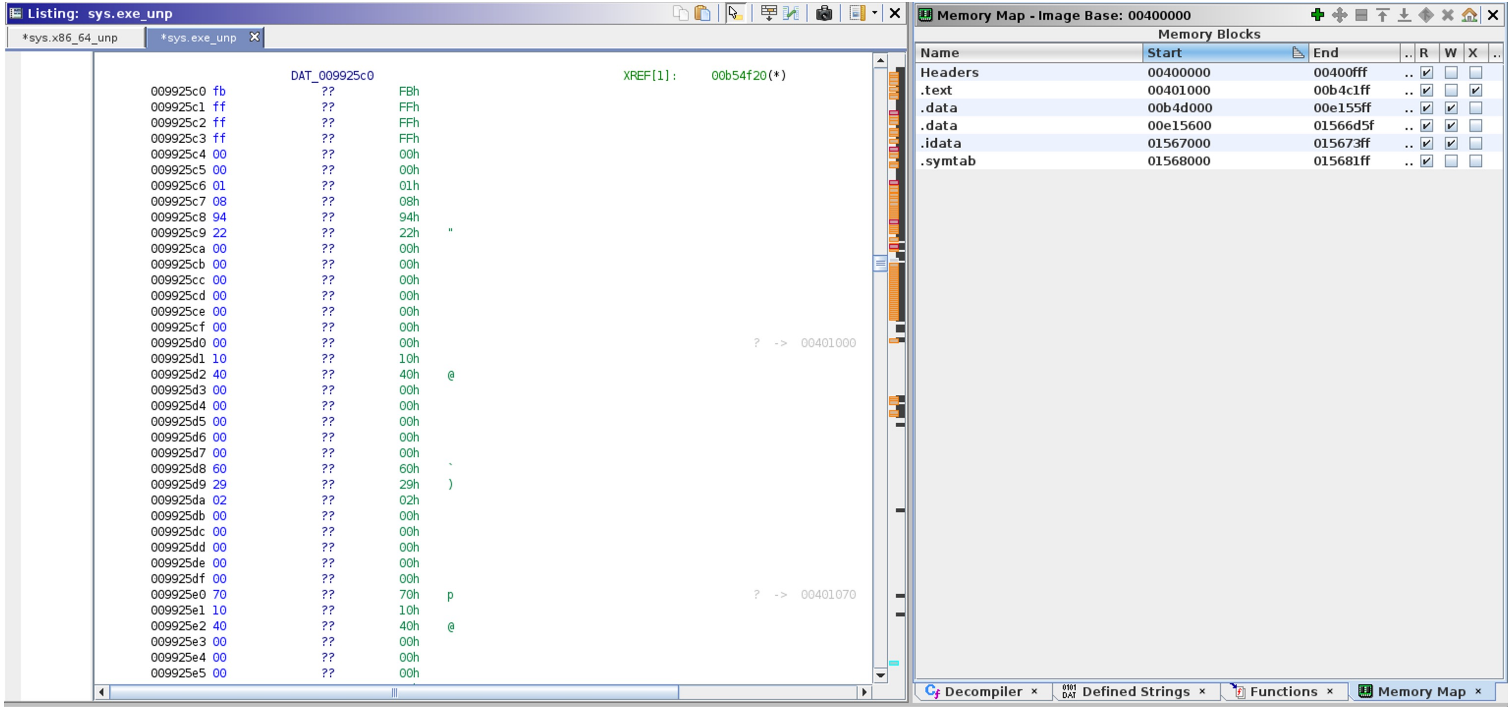
Function metadata pointers

[1] https://docs.google.com/document/d/1lyPIbmsYbXnpNj57a261hgOYVpNRcgydurVQIyZOz_o/pub

Recover function names

pclntab in Windows

- Not a separate section -> Look for the structure



```
                        DAT_009925c0                                   XREF[1]:     00b54f20(*)
        009925c0 fb           ??         FBh
        009925c1 ff           ??         FFh
        009925c2 ff           ??         FFh
        009925c3 ff           ??         FFh
        009925c4 00           ??         00h
        009925c5 00           ??         00h
        009925c6 01           ??         01h
        009925c7 08           ??         08h
        009925c8 94           ??         94h
        009925c9 22           ??         22h    "
        009925ca 00           ??         00h
        009925cb 00           ??         00h
        009925cc 00           ??         00h
        009925cd 00           ??         00h
        009925ce 00           ??         00h
        009925cf 00           ??         00h
        009925d0 00           ??         00h                                      ?  ->  00401000
        009925d1 10           ??         10h
        009925d2 40           ??         40h    @
        009925d3 00           ??         00h
        009925d4 00           ??         00h
        009925d5 00           ??         00h
        009925d6 00           ??         00h
        009925d7 00           ??         00h
        009925d8 60           ??         60h    `
        009925d9 29           ??         29h    )
        009925da 02           ??         02h
        009925db 00           ??         00h
        009925dc 00           ??         00h
        009925dd 00           ??         00h
        009925de 00           ??         00h
        009925df 00           ??         00h
        009925e0 70           ??         70h    p                                 ?  ->  00401070
        009925e1 10           ??         10h
        009925e2 40           ??         40h    @
        009925e3 00           ??         00h
        009925e4 00           ??         00h
        009925e5 00           ??         00h
```

Memory Blocks

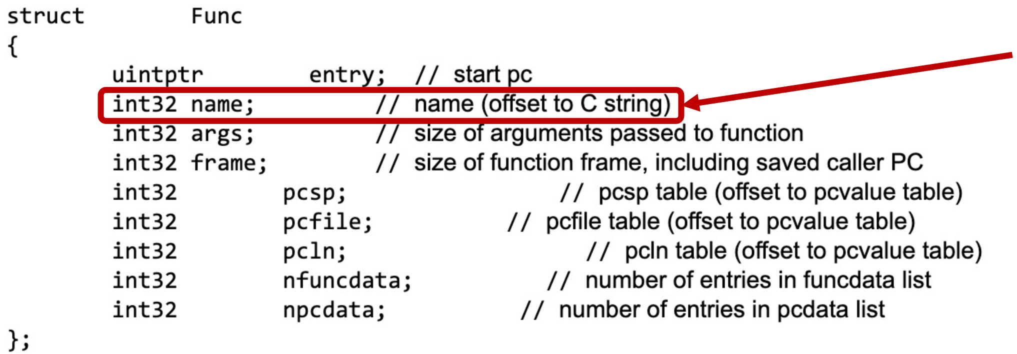
Name	Start	End	..	R	W	X	..
Headers	00400000	00400fff	..	☑	□	□	
.text	00401000	00b4c1ff	..	☑	□	☑	
.data	00b4d000	00e155ff	..	☑	☑	□	
.data	00e15600	01566d5f	..	☑	☑	□	
.idata	01567000	015673ff	..	☑	☑	□	
.symtab	01568000	015681ff	..	☑	□	□	

Binary: unpacked sys.exe (03f5f4470bc9cdeb62c3afecba326f3edf2c1ca46063490dffd5a3cede569bad)

Recover function names

pclntab

- Function metadata

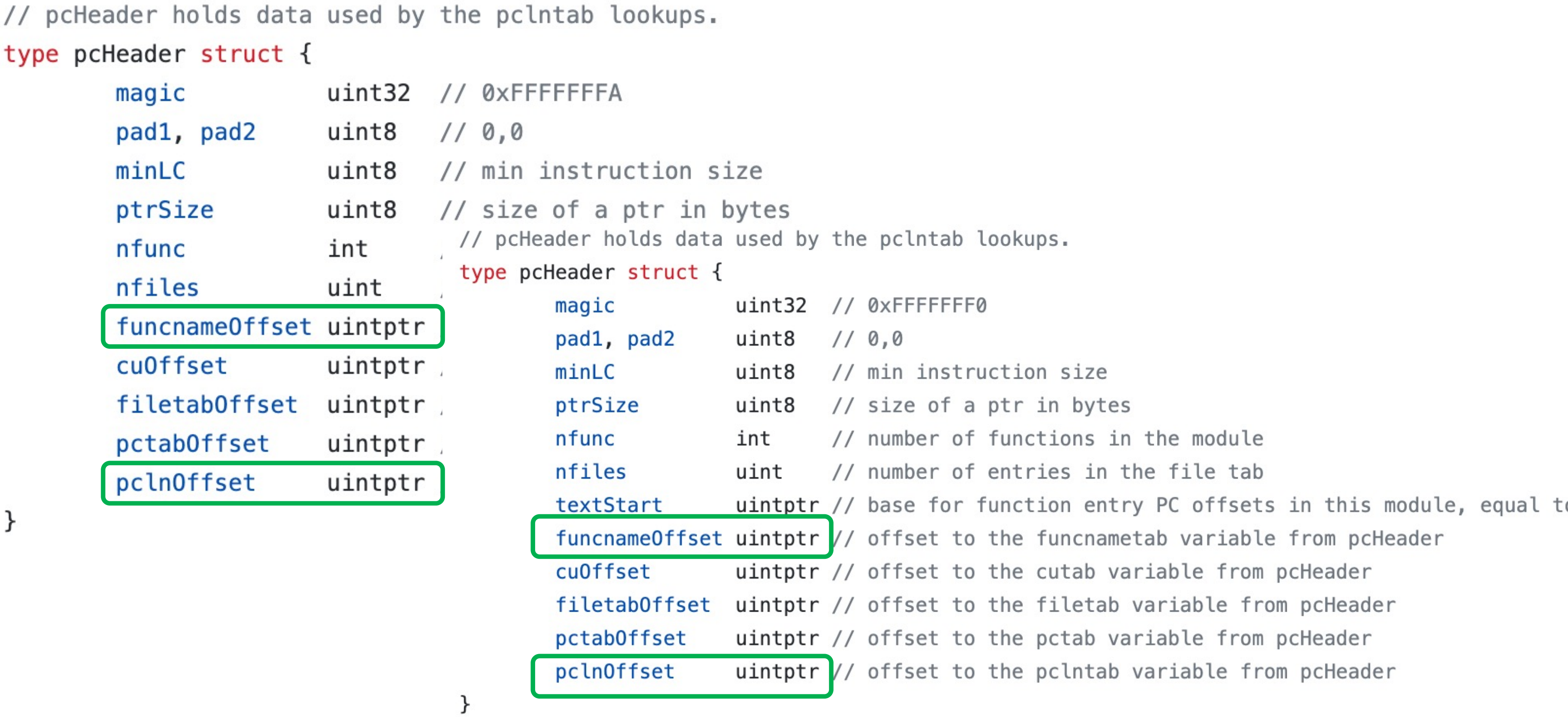


```
struct        Func
{
        uintptr        entry;  // start pc
        int32 name;            // name (offset to C string)
        int32 args;            // size of arguments passed to function
        int32 frame;           // size of function frame, including saved caller PC
        int32       pcsp;                 // pcsp table (offset to pcvalue table)
        int32       pcfile;           // pcfile table (offset to pcvalue table)
        int32       pcln;                   // pcln table (offset to pcvalue table)
        int32       nfuncdata;            // number of entries in funcdata list
        int32       npcdata;            // number of entries in pcdata list
};
```

Function name offset

Recover function names

pclntab (from go 1.16 and go 1.18)

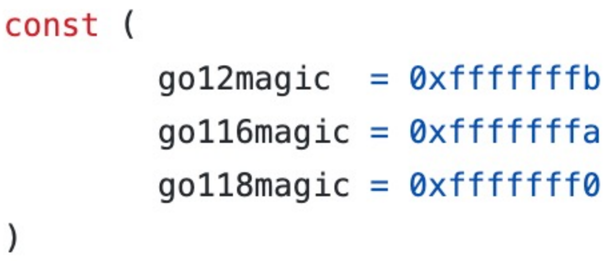


```
// pcHeader holds data used by the pclntab lookups.
type pcHeader struct {
        magic          uint32  // 0xFFFFFFFA
        pad1, pad2     uint8   // 0,0
        minLC          uint8   // min instruction size
        ptrSize        uint8   // size of a ptr in bytes
        nfunc          int
        nfiles         uint
        funcnameOffset uintptr
        cuOffset       uintptr
        filetabOffset  uintptr
        pctabOffset    uintptr
        pclnOffset     uintptr
}
```

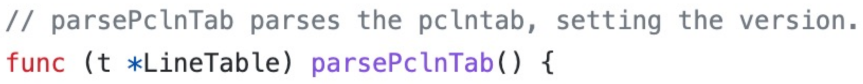
```
// pcHeader holds data used by the pclntab lookups.
type pcHeader struct {
        magic          uint32  // 0xFFFFFFF0
        pad1, pad2     uint8   // 0,0
        minLC          uint8   // min instruction size
        ptrSize        uint8   // size of a ptr in bytes
        nfunc          int     // number of functions in the module
        nfiles         uint    // number of entries in the file tab
        textStart      uintptr // base for function entry PC offsets in this module, equal t
        funcnameOffset uintptr // offset to the funcnametab variable from pcHeader
        cuOffset       uintptr // offset to the cutab variable from pcHeader
        filetabOffset  uintptr // offset to the filetab variable from pcHeader
        pctabOffset    uintptr // offset to the pctab variable from pcHeader
        pclnOffset     uintptr // offset to the pclntab variable from pcHeader
}
```

Recover function names

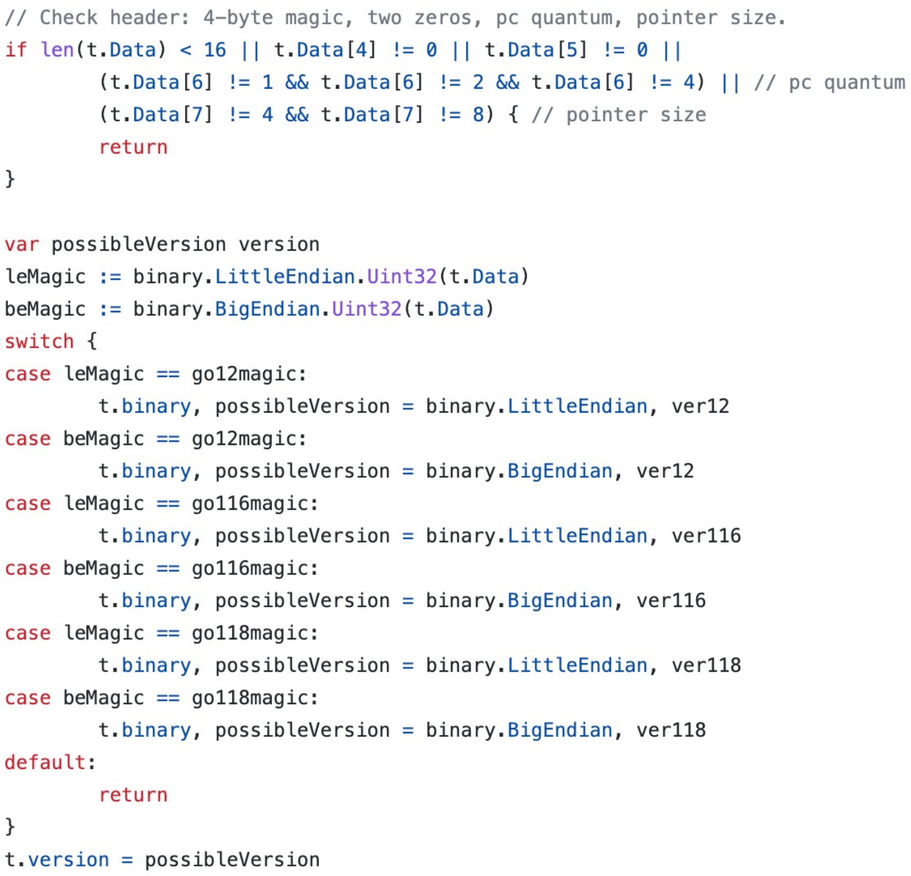
pclntab (from go 1.16 and go 1.18)



```
const (
        go12magic  = 0xfffffffb
        go116magic = 0xfffffffa
        go118magic = 0xfffffff0
)
```



```
// parsePclnTab parses the pclntab, setting the version.
func (t *LineTable) parsePclnTab() {
```



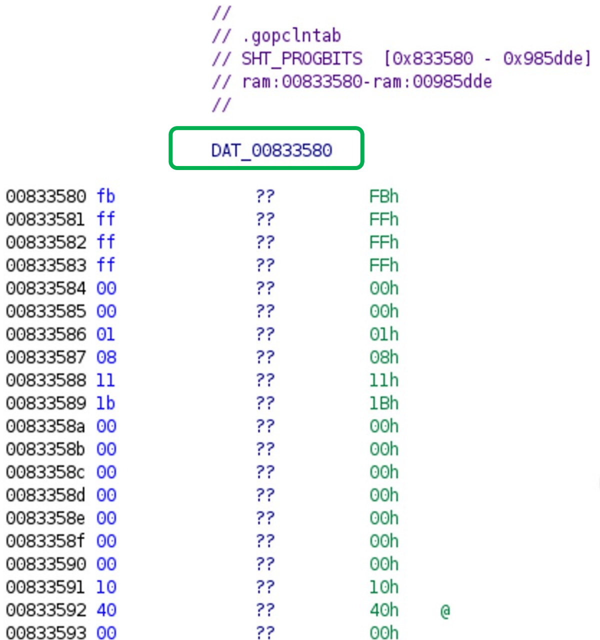
```
// Check header: 4-byte magic, two zeros, pc quantum, pointer size.
if len(t.Data) < 16 || t.Data[4] != 0 || t.Data[5] != 0 ||
        (t.Data[6] != 1 && t.Data[6] != 2 && t.Data[6] != 4) || // pc quantum
        (t.Data[7] != 4 && t.Data[7] != 8) { // pointer size
        return
}

var possibleVersion version
leMagic := binary.LittleEndian.Uint32(t.Data)
beMagic := binary.BigEndian.Uint32(t.Data)
switch {
case leMagic == go12magic:
        t.binary, possibleVersion = binary.LittleEndian, ver12
case beMagic == go12magic:
        t.binary, possibleVersion = binary.BigEndian, ver12
case leMagic == go116magic:
        t.binary, possibleVersion = binary.LittleEndian, ver116
case beMagic == go116magic:
        t.binary, possibleVersion = binary.BigEndian, ver116
case leMagic == go118magic:
        t.binary, possibleVersion = binary.LittleEndian, ver118
case beMagic == go118magic:
        t.binary, possibleVersion = binary.BigEndian, ver118
default:
        return
}
t.version = possibleVersion
```

Recover function names

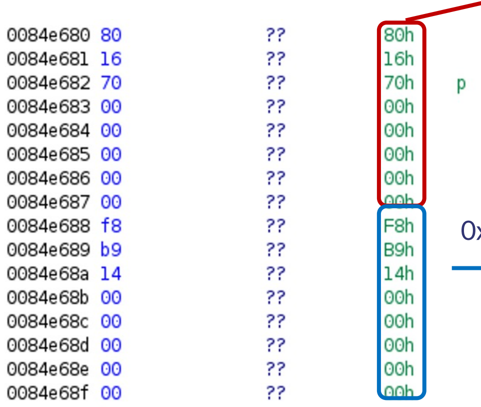
Idea

Function name recovery steps:

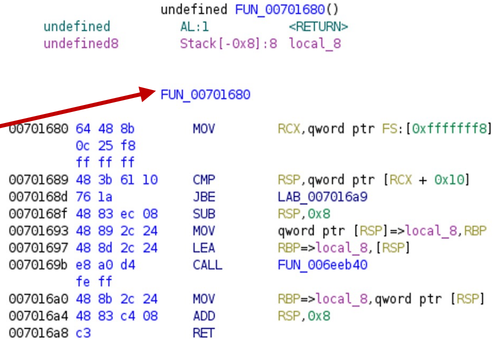
- Locate pclntab structure
- Extract function addresses
- Find function name offsets



```
//
// .gopclntab
// SHT_PROGBITS  [0x833580 - 0x985dde]
// ram:00833580-ram:00985dde
//

DAT_00833580

00833580 fb         ??         FBh
00833581 ff         ??         FFh
00833582 ff         ??         FFh
00833583 ff         ??         FFh
00833584 00         ??         00h
00833585 00         ??         00h
00833586 01         ??         01h
00833587 08         ??         08h
00833588 11         ??         11h
00833589 1b         ??         1Bh
0083358a 00         ??         00h
0083358b 00         ??         00h
0083358c 00         ??         00h
0083358d 00         ??         00h
0083358e 00         ??         00h
0083358f 00         ??         00h
00833590 00         ??         00h
00833591 10         ??         10h
00833592 40         ??         40h    @
00833593 00         ??         00h
```

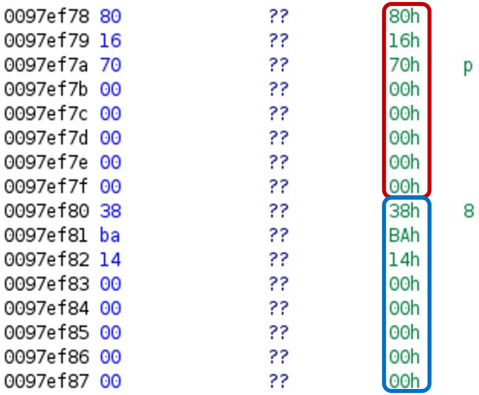


```
0084e680 80         ??         80h
0084e681 16         ??         16h
0084e682 70         ??         70h    p
0084e683 00         ??         00h
0084e684 00         ??         00h
0084e685 00         ??         00h
0084e686 00         ??         00h
0084e687 00         ??         00h
0084e688 f8         ??         F8h
0084e689 b9         ??         B9h
0084e68a 14         ??         14h
0084e68b 00         ??         00h
0084e68c 00         ??         00h
0084e68d 00         ??         00h
0084e68e 00         ??         00h
0084e68f 00         ??         00h
```



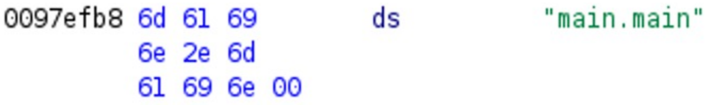
```
                    undefined FUN_00701680()
     undefined          AL:1             <RETURN>
     undefined8         Stack[-0x8]:8    local_8

                    FUN_00701680

00701680 64 48 8b       MOV        RCX,qword ptr FS:[0xfffffff8]
         0c 25 f8
         ff ff ff
00701689 48 3b 61 10    CMP        RSP,qword ptr [RCX + 0x10]
0070168d 76 1a          JBE        LAB_007016a9
0070168f 48 83 ec 08    SUB        RSP,0x8
00701693 48 89 2c 24    MOV        qword ptr [RSP]=>local_8,RBP
00701697 48 8d 2c 24    LEA        RBP=>local_8,[RSP]
0070169b e8 a0 d4       CALL       FUN_006eeb40
         fe ff
007016a0 48 8b 2c 24    MOV        RBP=>local_8,qword ptr [RSP]
007016a4 48 83 c4 08    ADD        RSP,0x8
007016a8 c3             RET
```

0x833580 + 0x14B9F8 = 0x97EF78



```
0097ef78 80         ??         80h
0097ef79 16         ??         16h
0097ef7a 70         ??         70h    p
0097ef7b 00         ??         00h
0097ef7c 00         ??         00h
0097ef7d 00         ??         00h
0097ef7e 00         ??         00h
0097ef7f 00         ??         00h
0097ef80 38         ??         38h    8
0097ef81 ba         ??         BAh
0097ef82 14         ??         14h
0097ef83 00         ??         00h
0097ef84 00         ??         00h
0097ef85 00         ??         00h
0097ef86 00         ??         00h
0097ef87 00         ??         00h
```

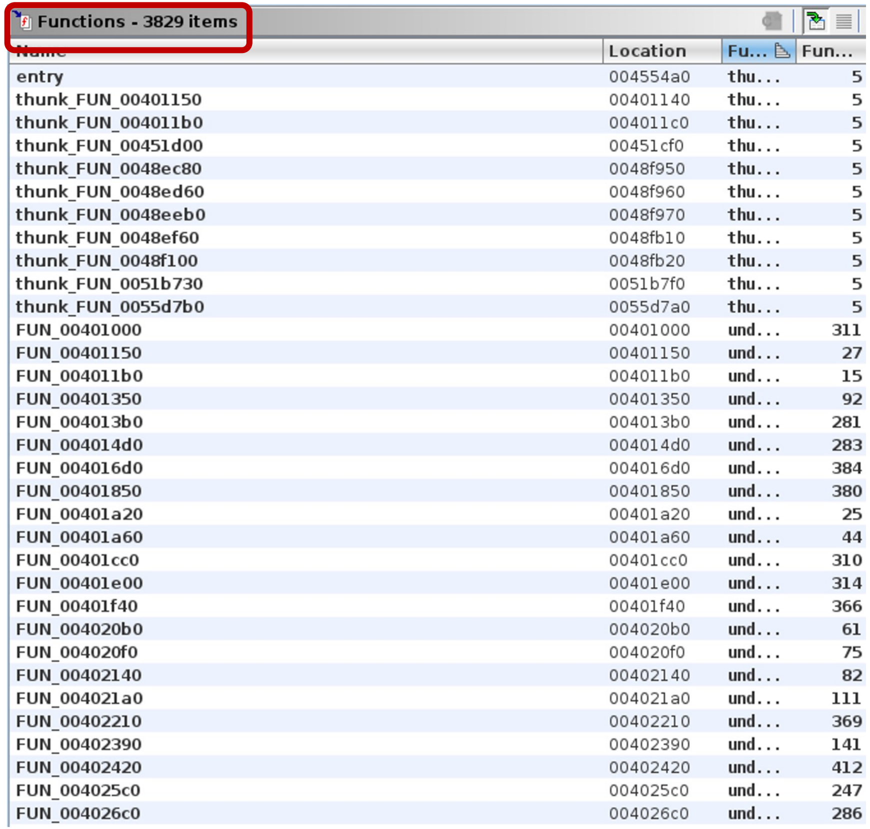
0x833580 + 0x14BA38 = 0x97EFB8



```
0097efb8 6d 61 69      ds      "main.main"
         6e 2e 6d
         61 69 6e 00
```

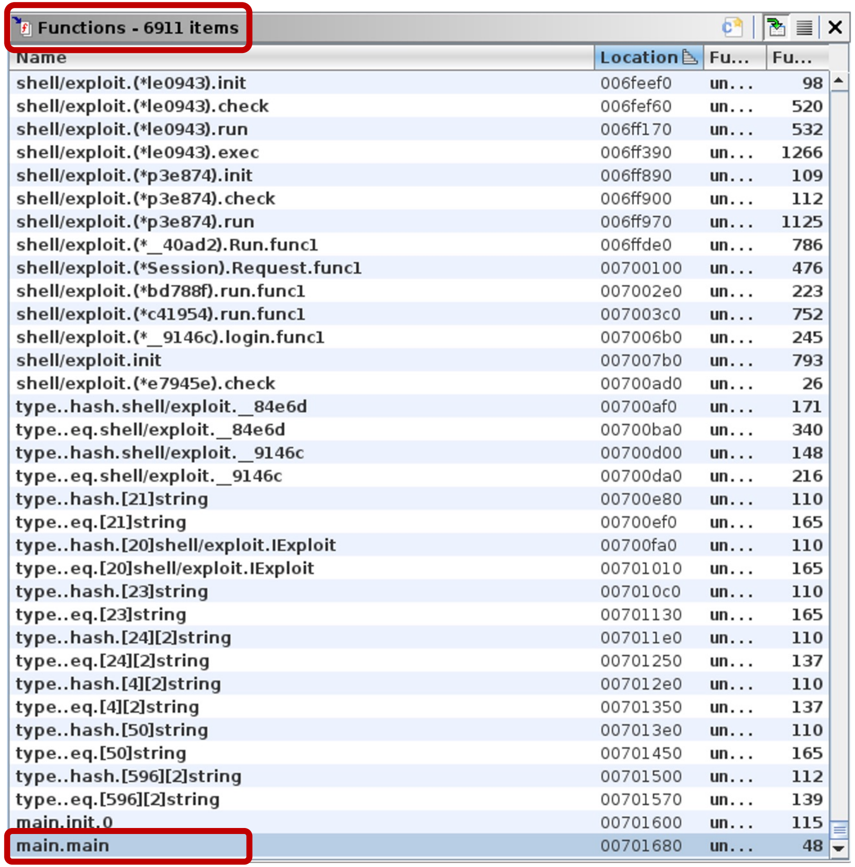
Recover function names

Executing our script



Functions - 3829 items

Name	Location	Fu...	Fun...
entry	004554a0	thu...	5
thunk_FUN_00401150	00401140	thu...	5
thunk_FUN_004011b0	004011c0	thu...	5
thunk_FUN_00451d00	00451cf0	thu...	5
thunk_FUN_0048ec80	0048f950	thu...	5
thunk_FUN_0048ed60	0048f960	thu...	5
thunk_FUN_0048eeb0	0048f970	thu...	5
thunk_FUN_0048ef60	0048fb10	thu...	5
thunk_FUN_0048f100	0048fb20	thu...	5
thunk_FUN_0051b730	0051b7f0	thu...	5
thunk_FUN_0055d7b0	0055d7a0	thu...	5
FUN_00401000	00401000	und...	311
FUN_00401150	00401150	und...	27
FUN_004011b0	004011b0	und...	15
FUN_00401350	00401350	und...	92
FUN_004013b0	004013b0	und...	281
FUN_004014d0	004014d0	und...	283
FUN_004016d0	004016d0	und...	384
FUN_00401850	00401850	und...	380
FUN_00401a20	00401a20	und...	25
FUN_00401a60	00401a60	und...	44
FUN_00401cc0	00401cc0	und...	310
FUN_00401e00	00401e00	und...	314
FUN_00401f40	00401f40	und...	366
FUN_004020b0	004020b0	und...	61
FUN_004020f0	004020f0	und...	75
FUN_00402140	00402140	und...	82
FUN_004021a0	004021a0	und...	111
FUN_00402210	00402210	und...	369
FUN_00402390	00402390	und...	141
FUN_00402420	00402420	und...	412
FUN_004025c0	004025c0	und...	247
FUN_004026c0	004026c0	und...	286



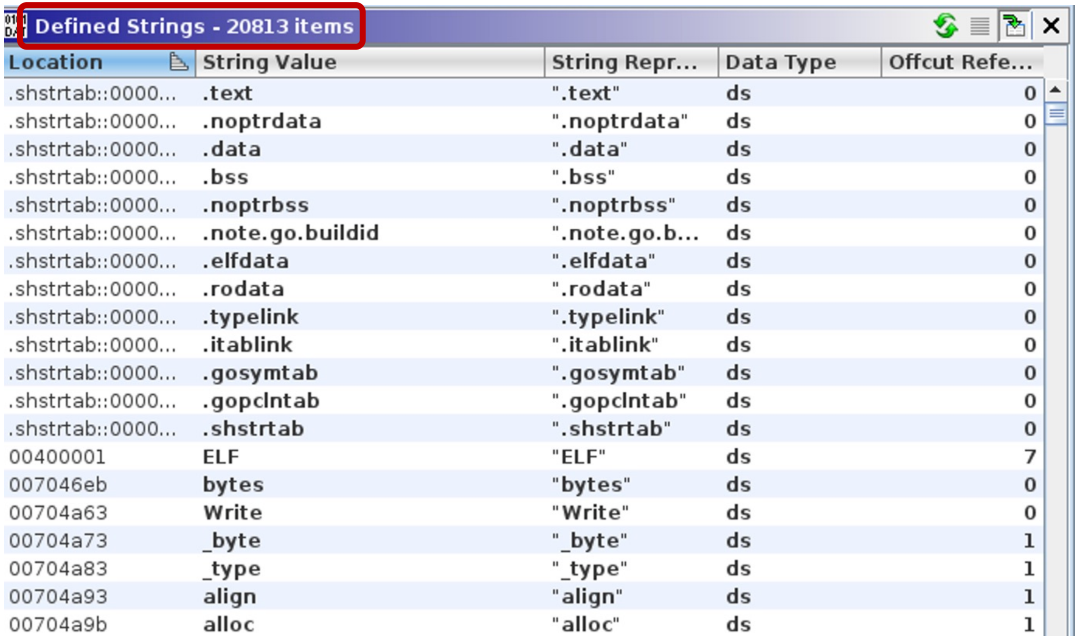
Functions - 6911 items

Name	Location	Fu...	Fu...
shell/exploit.(*le0943).init	006feef0	un...	98
shell/exploit.(*le0943).check	006fef60	un...	520
shell/exploit.(*le0943).run	006ff170	un...	532
shell/exploit.(*le0943).exec	006ff390	un...	1266
shell/exploit.(*p3e874).init	006ff890	un...	109
shell/exploit.(*p3e874).check	006ff900	un...	112
shell/exploit.(*p3e874).run	006ff970	un...	1125
shell/exploit.(*__40ad2).Run.func1	006ffde0	un...	786
shell/exploit.(*Session).Request.func1	00700100	un...	476
shell/exploit.(*bd788f).run.func1	007002e0	un...	223
shell/exploit.(*c41954).run.func1	007003c0	un...	752
shell/exploit.(*__9146c).login.func1	007006b0	un...	245
shell/exploit.init	007007b0	un...	793
shell/exploit.(*e7945e).check	00700ad0	un...	26
type..hash.shell/exploit.__84e6d	00700af0	un...	171
type..eq.shell/exploit.__84e6d	00700ba0	un...	340
type..hash.shell/exploit.__9146c	00700d00	un...	148
type..eq.shell/exploit.__9146c	00700da0	un...	216
type..hash.[21]string	00700e80	un...	110
type..eq.[21]string	00700ef0	un...	165
type..hash.[20]shell/exploit.IExploit	00700fa0	un...	110
type..eq.[20]shell/exploit.IExploit	00701010	un...	165
type..hash.[23]string	007010c0	un...	110
type..eq.[23]string	00701130	un...	165
type..hash.[24][2]string	007011e0	un...	110
type..eq.[24][2]string	00701250	un...	137
type..hash.[4][2]string	007012e0	un...	110
type..eq.[4][2]string	00701350	un...	137
type..hash.[50]string	007013e0	un...	110
type..eq.[50]string	00701450	un...	165
type..hash.[596][2]string	00701500	un...	112
type..eq.[596][2]string	00701570	un...	139
main.init.0	00701600	un...	115
main.main	00701680	un...	48

Strings in Ghidra

Go

- 20813 defined strings in Ghidra
- Hard to spot interesting ones
- Do we see everything?



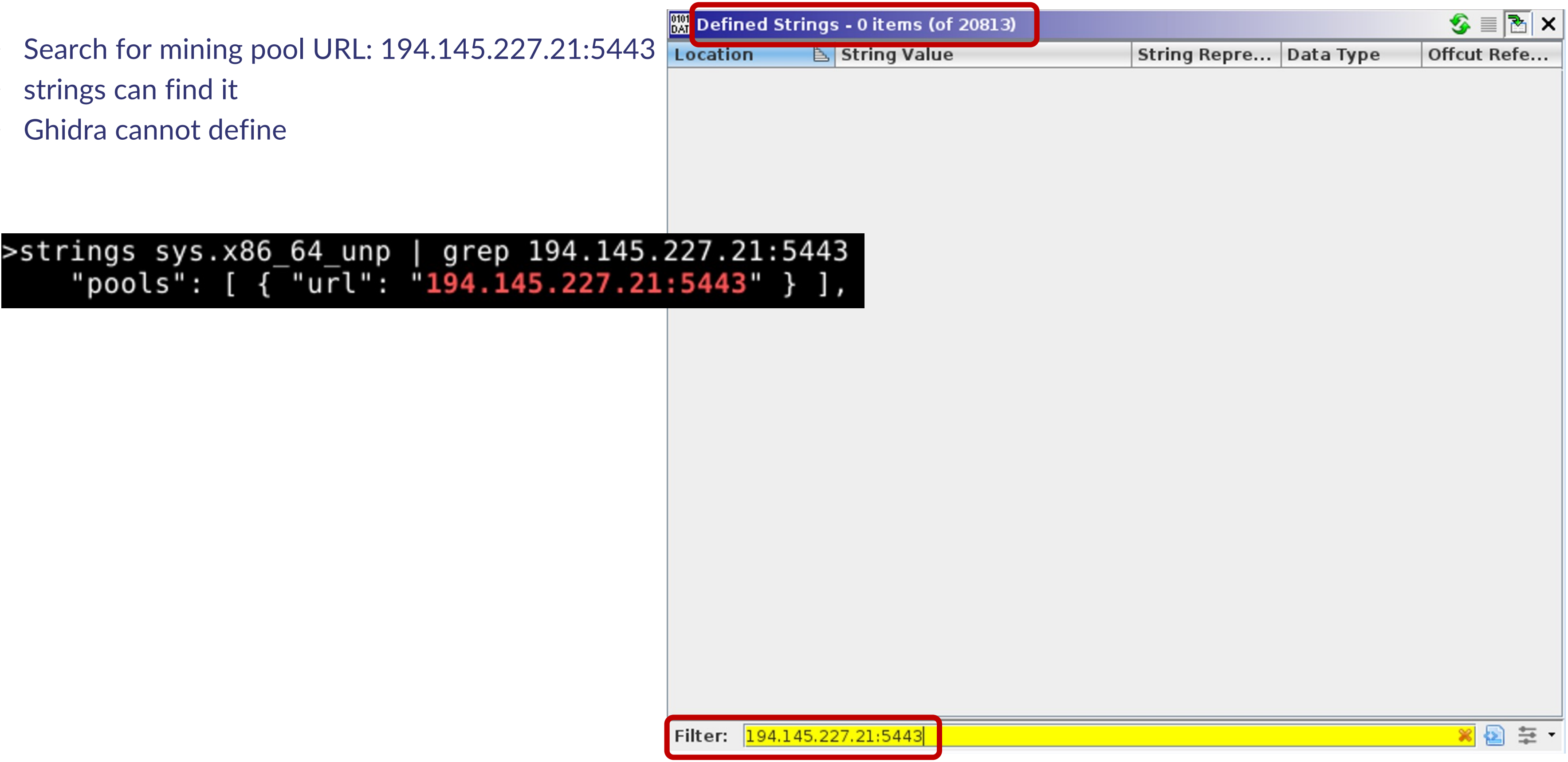
Defined Strings - 20813 items

Location	String Value	String Repr...	Data Type	Offcut Refe...
.shstrtab::0000...	.text	".text"	ds	0
.shstrtab::0000...	.noptrdata	".noptrdata"	ds	0
.shstrtab::0000...	.data	".data"	ds	0
.shstrtab::0000...	.bss	".bss"	ds	0
.shstrtab::0000...	.noptrbss	".noptrbss"	ds	0
.shstrtab::0000...	.note.go.buildid	".note.go.b...	ds	0
.shstrtab::0000...	.elfdata	".elfdata"	ds	0
.shstrtab::0000...	.rodata	".rodata"	ds	0
.shstrtab::0000...	.typelink	".typelink"	ds	0
.shstrtab::0000...	.itablink	".itablink"	ds	0
.shstrtab::0000...	.gosymtab	".gosymtab"	ds	0
.shstrtab::0000...	.gopclntab	".gopclntab"	ds	0
.shstrtab::0000...	.shstrtab	".shstrtab"	ds	0
00400001	ELF	"ELF"	ds	7
007046eb	bytes	"bytes"	ds	0
00704a63	Write	"Write"	ds	0
00704a73	_byte	"_byte"	ds	1
00704a83	_type	"_type"	ds	1
00704a93	align	"align"	ds	1
00704a9b	alloc	"alloc"	ds	1

Strings in Ghidra

Go

- Search for mining pool URL: 194.145.227.21:5443
- strings can find it
- Ghidra cannot define



```
>strings sys.x86_64_unp | grep 194.145.227.21:5443
    "pools": [ { "url": "194.145.227.21:5443" } ],
```

Defined Strings - 0 items (of 20813)

Location	String Value	String Repre...	Data Type	Offcut Refe...

Filter: 194.145.227.21:5443

String Representation

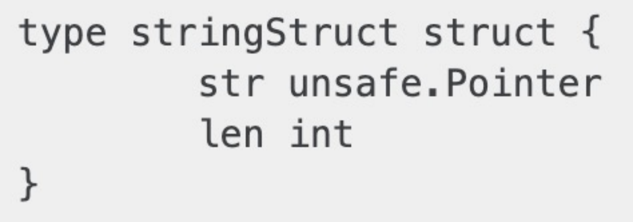
C vs Go

C

- sequence of characters terminated with a null character

Go

- sequence of bytes with a fixed length
- not null terminated
- str – sequence of bytes
- len – number of bytes
- https://golang.org/src/runtime/string.go
- Large string blobs from concatenated strings until null character
- Ghidra has a hard time defining strings in Go binaries



```
type stringStruct struct {
        str unsafe.Pointer
        len int
}
```

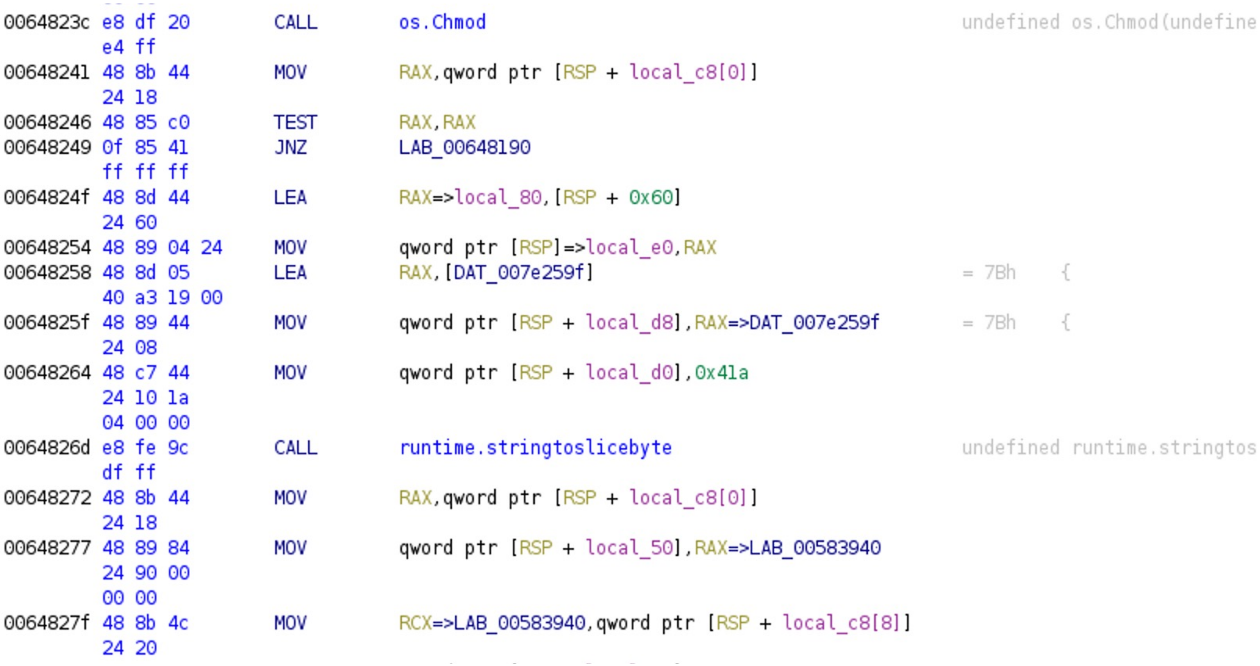
Idea: help Ghidra to find string structures

- Static vs dynamic allocation
- Per architecture (different instruction set)
- Multiple solution within one architecture
- Possible changes per Go version

Dynamically allocated string structure

x86

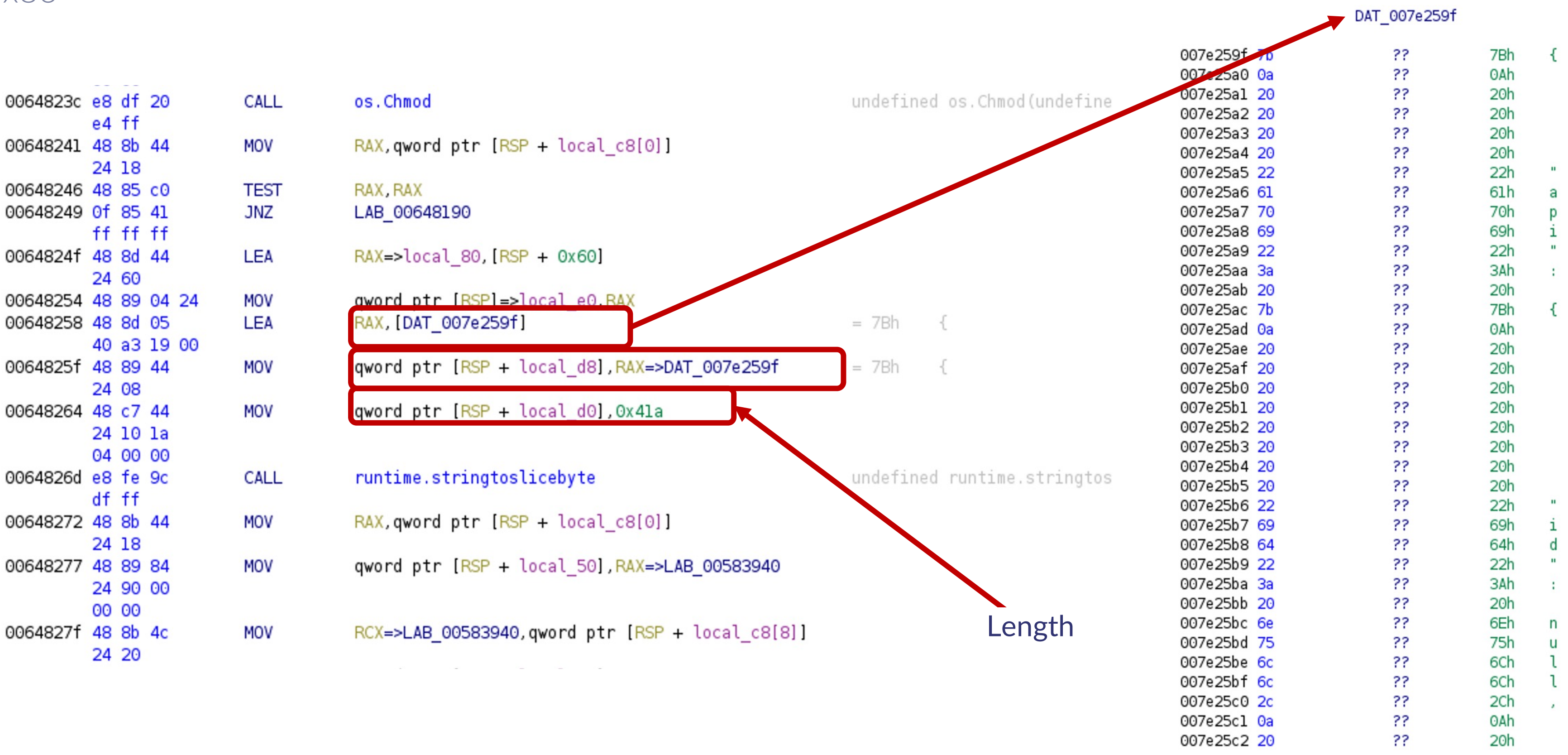
- String structures can be allocated runtime
- Several different scenarios
- Let’s take a look at the shell/miner.xmrRun function



```
0064823c e8 df 20       CALL       os.Chmod                                   undefined os.Chmod(undefine
         e4 ff
00648241 48 8b 44       MOV        RAX,qword ptr [RSP + local_c8[0]]
         24 18
00648246 48 85 c0       TEST       RAX,RAX
00648249 0f 85 41       JNZ        LAB_00648190
         ff ff ff
0064824f 48 8d 44       LEA        RAX=>local_80,[RSP + 0x60]
         24 60
00648254 48 89 04 24    MOV        qword ptr [RSP]=>local_e0,RAX
00648258 48 8d 05       LEA        RAX,[DAT_007e259f]                         = 7Bh    {
         40 a3 19 00
0064825f 48 89 44       MOV        qword ptr [RSP + local_d8],RAX=>DAT_007e259f  = 7Bh    {
         24 08
00648264 48 c7 44       MOV        qword ptr [RSP + local_d0],0x41a
         24 10 1a
         04 00 00
0064826d e8 fe 9c       CALL       runtime.stringtoslicebyte                  undefined runtime.stringtos
         df ff
00648272 48 8b 44       MOV        RAX,qword ptr [RSP + local_c8[0]]
         24 18
00648277 48 89 84       MOV        qword ptr [RSP + local_50],RAX=>LAB_00583940
         24 90 00
         00 00
0064827f 48 8b 4c       MOV        RCX=>LAB_00583940,qword ptr [RSP + local_c8[8]]
         24 20
```

Dynamically allocated string structure

x86



```
0064823c e8 df 20      CALL       os.Chmod                                          undefined os.Chmod(undefine
         e4 ff
00648241 48 8b 44      MOV        RAX,qword ptr [RSP + local_c8[0]]
         24 18
00648246 48 85 c0      TEST       RAX,RAX
00648249 0f 85 41      JNZ        LAB_00648190
         ff ff ff
0064824f 48 8d 44      LEA        RAX=>local_80,[RSP + 0x60]
         24 60
00648254 48 89 04 24   MOV        qword ptr [RSP]=>local_e0,RAX
00648258 48 8d 05      LEA        RAX,[DAT_007e259f]                                = 7Bh    {
         40 a3 19 00
0064825f 48 89 44      MOV        qword ptr [RSP + local_d8],RAX=>DAT_007e259f      = 7Bh    {
         24 08
00648264 48 c7 44      MOV        qword ptr [RSP + local_d0],0x41a
         24 10 1a
         04 00 00
0064826d e8 fe 9c      CALL       runtime.stringtoslicebyte                         undefined runtime.stringtos
         df ff
00648272 48 8b 44      MOV        RAX,qword ptr [RSP + local_c8[0]]
         24 18
00648277 48 89 84      MOV        qword ptr [RSP + local_50],RAX=>LAB_00583940
         24 90 00
         00 00
0064827f 48 8b 4c      MOV        RCX=>LAB_00583940,qword ptr [RSP + local_c8[8]]
         24 20
```

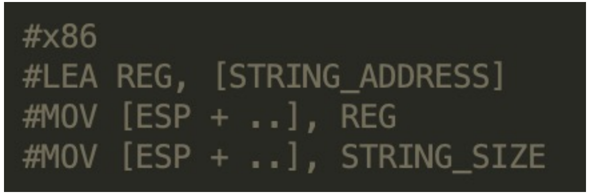
```
                    DAT_007e259f

007e259f 7b         ??       7Bh    {
007e25a0 0a         ??       0Ah
007e25a1 20         ??       20h
007e25a2 20         ??       20h
007e25a3 20         ??       20h
007e25a4 20         ??       20h
007e25a5 22         ??       22h    "
007e25a6 61         ??       61h    a
007e25a7 70         ??       70h    p
007e25a8 69         ??       69h    i
007e25a9 22         ??       22h    "
007e25aa 3a         ??       3Ah    :
007e25ab 20         ??       20h
007e25ac 7b         ??       7Bh    {
007e25ad 0a         ??       0Ah
007e25ae 20         ??       20h
007e25af 20         ??       20h
007e25b0 20         ??       20h
007e25b1 20         ??       20h
007e25b2 20         ??       20h
007e25b3 20         ??       20h
007e25b4 20         ??       20h
007e25b5 20         ??       20h
007e25b6 22         ??       22h    "
007e25b7 69         ??       69h    i
007e25b8 64         ??       64h    d
007e25b9 22         ??       22h    "
007e25ba 3a         ??       3Ah    :
007e25bb 20         ??       20h
007e25bc 6e         ??       6Eh    n
007e25bd 75         ??       75h    u
007e25be 6c         ??       6Ch    l
007e25bf 6c         ??       6Ch    l
007e25c0 2c         ??       2Ch    ,
007e25c1 0a         ??       0Ah
007e25c2 20         ??       20h
```

Length

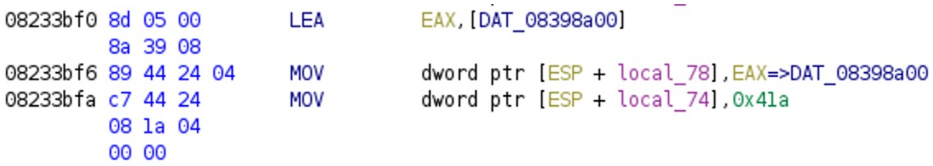
Dynamically allocated string structure

x86

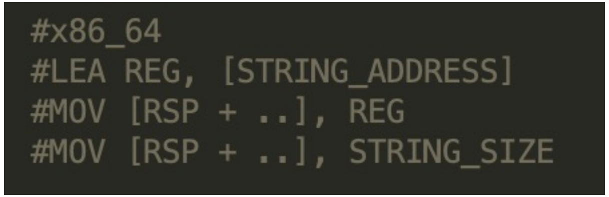
- Search for these instructions and define strings



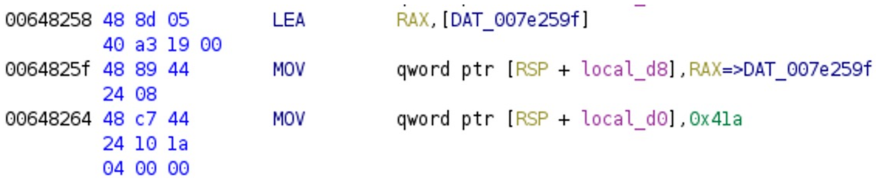
```
#x86
#LEA REG, [STRING_ADDRESS]
#MOV [ESP + ..], REG
#MOV [ESP + ..], STRING_SIZE
```



```
08233bf0 8d 05 00        LEA        EAX,[DAT_08398a00]
         8a 39 08
08233bf6 89 44 24 04     MOV        dword ptr [ESP + local_78],EAX=>DAT_08398a00
08233bfa c7 44 24        MOV        dword ptr [ESP + local_74],0x41a
         08 1a 04
         00 00
```



```
#x86_64
#LEA REG, [STRING_ADDRESS]
#MOV [RSP + ..], REG
#MOV [RSP + ..], STRING_SIZE
```

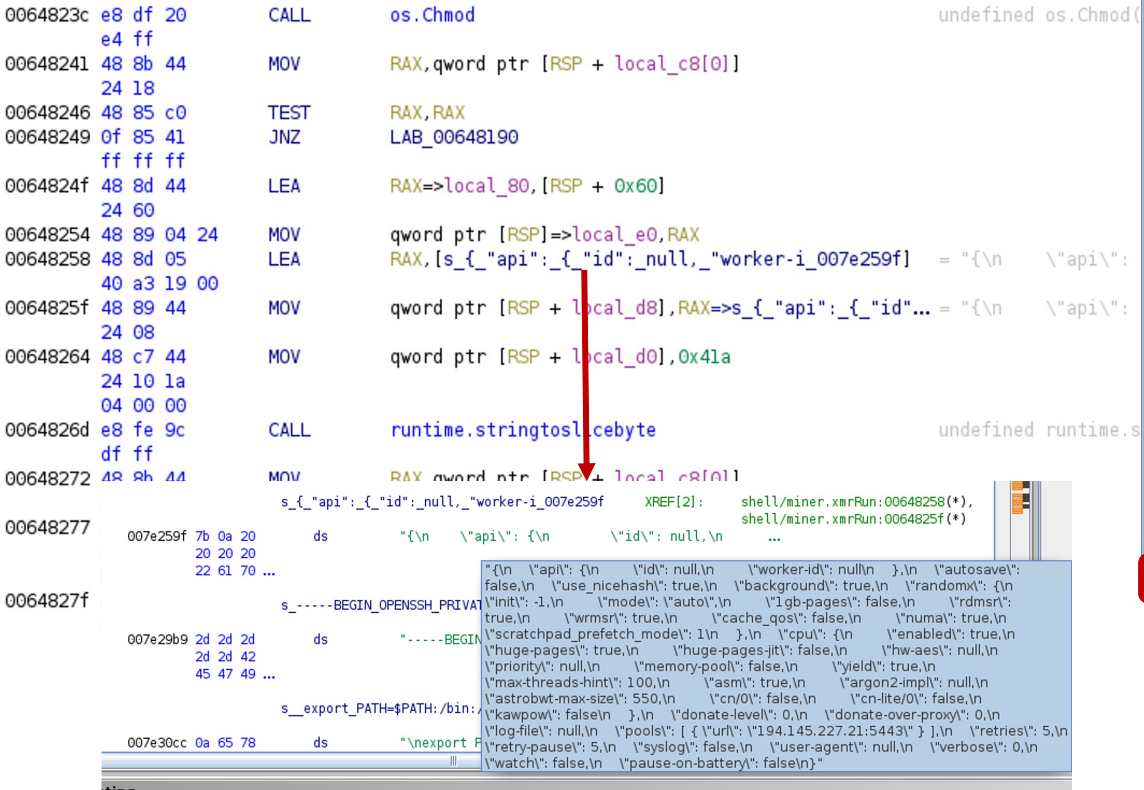


```
00648258 48 8d 05        LEA        RAX,[DAT_007e259f]
         40 a3 19 00
0064825f 48 89 44        MOV        qword ptr [RSP + local_d8],RAX=>DAT_007e259f
         24 08
00648264 48 c7 44        MOV        qword ptr [RSP + local_d0],0x41a
         24 10 1a
         04 00 00
```

Dynamically allocated string structure

x86

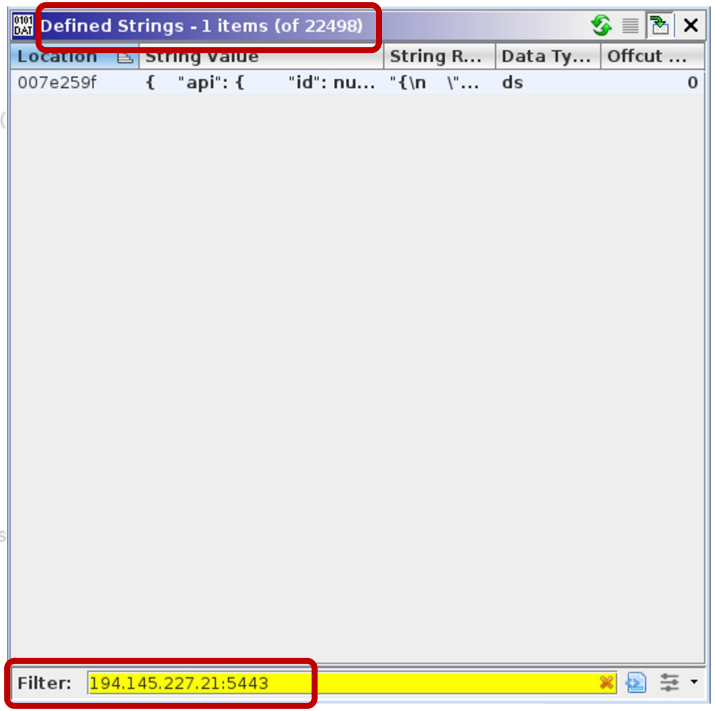
- Results after executing the script



```
0064823c e8 df 20       CALL     os.Chmod                                          undefined os.Chmod(
         e4 ff
00648241 48 8b 44       MOV      RAX,qword ptr [RSP + local_c8[0]]
         24 18
00648246 48 85 c0       TEST     RAX,RAX
00648249 0f 85 41       JNZ      LAB_00648190
         ff ff ff
0064824f 48 8d 44       LEA      RAX=>local_80,[RSP + 0x60]
         24 60
00648254 48 89 04 24    MOV      qword ptr [RSP]=>local_e0,RAX
00648258 48 8d 05       LEA      RAX,[s_{_"api":_{_"id":_null,_"worker-i_007e259f]   = "{\n     \"api\":
         40 a3 19 00
0064825f 48 89 44       MOV      qword ptr [RSP + local_d8],RAX=>s_{_"api":_{_"id"... = "{\n     \"api\":
         24 08
00648264 48 c7 44       MOV      qword ptr [RSP + local_d0],0x41a
         24 10 1a
         04 00 00
0064826d e8 fe 9c       CALL     runtime.stringtoslicebyte                         undefined runtime.s
         df ff
00648272 48 8b 44       MOV      RAX,qword ptr [RSP + local_c8[0]]

00648277

0064827f
```

```
                    s_{_"api":_{_"id":_null,_"worker-i_007e259f      XREF[2]:     shell/miner.xmrRun:00648258(*),
                                                                                  shell/miner.xmrRun:0064825f(*)
007e259f 7b 0a 20       ds          "{\n     \"api\": {\n         \"id\": null,\n       ...
         20 20 20
         22 61 70 ...

                    s_-----BEGIN_OPENSSH_PRIVAT

007e29b9 2d 2d 2d       ds          "-----BEGIN
         2d 2d 42
         45 47 49 ...

                    s__export_PATH=$PATH:/bin:/

007e30cc 0a 65 78       ds          "\nexport P
```

```
"{\n   \"api\": {\n       \"id\": null,\n       \"worker-id\": null\n   },\n   \"autosave\":
false,\n   \"use_nicehash\": true,\n   \"background\": true,\n   \"randomx\": {\n
\"init\": -1,\n       \"mode\": \"auto\",\n       \"1gb-pages\": false,\n       \"rdmsr\":
true,\n       \"wrmsr\": true,\n       \"cache_qos\": false,\n       \"numa\": true,\n
\"scratchpad_prefetch_mode\": 1\n   },\n   \"cpu\": {\n       \"enabled\": true,\n
\"huge-pages\": true,\n       \"huge-pages-jit\": false,\n       \"hw-aes\": null,\n
\"priority\": null,\n       \"memory-pool\": false,\n       \"yield\": true,\n
\"max-threads-hint\": 100,\n       \"asm\": true,\n       \"argon2-impl\": null,\n
\"astrobwt-max-size\": 550,\n       \"cn/0\": false,\n       \"cn-lite/0\": false,\n
\"kawpow\": false\n   },\n   \"donate-level\": 0,\n   \"donate-over-proxy\": 0,\n
\"log-file\": null,\n   \"pools\": [ { \"url\": \"194.145.227.21:5443\" } ],\n   \"retries\": 5,\n
\"retry-pause\": 5,\n   \"syslog\": false,\n   \"user-agent\": null,\n   \"verbose\": 0,\n
\"watch\": false,\n   \"pause-on-battery\": false\n}"
```



Defined Strings - 1 items (of 22498)

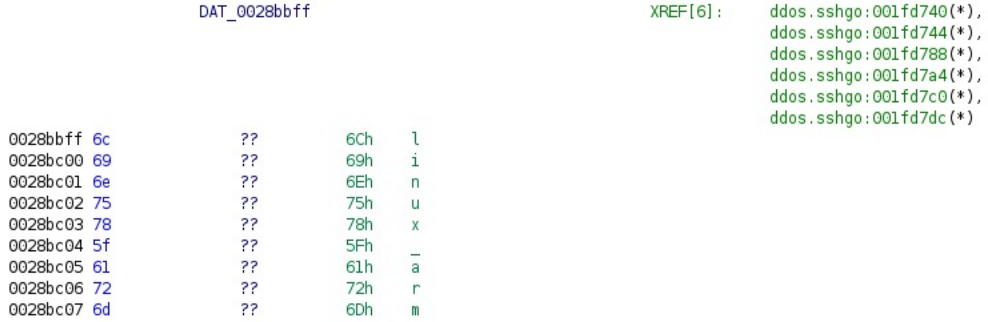
Location	String Value	String R...	Data Ty...	Offcut ...
007e259f	{ "api": { "id": nu...	"{\n \"...	ds	0

Filter: 194.145.227.21:5443

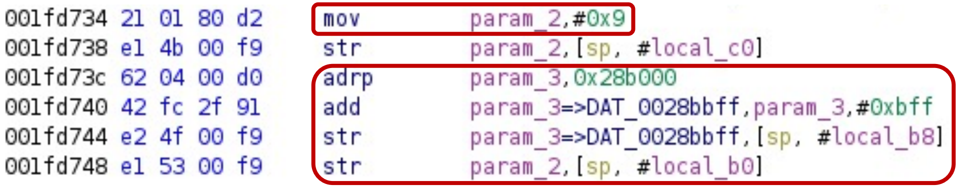
Dynamically allocated string structure

Challenges

- Different instruction sets
- Can be implemented in different ways within the same architecture
- Easy to break intentionally



```
                 DAT_0028bbff                                 XREF[6]:    ddos.sshgo:001fd740(*),
                                                                          ddos.sshgo:001fd744(*),
                                                                          ddos.sshgo:001fd788(*),
                                                                          ddos.sshgo:001fd7a4(*),
                                                                          ddos.sshgo:001fd7c0(*),
                                                                          ddos.sshgo:001fd7dc(*)
0028bbff 6c          ??         6Ch    l
0028bc00 69          ??         69h    i
0028bc01 6e          ??         6Eh    n
0028bc02 75          ??         75h    u
0028bc03 78          ??         78h    x
0028bc04 5f          ??         5Fh    _
0028bc05 61          ??         61h    a
0028bc06 72          ??         72h    r
0028bc07 6d          ??         6Dh    m
```



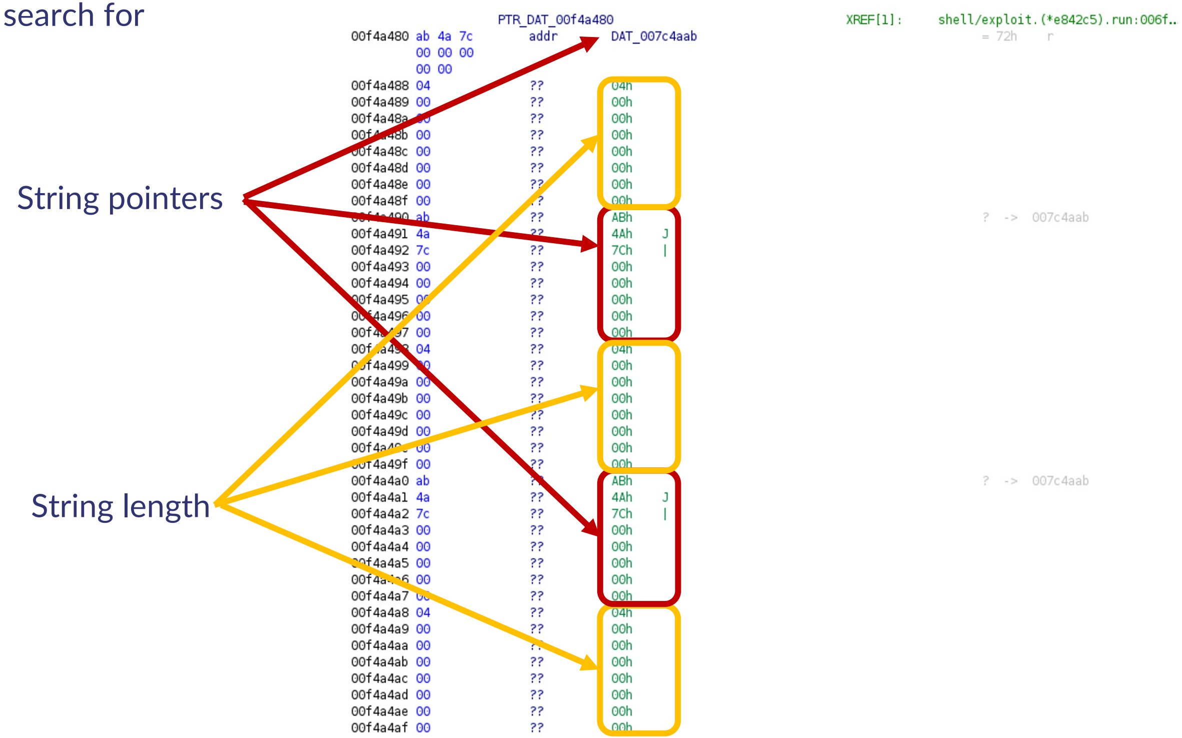
```
001fd734 21 01 80 d2    mov        param_2,#0x9
001fd738 e1 4b 00 f9    str        param_2,[sp, #local_c0]
001fd73c 62 04 00 d0    adrp       param_3,0x28b000
001fd740 42 fc 2f 91    add        param_3=>DAT_0028bbff,param_3,#0xbff
001fd744 e2 4f 00 f9    str        param_3=>DAT_0028bbff,[sp, #local_b8]
001fd748 e1 53 00 f9    str        param_2,[sp, #local_b0]
```

Binary: Kaiji – ARM

Statically allocated string structure

Idea

- Look for pointer to string followed by possible length value
- To eliminate FPs limit string length and search for printable characters only
- Check only in data sections
- Not architecture specific



String pointers

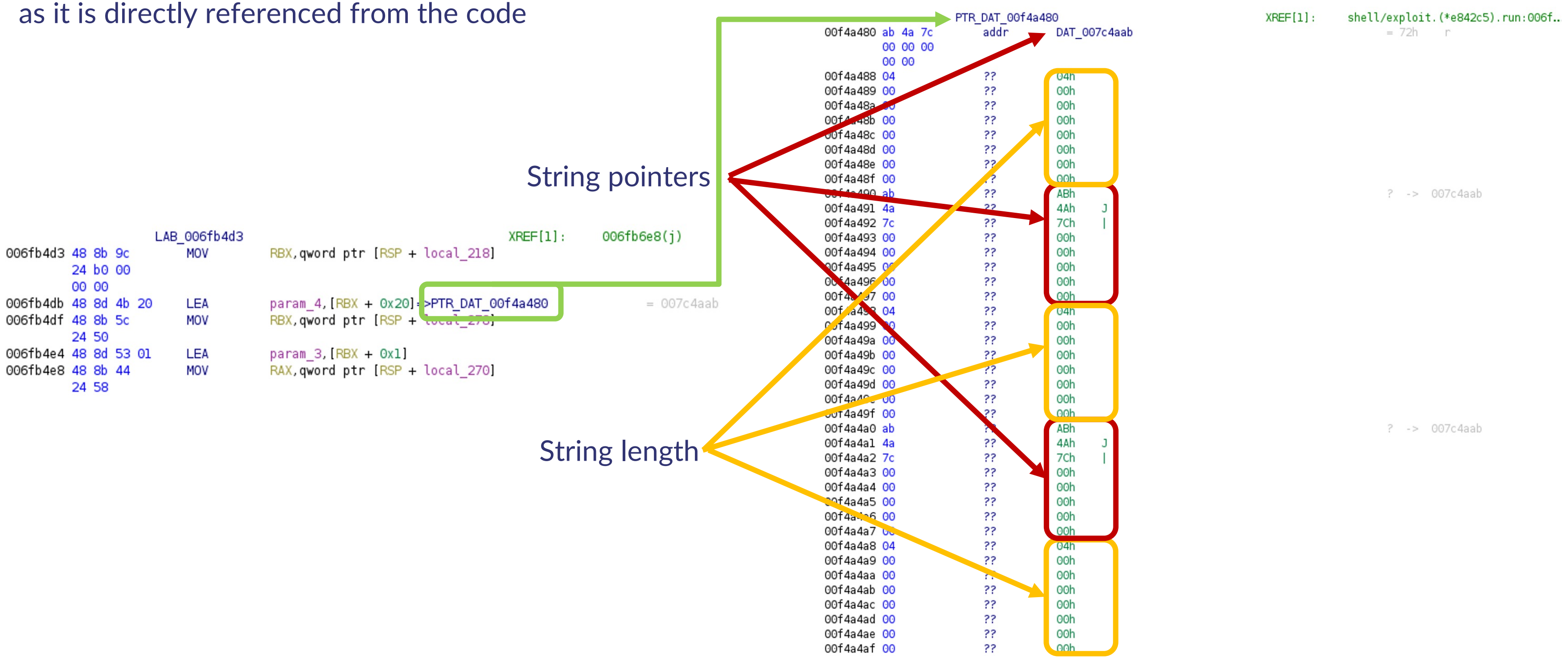
String length

```
                        PTR_DAT_00f4a480                           XREF[1]:     shell/exploit.(*e842c5).run:006f..
00f4a480 ab 4a 7c          addr       DAT_007c4aab                                 = 72h     r
         00 00 00
         00 00
00f4a488 04                ??         04h
00f4a489 00                ??         00h
00f4a48a 00                ??         00h
00f4a48b 00                ??         00h
00f4a48c 00                ??         00h
00f4a48d 00                ??         00h
00f4a48e 00                ??         00h
00f4a48f 00                ??         00h
00f4a490 ab                ??         ABh                                          ?  ->  007c4aab
00f4a491 4a                ??         4Ah     J
00f4a492 7c                ??         7Ch     |
00f4a493 00                ??         00h
00f4a494 00                ??         00h
00f4a495 00                ??         00h
00f4a496 00                ??         00h
00f4a497 00                ??         00h
00f4a498 04                ??         04h
00f4a499 00                ??         00h
00f4a49a 00                ??         00h
00f4a49b 00                ??         00h
00f4a49c 00                ??         00h
00f4a49d 00                ??         00h
00f4a49e 00                ??         00h
00f4a49f 00                ??         00h
00f4a4a0 ab                ??         ABh                                          ?  ->  007c4aab
00f4a4a1 4a                ??         4Ah     J
00f4a4a2 7c                ??         7Ch     |
00f4a4a3 00                ??         00h
00f4a4a4 00                ??         00h
00f4a4a5 00                ??         00h
00f4a4a6 00                ??         00h
00f4a4a7 00                ??         00h
00f4a4a8 04                ??         04h
00f4a4a9 00                ??         00h
00f4a4aa 00                ??         00h
00f4a4ab 00                ??         00h
00f4a4ac 00                ??         00h
00f4a4ad 00                ??         00h
00f4a4ae 00                ??         00h
00f4a4af 00                ??         00h
```

Statically allocated string structure

Idea

- One pointer was successfully identified as it is directly referenced from the code



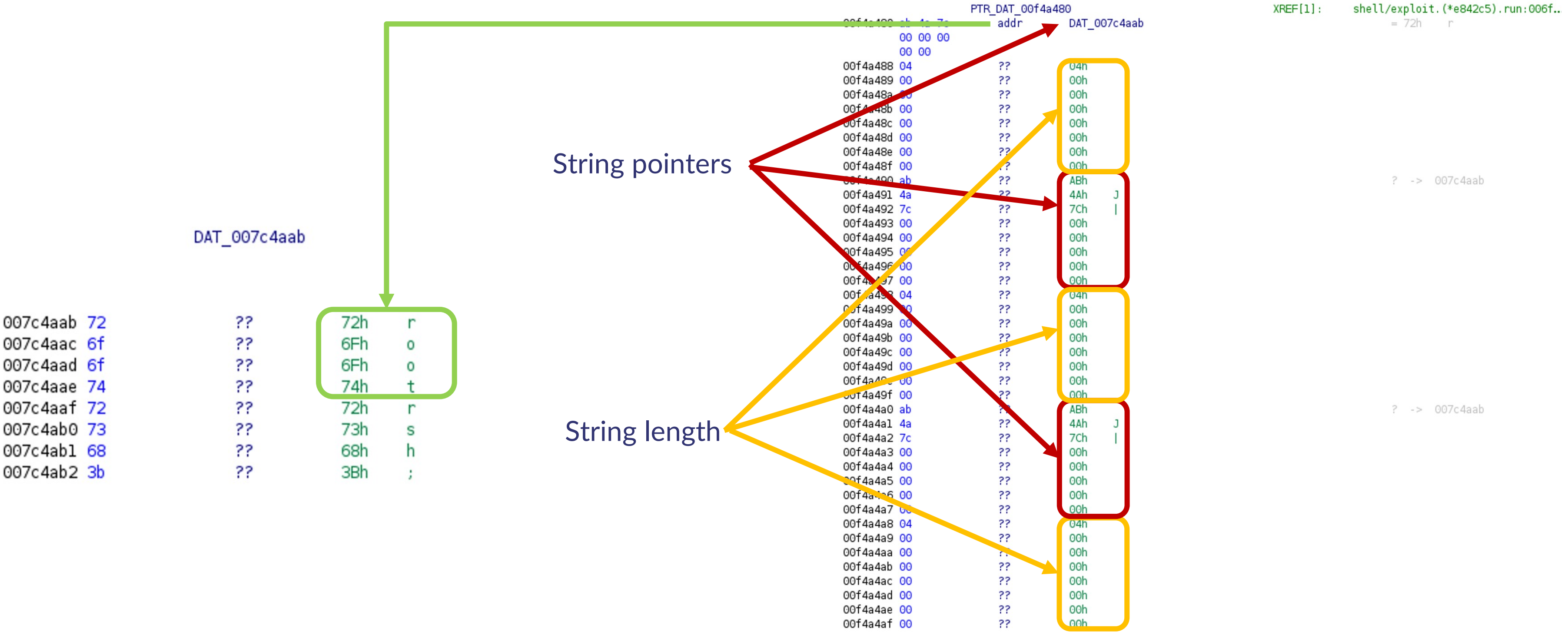
```
                      LAB_006fb4d3                          XREF[1]:     006fb6e8(j)
006fb4d3 48 8b 9c        MOV        RBX,qword ptr [RSP + local_218]
         24 b0 00
         00 00
006fb4db 48 8d 4b 20     LEA        param_4,[RBX + 0x20]=>PTR_DAT_00f4a480              = 007c4aab
006fb4df 48 8b 5c        MOV        RBX,qword ptr [RSP + local_278]
         24 50
006fb4e4 48 8d 53 01     LEA        param_3,[RBX + 0x1]
006fb4e8 48 8b 44        MOV        RAX,qword ptr [RSP + local_270]
         24 58
```

```
                     PTR_DAT_00f4a480                         XREF[1]:     shell/exploit.(*e842c5).run:006f..
00f4a480 ab 4a 7c        addr       DAT_007c4aab                                     = 72h    r
         00 00 00
         00 00
00f4a488 04              ??         04h
00f4a489 00              ??         00h
00f4a48a 00              ??         00h
00f4a48b 00              ??         00h
00f4a48c 00              ??         00h
00f4a48d 00              ??         00h
00f4a48e 00              ??         00h
00f4a48f 00              ??         00h
00f4a490 ab              ??         ABh                                              ?  ->  007c4aab
00f4a491 4a              ??         4Ah     J
00f4a492 7c              ??         7Ch     |
00f4a493 00              ??         00h
00f4a494 00              ??         00h
00f4a495 00              ??         00h
00f4a496 00              ??         00h
00f4a497 00              ??         00h
00f4a498 04              ??         04h
00f4a499 00              ??         00h
00f4a49a 00              ??         00h
00f4a49b 00              ??         00h
00f4a49c 00              ??         00h
00f4a49d 00              ??         00h
00f4a49e 00              ??         00h
00f4a49f 00              ??         00h
00f4a4a0 ab              ??         ABh                                              ?  ->  007c4aab
00f4a4a1 4a              ??         4Ah     J
00f4a4a2 7c              ??         7Ch     |
00f4a4a3 00              ??         00h
00f4a4a4 00              ??         00h
00f4a4a5 00              ??         00h
00f4a4a6 00              ??         00h
00f4a4a7 00              ??         00h
00f4a4a8 04              ??         04h
00f4a4a9 00              ??         00h
00f4a4aa 00              ??         00h
00f4a4ab 00              ??         00h
00f4a4ac 00              ??         00h
00f4a4ad 00              ??         00h
00f4a4ae 00              ??         00h
00f4a4af 00              ??         00h
```

Statically allocated string structure

Example – before executing the script

- Strings are not defined



String pointers

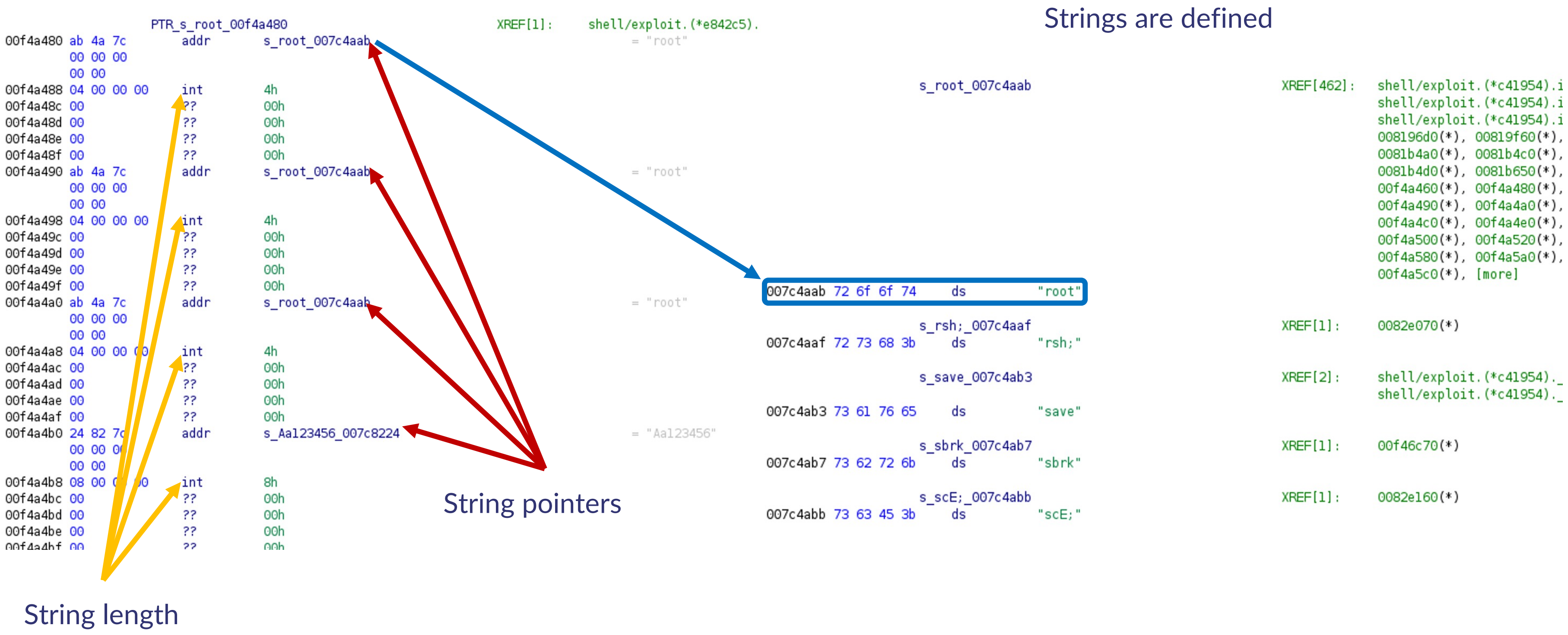
String length

```
                        DAT_007c4aab

007c4aab 72         ??        72h    r
007c4aac 6f         ??        6Fh    o
007c4aad 6f         ??        6Fh    o
007c4aae 74         ??        74h    t
007c4aaf 72         ??        72h    r
007c4ab0 73         ??        73h    s
007c4ab1 68         ??        68h    h
007c4ab2 3b         ??        3Bh    ;
```

```
                  PTR_DAT_00f4a480                   XREF[1]:     shell/exploit.(*e842c5).run:006f..
00f4a480 ab 4a 7c     addr      DAT_007c4aab                                = 72h     r
         00 00 00
         00 00
00f4a488 04           ??        04h
00f4a489 00           ??        00h
00f4a48a 00           ??        00h
00f4a48b 00           ??        00h
00f4a48c 00           ??        00h
00f4a48d 00           ??        00h
00f4a48e 00           ??        00h
00f4a48f 00           ??        00h
00f4a490 ab           ??        ABh                                         ?  ->  007c4aab
00f4a491 4a           ??        4Ah      J
00f4a492 7c           ??        7Ch      |
00f4a493 00           ??        00h
00f4a494 00           ??        00h
00f4a495 00           ??        00h
00f4a496 00           ??        00h
00f4a497 00           ??        00h
00f4a498 04           ??        04h
00f4a499 00           ??        00h
00f4a49a 00           ??        00h
00f4a49b 00           ??        00h
00f4a49c 00           ??        00h
00f4a49d 00           ??        00h
00f4a49e 00           ??        00h
00f4a49f 00           ??        00h
00f4a4a0 ab           ??        ABh                                         ?  ->  007c4aab
00f4a4a1 4a           ??        4Ah      J
00f4a4a2 7c           ??        7Ch      |
00f4a4a3 00           ??        00h
00f4a4a4 00           ??        00h
00f4a4a5 00           ??        00h
00f4a4a6 00           ??        00h
00f4a4a7 00           ??        00h
00f4a4a8 04           ??        04h
00f4a4a9 00           ??        00h
00f4a4aa 00           ??        00h
00f4a4ab 00           ??        00h
00f4a4ac 00           ??        00h
00f4a4ad 00           ??        00h
00f4a4ae 00           ??        00h
00f4a4af 00           ??        00h
```

Statically allocated string structure

Example – after executing the script



Strings are defined

String pointers

String length

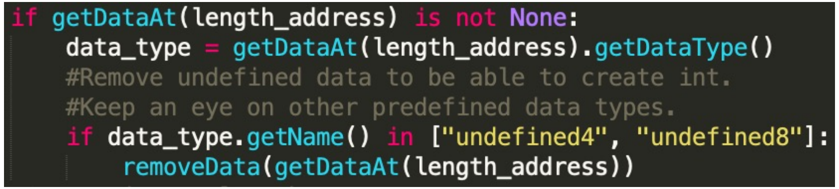
```
                        PTR_s_root_00f4a480                       XREF[1]:     shell/exploit.(*e842c5).
00f4a480 ab 4a 7c          addr         s_root_007c4aab                         = "root"
         00 00 00
         00 00
00f4a488 04 00 00 00       int          4h
00f4a48c 00                ??           00h
00f4a48d 00                ??           00h
00f4a48e 00                ??           00h
00f4a48f 00                ??           00h
00f4a490 ab 4a 7c          addr         s_root_007c4aab                         = "root"
         00 00 00
         00 00
00f4a498 04 00 00 00       int          4h
00f4a49c 00                ??           00h
00f4a49d 00                ??           00h
00f4a49e 00                ??           00h
00f4a49f 00                ??           00h
00f4a4a0 ab 4a 7c          addr         s_root_007c4aab                         = "root"
         00 00 00
         00 00
00f4a4a8 04 00 00 00       int          4h
00f4a4ac 00                ??           00h
00f4a4ad 00                ??           00h
00f4a4ae 00                ??           00h
00f4a4af 00                ??           00h
00f4a4b0 24 82 7c          addr         s_Aa123456_007c8224                     = "Aa123456"
         00 00 00
         00 00
00f4a4b8 08 00 00 00       int          8h
00f4a4bc 00                ??           00h
00f4a4bd 00                ??           00h
00f4a4be 00                ??           00h
00f4a4bf 00                ??           00h
```

```
                  s_root_007c4aab                         XREF[462]:   shell/exploit.(*c41954).i
                                                                       shell/exploit.(*c41954).i
                                                                       shell/exploit.(*c41954).i
                                                                       008196d0(*), 00819f60(*),
                                                                       0081b4a0(*), 0081b4c0(*),
                                                                       0081b4d0(*), 0081b650(*),
                                                                       00f4a460(*), 00f4a480(*),
                                                                       00f4a490(*), 00f4a4a0(*),
                                                                       00f4a4c0(*), 00f4a4e0(*),
                                                                       00f4a500(*), 00f4a520(*),
                                                                       00f4a580(*), 00f4a5a0(*),
                                                                       00f4a5c0(*), [more]
007c4aab 72 6f 6f 74     ds          "root"
                  s_rsh;_007c4aaf                         XREF[1]:     0082e070(*)
007c4aaf 72 73 68 3b     ds          "rsh;"
                  s_save_007c4ab3                         XREF[2]:     shell/exploit.(*c41954)._
                                                                       shell/exploit.(*c41954)._
007c4ab3 73 61 76 65     ds          "save"
                  s_sbrk_007c4ab7                         XREF[1]:     00f46c70(*)
007c4ab7 73 62 72 6b     ds          "sbrk"
                  s_scE;_007c4abb                         XREF[1]:     0082e160(*)
007c4abb 73 63 45 3b     ds          "scE;"
```

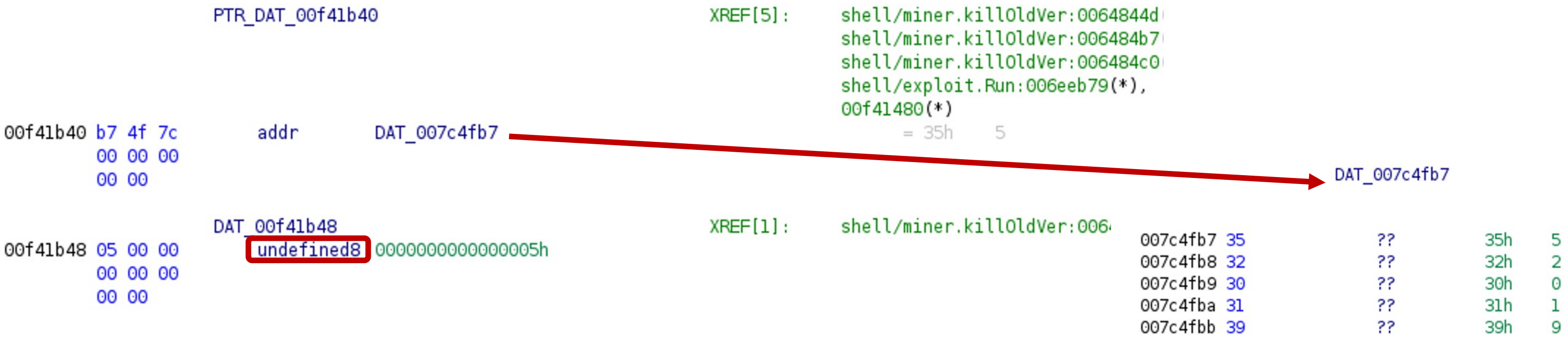
String recovery challenges

Falsely defined data types by Ghidra

- undefined4 or undefined8 (depends on pointer size)
- Already defined data types cannot be redefined
  (undifined4 and undifined8 are defined data types)
- First the data type has to be removed
- Then the new data type can be defined



```
if getDataAt(length_address) is not None:
    data_type = getDataAt(length_address).getDataType()
    #Remove undefined data to be able to create int.
    #Keep an eye on other predefined data types.
    if data_type.getName() in ["undefined4", "undefined8"]:
        removeData(getDataAt(length_address))
```



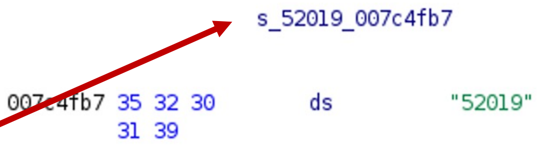
```
                 PTR_DAT_00f41b40                 XREF[5]:     shell/miner.killOldVer:0064844d
                                                               shell/miner.killOldVer:006484b7
                                                               shell/miner.killOldVer:006484c0
                                                               shell/exploit.Run:006eeb79(*),
                                                               00f41480(*)
00f41b40 b7 4f 7c       addr       DAT_007c4fb7                    = 35h     5
         00 00 00
         00 00
                 DAT_00f41b48                     XREF[1]:     shell/miner.killOldVer:006
00f41b48 05 00 00       undefined8 0000000000000005h
         00 00 00
         00 00
```

```
DAT_007c4fb7
007c4fb7 35         ??        35h    5
007c4fb8 32         ??        32h    2
007c4fb9 30         ??        30h    0
007c4fba 31         ??        31h    1
007c4fbb 39         ??        39h    9
```

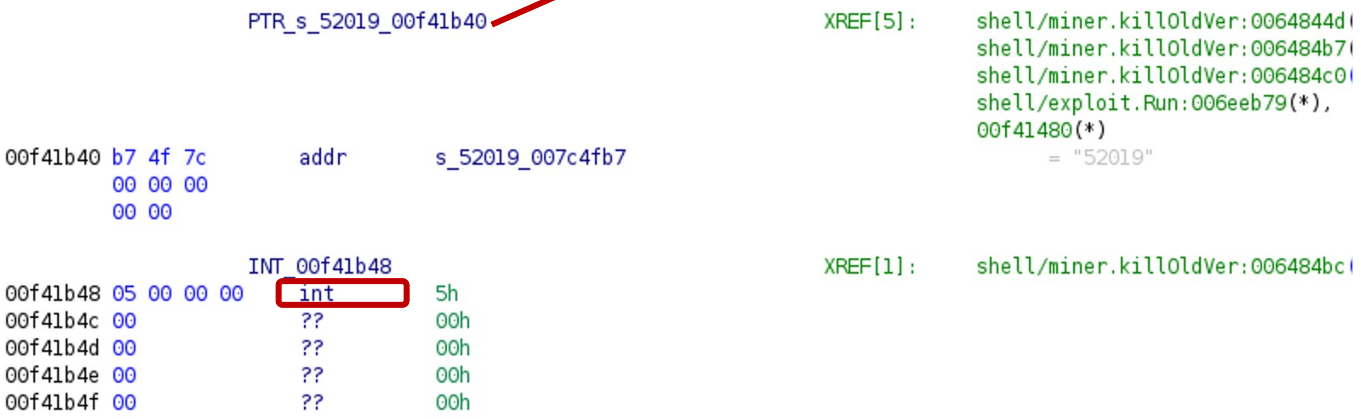
String recovery challenges

Falsely defined data types by Ghidra

- undefined4 or undefined8 (depends on pointer size)
- Already defined data types cannot be redefined
  (undifined4 and undifined8 are defined data types)
- First the data type has to be removed
- Then the new data type can be defined



```
                         s_52019_007c4fb7

007c4fb7 35 32 30        ds            "52019"
         31 39
```

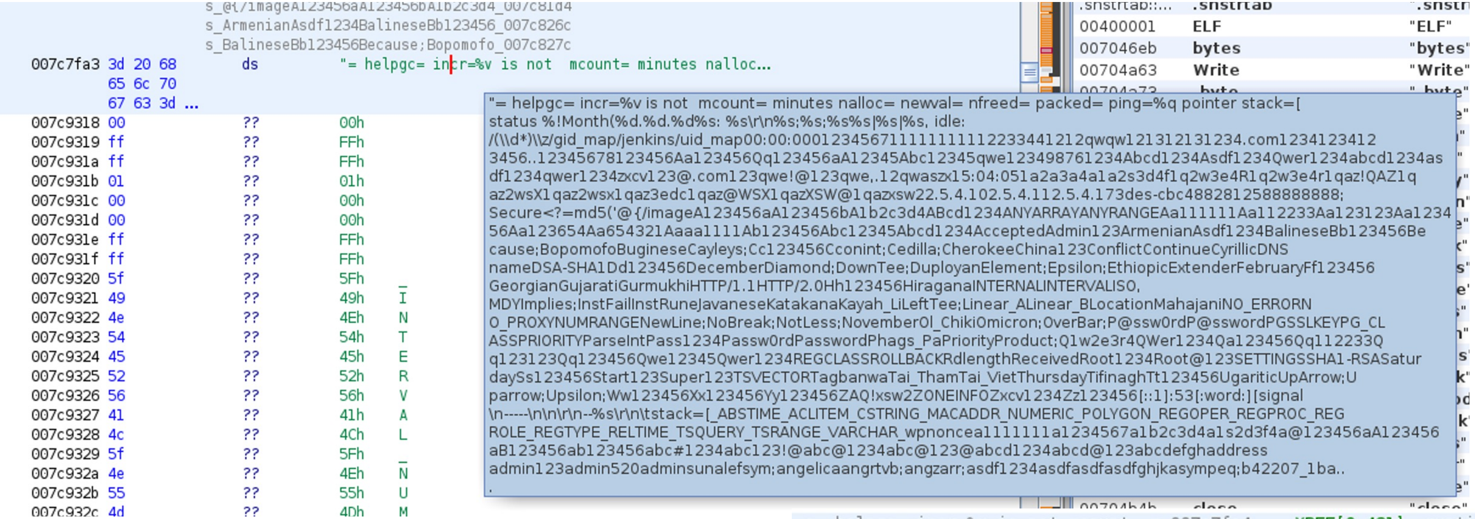


```
                 PTR_s_52019_00f41b40                            XREF[5]:     shell/miner.killOldVer:0064844d(
                                                                              shell/miner.killOldVer:006484b7(
                                                                              shell/miner.killOldVer:006484c0(
                                                                              shell/exploit.Run:006eeb79(*),
                                                                              00f41480(*)
00f41b40 b7 4f 7c        addr       s_52019_007c4fb7                              = "52019"
         00 00 00
         00 00

                 INT_00f41b48                                    XREF[1]:     shell/miner.killOldVer:006484bc(
00f41b48 05 00 00 00     int        5h
00f41b4c 00              ??         00h
00f41b4d 00              ??         00h
00f41b4e 00              ??         00h
00f41b4f 00              ??         00h
```

String recovery challenges

Falsely defined data types by Ghidra

- A large string blob (containing multiple strings) defined as one string

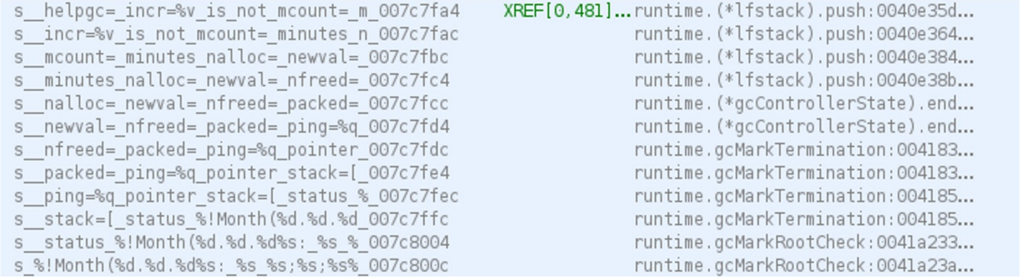


```
                                s_@{/imageA123456aA123456bA1b2c3d4_007c81d4
                                s_ArmenianAsdf1234BalineseBb123456_007c826c
                                s_BalineseBb123456Because;Bopomofo_007c827c
007c7fa3 3d 20 68        ds          "= helpgc= incr=%v is not  mcount= minutes nalloc…
         65 6c 70
         67 63 3d …
007c9318 00              ??          00h
007c9319 ff              ??          FFh
007c931a ff              ??          FFh
007c931b 01              ??          01h
007c931c 00              ??          00h
007c931d 00              ??          00h
007c931e ff              ??          FFh
007c931f ff              ??          FFh
007c9320 5f              ??          5Fh    _
007c9321 49              ??          49h    I
007c9322 4e              ??          4Eh    N
007c9323 54              ??          54h    T
007c9324 45              ??          45h    E
007c9325 52              ??          52h    R
007c9326 56              ??          56h    V
007c9327 41              ??          41h    A
007c9328 4c              ??          4Ch    L
007c9329 5f              ??          5Fh    _
007c932a 4e              ??          4Eh    N
007c932b 55              ??          55h    U
007c932c 4d              ??          4Dh    M
```

```
"= helpgc= incr=%v is not  mcount= minutes nalloc= newval= nfreed= packed= ping=%q pointer stack=[
status %!Month(%d.%d.%d%s: %s\r\n%s;%s;%s%s|%s|%s, idle:
/(\\d*)\\z/gid_map/jenkins/uid_map00:00:0001234567111111111223344121 2qwqw121312131234.com123412341 2
3456..12345678123456Aa123456Qq123456aA12345Abc12345qwe123498761234Abcd1234Asdf1234Qwer1234abcd1234as
df1234qwer1234zxcv123@.com123qwe!@123qwe,.12qwaszx15:04:051a2a3a4a1a2s3d4f1q2w3e4R1q2w3e4r1qaz!QAZ1q
az2wsX1qaz2wsx1qaz3edc1qaz@WSX1qazXSW@1qazxsw22.5.4.102.5.4.112.5.4.173des-cbc4882812588888888;
Secure<?=md5('@{/imageA123456aA123456bA1b2c3d4ABcd1234ANYARRAYANYRANGEAa111111Aa112233Aa123123Aa1234
56Aa123654Aa654321Aaaa1111Ab123456Abc12345Abcd1234AcceptedAdmin123ArmenianAsdf1234BalineseBb123456Be
cause;BopomofoBugineseCayleys;Cc123456Cconint;Cedilla;CherokeeChina123ConflictContinueCyrillicDNS
nameDSA-SHA1Dd123456DecemberDiamond;DownTee;DuployanElement;Epsilon;EthiopicExtenderFebruaryFf123456
GeorgianGujaratiGurmukhiHTTP/1.1HTTP/2.0Hh123456HiraganaINTERNALINTERVALISO,
MDYImplies;InstFailInstRuneJavaneseKatakanaKayah_LiLeftTee;Linear_ALinear_BLocationMahajaniNO_ERRORN
O_PROXYNUMRANGENewLine;NoBreak;NotLess;NovemberOl_ChikiOmicron;OverBar;P@ssw0rdP@sswordPGSSLKEYPG_CL
ASSPRIORITYParseIntPass1234Passw0rdPasswordPhags_PaPriorityProduct;Q1w2e3r4QWer1234Qa123456Qq112233Q
q123123Qq123456Qwe12345Qwer1234REGCLASSROLLBACKRdlengthReceivedRoot1234Root@123SETTINGSSHA1-RSASatur
daySs123456Start123Super123TSVECTORTagbanwaTai_ThamTai_VietThursdayTifinaghTt123456UgariticUpArrow;U
parrow;Upsilon;Ww123456Xx123456Yy123456ZAQ!xsw2ZONEINFOZxcv1234Zz123456[::1]:53[:word:][signal
\n-----\n\n\r\n--%s\r\n\tstack=[_ABSTIME_ACLITEM_CSTRING_MACADDR_NUMERIC_POLYGON_REGOPER_REGPROC_REG
ROLE_REGTYPE_RELTIME_TSQUERY_TSRANGE_VARCHAR_wpnoncea111111a1234567a1b2c3d4a1s2d3f4a@123456aA123456
aB123456ab123456abc#1234abc123!@abc@1234abc@123@abcd1234abcd@123abcdefghaddress
admin123admin520adminsunalefsym;angelicaangrtvb;angzarr;asdf1234asdfasdfasdfghjkasympeq;b42207_1ba..
.
```

```
.shstrtab:...   .shstrtab   ".shstr
00400001        ELF         "ELF"
007046eb        bytes       "bytes"
00704a63        Write       "Write"
00704a73        byte        "_byte"
00704b4b        close       "close"
```

Offcut references

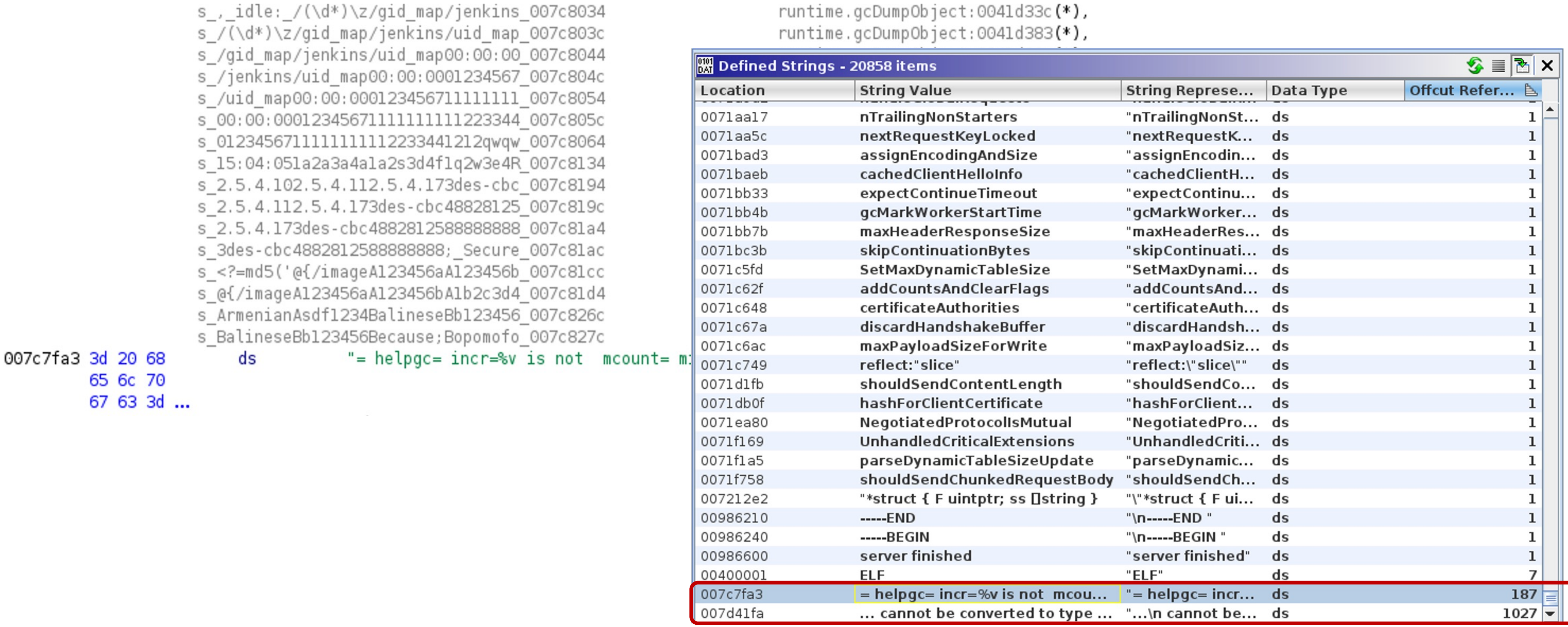


```
s__helpgc=_incr=%v_is_not_mcount=_m_007c7fa4       XREF[0,481]...runtime.(*lfstack).push:0040e35d...
s__incr=%v_is_not_mcount=_minutes_n_007c7fac                    runtime.(*lfstack).push:0040e364...
s__mcount=_minutes_nalloc=_newval=_007c7fbc                     runtime.(*lfstack).push:0040e384...
s__minutes_nalloc=_newval=_nfreed=_007c7fc4                     runtime.(*lfstack).push:0040e38b...
s__nalloc=_newval=_nfreed=_packed=_007c7fcc                     runtime.(*gcControllerState).end...
s__newval=_nfreed=_packed=_ping=%q_007c7fd4                     runtime.(*gcControllerState).end...
s__nfreed=_packed=_ping=%q_pointer_007c7fdc                     runtime.gcMarkTermination:004183...
s__packed=_ping=%q_pointer_stack=[_007c7fe4                     runtime.gcMarkTermination:004183...
s__ping=%q_pointer_stack=[_status_%_007c7fec                    runtime.gcMarkTermination:004185...
s__stack=[_status_%!Month(%d.%d.%d_007c7ffc                     runtime.gcMarkTermination:004185...
s__status_%!Month(%d.%d.%d%s:_%s_%_007c8004                     runtime.gcMarkRootCheck:0041a233...
s_%!Month(%d.%d.%d%s:_%s_%s;%s;%s%_007c800c                     runtime.gcMarkRootCheck:0041a23a...
```

String recovery challenges

Falsely defined data types by Ghidra

- A large string blob (containing multiple strings) defined as one string



```
                    s_,_idle:_/(\d*)\z/gid_map/jenkins_007c8034
                    s_/(\d*)\z/gid_map/jenkins/uid_map_007c803c
                    s_/gid_map/jenkins/uid_map00:00:00_007c8044
                    s_/jenkins/uid_map00:00:0001234567_007c804c
                    s_/uid_map00:00:000123456711111111_007c8054
                    s_00:00:0001234567111111111223344_007c805c
                    s_01234567111111111223344l212qwqw_007c8064
                    s_15:04:05la2a3a4ala2s3d4flq2w3e4R_007c8134
                    s_2.5.4.102.5.4.112.5.4.173des-cbc_007c8194
                    s_2.5.4.112.5.4.173des-cbc48828125_007c819c
                    s_2.5.4.173des-cbc488281258888888_007c81a4
                    s_3des-cbc4882812588888888;_Secure_007c81ac
                    s_<?=md5('@{/imageA123456aA123456b_007c81cc
                    s_@{/imageA123456aA123456bA1b2c3d4_007c81d4
                    s_ArmenianAsdf1234BalineseBb123456_007c826c
                    s_BalineseBb123456Because;Bopomofo_007c827c
007c7fa3 3d 20 68         ds         "= helpgc= incr=%v is not  mcount= mi
         65 6c 70
         67 63 3d ...
```

```
runtime.gcDumpObject:0041d33c(*),
runtime.gcDumpObject:0041d383(*),
```

Defined Strings - 20858 items

Location	String Value	String Represe...	Data Type	Offcut Refer...
0071aa17	nTrailingNonStarters	"nTrailingNonSt...	ds	1
0071aa5c	nextRequestKeyLocked	"nextRequestK...	ds	1
0071bad3	assignEncodingAndSize	"assignEncodin...	ds	1
0071baeb	cachedClientHelloInfo	"cachedClientH...	ds	1
0071bb33	expectContinueTimeout	"expectContinu...	ds	1
0071bb4b	gcMarkWorkerStartTime	"gcMarkWorker...	ds	1
0071bb7b	maxHeaderResponseSize	"maxHeaderRes...	ds	1
0071bc3b	skipContinuationBytes	"skipContinuati...	ds	1
0071c5fd	SetMaxDynamicTableSize	"SetMaxDynami...	ds	1
0071c62f	addCountsAndClearFlags	"addCountsAnd...	ds	1
0071c648	certificateAuthorities	"certificateAuth...	ds	1
0071c67a	discardHandshakeBuffer	"discardHandsh...	ds	1
0071c6ac	maxPayloadSizeForWrite	"maxPayloadSiz...	ds	1
0071c749	reflect:"slice"	"reflect:\"slice\""	ds	1
0071d1fb	shouldSendContentLength	"shouldSendCo...	ds	1
0071db0f	hashForClientCertificate	"hashForClient...	ds	1
0071ea80	NegotiatedProtocolIsMutual	"NegotiatedPro...	ds	1
0071f169	UnhandledCriticalExtensions	"UnhandledCriti...	ds	1
0071f1a5	parseDynamicTableSizeUpdate	"parseDynamic...	ds	1
0071f758	shouldSendChunkedRequestBody	"shouldSendCh...	ds	1
007212e2	"*struct { F uintptr; ss []string }	"\"*struct { F ui...	ds	1
00986210	-----END	"\n-----END "	ds	1
00986240	-----BEGIN	"\n-----BEGIN "	ds	1
00986600	server finished	"server finished"	ds	1
00400001	ELF	"ELF"	ds	7
007c7fa3	= helpgc= incr=%v is not mcou...	"= helpgc= incr...	ds	187
007d41fa	... cannot be converted to type ...	"...\n cannot be...	ds	1027

Other researcher’s work

Links

IDA Pro

- https://github.com/sibears/IDAGolangHelper
- https://github.com/strazzere/golang_loader_assist

radare2 / Cutter

- https://github.com/f0rki/r2-go-helpers
- https://github.com/JacobPimental/r2-gohelper/blob/master/golang_helper.py
- https://github.com/CarveSystems/gostringsr2

Binary Ninja

- https://github.com/f0rki/bn-goloader

Ghidra

- https://github.com/felberj/gotools
  Only handles linux/x86_64 binaries.
- https://github.com/ghidraninja/ghidra_scripts/blob/master/golang_renamer.py

References, additional reading

Sysrv blog posts and other Go malware research

- https://www.intezer.com/blog/research/new-golang-worm-drops-xmrig-miner-on-servers/
- https://help.aliyun.com/document_detail/196163.html
- https://s.tencent.com/research/report/1259.html
- https://blogs.juniper.net/en-us/threat-research/sysrv-botnet-expands-and-gains-persistence
- https://www.lacework.com/blog/sysrv-hello-expands-infrastructure/
- https://blog.netlab.360.com/threat-alert-new-update-from-sysrv-hello-now-infecting-victims-webpages-to-push-malicious-exe-to-end-users/
- https://community.riskiq.com/article/98f391f9
- https://developer.aliyun.com/article/780758
- https://digital.nhs.uk/cyber-alerts/2021/cc-3838
- https://braintrace.com/wp-content/uploads/2021/06/Threat-Advisory-Report-6-17-2021.pdf
- https://rednaga.io/2016/09/21/reversing_go_binaries_like_a_pro/
- https://2016.zeronights.ru/wp-content/uploads/2016/12/GO_Zaytsev.pdf
- https://carvesystems.com/news/reverse-engineering-go-binaries-using-radare-2-and-python/
- https://www.pnfsoftware.com/blog/analyzing-golang-executables/
- https://github.com/strazzere/golang_loader_assist/blob/master/Bsides-GO-Forth-And-Reverse.pdf
- https://github.com/radareorg/r2con2020/blob/master/day2/r2_Gophers-AnalysisOfGoBinariesWithRadare2.pdf

CUJOAI

Botconf
April 2022

Dorka Palotay
Senior Threat Researcher, CUJO AI
@pad0rka

Gyorgy Luptak
Junior Threat Researcher, CUJO AI
@gyluptak

CUJO AI Labs
https://github.com/getCUJO/ThreatIntel
@CujoaiLabs