Quickstart: RouterOS jailbreaking and security research

19 & 20 JUNE

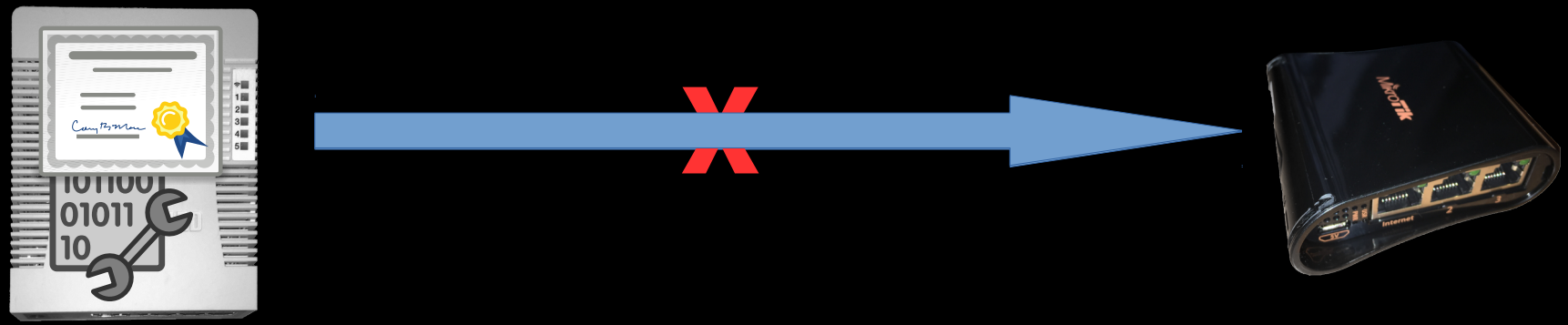
Hack in Paris

Author

- Lead researcher at Possible Security, Latvia
- Author of RouterOS jailbreaks
- CCC, Hack in the Box, Nullcon, BalCCon, CONFidence, TyphoonCon....
- Trainer: CEI, CEH, CASP, CySA+, PenTest+
- twitter / @KirilsSolovjovs

Legal disclaimer

Goal of this research is to achieve the interoperability of computer programs (i.e. software running on MikroTik routers) with other computer programs.



Plan for today

- Set-up
- Jailbreak
- RouterOS internals
  - NPK
  - Backup files
  - Config files
  - supout

1,5h + 1,5h

Set up

Let's get started

- Network:
  -
  -
- http://eja.lv/3ea
  - RouterOS 6.44.3 ISO; install ALL pckgs
- http://eja.lv/3eb
  - VirtualBox if you ain't got it; Network!
- https://github.com/Oki/mikrotik-tools
  - zero — kilo — india

Mikrotik RouterOS

- Linux
  - old
- Startup scripts
- Nova binaries
- Config

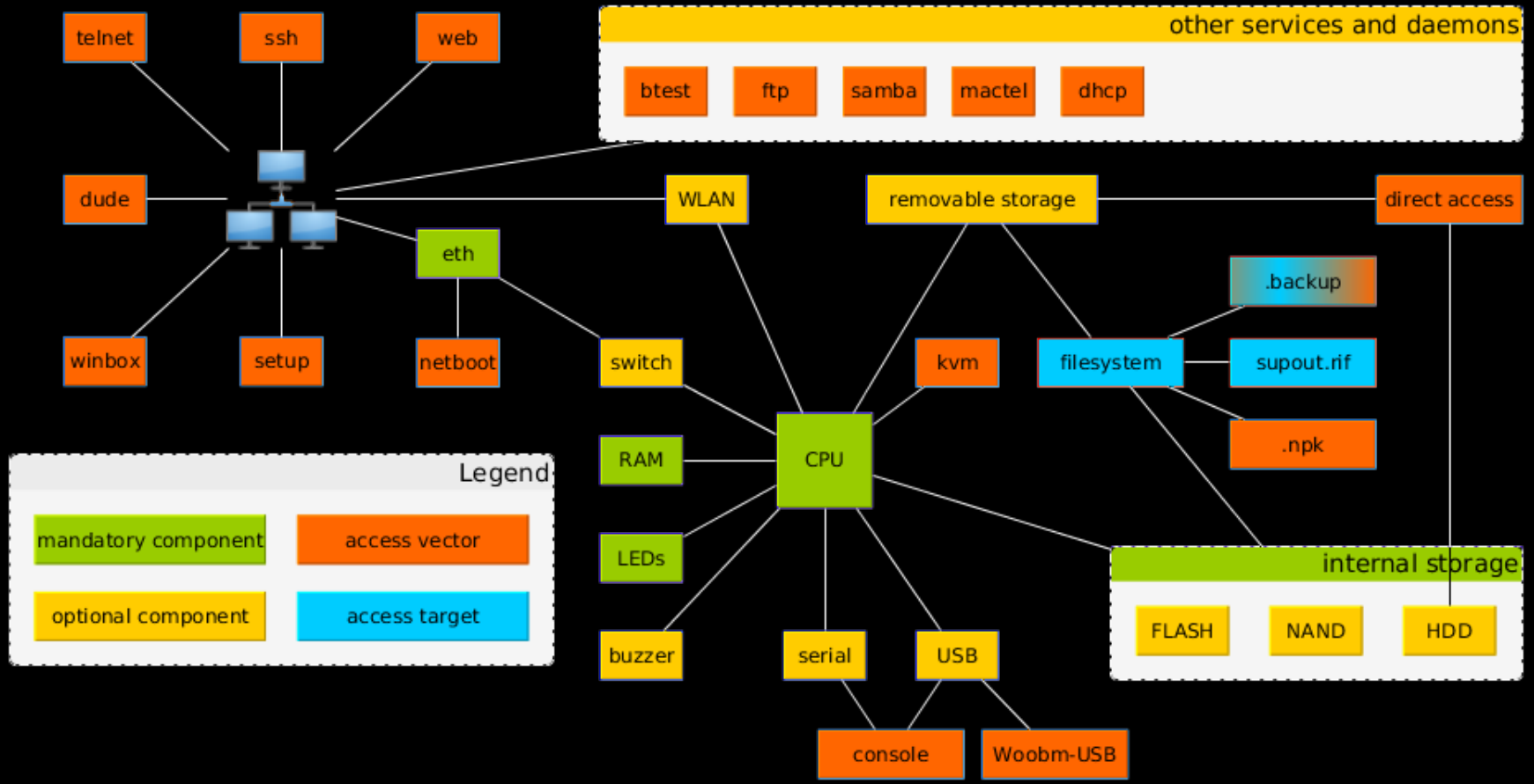
```
11   advanced-tools          6.44.3

# uname -a
Linux MikroTik 3.3.5 #1 Tue May 14 11:46:38 UTC 2019 i686 unknown
#
```

Secure | https://www.kernel.org/pub/linux/kernel/v3.x/

linux-3.3.4.tar.xz	27-Apr-2012 17:46
linux-3.3.5.tar.bz2	07-May-2012 16:15
linux-3.3.5.tar.gz	07-May-2012 16:15
linux-3.3.5.tar.sign	07-May-2012 16:15
linux-3.3.5.tar.xz	07-May-2012 16:15


DEMO

Ecosystem. Possible points of entry.



Jailbreaking

Jailbreak

- Use exploit-backup for versions up to 6.41
- Use exploit-defconf for versions starting with 6.41
  - Supports all current versions up to at least 6.44.3
  - http://02.lv/f/2019/06/19/magic_usb.vdi.zip

Jailbreaking history

- 1999 MikroTik™ v2.0 Router Software released
- 2005 2.9.8 option package & /nova/etc/devel-login introduced
- 2009 3.22 NPK signing added
- 2009 3.30 first jailbreak hints published (that I could find)
  - http://bbs.routerclub.com/thread-67904-1-1.html
- 2017 `mikrotik-tools` published
- 2017 5.x - 6.40.x first fully automated jailbreak tool
- 2017 6.41rc61 devel-login removed; only /pckg/option/ remains
- 2018 defconf-option jailbreak released (still works)

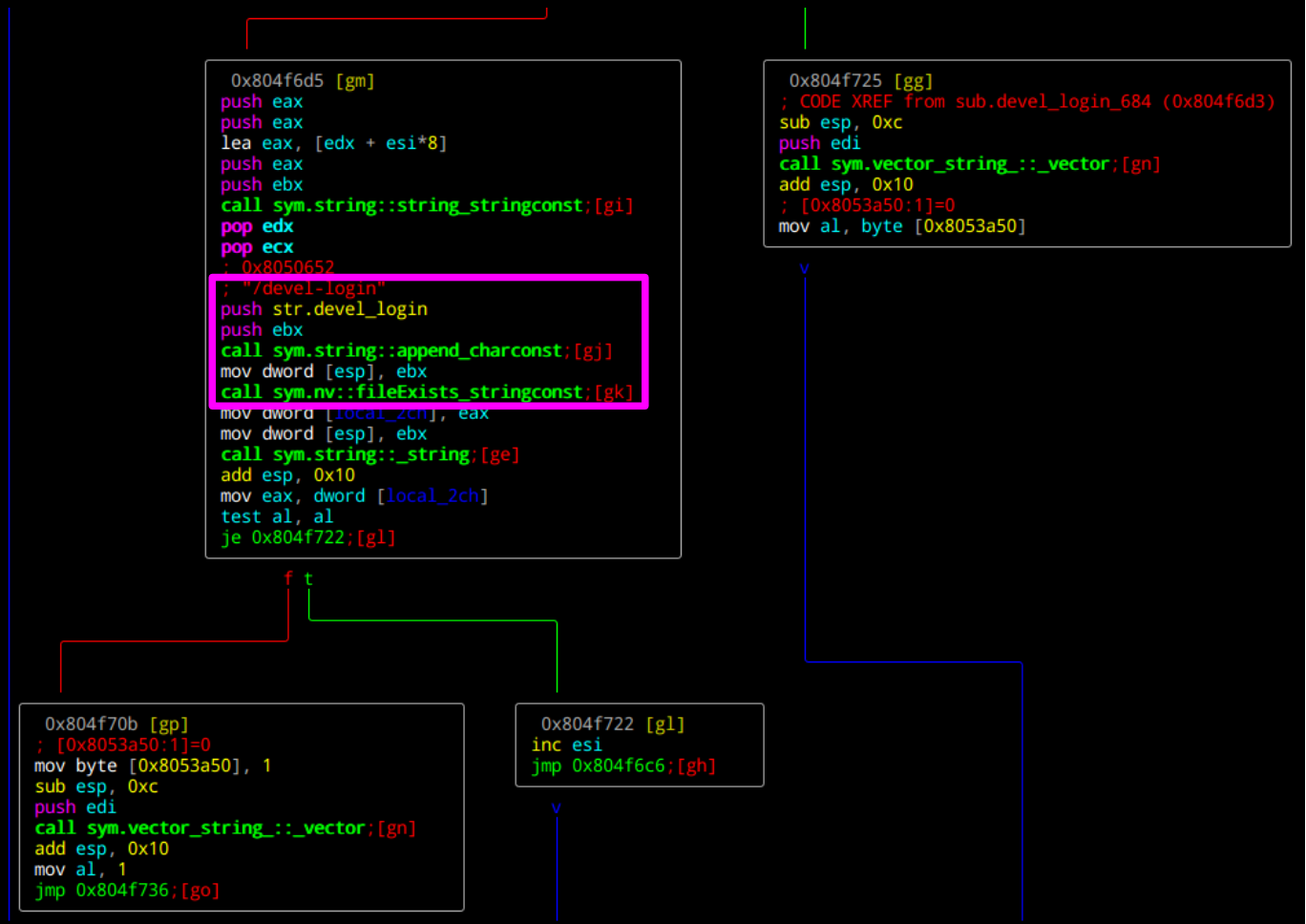
devel-login based jailbreak

- Authenticated root-level access

```
[ -f /nova/etc/devel-login
  && username == devel
  && password == admin.password ]
                          && /bin/ash
```

- /nova/bin/login
- Fixed in 6.41 (not backported)

devel-login


0x804f6d5 [gm]
push eax
push eax
lea eax, [edx + esi*8]
push eax
push ebx
call sym.string::string_stringconst;[gi]
pop edx
pop ecx
; 0x8050652
; "/devel-login"
push str.devel_login
push ebx
call sym.string::append_charconst;[gj]
mov dword [esp], ebx
call sym.nv::fileExists_stringconst;[gk]
mov dword [local_2ch], eax
mov dword [esp], ebx
call sym.string::_string;[ge]
add esp, 0x10
mov eax, dword [local_2ch]
test al, al
je 0x804f722;[gl]
f
t
0x804f725 [gg]
; CODE XREF from sub.devel_login_684 (0x804f6d3)
sub esp, 0xc
push edi
call sym.vector_string_::_vector;[gn]
add esp, 0x10
; [0x8053a50:1]=0
mov al, byte [0x8053a50]
0x804f70b [gp]
; [0x8053a50:1]=0
mov byte [0x8053a50], 1
sub esp, 0xc
push edi
call sym.vector_string_::_vector;[gn]
add esp, 0x10
mov al, 1
jmp 0x804f736;[go]
0x804f722 [gl]
inc esi
jmp 0x804f6c6;[gh]

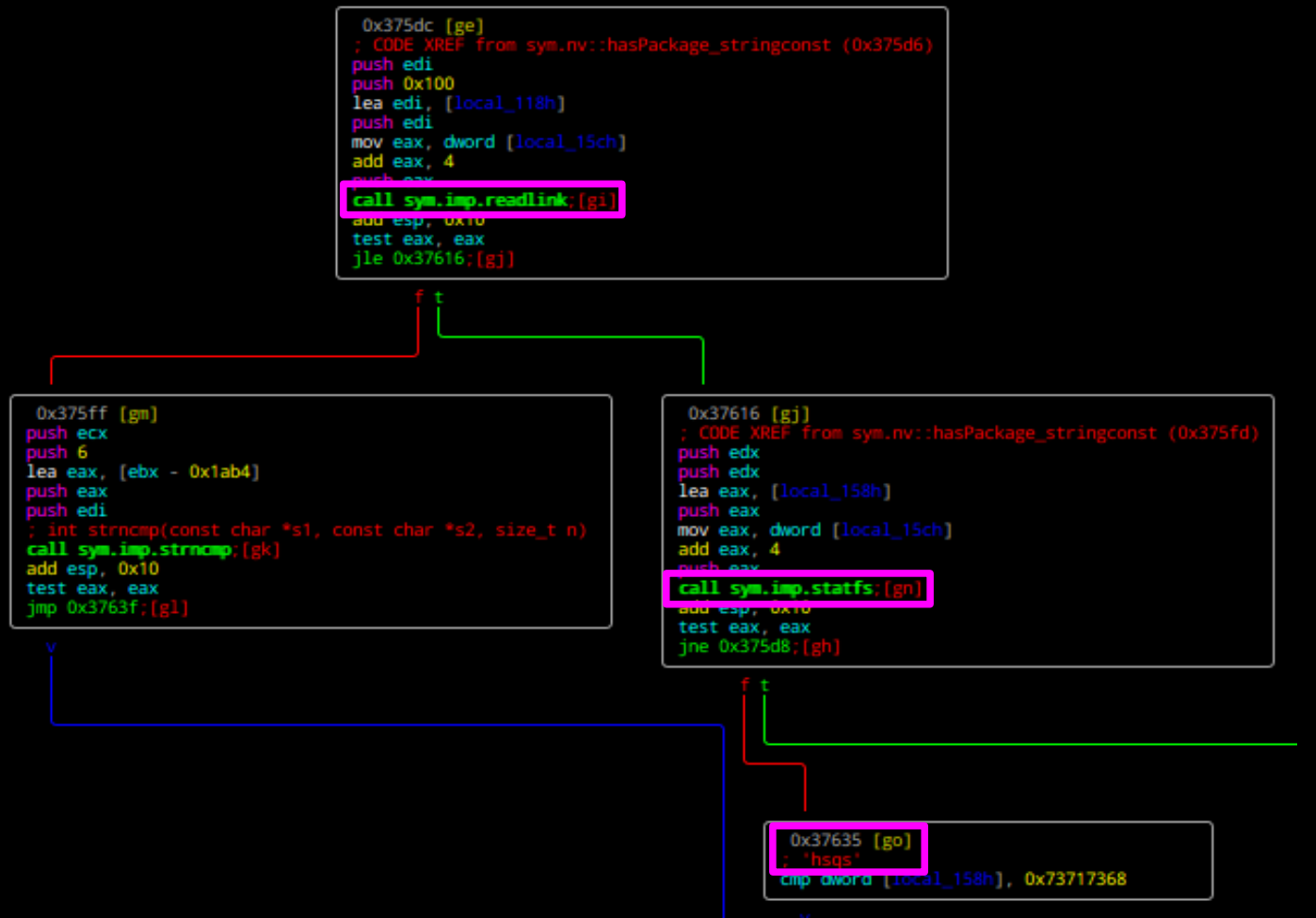
devel-login


call sub.devel_login_684;[gDi]
test al, al
je 0x804bd58;[gDh]
f t
0x804bd25 [gDm]
push edx
push edx
; "admin"
push str.admin
lea edi, [local_110h]
push edi
call sym.string::string_charconst;[gAv]
add esp, 0xc
push edi
push 0x20000001
push esi
call sym.nv::message::insert_string_unsignedint_stringconst;[gDk]
mov dword [esp], edi
call sym.string::_string;[gAz]
; [0x8053a45:1]=0
mov byte [0x8053a45], 1
jmp 0x804bd8b;[gDl]
0x804bd58 [gDh]
; CODE XREFS from main (0x804bd1a, 0x804bd23)
push eax
push eax
push ebx
lea esi, [local_110h]
push esi
call sym.string::string_charconst;[gAv]
add esp, 0xc
push esi
push 0x20000001
lea eax, [local_15ch]
push eax
call sym.nv::message::insert_string_unsignedint_stringconst;[gDk]
mov dword [esp], esi
call sym.string::_string;[gAz]
; [0x8053a45:1]=0
mov byte [0x8053a45], 0

exploit-backup based jailbreak

- mkdir -p pathname(“/flash/rw/store/”+filename)
- write idx to “/flash/rw/store/”+filename+“.idx”
- write dat to “/flash/rw/store/”+filename+“.dat”

package/option based jailbreak

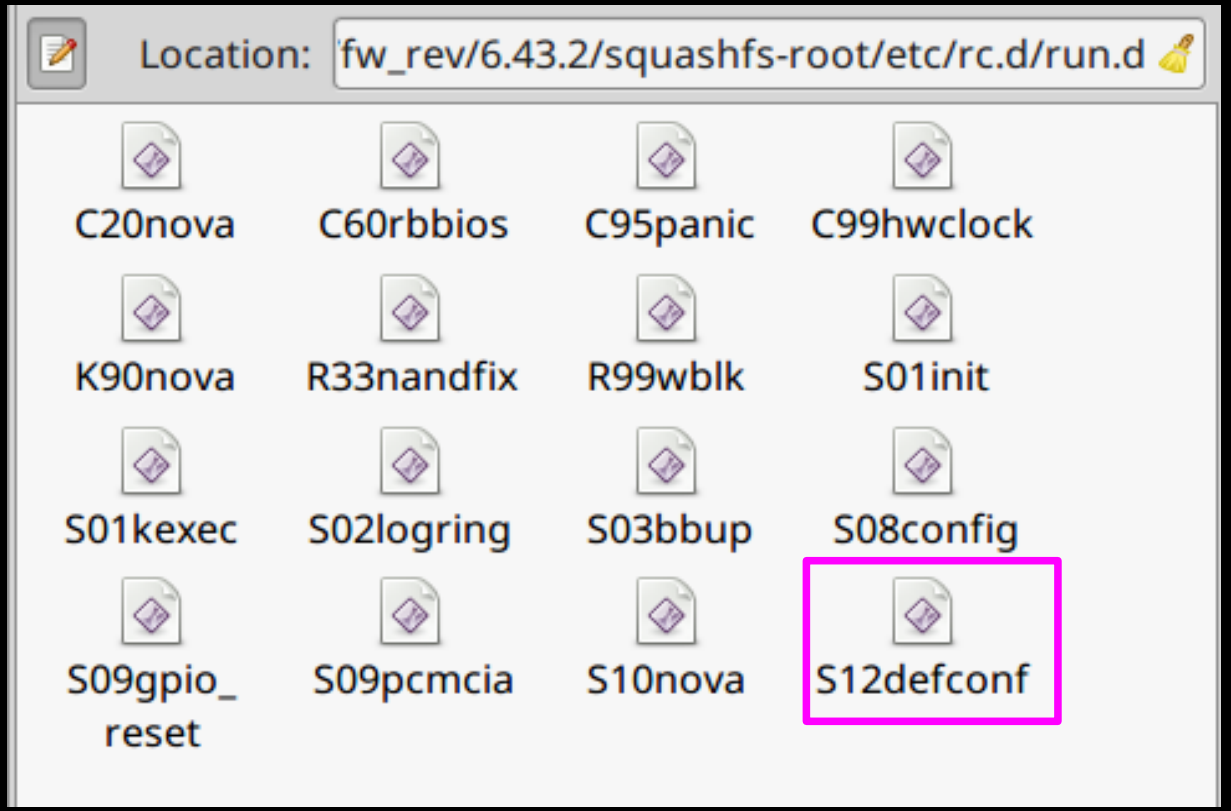


package/option based jailbreak

- lib/libumsg.so
- nv::hasPackage(“option”)
- nv::hasPackage checks if
  - /pckg/<name> exists
  - if it’s not a symlink
  - if fs is squashfs

¯_(ツ)_/¯

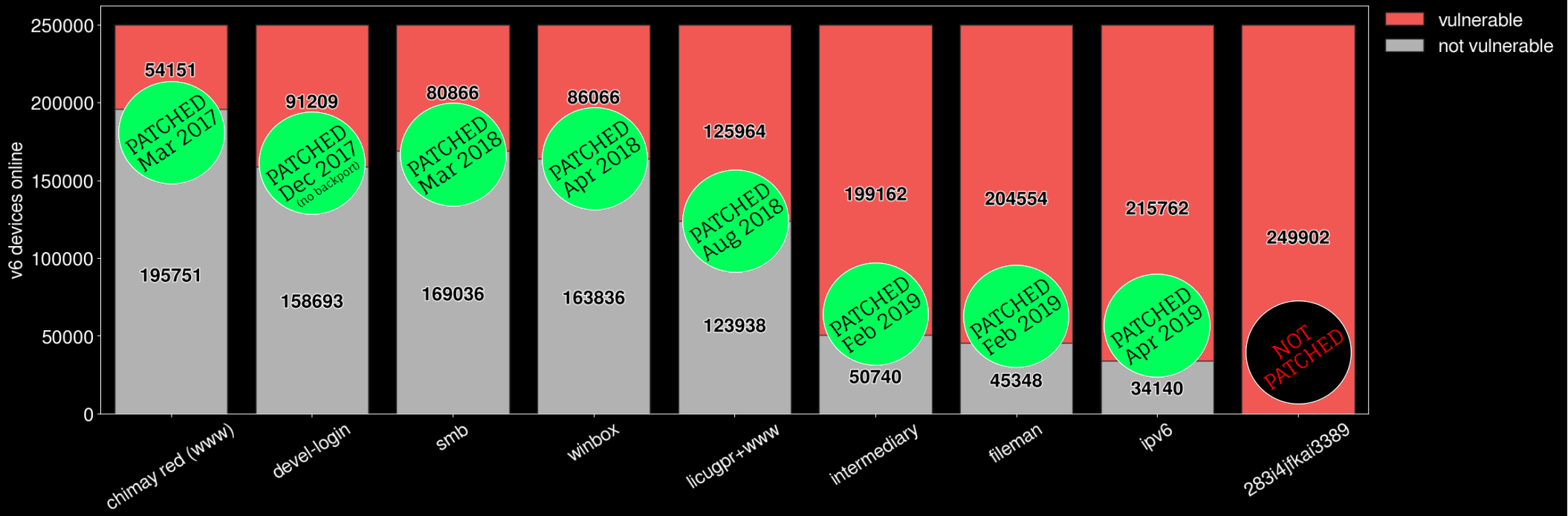
- mkdir /pckg/option
- mount -o bind /pckg/dude/ /pckg/option


Location:
fw_rev/6.43.2/squashfs-root/etc/rc.d/run.d
C20nova
C60rbbios
C95panic
C99hwclock
K90nova
R33nandfix
R99wblk
S01init
S01kexec
S02logring
S03bbup
S08config
S09gpio_
reset
S09pcmcia
S10nova
S12defconf

```
elif [ -f /rw/DEFCONF ]; then
    usleep 3000000
    /nova/bin/sendmsg 0xfe0000 48
    confirm=/ram/DEFCONF_CONFIRM
    if [ ! -s /rw/DEFCONF ]; then
    /nova/lib/defconf/choose >> /rw/DEFCONF
    confirm=/rw/DEFCONF_CONFIRM
    fi
    /nova/bin/autoupdate
    defcf=$(cat /rw/DEFCONF)
    echo > /ram/defconf-params
    if [ -f /nova/bin/flash ]; then
    /nova/bin/flash --fetch-defconf-params /ram/defconf-params
    fi
    (eval $(cat /ram/defconf-params) action=apply /bin/gosh $defcf;
     cp $defcf $confirm; rm /rw/DEFCONF /ram/defconf-params) &
fi
```

Please, patch!

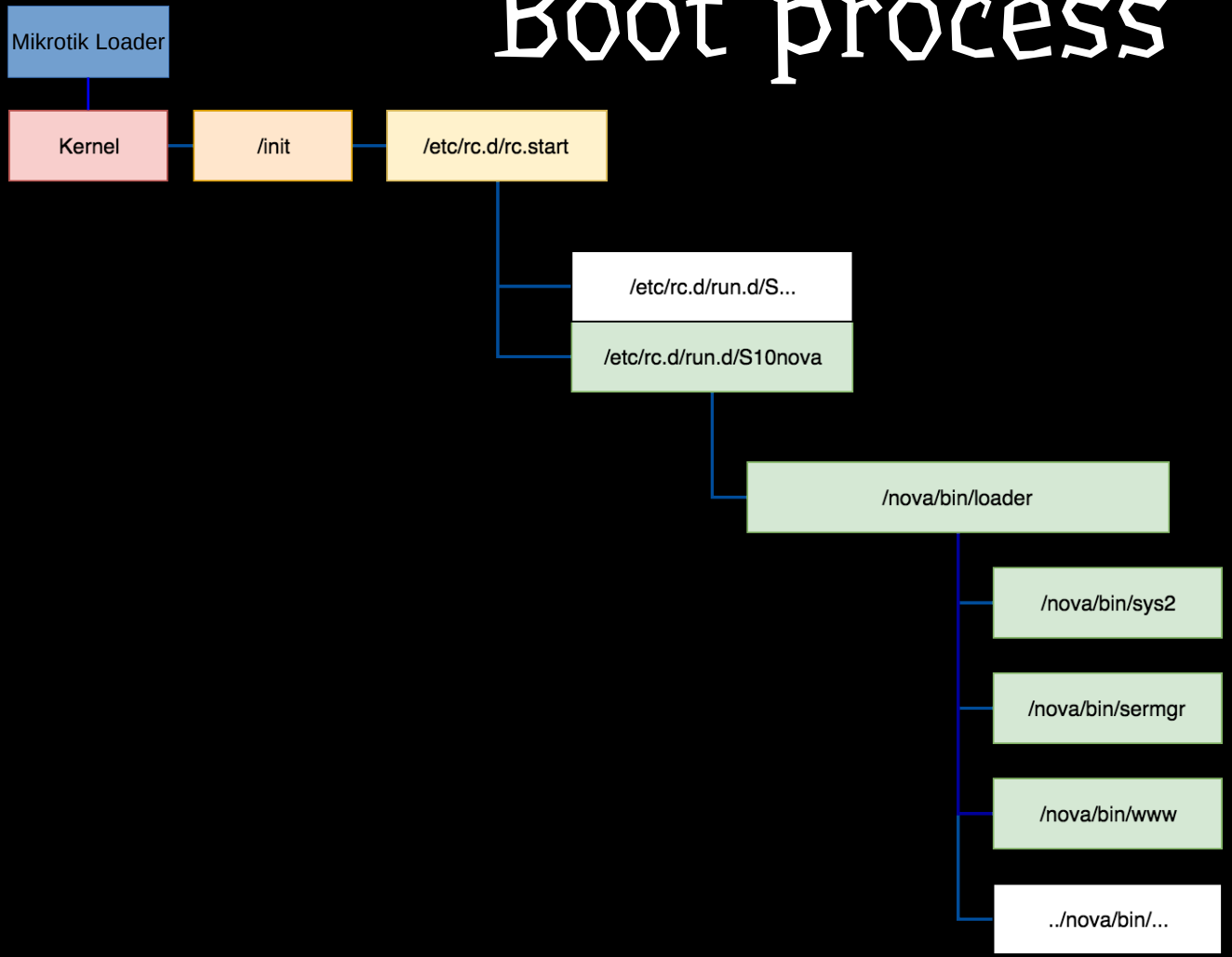
Do you even patch, bro?



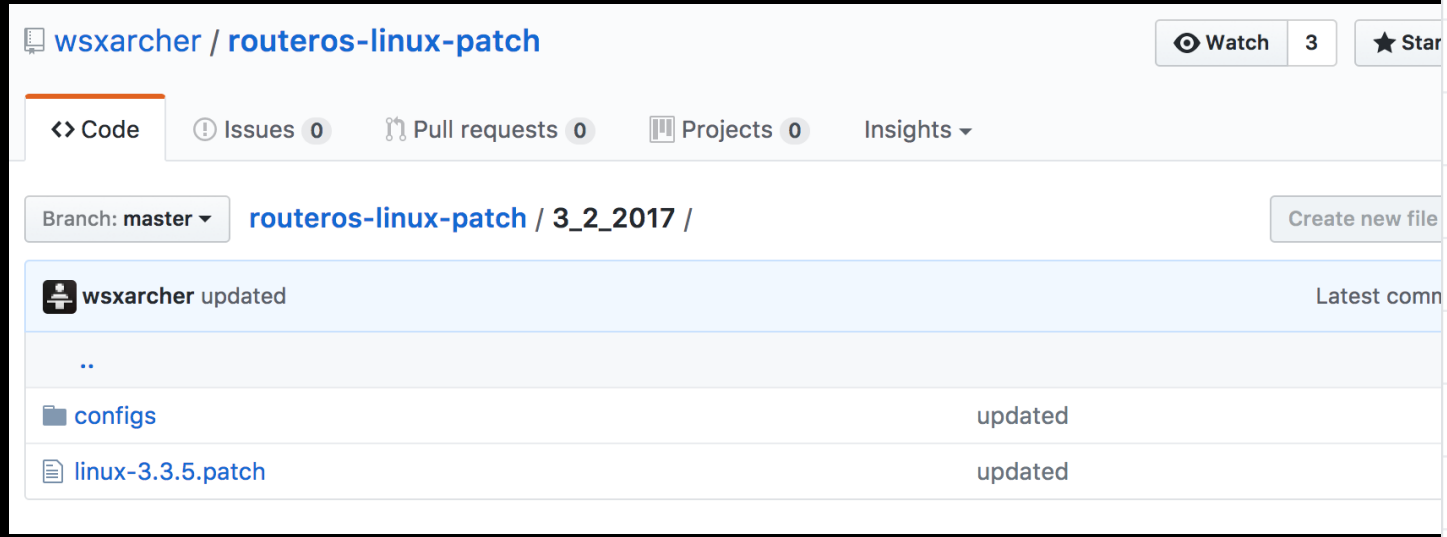
June 12 2019

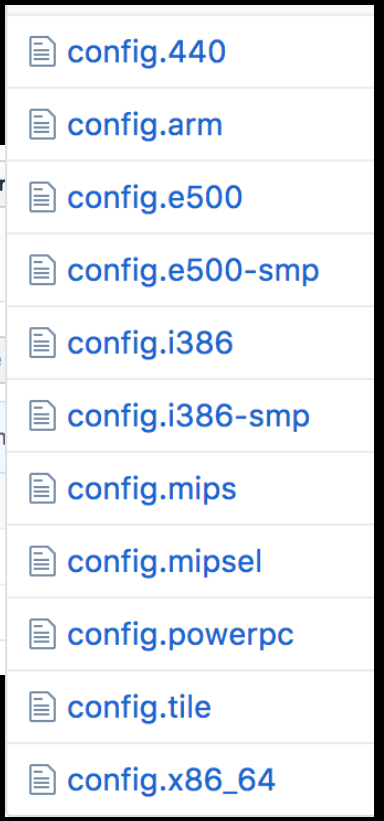
RouterOS internals

Boot process



Kernel patches





https://github.com/wsxarcher/routeros-linux-patch

Hacking RouterOS

NPK file sourcing

- getnpk.sh
  - deps: wget
- reversenpk.sh
  - deps: unsquashfs (squashfs-tools), unnpk
    - https://github.com/rsa9000/npk-tools
    - http://02.lv/f/2019/06/19/unnpk

Get ready to take a look inside

- Download some NPKs
- getnpk.sh 6.44
- getnpk.sh -calea-6.44
- getnpk.sh -mikrotik-6.43.iso
- More:
  - 6.38.4 and 6.38.5 (chimay_red)

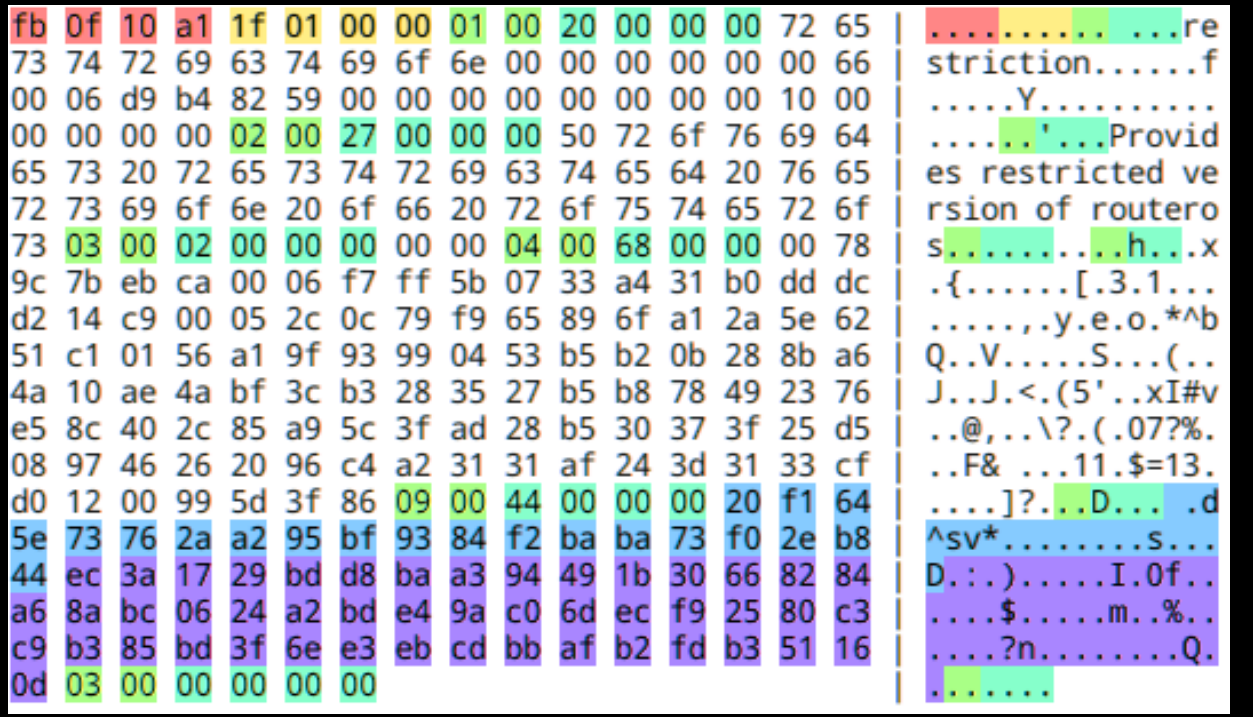
NPK packages

Now take a look inside

- reversenpk.sh

NPK format

- Nova PacKage
- Numeric values are unsigned little endian
- File consists of header, file size, and parts.
- File size is 8B less
- Each part consist of:
  - part type (short)
  - payload size (long)
  - payload



NPK format

- At least two types of current NPKs:
  - package
    - 0..3 header 1E F1 D0 BA
  - restriction (invisible package)
    - 0..3 header FB 0F 10 A1

```
[admin@MikroTik] > /system package
[admin@MikroTik] /system package> print
Flags: X - disabled
 #   NAME                      VERSION                          SCHEDULED
 0   system                    6.38.4
 1 X restriction               6.0
[admin@MikroTik] /system package>
```

Part types

N	Type	Meaning	First seen	Last seen	Mandatory
1	01 00	Part info	forever	now	yes
2	02 00	Part description	forever	now	yes
3	03 00	Dependencies	forever	now	yes
4	04 00	File container	forever	now	no
5	05 00	Install script (libinstall)	forever	2.7.xx	no
6	06 00	? Uninstall script (libinstall)	never	never	no
7	07 00	Install script (bash)	forever	now	no
8	08 00	Uninstall script (bash)	forever	now	no
9	09 00	Signature	3.22	now	yes
10	0a 00	unused	never	never	no
11	0b 00	unused	never	never	no
12	0c 00	unused	never	never	no
13	0d 00	unused	never	never	no
14	0e 00	unused	never	never	no
15	0f 00	unused	never	never	no
16	10 00	Architecture	2.9	now	yes
17	11 00	Package conflicts	3.14	3.22	no
18	12 00	Package info	2.9	now	no
19	13 00	Part features	2.9	now	no
20	14 00	Package features	2.9	now	no
21	15 00	SquashFS block	6.0beta3	now	package only
22	16 00	Zero padding	6.0beta3	now	no
23	17 00	Digest	6.30	now	package only
24	18 00	Channel	6.33	now	package only

Nova binaries (1)

- /nova/bin/loader
  - Spawns processes and manages communication between them
- /nova/bin/watchdog
  - Restarts the device if a critical process stops working
- /nova/bin/sys2
  - Manages device settings and parses received commands
- /nova/bin/sermgr (kind of like inetd)
  - Super-server daemon that provides internet services

Nova binaries (2)

- /nova/bin/net
  - Deals with network configuration, tunnels, AT commands
- /nova/bin/moduler
  - Manages loading of firmware for external devices
    - e.g. usb2serial adpters, 3G modems
- /nova/bin/modprobed
  - Symlink to moduler, used for loading kernel modules
- /nova/bin/manager
  - Manages loading of firmware for external devices
    - e.g. usb2serial adpters, 3G modems

Nova binaries (3)

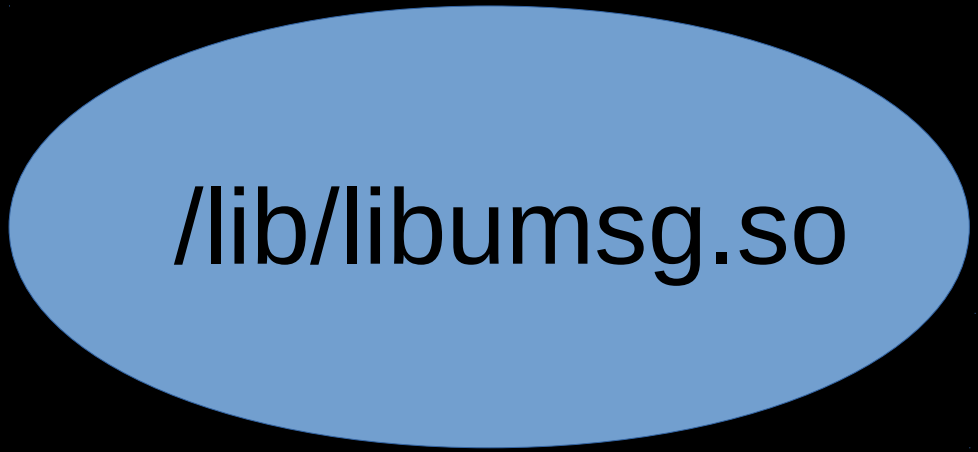
- /nova/bin/log
  - Log daemon
- /nova/bin/mproxy
  - Winbox daemon
- /nova/bin/quickset
  - Separate daemon for management of quickset settings
- /nova/bin/undo
  - Safe mode support
- /nova/bin/www
  - Web interface daemon

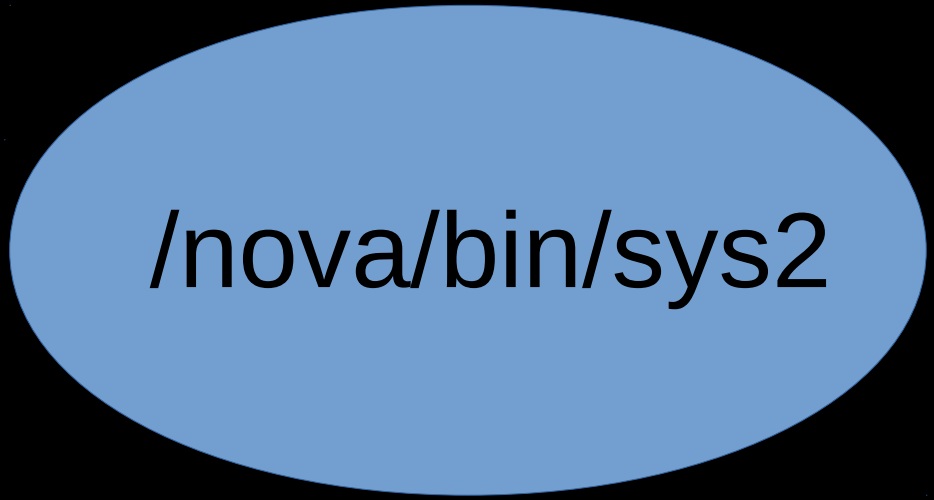
Take a look at www (6.38.4)

- r2 -A nova/bin/www
  - s sym.Request::readPostData_string__unsignedint_const
  - pdf


```
0x08055a04      55              push ebp
0x08055a05      89e5            mov ebp, esp
0x08055a07      57              push edi
0x08055a08      56              push esi
0x08055a09      53              push ebx
0x08055a0a      83ec24          sub esp, 0x24                ; '$'
0x08055a0d      8b7d10          mov edi, dword [arg_10h]     ; [0x10:4]=-1 ; 16
0x08055a10      c745e4000000.   mov dword [local_1ch], 0
0x08055a17      683dac0508      push str.content_length      ; 0x805ac3d ; "content-
0x08055a1c      8d75e0          lea esi, [local_20h]
0x08055a1f      56              push esi
0x08055a20      e80bb5ffff      call sym.string::string_charconst
0x08055a25      83c40c          add esp, 0xc
0x08055a28      8d45e4          lea eax, [local_1ch]
0x08055a2b      50              push eax
0x08055a2c      56              push esi
0x08055a2d      ff7508          push dword [arg_8h]
0x08055a30      e8d1160000      call sym.Headers::getHeader_stringconst__unsignedint
0x08055a35      88c3            mov bl, al
0x08055a37      893424          mov dword [esp], esi
0x08055a3a      e8c1a8ffff      call sym.string::_string
0x08055a3f      83c410          add esp, 0x10
0x08055a42      84db            test bl, bl
0x08055a44      7504            jne 0x8055a4a
; CODE XREFS from sym.Request::readPostData_string__unsignedint_const (0x8055a51, 0
0x08055a46      31db            xor ebx, ebx
0x08055a48      eb57            jmp 0x8055aa1
; CODE XREF from sym.Request::readPostData_string__unsignedint_const (0x8055a44)
0x08055a4a      85ff            test edi, edi
0x08055a4c      7405            je 0x8055a53
0x08055a4e      3b7de4          cmp edi, dword [local_1ch]
0x08055a51      72f3            jb 0x8055a46
; CODE XREF from sym.Request::readPostData_string__unsignedint_const (0x8055a4c)
0x08055a53      8b55e4          mov edx, dword [local_1ch]
0x08055a56      8d4210          lea eax, [edx + 0x10]        ; 16
0x08055a59      83e0f0          and eax, 0xfffffff0
0x08055a5c      29c4            sub esp, eax
0x08055a5e      89e7            mov edi, esp
0x08055a60      50              push eax
0x08055a61      52              push edx
```

Messaging in RouterOS





Custom binaries

Upload the good stuff

- scp exploit-backup/busybox-arch admin@0.0.0.0:/ ← run from Linux box
- run in jailbroken shell:
- mv /flash/rw/disk/busybox /rw/tmp
- cd /rw/tmp
- chmod a+x ./busybox
- ./busybox --install -s .
- export PATH=$PATH:/rw/tmp

Look around

- `netstat -apn`

Backup files

Backup file layout

- Header (long)
  - 0x88ACA1B1 – backup
  - 0xEFA89172 – encrypted backup
- Length of backup file (long)
- Records of:
  - Path name, idx contents, dat contents
- Each record consists of length (long) and binary data

Take a look inside a backup

- /user
  - add ...
  - set ...
- /system backup save dont-encrypt=yes
- decode_backup.py

Config files

Configuration

- Config is stored in /rw/store as pairs of files
  - IDX = index
  - DAT = data

```
batman
bgconf
bserv.dat
bserv.idx
cerm.dat
cerm.idx
cert
cmanifacecf
cmanifacecf
command
dhcp
diskd
dude
echosave
graphing
group.dat
group.idx
hotspot
igmpproxy
log-actions.idx
log-rules.dat
smbusers.idx
snmp-communities.dat
snmp-communities.idx
snmpd.dat
snmpd.idx
ssh
sstp
system.dat
system.idx
tftp.dat
tftp.idx
um4.dat
um4.idx
unicl
user
user.dat
user.idx
webproxy
wireless.dat
wireless.idx
```

IDX format

- Record ID (long)
  - if ID is 0xFFFFFFFF, field has no content
  - used for offsetting
- length (long)
- separator (long)
  - usually 0x05000000

DAT format

- LENGTH (short)
- M2 RECORD of length
  - Config ID (3 bytes)
  - type (1 byte)
    - content depends on to type

```
btype ........
      0000000, - boolean
      ,,1,1,,, - M2 block (len = short)
      ,,11,,,, - binary data block (len = short)
      ,,,,,,,1 - one byte
      ,,,,,,1, - ???
      ,,,,,1,, - ???
      ,,,11,,, - 128bit int
      ,,,,1,,, - int (four bytes)
      ,,,1,,,, - long (8 bytes)
      ,,1,,,,, - string
      ,1,,,,,, - ??? unused? or long array of?
      1,,,,,,, - short array of

types (MT notation)
    (CAPITAL X = list of x)

    a,A, (0x18) IPv6 address (or duid)
    b,B, bool
      M, multi
    q,Q, (0x10) big number
    r,R, (0x31) mac address
    s,S, (0x21) string
    u,U, unsigned integer
```

Peculiarities / features

- Field IDs shared with web
- Winbox protocol derived from DAT format
  - “Must be dangerous” —me, 2017

Let's decode some config

- mt_dat_decoder.py

```
from mt_dat_decoder import MTConfig

conf = MTConfig("disks.dat","disks.idx")
conf.mapBlockNames( {0xb:"permissions"} )

for record in conf:
   print(record)
```

Where's my password?

- Calm down! It's encrypted!

```
# ls /rw/store/
batman                  log-actions.idx         smbusers.idx
bfd                     log-rules.dat           snmp-communities.dat
bgconf                  log-rules.idx           snmp-communities.idx
bserv.dat               mactel.dat              snmpd.dat
bserv.idx               mactel.idx              snmpd.idx
cerm.dat                mcast                   ssh
cerm.idx                mpls                    sstp
cert                    mproxy.dat              system.dat
cmanifacecfg.dat        mproxy.idx              system.idx
cmanifacecfg.idx        net                     tftp.dat
command                 ospfconf                tftp.idx
dhcp                    ospfv3                  um4.dat
diskd                   ovpn                    um4.idx
dude                    ppp                     unicl
echosave                pptp                    user.dat
graphing                radius                  user.idx
group.dat               radius.dat              wireless.dat
group.idx               radius.idx              wireless.idx
hotspot                 radvd
igmpproxy               resolver
```

The password is

- hashed
- salted
- md5

- Oh, wait, no. That's the key.


'MEMBER ME?
key = md5(username + "283i4jfkai3389")
password_e = password xor key


esc
F1
F2
F3
F4
F5
F6
F7
F8
F9
F10
F11
F12
~ `
! 1
@ 2
# 3
$ 4
% 5
^ 6
& 7
* 8
(9
) 0
_ -
+ =
delete
tab
Q
W
E
R
T
Y
U
I
O
P
{ [
}]
| \
caps lock
A
S
D
F
G
H
J
K
L
: ;
" '
enter
return
shift
Z
X
C
V
B
N
M
< ,
> .
? /
shift
fn
control
alt
option
command
command
alt
option

Passwords?

- decode_user.py

supout.rif

What is supout.rif?

- Support output
  - ridiculously intricate format
  - or RouterOS information file, maybe, idk ¯_(ツ)_/¯

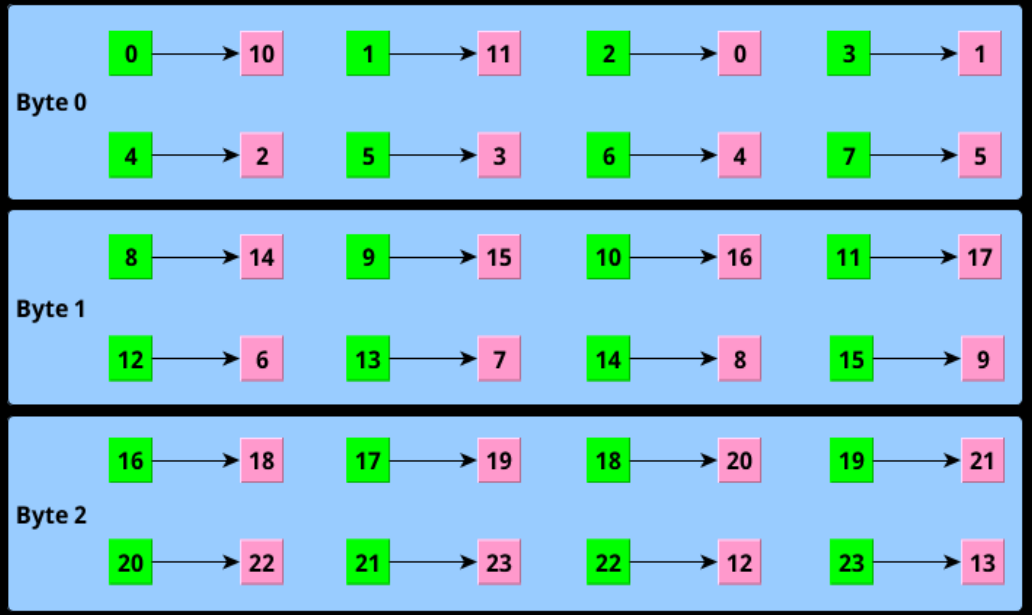
```
[admin@MikroTik] > /system sup-output
  created: 1%
-- [Q quit|D dump|C-z pause]
```

supout.rif from outside

```
...BEGIN ROUTEROS SUPOUT SECTION
oVWYsRHaAgHnjXuAAAgJAgB=
--END ROUTEROS SUPOUT SECTION
--BEGIN ROUTEROS SUPOUT SECTION
w9WZt8Wd0BAecukSMFFSO/czNx8SRh8SM3UVog8TVBNyJz8SVBjBKROlmbeKYkxayFAAccOD1D==
--END ROUTEROS SUPOUT SECTION
--BEGIN ROUTEROS SUPOUT SECTION
sNGZ09WdjhGA4x58xZXUwdX9z1gc0HFC21QCxT/cPYe5KpETRhkzP3cTMvUUIvEzNVFyJ5UUQjcy
MvUVwEgSkTp5mnCmoJXAAsy1S0E=
--END ROUTEROS SUPOUT SECTION
```

supout.rif section decoding

- swap bits around
  - per three bytes
- base64
- section decodes to:
  - name + ‘\0’ + zlib_compressed_content



supout.rif section decoding

```
tribitmap=[10,11,0,1,2,3,4,5,14,15,16,17,6,7,8,9,18,19,20,21,22,23,12,13]

def tribit(content):
    result=""
    for i in xrange(0, len(content) - 1,3):
        goodtribit=0
        badtribit=ord(content[i])*0x10000+ord(content[i+1])*0x100+ord(content[i+2])
        for mangle in tribitmap:
            goodtribit = (goodtribit<<1) + (1 if ((badtribit & (0x800000>>mangle))>0) else 0)

        for move in [16,8,0]:
            result=result+chr((goodtribit >> move)& 0xff)

    return result
```

supout.rif from inside

- What does it contain?
  - your whole configuration
  - /proc/ folder
  - memory addresses
  - your log
  - and more

```
$ ls supout.rif_contents/
01_.debug       16_arp             31_profile       46_wirelesselog
02_.profile     17_ip              32_dhcp          47_bfd
03_.proc        18_nexthop         33_neighbor      48_bgp
04_.startup     19_route           34_dhcp6         49_mme
05_.livetrace   20_user            35_license       50_mpls
06_resource     21_firewall        36_package       51_ntp-client
07_pci          22_firewall-stats  37_instchk       52_ospf
08_usb          23_bridge          38_oops          53_ppp
09_log          24_mesh            39_backtrace     54_ipsec
10_export       25_queue           40_store         55_health
11_interface    26_queue-packets   41_hotspot       56_poe-out
12_ethernet     27_queue-bytes     42_routerboard   57_lcdtouch
13_switch       28_queue-stats     43_webproxy
14_address      29_ippool          44_wireless
15_port         30_certificate     45_wirelessdump
$
```

Playing around with supout files

- decode_supout.py
- modify
- encode_supout.py
- upload it to
  - https://mikrotik.com/client/supout
  - DO NOT try to hack their server!

Final boss task

- Requirements:
  - radare, gdb, ghidra or IDA pro
- Take a look at:
  - **diff -R** two recent versions
  - **r2 -g** vulnerable and non-vulnerable binary
  - Take a look at:
    - bash, cloud, kidcontrol, licupgr

Thank you!

@KirilsSolovjovs