SYNACKTIV
DIGITAL SECURITY

How Sercomm saved my Easter!

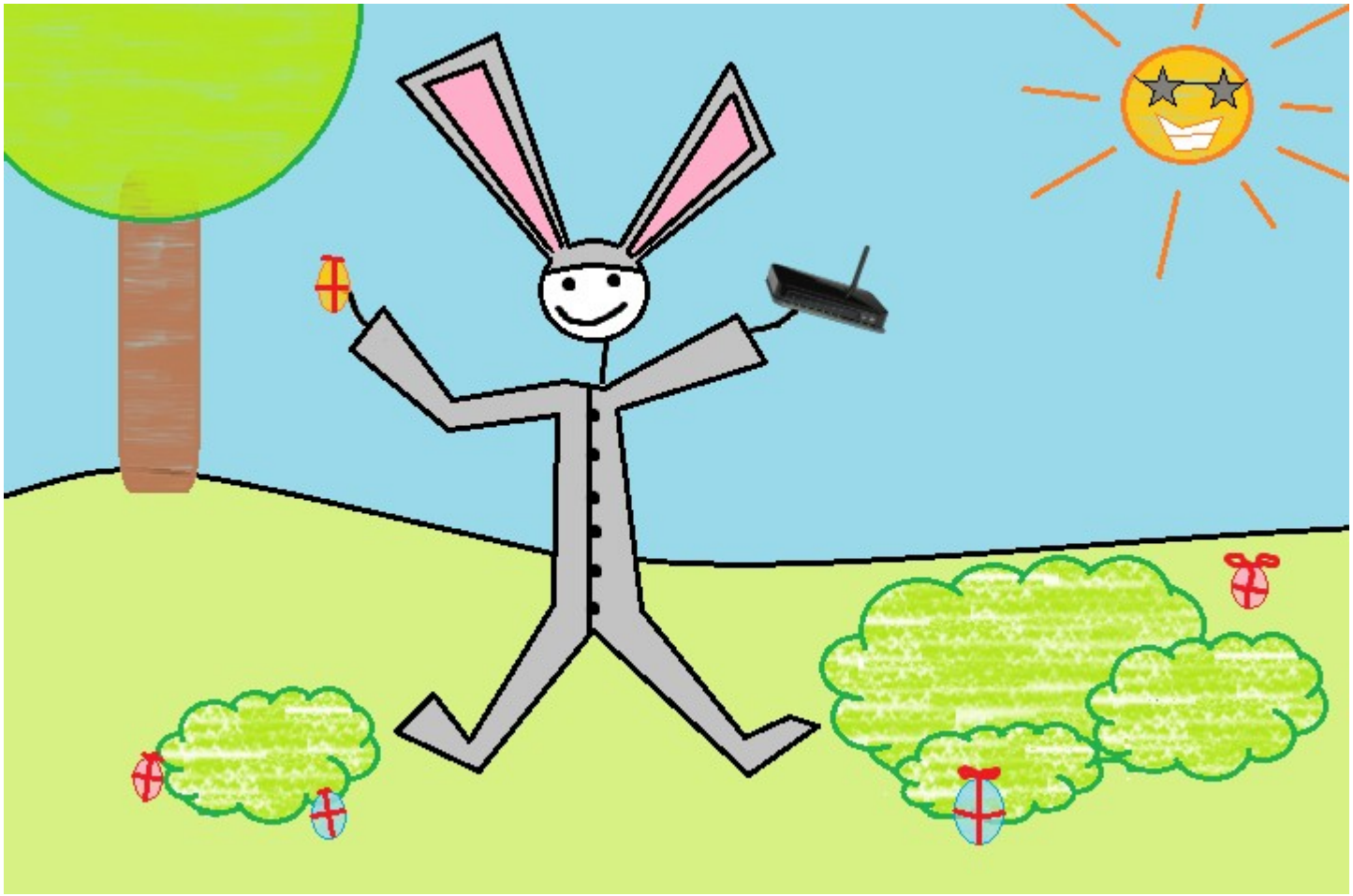
Another backdoor in my router:
when Christmas is NOT enough!

Released 18/04/2014

By Eloi Vanderbeken - Synacktiv

I don't know about you, but I love Easter!

- **And with Sercomm, it's Easter every day!**



Remember the TCP/32764 router backdoor?

- **Introduced by Sercomm**
- **Gives root shell, no authentication**
- **Dump entire configuration**
- **4 affected manufacturers (Cisco, Linksys, NetGear, Diamond)**
- **24 router models confirmed vulnerable**
- **6000 vulnerable routers on the Internet**
- **(more info: https://github.com/elvanderb/TCP-32764)**

It was patched!

zmaile commented 11 days ago

I brought this issue up with netgear support (2014/01/17), and just in the last few days they have released a new firmware version that resolves the port 32764 issue. The new firmware is available on their website (http://downloadcenter.netgear.com/other/)

I've confirmed that the below version works correctly.
http://www.downloads.netgear.com/files/GDC/DGN1000/DGN1000-V1.1.00.49WW.zip

If the original backdoor was a planned 'feature', then its possible that there is a knocking sequence required to unlock port 32764 (that is, port 32764 opens after trying port 5000, then 8000 before 32764 as an example).

No, it can't be a *feature*!
It was a simple mistake... wasn't it?

zmaile commented 11 days ago

I brought this issue up with netgear support (2014/01/17), and just in the last few days they have released a new firmware version that resolves the port 32764 issue. The new firmware is available on their website (http://downloadcenter.netgear.com/other/)

I've confirmed that the below version works correctly.
http://www.downloads.netgear.com/files/GDC/DGN1000/DGN1000-V1.1.00.49WW.zip

If the original backdoor was a planned 'feature', then its possible that there is a knocking sequence required to unlock port 32764 (that is, port 32764 opens after trying port 5000, then 8000 before 32764 as an example).

Let's have a look!

- **'binwalk -e' to extract the file system**
- ***scfgmgr* (the backdoor binary) is still present...**
- **But it's now started with a new -l option**

```
:~/_DGN1000_1.1.00.55_NA.img.extracted$ find . -name scfgmgr
./squashfs-root/usr/sbin/scfgmgr
:~/_DGN1000_1.1.00.55_NA.img.extracted$ grep -r scfgmgr .
./squashfs-root/usr/etc/rcS.MTCODE:/usr/sbin/scfgmgr
./squashfs-root/usr/etc/rcS.IPV6:/usr/sbin/scfgmgr
./squashfs-root/usr/etc/rcS:/usr/sbin/scfgmgr -l &
./squashfs-root/usr/etc/lib_md5:b36d99bad4758881cd62d87ad11bec3c  ./usr/sbin/scfgmgr
```

What's this -l option?

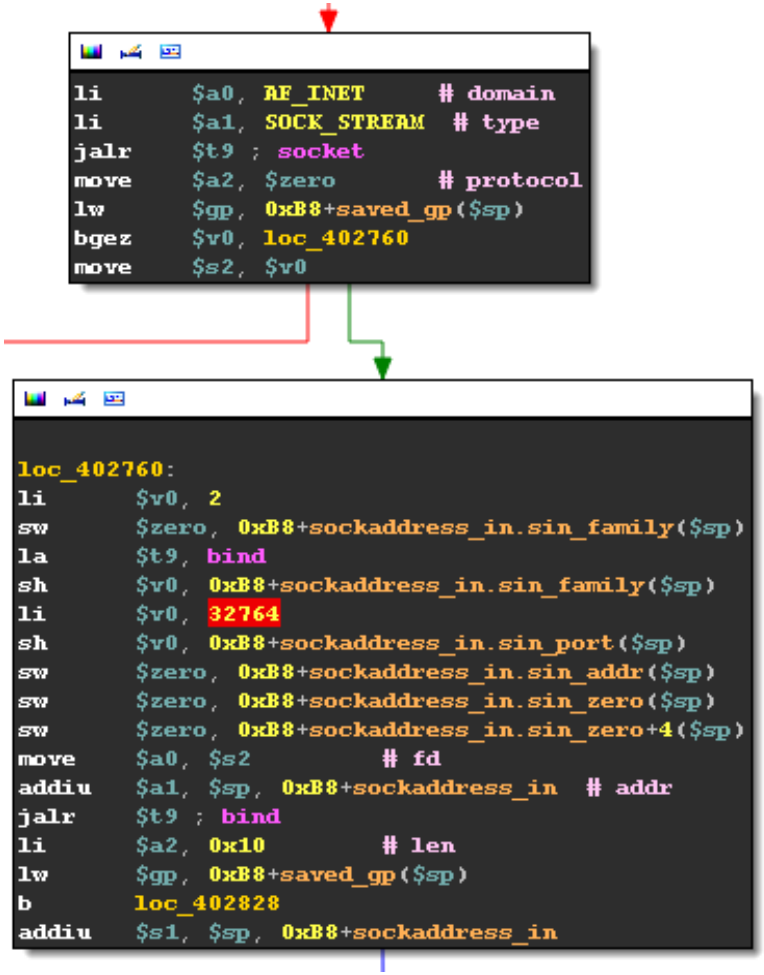
- ***scfgmgr* now listens on a Unix domain socket :'(**

```
loc_4027A0:                             # type
li      $a1, SOCK_STREAM
jalr    $t9 ; socket
move    $a2, $zero                      # protocol
lw      $gp, 0xB8+saved_gp($sp)
bltz    $v0, loc_402738
move    $s2, $v0
```

```
la      $t9, memset
addiu   $s0, $sp, 0xB8+sockaddress_un
move    $a0, $s0                        # s
move    $a1, $zero                      # c
jalr    $t9 ; memset
li      $a2, 0x6E                       # n
lw      $gp, 0xB8+saved_gp($sp)
addiu   $s1, $sp, 0xB8+sockaddress_un.sun_path
la      $a1, 0x400000
la      $t9, strncpy
addiu   $a1, (aTmpScfgmgr_soc - 0x400000)  # "/tmp/scfgmgr_socket"
li      $a2, 0x6B                       # n
move    $a0, $s1                        # dest
jalr    $t9 ; strncpy
sh      $s3, 0xB8+sockaddress_un($sp)
lw      $gp, 0xB8+saved_gp($sp)
move    $a0, $s1                        # name
la      $t9, unlink
nop
jalr    $t9 ; unlink
move    $s1, $s0
lw      $gp, 0xB8+saved_gp($sp)
move    $a0, $s2                        # fd
la      $t9, bind
move    $a1, $s0                        # addr
jalr    $t9 ; bind
li      $a2, 0x6E                       # len
lw      $gp, 0xB8+saved_gp($sp)
```

Wait... what?

- **There is an alternate option: -f that makes *scfgmgr* listen on TCP**



Let's see if it's used...

```
:~/_DGN1000_1.1.00.55_NA.img.extracted$ grep -r "scfgmgr -f" .
./squashfs-root/usr/sbin/ft_tool
```

What's this 'ft_tool'?

- **Opens a raw socket**
- **Waits for packets**
  - with ethertype = 0x8888
  - coming from the Ethernet card or broadcasted (check of the destination MAC address)
- **Packet format**

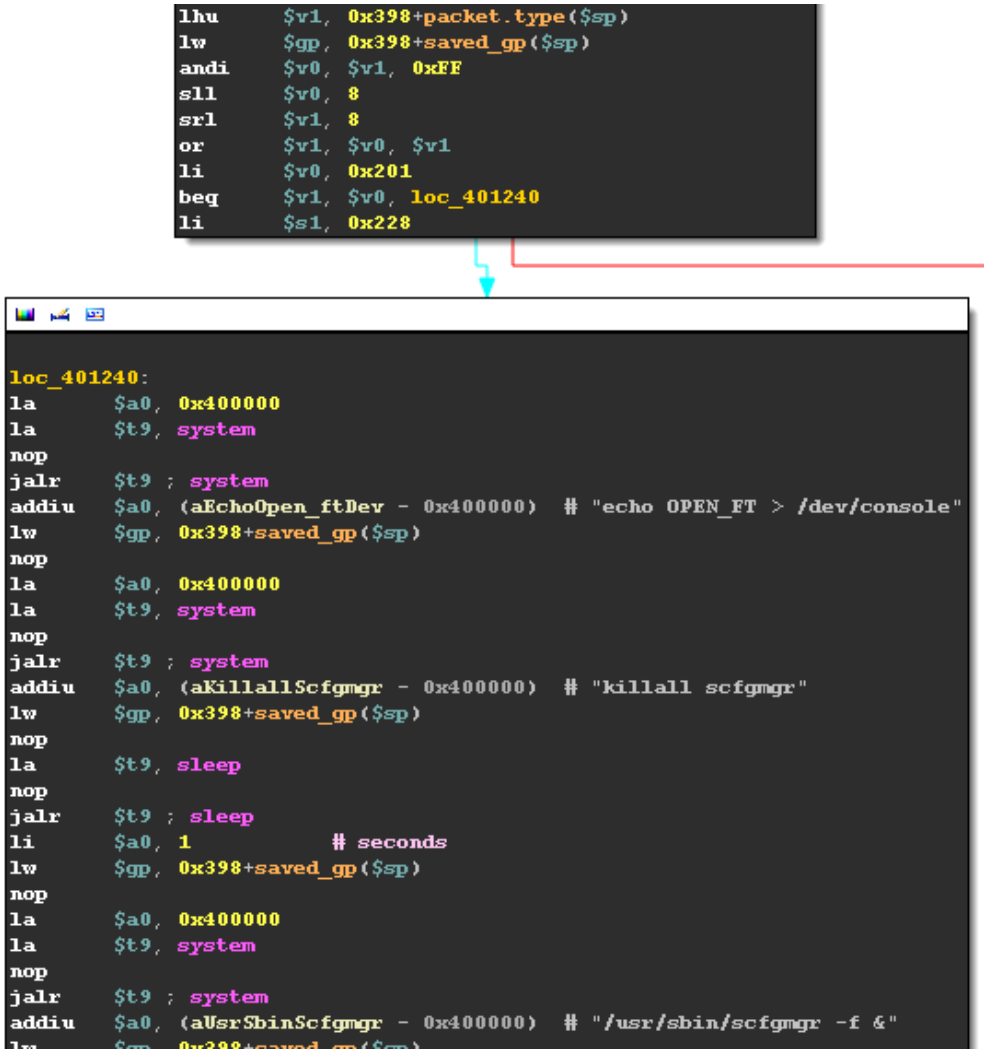
```
la      $t9, socket
li      $a0, AF_INET      # domain
li      $a1, SOCK_PACKET  # type
jalr    $t9 ; socket
li      $a2, 0x8888       # protocol
```

```
00000000 packet_struct   struc  # (sizeof=0x228)
00000000 header:         ether_header ?
0000000E type:           .half ?
00000010 sequence:       .half ?
00000012 offset:         .half ?
00000014 chunk:          .half ?
00000016 payload_len:    .half ?
00000018 payload:        .byte 528 dup(?)
00000228 packet_struct   ends
```

If payload == md5("DGN1000")...

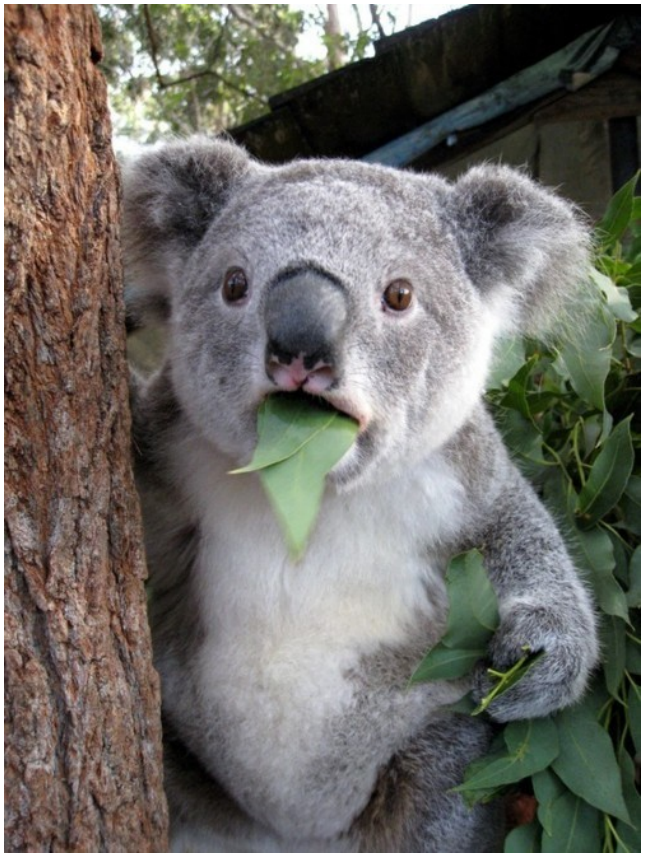
```
la      $v1, 0x400000
la      $t9, strlen
addiu   $v0, $v1, (aDgn1000 - 0x400000)  # "DGN1000"
lw      $a0, (aDgn1000+4 - 0x4028FC)($v0)
lw      $v0, (aDgn1000 - 0x400000)($v1)  # "DGN1000"
addiu   $s2, $sp, 0x398+cpy_DGN1000
sw      $a0, 0x398+cpy_DGN1000+4($sp)
sw      $v0, 0x398+cpy_DGN1000($sp)
jalr    $t9 ; strlen
move    $a0, $s2          # s
lw      $gp, 0x398+saved_gp($sp)
addiu   $s0, $sp, 0x398+md5_ctx
la      $t9, MD5Init
move    $a0, $s0
jalr    $t9 ; MD5Init
move    $s1, $v0
lw      $gp, 0x398+saved_gp($sp)
move    $a2, $s1
la      $t9, MD5Update
move    $a0, $s0
jalr    $t9 ; MD5Update
move    $a1, $s2
lw      $gp, 0x398+saved_gp($sp)
addiu   $s1, $sp, 0x398+var_88
la      $t9, MD5Final
move    $a1, $s0
jalr    $t9 ; MD5Final
move    $a0, $s1
lw      $gp, 0x398+saved_gp($sp)
addiu   $a0, $sp, 0x398+packet.payload  # s1
la      $t9, memcmp
move    $a1, $s1          # s2
jalr    $t9 ; memcmp
li      $a2, 0x10         # n
lw      $gp, 0x398+saved_gp($sp)
bnez    $v0, main_loop
addiu   $a1, $sp, 0x398+fd_set
```

And if packet type == 0x201...



```
lhu     $v1, 0x398+packet.type($sp)
lw      $gp, 0x398+saved_gp($sp)
andi    $v0, $v1, 0xFF
sll     $v0, 8
srl     $v1, 8
or      $v1, $v0, $v1
li      $v0, 0x201
beq     $v1, $v0, loc_401240
li      $s1, 0x228
```

```
loc_401240:
la      $a0, 0x400000
la      $t9, system
nop
jalr    $t9 ; system
addiu   $a0, (aEchoOpen_ftDev - 0x400000)  # "echo OPEN_FT > /dev/console"
lw      $gp, 0x398+saved_gp($sp)
nop
la      $a0, 0x400000
la      $t9, system
nop
jalr    $t9 ; system
addiu   $a0, (aKillallScfgmgr - 0x400000)  # "killall scfgmgr"
lw      $gp, 0x398+saved_gp($sp)
nop
la      $t9, sleep
nop
jalr    $t9 ; sleep
li      $a0, 1          # seconds
lw      $gp, 0x398+saved_gp($sp)
nop
la      $a0, 0x400000
la      $t9, system
nop
jalr    $t9 ; system
addiu   $a0, (aUsrSbinScfgmgr - 0x400000)  # "/usr/sbin/scfgmgr -f &"
```

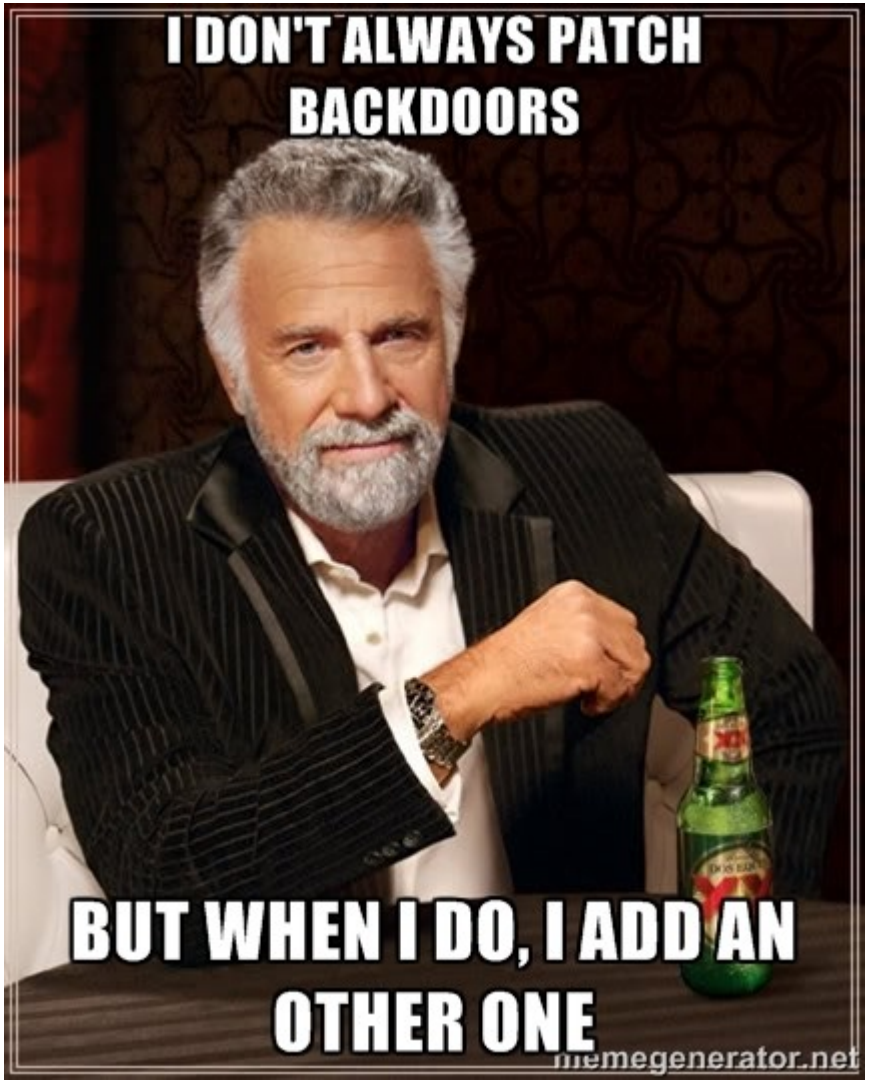


system("scfgmgr -f &")!!!

So you can reactivate the backdoor again...

- **If you're on the LAN**
- **Or if you're an Internet provider (if you're one-hop away, you can craft Ethernet headers)**
- **It's DELIBERATE**
- **You can also use the 0x200 packet type to ping the router (it will respond with its MAC address) and 0x202 to change its LAN IP address**

I don't always patch backdoors...



Because a root shell is not enough...

- **You can now (among other things) make the router LEDs flash with the 33, 34 and 35th message :)**

```
jalr    $t9 ; set_led_on
addiu   $a0, (aPower_green - 0x400000)  # "power_green"
lw      $gp, 0x10698+var_10678($sp)
nop
la      $a0, 0x400000
la      $t9, set_led_on
nop
jalr    $t9 ; set_led_on
addiu   $a0, (aPower_red - 0x400000)  # "power_red"
lw      $gp, 0x10698+var_10678($sp)
nop
la      $a0, 0x400000
la      $t9, set_led_on
nop
jalr    $t9 ; set_led_on
addiu   $a0, (aInternet_green - 0x400000)  # "internet_gre
lw      $gp, 0x10698+var_10678($sp)
nop
la      $a0, 0x400000
la      $t9, set_led_on
nop
jalr    $t9 ; set_led_on
addiu   $a0, (aInternet_red - 0x400000)  # "internet_red"
lw      $gp, 0x10698+var_10678($sp)
```

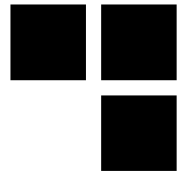
```
led_off:                # jumptable 00401284 case 34
la      $a0, 0x400000
la      $t9, set_led_off
li      $v0, 1
sw      $v0, (console_mode - 0x10000030)($s6)
jalr    $t9 ; set_led_off
addiu   $a0, (aPower_green - 0x400000)  # "power_green"
lw      $gp, 0x10698+var_10678($sp)
nop
la      $a0, 0x400000
la      $t9, set_led_off
nop
jalr    $t9 ; set_led_off
addiu   $a0, (aPower_red - 0x400000)  # "power_red"
lw      $gp, 0x10698+var_10678($sp)
nop
la      $a0, 0x400000
la      $t9, set_led_off
nop
jalr    $t9 ; set_led_off
addiu   $a0, (aInternet_green - 0x400000)  # "internet_green"
lw      $gp, 0x10698+var_10678($sp)
nop
la      $a0, 0x400000
la      $t9, set_led_off
nop
jalr    $t9 ; set_led_off
addiu   $a0, (aInternet_red - 0x400000)  # "internet_red"
```

```
addiu   $a0, (aPower_red - 0x400000)  # "power_red"
li      $a2, 0xFFFFFFFF
li      $a3, 5
jalr    $t9 ; set_led_blink
sw      $s0, 0x10698+var_10688($sp)
lw      $gp, 0x10698+var_10678($sp)
li      $a1, 1
la      $a0, 0x400000
la      $t9, set_led_blink
addiu   $a0, (aInternet_green - 0x400000)  # "internet_green"
li      $a2, 0xFFFFFFFF
li      $a3, 5
jalr    $t9 ; set_led_blink
sw      $s0, 0x10698+var_10688($sp)
lw      $gp, 0x10698+var_10678($sp)
li      $a1, 1
la      $a0, 0x400000
la      $t9, set_led_blink
addiu   $a0, (aInternet_red - 0x400000)  # "internet_red"
li      $a2, 0xFFFFFFFF
li      $a3, 5
jalr    $t9 ; set_led_blink
sw      $s0, 0x10698+var_10688($sp)
lw      $gp, 0x10698+var_10678($sp)
li      $a1, 1
la      $a0, 0x400000
la      $t9, set_led_blink
addiu   $a0, (aDsl - 0x400000)  # "dsl"
li      $a2, 0xFFFFFFFF
li      $a3, 5
jalr    $t9 ; set_led_blink
sw      $s0, 0x10698+var_10688($sp)
lw      $gp, 0x10698+var_10678($sp)
li      $a1, 1
la      $a0, 0x400000
la      $t9, set_led_blink
addiu   $a0, (aUsb - 0x400000)  # "usb"
```

But where does it come from?

- **The 0x8888 ethertype and packet structure is used in an old Sercomm update tool:**
  http://wiki.openwrt.org/_media/toh/netgear/dg834.g.v4/nftp.c
  - lazy guys, they didn't even code their new *backdoor* from scratch ;)
- **It may be present in other hardware but hard to tell:**
  - No easy way to scan
  - MD5 signature will certainly be different as it's based on the router commercial name

How to detect it?



- **For DGN1000, simply use the PoC from your LAN**

- **For other routers, the simplest way is to:**
  - Use 'binwalk -e' to extract the file system
  - Search for 'ft_tool' or grep -r 'scfgmgr -f'
  - Use IDA to confirm

We hope you enjoyed this presentation :)

- **PoC is available here:**

  http://synacktiv.com/ressources/ethercomm.c

SYNACKTIV
DIGITAL SECURITY