

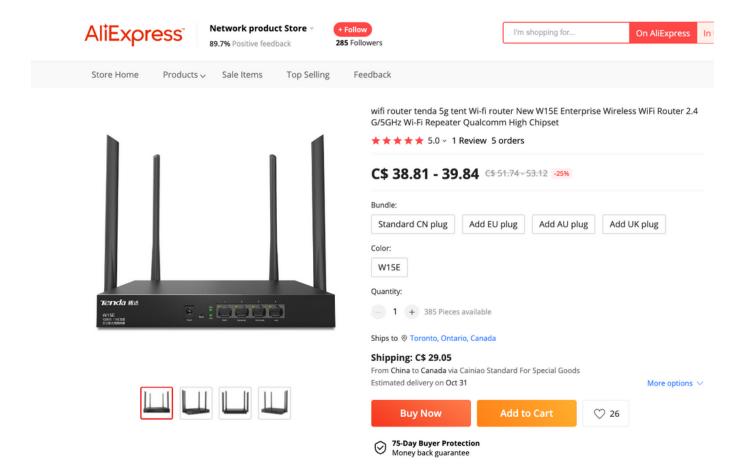
**Reverse Engineering** 

# Vulnerabilities in Tenda's W15Ev2 AC1200 Router

Reverse engineering Tenda's W15Ev2 AC1200 SOHO router discovering 10 0-days (10 CVEs)



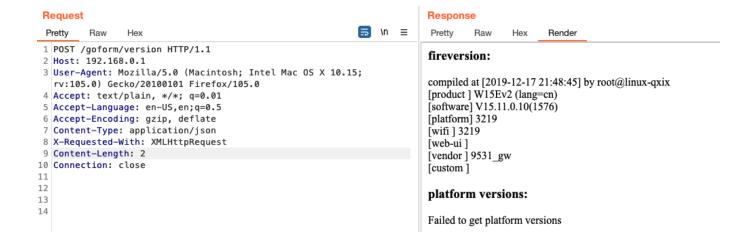
Lately, after work, I've really enjoyed hacking and reverse engineering funky IoT devices. In early May of 2022, I went on a little Amazon/AliExpress shopping spree and bought ~15 cheap IoT devices. Among them was this compact SOHO router by Tenda.



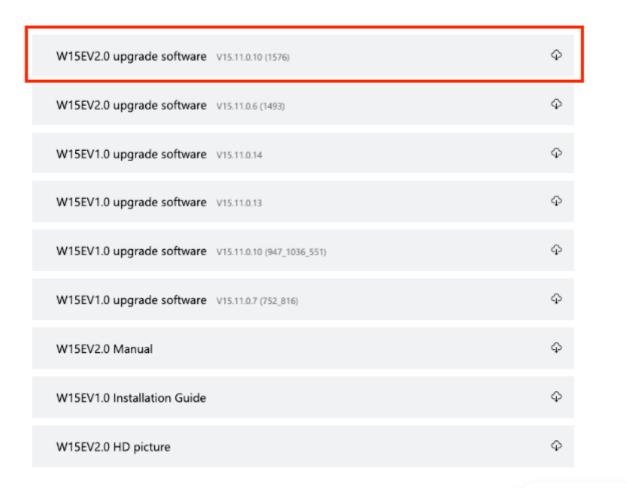
Shenzhen Tenda Technology Co., Ltd. hereafter referred to as Tenda is one of the leading suppliers of networking devices and equipment. Tenda has committed to delivering easy-to-install and affordable networking solutions, offering innovative, cutting-edge products to realize people's intelligent life. They are a Chinese company based in Shenzhen, China with R&D centers, in Shenzhen and Chengdu. Reportedly with over ~1000 employees they're fairly well established in the industry and offer a wide variety of products. The company's official website is <a href="https://www.tendacn.com">www.tendacn.com</a>.

The router I purchased was the AC1200 SOHO router model <u>W15Ev2</u> with the <u>firmware V15.11.0.10(1576)</u>. The chip model for the device is <u>qca9531</u> (qca9531\_wifi) by Qualcomm which is a 2.4 GHz System-on-a-Chip (SoC) that leverages a MIPS 24Kc processor.

Performing a GET request on /goform/version or /common/macro\_config.js identifies the firmware version. Out-of-the-box mine was CONFIG\_FIP Subscribe



Going to Tenda's firmware <u>downloads</u> page confirms we're on their latest Chinese deployment. Due to the seeming lack of interest, I will only be testing the V15.11.0.10(1576) firmware.



Multiple attempts were made through numerous communication channels to contact this vendor, but all failed. I try my best to act in a transparent, responsible, and consistent manner, with the goal of building a safer online space. In order to achieve this, I adhere to a 90-day time frame from the initial contact to public disclosure. If you have any ethical dilemmas or moral qualms as a result of this publication, I urge you to read the section entitled "Thoughts On Responsible Disclosure" situated towards the end of the blog to better understand the events, my perspective, and reasons for disclosure.

#### **Disclosure Timeline**

July 18, 2022: Vulnerability discovered and first reported

**August 6, 2022**: Second attempt to make contact, further informing the vendor of the severity of the vulnerability

August 13, 2022: WeChat & QQ attempt to contact the vendor

September 2, 2022: A Third attempt to contact the vendor

September 13, 2022: File for CVE assignment through Mitre.org

September 19, 2022: File for additional CVE assignment through Mitre.org

September 26, 2022: File for additional CVE assignment through Mitre.org

**September 26, 2022:** Forth attempt to contact vendor, informing them of assigned 5 CVEs

**October 3, 2022**: Fifth attempt to make contact, further informing vendors of the severity & number of vulnerabilities

October 10, 2022: Sixth attempt to contact the vendor

**October 17, 2022:** Final contact attempted & warning of public disclosure & confirming 10 CVEs had been assigned to their product

October 19, 2022: Public disclosure

#### **Vulnerabilities List**

A total of 11 vulnerabilities were identified in Tenda's AC1200 W15Ev2 router and IoT cloud gateway peripheral:

Subscribe

```
    Improper Authorization (CVE-2022-40843)
    Password Disclosure (CVE-2022-40845)
    Improper Access Control
    OS Command Injection #1 (CVE-2022-40847)
    OS Command Injection #2 (CVE-2022-41396)
    OS Command Injection #3 (CVE-2022-41395)
    OS Command Injection #4 (CVE-2022-42053)
    Stack-based Buffer Overflow #1 (CVE-2022-42058)
    Stack-based Buffer Overflow #2 (CVE-2022-42060)
    Stored XSS in Website Filtering (CVE-2022-40844)
    Stored XSS in Online Device Names (CVE-2022-40846)
```

# **Web Application**

#### **Improper Authorization**

The Tenda AC1200 V-W15Ev2 router is affected by improper authorization/improper session management. The software does not perform or incorrectly perform an authorization check when a user attempts to access a resource or perform an action. This allows the router's login page to be bypassed. The improper validation of user sessions/authorization can lead to unauthenticated attackers having the ability to read the router's <code>syslog.log</code> file, which contains the MD5 password of the Administrator's user account. This vulnerability exists within the local web and hosted remote management console. The vulnerability affects version V15.11.0.10(1576).

## **Vulnerability Details**

```
CVE ID: CVE-2022-40843

Access Vector: Remote/Local
Security Risk: Critical
Vulnerability: CWE-287 / CWE-285 / CWE-228

CVSS Base Score: 9.9

CVSS Vector: CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:L/A:L
```

This vulnerability is present locally out-of-the-box without any user/configuration and remotely once an administrator user has enabled the

Management" configuration as seen below.
Enabling this setting would allow the router's owner to manage the d anywhere in the world via the provided public URL. Enabling this configuration

arostas a narak tunnal to a prodofined and the tondom not subdomain

creates a <u>ligion</u> tuiller to a predefined | couta.tendach.net | subdomain.



Once enable the router can be accessed through the public web as seen below.

If you're wondering how many of these devices are configured like that thankfully not many. With some inefficient Google dorking

"\*.cloud.tendacn.net:8080" I was able to find [~18] devices configured with the cloud gateway. There is no way of ethically knowing what these devices are.

#### DISCLAIMER

If you go hunting for these hosts with any malice and start testing what you see below, I'd consider your activities as active intrusion. Leveraging vulnerabilities against hosts you do not own is NOT something I endorse & is unethical. Subsequently, all traffic that passed through 6nyz4sf4.cloud.tendacn.net did not target anything other than my own device. I was not directly attacking or affecting the availability & integrity of the public/reserved gateway.

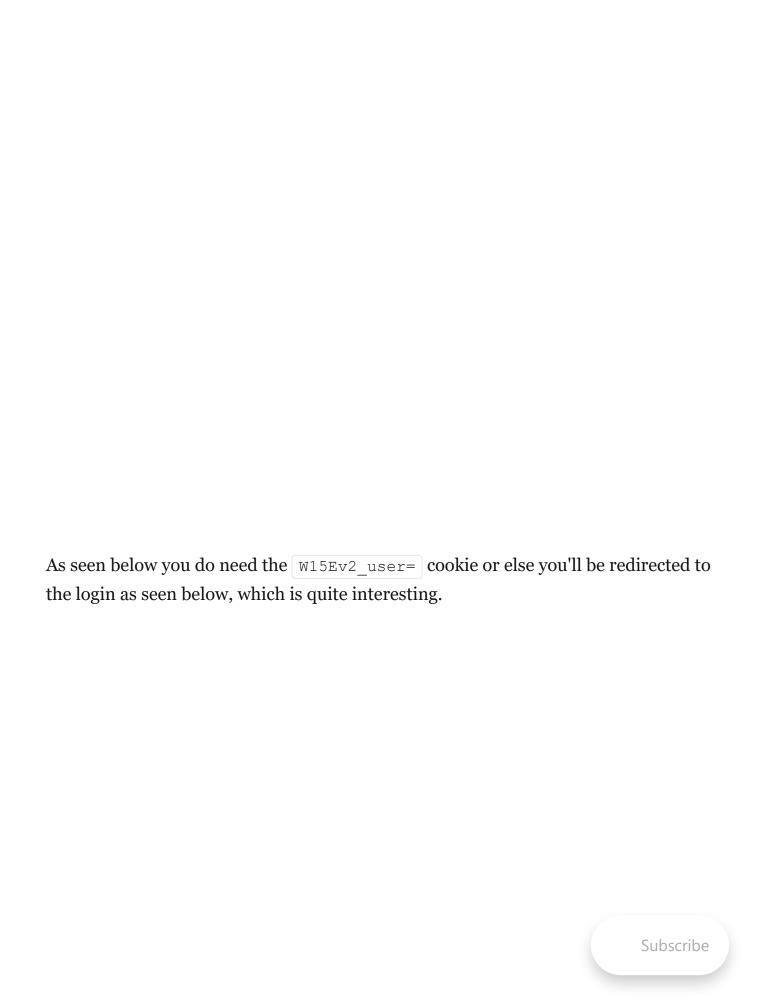
\*.cloud.tendacn.net is only a public-facing tunnel to the router I own. Traffic passing through this public-facing proxy is simply passed as regular traffic/requests. Additionally, I never attempted or performed any reconnaissance or automated tests against cloud.tendacn.net.

The vulnerability in & of itself stems from a complete lack of proper session token generation & the presence of authentication levels that should be expected from a cookie. The following images will help demonstrate this.

After logging into the device, the user is provided with the following cookies.

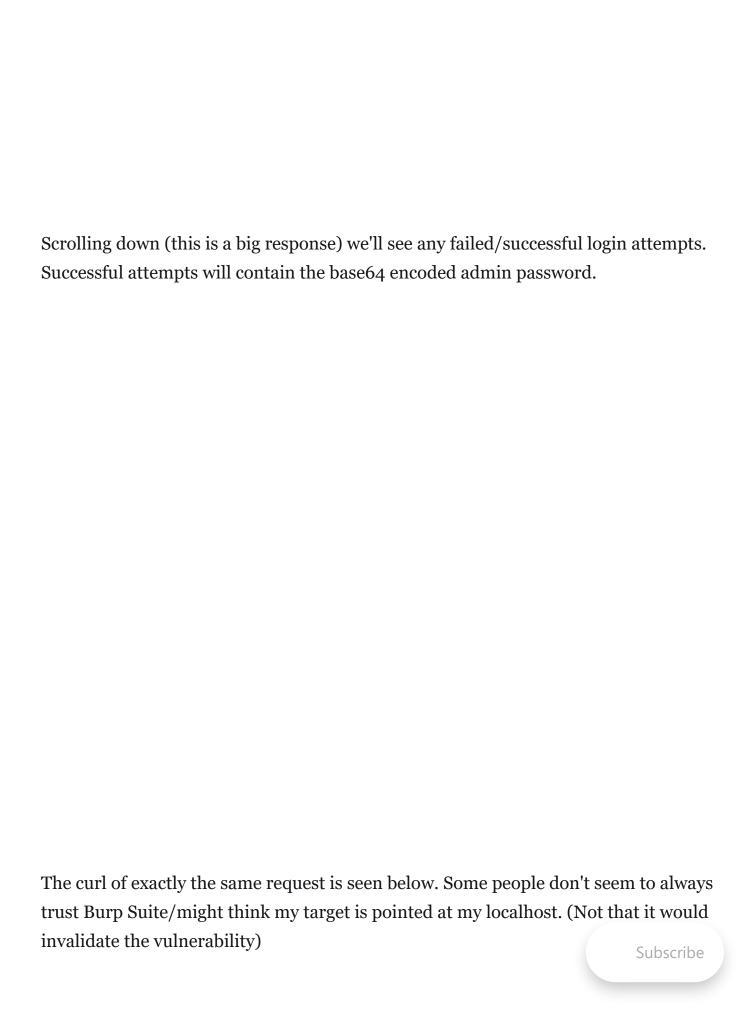
```
Cookie: bLanguage=cn; _:USERNAME:_=admin; W15Ev2_user=admin
```

However, both the blanguage=cn; and the \_:USERNAME:=admin; cookies are not required. Additionally W15Ev2\_user= can simply be left blank.



This obviously isn't good. As seen in 2 images above, I was able to fetch the /syslog.log which should only be viewed/accessible once authenticated.

This endpoint can leak the admin login event containing the base64 password of the administrator user. This can easily be leveraged to gain access to the device. As seen
below we have the ability to perform this both locally or remotely via our assigned
Remote Web Management endpoint. To demo, we'll perform a GET request on
/goform/downloadSyslog/syslog.log with the cookie set to W15Ev2_user= .
Subscribe



If you take the time to decompile the <a href="httpd">httpd</a> binary the culprit is the <a href="httpd">authSecurityHandler</a> function that I won't be going into here & will leave as an exercise to the reader.

You might be thinking "yeah, but who cares"? With access to the router, you're able to perform internal network scans, obtain the wifi's PSK, set up your own port forwarding, etc,. Other attacks can be chained with this "improper session management" to perform command injections and obtain a shell on the device. Although unlikely, it would suck to see a small-medium-sized company use one of these devices as a random & cheap AP/Switch for a small cluster of employees, thinking it was inherently safe.

### **Password Exposure**

The Tenda AC1200 V-W15Ev2 router is affected by a password exposure vulnerability. When combined with the improper authorization/improper session management vulnerability, an attacker with access to the router may be able to expose sensitive information which they're not explicitly authorized to have. This is performed via a direct request to <code>/cgi-bin/DownloadCfg/RouterCfm.cfg</code> without authorization. This vulnerability exists within the local web interface remote web management console. An attacker would be able to retriev passwords via this technique. The vulnerability affects version V15.11.0.10(1576).

# **Vulnerability Details**

```
CVE ID: CVE-2022-40845
Access Vector: Remote/Local
Security Risk: High
Vulnerability: CWE-200
CVSS Base Score: 8.8
CVSS Vector: CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H
```

Like the previous vulnerability, this can be performed both locally or remotely through the cloud.tendacn.net proxy tunnel. Generally, downloading backups could only be performed once authenticated.

However, there isn't a need to have the device initiated to perform th' backup file containing hashes of users' passwords, PSK's, etc. can eash, downloaded. This is generally considered bad hygiene/practice. Realistically, the

impact on organizations depends on many factors that are unique to each organization. You can decide this for yourself when presented with the information a little further down. I was a little confused at first, I didn't think this router had a CGI interface. Taking a step back, I dug up this endpoint when scouring through the firmware dump. It's a weird subset that is mainly defined to make executables on the server respond to web requests. For all three requests seen in the images below this is more-or-less the action's performed in the backend. Here is the attack being performed locally with Burp Suite.

Remotely with Burp Suite. Remotely with the curl command.

The crucial information obtained is listed below.

```
sys.rzadmin.username= rzadmin
sys.rzadmin.password= cnphZG1pbg== (rzadmin)
sys.rzadmin.password.decoded= r***n
wlan0.0_bss_wpapsk_key= this_is_my_super_secret_ssid_password
sys.admin.username= admin
sys.admin.password.decoded= t***d
sys.admin.password= base64 PSK
usb.share.gst0.user= guest
usb.share.gst0.pwd= guest
sys.guest.password.decoded= g***t
sys.guest.password= Z3Vlc3Q= (guest)
```

#### **Improper Access Control**

The Tenda AC1200 V-W15Ev2 router is affected by improper access control. An attacker can start telnet without authorization by performing a <code>GET</code> request to the <code>/goform/telnet</code> endpoint, spawning the <code>telnetd</code> service on the device. This service is password protected; however, the default account credentials were identified as <code>root:Fireitup</code> which can be used to successfully login. The vulnerability affects version V15.11.0.10(1576).

## **Vulnerability Details**

```
CVE ID: N/A (Already assigned to CVE-2018-5770)

Access Vector: Remote/Local
Security Risk: High

Vulnerability: CWE-287 / CWE-284

CVSS Base Score: 8.4

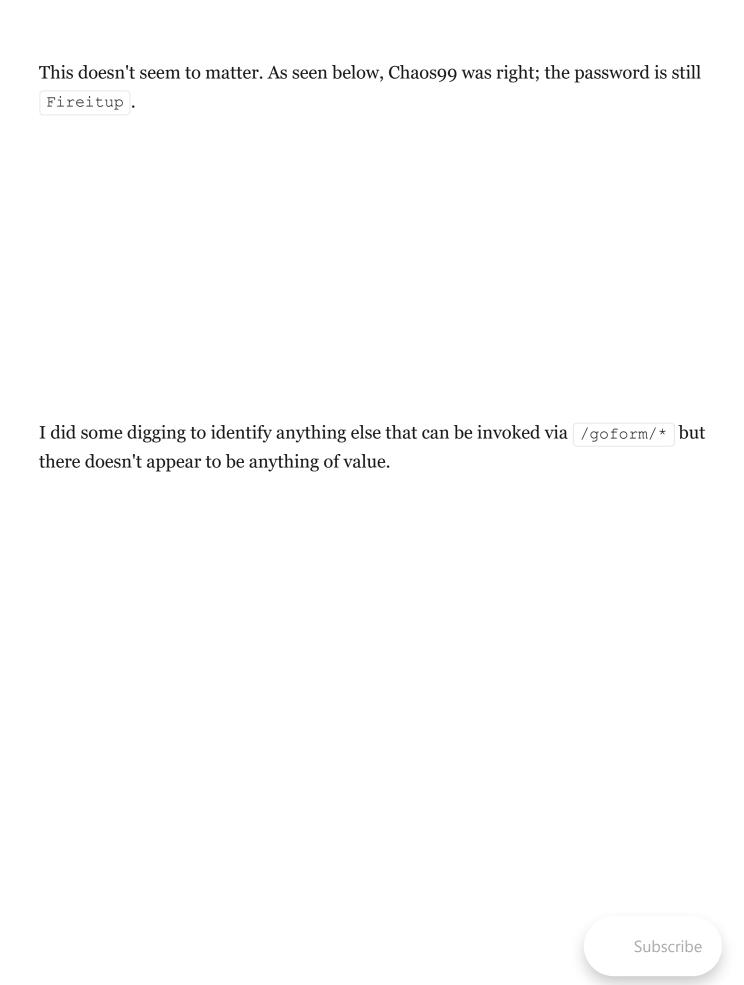
CVSS Vector: CVSS:3.1/AV:L/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H

Subscribe
```

There's not much to explain here. As seen below the device doesn't ship with telnet



As seen below telnet is now up and accessible.	
I did some digging to identify the telnet password. The earliest mentidates back to 2017 from this forum by user Chaos99. This was for the	
got an AC1200.	
	Subscribe



# **OS Command Injection #1**

The Tenda AC1200 V-W15Ev2 router is affected by an OS Command Injection vulnerability that allows an authenticated attacker to run arbitrary commands on the affected system as the application user. This vulnerability exists within the ping function <code>formSetFixTools</code>. This is the result of improper input validation passed through the <code>hostName</code> parameter. This vulnerability exists within the local web interface and the hosted remote web management console. The vulnerability affects version V15.11.0.10(1576).

## **Vulnerability Details**

```
CVE ID: CVE-2022-40847

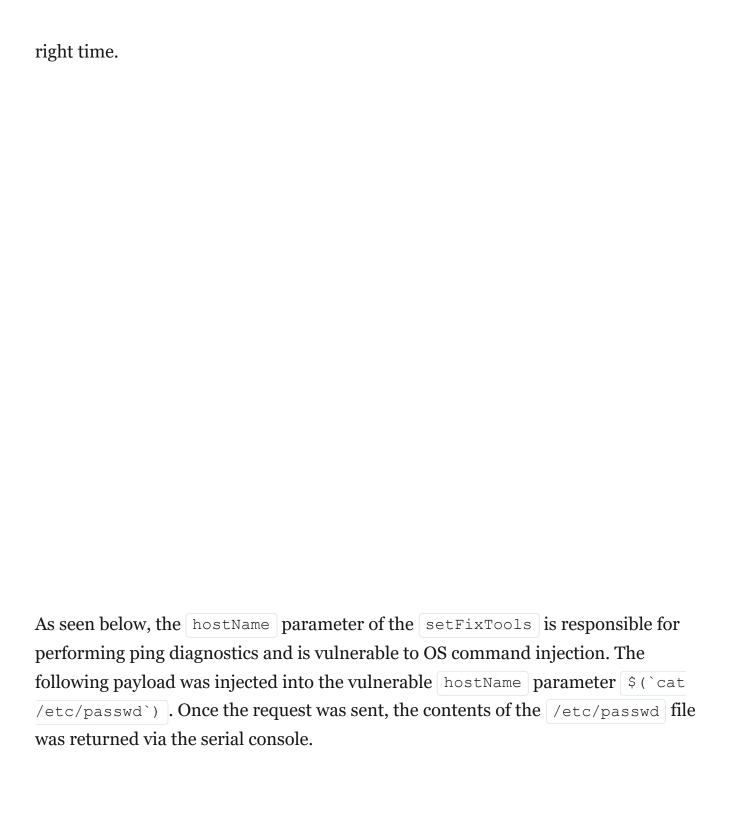
Access Vector: Remote/Local
Security Risk: High
Vulnerability: CWE-78

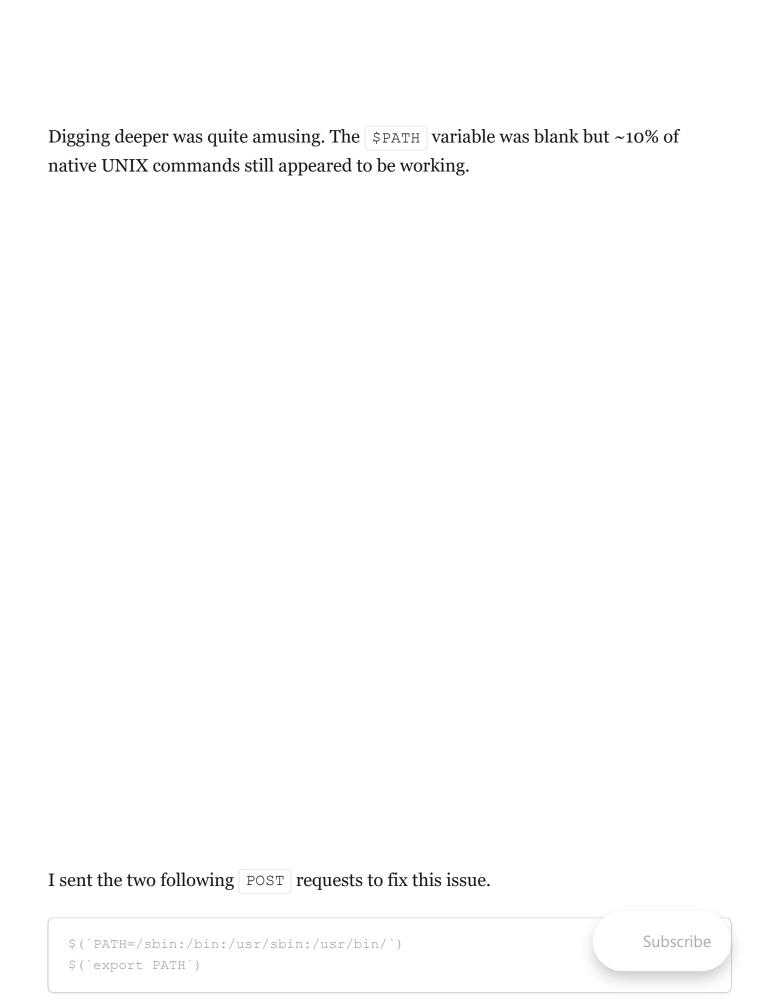
CVSS Base Score: 7.8

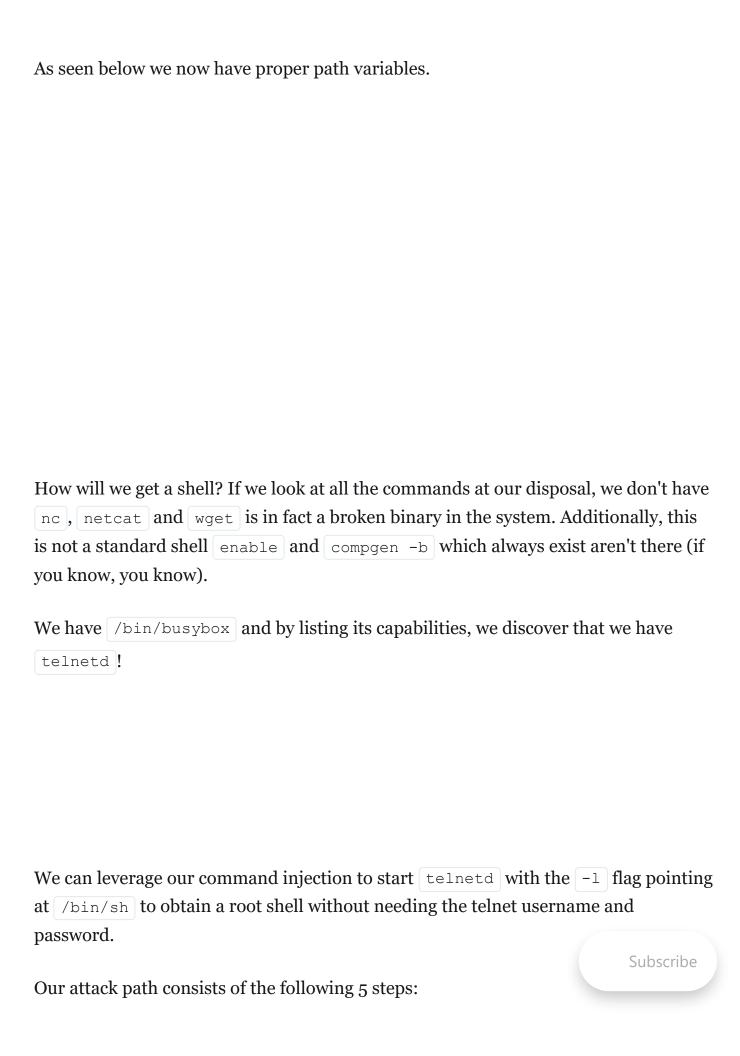
CVSS Vector: CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H
```

This command injection vulnerability was discovered through the sering to the router's UART 1

If you're wondering why the serial interface is sending out all of our logs and actions, I'm not 100% sure. I'm assuming it's so that application-specific errors can be sent to the log using some API methods. Without this serial logging, I'm not sure I would've discovered this command injection vulnerability.
I somewhat got lucky and have a methodology that leads to these discoveries. I initially fizzed the hostName parameter with Burp Suite intruder us; custom and common command injections mainly found in this <u>GitHu</u> just so happened that I was paying attention to my serial connection terminal at the







```
Step #1 - identify telnet pid - we can't use `|` or `&&` or `;`
Payload 1: 'ps >/tmp/a'

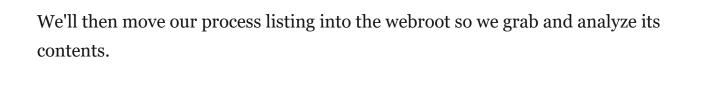
Step #2 - move it to webroot so we can get it
Payload 2: 'cp /tmp/a /var/webroot'

Step #3 - kill telnetd if its alive
Payload 3: 'kill -9 PID'

Step #4 - copy sh executable in pwd
Payload 4: 'cp /bin/sh .'

Step #5 - restart telnetd binded on sh
Payload 5: 'telnetd -1 sh'
```

First, we'll pipe the contents of the systems processes into a file in /tmp.



Go to 192.168.0.1/a to retrieve the file.

Telnet is currently running with a PID of 3718. However, this telnet is no good because we need to know the credentials for it. So we'll kill this process.

We'll then move the sh binary to our current working directory.

OS Command Injection #2	Subscribe
Tada.	
Now we should just be able to telnet without requiring a password.	
iogin/ connection attempt.	

The Tenda AC1200 V-W15Ev2 router is affected by an OS Command Injection

vulnerability that allows authenticated attackers to run arbitrary commands on the affected system as the application's user. This vulnerability exists within the IPsec router configuration, specifically via the <code>/goform/module</code> function <code>setIPsecTunnelList</code>. This vulnerability in part requires the router's VPN server to be enabled (which it is by default but can be disabled). At its core, the <code>httpd</code> binary contains the vulnerability and can occur when a malicious <code>formSetIpSecTunnel</code> parameter gets processed. During this process, inner process communication is established through the <code>netctrl</code> binary. This vulnerability is the result of improper input validation passed through the <code>IPsecLocalNet</code> or <code>IPsecRemoteNet</code> parameter. This vulnerability exists within the local web interface and the hosted remote web management console. The vulnerability affects version V15.11.0.10(1576).

#### **Vulnerability Details**

```
CVE ID: CVE-2022-41396

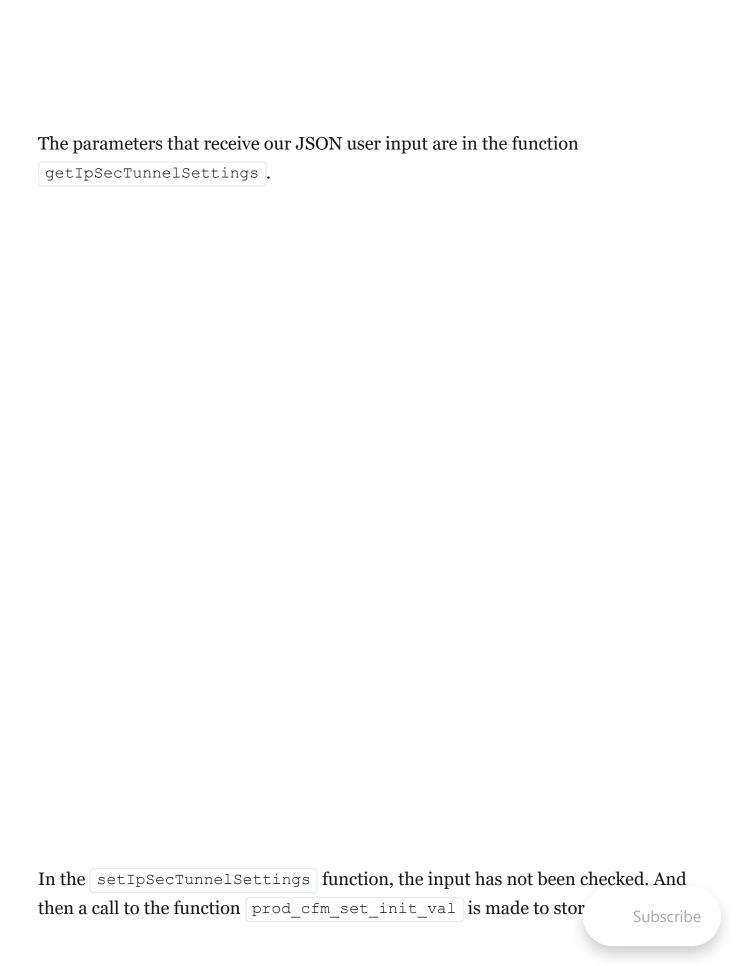
Access Vector: Remote/Local
Security Risk: High
Vulnerability: CWE-78

CVSS Base Score: 7.8

CVSS Vector: CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H
```

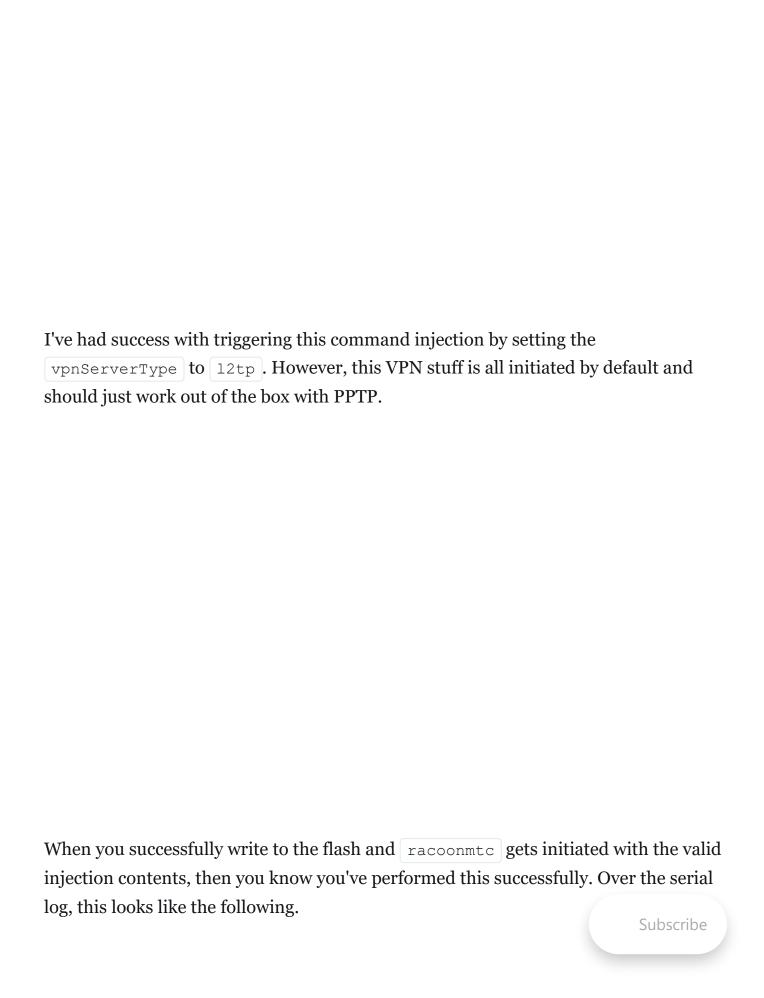
In the formSetIpSecTunnel function, both the IPsecLocalNet and IPsecRemoteNet parameters are passed directly to the application, which can be manipulated by the user. The Local (IPsecLocalNet) and Remote (IPsecRemoteNet) network parameters can be maliciously injected into.

To understand what's going on, we need to reverse the application a little. This vulnerability occurs in the setIpSecTunnel function. Below, we enter the function getIpSecTunnelSettings and the rabbit hole begins.



These are the function parameters passed through the <code>POST</code> command. But what's actually happening in <code>setIpSecTunnelSetting</code>? It isn't immediately apparent, but at the end of this decompilation there's a pretty significant block of code related to, and required for inter-process communication. This all happens in another binary that handles the passed input, leveraging the contents through other functions that write this new data and changes to the flash. Therefore, to see how we can inject, we need to look at that binary. After a little digging, it was discovered that the vulnerability would be triggered in <code>netctrl</code>.

If we want our <code>IPsecLocalNet</code> or <code>IPsecRemoteNet</code> input to be extracted, we have to have one of the following protocols initiated.

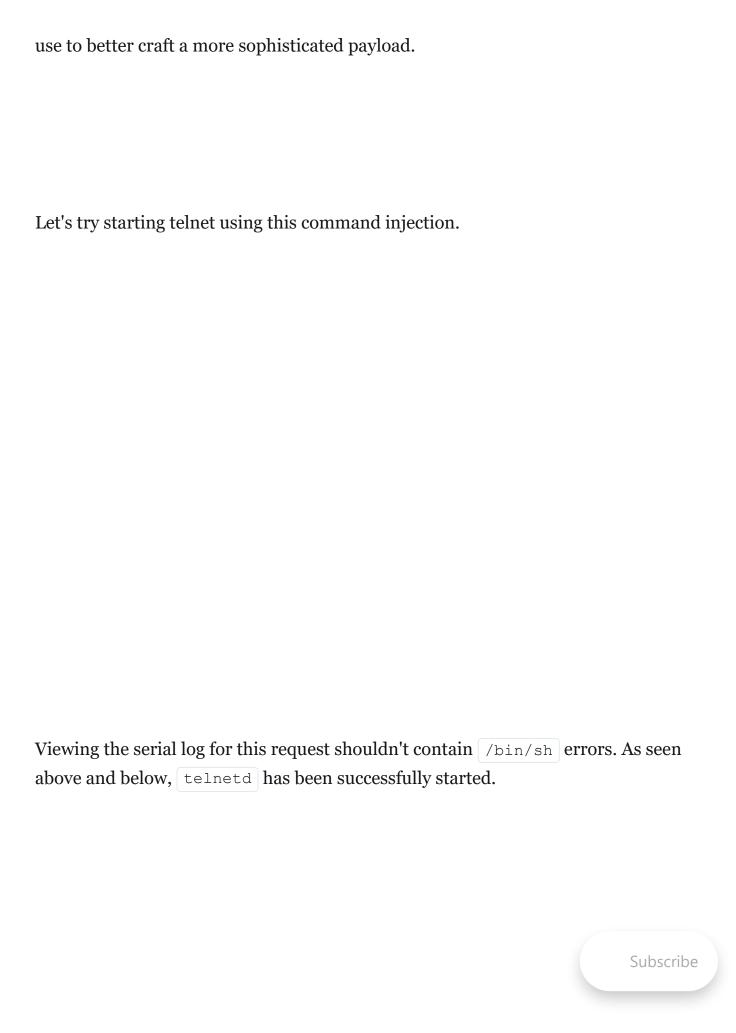


Inside the <code>netctrl</code> binary, looking at the <code>vpn\_ipsec\_ini</code> function, the initial input will be extracted and injected into <code>%s</code>, which is how we will achieve our code execution. We can further use command separators to escape the <code>iptabes</code> command.

Let's visualize this by moving to a GUI. The front-end vulnerable input is tagged as "Remote Internet" and "Local Internet" which maps to IPsecLocalNet and IPsecRemoteNet respectively.

To test that we have command injection let's enter a random string and see attempt to execute it.	/bin/sh
	Subscribe







#### **OS Command Injection #3**

The Tenda AC1200 V-W15Ev2 router is affected by an OS Command Injection vulnerability that allows authenticated attackers to run arbitrary commands on the affected system as the application's user. This vulnerability exists within the DMZ router configuration, specifically via the <code>/goform/module</code> function <code>setDMZ</code>. The vulnerability was identified in the <code>netctrl</code> binary tracing preprocessing calls from the <code>httpd</code> binary. This vulnerability is the result of improper input validation passed through the <code>dmzHost</code> parameter. This vulnerability exists within the local

web interface and the hosted remote web management console. The vulnerability affects version V15.11.0.10(1576).

### **Vulnerability Details**

```
CVE ID: CVE-2022-41395

Access Vector: Remote/Adjacent
Security Risk: High
Vulnerability: CWE-78

CVSS Base Score: 7.8

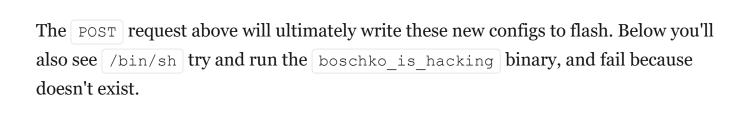
CVSS Vector: CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H
```

Continuing the investigation into <code>netctrl</code>, the culprit for the injection above, I ended up finding more injections.

In the <code>formSetDMZ</code> function, <code>dmzHost</code> can be directly passed by the attacker. If we can control the <code>dmzHost</code> input, presumably we can attack the system. Looking into the <code>httpd</code> binary seen below, the JSON-provided input doesn't get checked or sanitized. After that, it calls the function <code>prod\_cfm\_set\_value</code> to s' Subscribe So the parameter <code>dmzIp</code> is what we're looking to inject into it.

The netctrl binary advance\_set\_dmz\_cfg function's input will grab the provided input passed to it by httpd.

Eventually, in the <code>advance\_set\_dmz\_cfg</code> function, this unchecked input could be leveraged to perform command injection when passed through <code>%s</code>. Looking at this in Burp Suite, we'll modify our injectable <code>dmzHost</code> parameter. We're testing with fake binary to see what gets acted on it/the errors we receive.



This is great information, we more or less know we should be able to spawn telnetd. As seen below telnet wasn't running a few seconds prior and then was. Disregard the date I'm too lazy to go change it.

#### **OS Command Injection #4**

The Tenda AC1200 V-W15Ev2 router is affected by an OS Command Injection vulnerability that allows authenticated attackers to run arbitrary commands on the affected system as the application's user. This vulnerability exists within the Port Mapping router configuration, specifically via the <code>/goform/module</code> function <code>setPortMapping</code>. The vulnerability was identified in the <code>netctrl</code> binary tracing preprocessing calls from the <code>httpd</code> binary. This vulnerability is the result of improper input validation passed through the <code>portMappingServer</code> parameter. This vulnerability exists within the local web interface and the hosted remote web management console. The vulnerability affects version V15.11.0.10(15

Subscribe

```
CVE ID: CVE-2022-42053
Access Vector: Remote/Local
Security Risk: High
Vulnerability: CWE-78
CVSS Base Score: 7.8
CVSS Vector: CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H
```

At this point, you get the idea. I just kept looking at the <code>netctrl</code> binary and stumbled onto the <code>advance\_set\_port\_map\_cfg</code> function which looks a lot like our previous command injections.

The port mapping front-end looks like this, it's nothing crazy, so let's have a look at this request in Burp Suite.
We're going to skip reversing the preprocessing from <code>httpd</code> to figure out which parameters are getting passed. We can use our better judgment to make an educated
guess that the portMappingServer might be injectable.
Subscribe

In the image above we're starting the <code>telnetd</code> , which was previseconds before. Looking at the serial log, we're presented with far messages.	
	Subscribe

Fun little fact. If we didn't own the hardware and were emulating <code>httpd</code> off of qemu it would be tedious work to find the proper web request and would require properly reversing the <code>httpd</code> binary. However, I actually found this enterprise
documentation which perhaps unintentionally leaks the router functions?
Regardless, this is nice to have.

#### **Stack Buffer Overflow #1**

The Tenda AC1200 V-W15Ev2 router is affected by a stack-based buf Subscribe vulnerability that allows an unauthenticated attacker to run arbitrary the affected system or cause a denial of service. This vulnerability exists within the

"Remote WEB Management" router application, specifically via the <code>/goform/module</code> function <code>setRemoteWebManage</code>. The vulnerability was identified in the <code>netctrl</code> binary tracing preprocessing calls from the <code>httpd</code> binary via the vulnerable <code>remoteIP</code> parameter. This vulnerability is the result of mismanaged memory allocation. This vulnerability exists within the local web interface and the hosted remote web management console. The vulnerability affects version V15.11.0.10(1576).

#### **Vulnerability Details**

```
CVE ID: CVE-2022-42058

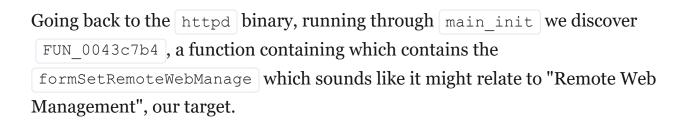
Access Vector: Remote/Local
Security Risk: Medium

Vulnerability: CWE-121

CVSS Base Score: 5.3

CVSS Vector: CVSS:3.1/AV:N/AC:H/PR:L/UI:N/S:U/C:N/I:N/A:H
```

The entire remote web management gateway piqued my interest early on. I decided to dig into it thinking that I might find a command injection vulnerability.

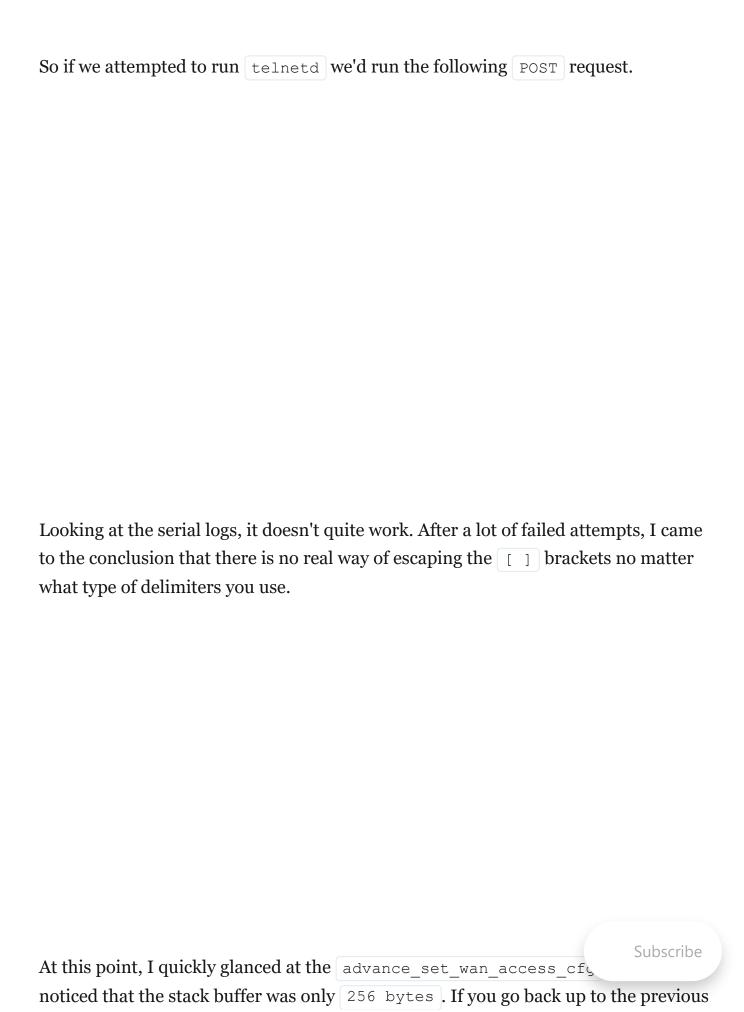


As seen below, in the <code>formSetSafeWanWebMan</code> function there exists the <code>remoteIp</code> parameter which is passing whatever is supplied. At first, I thought that this was a good start, if we can control the <code>remoteIp</code> value there's a potential to subscribe command injection.

As you can see above, the input isn't getting checked. After which a call to the function SetValue stores the input into wans.wanwebip. Let's figure out how this is all handled in the netctrl binary.

After a bit of digging, I discovered that the <code>[advance\_get\_wan\_access\_cfg]</code> function is where the input will be extracted.

After this, the <code>advance\_set\_wan\_access\_cfg</code> will use the input to write the contents of the <code>POST</code> request to the flash. As you can see below the input from <code>remoteIp</code> should be passed into <code>%s</code> copying our command to the stack buffer.



screenshot you'll notice that the function doesn't check the value of iVar2 and simply calls it, which copies the POST parameter remoteIp to stack buffer acStack520.

We should be able to crash this. I ended up overwriting the buffer with ~2000 characters the image below is 629 characters and intended to be more legible to the reader.

Subscribe

This causes a crash as seen below, the banner \*\*\*\*\* WeLoveLinux

displayed every time the device boots	up, and crash	ed immediately	after.	
				Subscribe

m
l rd.
1

As you can see above, we have a little problem with <code>cfms\_mib\_proc\_handle</code>. When running the program directly, it will output WeLoveLinux, and then it will crash/not initiate. I'm guessing that it has entered a state of suspended animation.

I'll use IDA for this next part because if I have to patch it I don't know how to do it with Ghidra. After a bit of digging to find the position where the string is used and set a breakpoint, then stepping through until something weird happens. I'm speculating that the culprit is this function, which is connected to a service that should return 1 under normal circumstances.

Subscribe

connect to the relevant service. We need to patch manually. I did this in IDA using the <u>keypatch</u> plugin to patch the return value to 1.

However, even with this patch set, it's not working. I'm almost 100% sure it should.

Regardless, although upsetting, I decided not to peruse this any further.

#### Stack Buffer Overflow #2

The Tenda AC1200 V-W15Ev2 router is affected by a stack-based buffer overflow vulnerability that allows an unauthenticated attacker to run arbitrary commands on the affected system or cause a denial of service. This vulnerability exists within the Internet Settings, Interface Types application. It's leveraged via the <code>/goform/module</code> function <code>setNetwork</code>. This vulnerability is the result of mismanaged memory allocation. The vulnerable parameters are <code>wanUser</code> and <code>wanPwd</code>. This vulnerability exists within the local web interface and the hosted remote web management console. The vulnerability affects version V15.11.0.10(1576).

### **Vulnerability Details**

```
CVE ID: CVE-2022-42060

Access Vector: Remote/Local
Security Risk: Medium

Vulnerability: CWE-121

CVSS Base Score: 5.3

CVSS Vector: CVSS:3.1/AV:N/AC:H/PR:L/UI:N/S:U/C:N/I:N/A:H
```

I'd like to start by saying that this overflow was quite tricky to understand given that I'm doing this statically. It would be much easier to debug this dynamically in my QEMU + GDB lab. As seen in the previous overflow, I wasn't "properly" patching ConnectCfm, so I'm stuck with this suboptimal method.

This stack buffer overflow is part of the Broadband Dialup configuration, which is part of the WAN device configuration. Point-to-Point Protocol over Ethernet (PPPoE) is a type of broadband connection and therefore is referenced as such. Initially, I thought the overflow stemmed from the screenshot below. From the <code>setWanPppoe</code> function, where the application reads user input into <code>uVar1</code> and <code>uVar2</code> without doing any proper length checks.

This turned out not to be the root cause. looking at the above, a crash could happen when it takes the address of <code>local24</code> and stores our user-provided JSON string into it, overflowing the <code>local\_20</code>, <code>local\_1c</code>, <code>local\_1s</code>, etc. I presume that <code>ivar5-9</code> are all status code checks. However, if <code>prod\_cfm\_set\_val</code> returns <code>0</code> on success, this check seems to be irrelevant. I later came. to the conclusion that this wasn't the case, and that something else was responsible for the behavior observed further down.

Going along with this initial presumption, all of this information is called upon when the configs are pushed to flash, which is performed as a result of the following POST request with the below UI as input.

Subscribe

So, the device crashed and rebooted not long after. What could be happening here? There are at least two ways the overflow could occur. It could occur if the password ever gets decoded. It could also happen if the program reads the user input into a variable without proper length checks, and writes it into memory.

Again, without the ability to attach GDB to the httpd process, it's really hard to

Subscribe

reverse. I'd also like to mention that there are confusing behaviors in the

add pppoe config function that occurs as part of formGetNetwor

have obstructed my understanding of the entire BOF chain.

My final verdict is as follows, and it will make more sense when watching the POC video. The overflow process is:

```
user input => setWanPppoe => getPPPoEConnectStatus => crash
```

A boiled-down explanation of what happens is:

- 1. JSON user-provided inputs get passed into the WAN config setWanPppoe a function that reads user input into uVarl and uVarl without doing any proper length checks.
- 2. At some point <code>getPPPoEConnectStatus</code> accepts input from <code>(presumably from setWanPppoe via prod\_cfm\_set\_val)</code> which reads the unchecked user input during which it calls <code>FUN\_00447674</code> (renamed <code>CredentialCompareFunction</code>) that stores the user-provided input with <code>sprintf</code> to the char buffer.
- 3. A check on the validity of the credentials is performed at which point the overflow happens.

It actually takes a dword parameter WanID (which is a value wanNum=3) and stores it into acstack264 (renamed UserPasswordString) this WanID stems from another function, FUN\_00447828 which is also invoked by the main credential check of the getPPPoEConnectStatus control flow specific to dialup initiation/integrity checks from a generated /etc/ppp/wan%d\_ppoe\_auth\_code file at startup.

Here's a video POC where you can see this in action. This BOF acts as a DOS and completely debilitates the router once the function is processed ~10 seconds after boot/login.



I won't be including a Golang or Python POC. This is purely a DoS and my aim isn't to enable and commoditize DoS attacks. Just like in the video, as soon as the device is logged in, it crashes and reboots in a perpetual never-ending cycle. Subscribe



reversing all the more confusing.

The blog is getting ridiculously long, so I will stop digging for these. If you follow prod\_cfm\_set\_val you're bound to find another ~10 buffer overflows. It's honestly a clown fiesta.

# **Stored Cross-Site Scripting**

The Tenda AC1200 V-W15Ev2 router does not perform proper validation on user-supplied input and is vulnerable to cross-site scripting attacks via the homepage's connected application hostname field. If a proper authorization mechanism was implemented, this vulnerability could be leveraged to perform actions on behalf of another user or the administrator. This vulnerability exists within the local web interface and the hosted remote web management console. The vulnerability affects version V15.11.0.10(1576).

# **Vulnerability Details**

```
CVE ID: CVE-2022-40846

Access Vector: Network
Security Risk: Medium
Vulnerability: CWE-79
CVSS Base Score: 6.5
CVSS Vector: CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:N/A:N
```

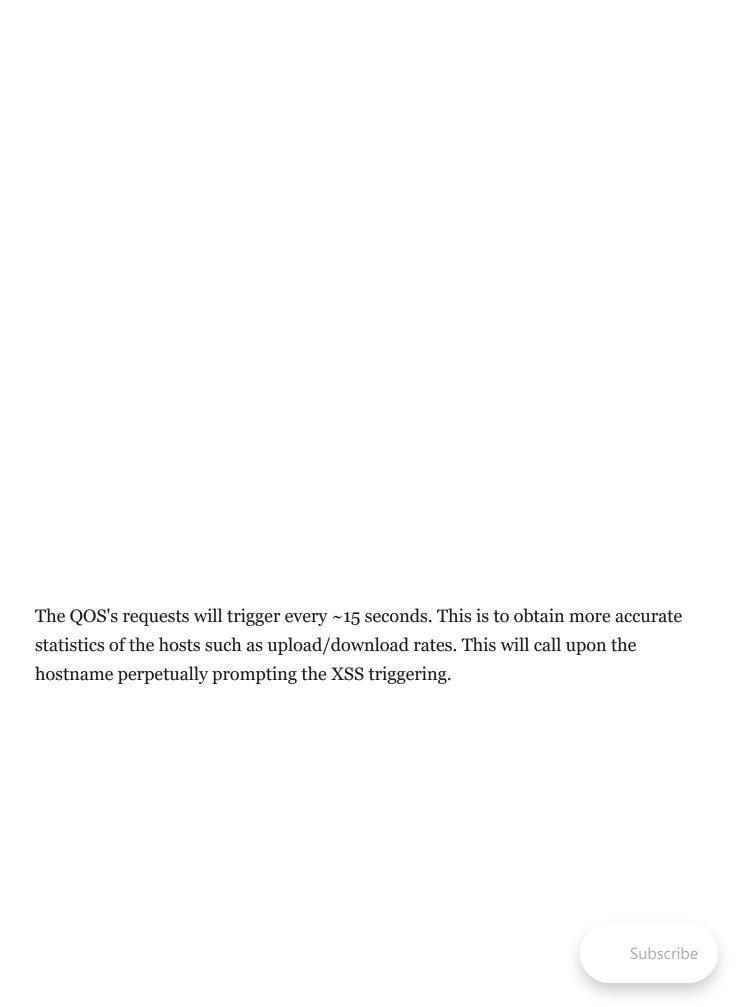
Subscribe

On the homepage, an administrator will have the ability to see the top 5 consume the most network resources. They'll also have the ability to rename these

hosts. I attempted the following payload, which failed. <script>alert(1)</script> However, if you look at the page's source, it would be reasonable to assume that you could trigger a valid XSS by prepending "> to your payload. Since I'm testing in Firefox ill be using marquee tags, my payload can be seen below. "><marquee onstart=alert(document.cookie)> In the page source the title string is now "handled" properly and closed leaving our malicious javascript outside of the element. Subscribe

Here's a little Burp Suite visual differentiating the two requests.

Subscribe



As seen below the XSS successfully triggers returning of the session cookies.
Stored Cross-Site Scripting
The Tenda AC1200 V-W15Ev2 router does not perform proper validation on user- supplied input and is vulnerable to cross-site scripting attacks through the web
filtering group body. If the proper authorization was implemented, this vulnerability could be leveraged to perform actions on behalf of another user or the administrator.

This vulnerability exists within the local web interface and the hosted remote web

Subscribe

management console. The vulnerability affects version V15.11.0.10(1576).

**Vulnerability Details** 

```
CVE ID: CVE-2022-40844

Access Vector: Network

Security Risk: Medium

Vulnerability: CWE-79

CVSS Base Score: 6.5

CVSS Vector: CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:N/A:N
```

There's another stored XSS in the Website filtering functionality. Navigating to the panel of "Website Filtering" you'll see the following.

Subscribe

Clicking on the "URL management" panel will pop-up the following window. Subscribe Anything injected within the URL body will reflect once its associated group name is clicked in the panel before. Pretty much any javascript here will work.

When I test in Firefox I personally prefer using Firefox-specific payloads such as <marquee onstart=alert(1)> for testing.

Since I don't blog a lot, I'll share my favorite payload which leverages the movie/video content tags which are never really filtered. Some of my goto payloads can be seen below and all work within this application.

```
<video controls oncanplay="alert(document.cookie)"><source src="http://mirrors.

<img src="test" OnError=prompt(document.cookie)>

<img src=x onerror=&#x61;&#x6C;&#x65;&#x72;&#x74;&#x28;&#x64;&#x6f;&#x63;&#x75;</pre>
```

Once the popup is saved, one way to trigger it is to single-click the element as seen below or set it to an IP group config.

# When Will it be Enough?

If I had to make an educated guess, I'd say there are roughly another ~10-15 CVEs laying around in the application. I won't personally be perusing them. Why?

- 1. I'd like to spend my time reversing something a little more entertaining.
- 2. Due to not having been able to establish a channel of communication with the vendor, I don't see the value in further research & potential additional impact on end-users given the nature of my disclosure.

If you want to land some easy CVEs give this application a look. There are roughly another 5 command injections at large (mainly in the hotel mode, the Subscribe initiation). There are also a lot of bad memset that I presume can lead and crashes. I'm not sure how realistic obtaining a stable root shell through a

carefully constructed payload would be. As a freebie, I'm 99% sure there is a buffer overflow in <code>GetWanAddress</code> via the <code>wanDns1</code> and <code>wanDns2</code> parameters that copy the user-provided input to the stack using <code>sprintf</code> where <code>%s</code> can't limit copy length, so it's possible to make the stack overflow there as well.

# **Thoughts on Responsible Disclosure**

I genuinely try my hardest to adhere to clear and transparent ethical principles in responsible vulnerability disclosure (RVD). In every case, I place top priority on the safety and security of end-users. I've written a handful of blogs that will never see the light of day because of vendor negligence, which would cause serious real-life end-user impact.

My blog is not intended to be malicious and isn't set out to damage brands or their reputations. I genuinely make these to educate, inform, and attempt to make the internet a safer place.

As mentioned above, I try my best to act in a transparent, responsible, and consistent manner. In order to achieve this, I adhere to 90 days from the initial contact disclosure deadline. This can be shortened if the vendor remediates the vulnerability sooner, and lengthened if the vendor requires additional time to patch the vulnerability.

- I will make at minimum 5 attempts to establish confidential and detailed communication with the vendor. This includes but is not limited to the vendor's "contact us" email address, key employee LinkedIn, Twitter, company telephone, etc,. I also make an effort to provide the vendor with multiple channels of communication (WeChat, LinkedIn, Email, Telegram, WhatsApp, and Telephone) for their overall convenience.
- Once in contact with the vendor's security team or designated corcontact, I provide the details of the vulnerabilities. During the fix also discuss my intentions to work along sides them / and help them with

remediation & retesting until the vulnerability is patched. Lastly, I discuss my intentions for public disclosure.

- Once everything has been agreed upon, a secure communications channel will be established. Over this channel, all details and materials needed to reproduce the vulnerability are provided.
- I try my best to provide reasonable assistance to the vendor to ensure the understanding of the significance of the discovered vulnerability.
- Once a patch has been released, I'll retest the patch out of good faith.

With regard to this blog, there were two decisions I could take. Either disclose or don't. "Who cares, just post it," you may be thinking. I do. And my ethical dilemma was as follows:

Option A: I don't disclose. As a consequence, there isn't a chance for the vendor or affected customers to be made aware of the underlying bugs. Furthermore, an

unethical individual may find the issue first and exploit it, potentially compromising dozens of utilities.

Option B: I publicly disclose the vulnerability, exposing the flaws in the product, and this blog-generated visibility prompts the vendor to take action (hopefully). As a consequence, low-life humans try and leverage these vulnerabilities against (in this case) a very small potential list of affected customers. Additionally, this blog could also rush affected customers to take their vulnerable systems down.

<u>Underlying issue</u>: The vendor has no control over the remote devices, other than publishing firmware updates and alerting customers the vendor can't do much.

As mentioned multiple times, the goal/aim of my research has always been to make the internet a safer place. In this case, after evaluating the vendor's behavior thereof, achieving my goal would require the vendor to change their (Subscribe & work towards establishing fluid channels of communication with researchers. This may require the publication of blogs such as those. Having weighed the present constitution of blogs such as those. Having weighed the present constitution of blogs such as those.

may require the publication of plogs such as these. Having weighed the pros and cons, I've decided to publicly disclose. Hopefully, this will help the vendor go through the change it needs to better handle researchers and disclosure.

It should be <u>blatantly</u> obvious that any past, present, or future employer cannot be held responsible for any personal research and disclosure conducted by me under my own name.

I find it unfortunate that a small majority of people remain hyper-sensitive to vulnerabilities and public disclosure. Therefore, I feel the need to add this disclaimer. I know my heart is in the right place, and that blogs far more damning than this go up all the time, without the author being judged and shamed.

#### **DISCLAIMER**

Any misuse derived from the information contained within this blog will be the exclusive responsibility of the person who carries it out. The owner of the blog, all present, past, and future employers, ghost.io, and any of its associates will not be

liable for any consequences or damages resulting from reckless and malicious use. The vendor is solely responsible for its security errors, and any damage that may be caused to the user's computer systems (hardware and software), or to the files & documents stored on them, as a consequence.

Shenzhen Tenda Technology Co., Ltd., failed to establish a clear channel of communication despite my thorough and consistent attempts. They were given ample opportunity to respond to my contact attempts. Responsible disclosure allows vendors between 60 and 120 business days to patch a vulnerability. 90 days have passed since my initial contact attempts.

Summary: Subscribe

I hope you liked the blog post. I apologize for getting political toward the end.

Follow me on <u>twitter</u> I sometimes post interesting stuff there too. Personally, I'd strongly recommend going on Amazon, Alibaba, or Aliexpress & buying a bunch of odd IoT devices & tearing them down. You never know what you will find:)

If you have any ethical dilemmas, hold anything against me, or disapprove of the contents of this blog, send me a DM, I'd love to see it from your perspective. Every day is a school day.

## Thank you for reading!

# Sign up for more like this.

Enter your email Subscribe

# **GL.iNET GL-MT300N-V2 Router Vulnerabilities and Hardware Teardown**

Hacking GL.iNET MT300N router & hardware teardown (3 CVEs)

Oct 26, 2022 32 min read

Boschko Security Blog © 2022

Powered by Ghost

Subscribe