Security Analysis of Router Firmware

Rakshith Raj Gurupura Puttaraju
M.E. Information Systems Security
Concordia University
Montreal, Quebec, Canada
rakshithraj.gp11@gmail.com

Nicholas Simo
M.E. Information Systems Security
Concordia University
Montreal, Quebec, Canada
simo.nicholas971@gmail.com

Mukul Prabhu
M.E. Information Systems Security
Concordia University
Montreal, Quebec, Canada
mukulsp99@gmail.com

Sanchit Smarak Behera
M.E. Information Systems Security
Concordia University
Montreal, Quebec, Canada
sanchitsmarak8@gmail.com

Rahul Ravi Hulli
M.E. Information Systems Security
Concordia University
Montreal, Quebec, Canada
rahulravi.hulli@gmail.com

***Abstract*— This project will introduce the concept of firmware vulnerability analysis (Dynamic and Static). Post which, seven selected firmware will be analyzed for vulnerability and the results of vulnerability analysis will be shared. The project also conducts static analysis on selected number of firmware files and shows the statistics of the findings of this analysis. The project also goes out of scope to fix a stated firmware analysis tool for future use.**

I. Introduction

A router's firmware is the core software that makes it possible for its hardware parts to operate as intended. It includes all the device drivers, configuration options, and operating system required for the router to function [1]. The router's user interface, security protocols, and feature set are all controlled by the firmware. Frequent firmware upgrades are crucial since they frequently include new features and include security patches and bug fixes [2]. Firmware security updates are especially important for protecting the router and the network it supports. Firmware also enables remote management capabilities, setting customization, and compatibility with emerging networking standards. Updating the firmware of the router regularly guarantees that it will continue to be resistant to new security risks, work with contemporary devices, and offer the best user experience.

Despite being an essential part of the device, router firmware is not impervious to cyberattacks. It is vulnerable to attacks that take advantage of holes in its coding, which could jeopardize the security of the entire network. Attackers may use firmware flaws to change settings, obtain illegal access, or carry out more complex attacks. Typical strategies include utilizing default credentials, injecting malicious code, and taking advantage of unpatched vulnerabilities. The router can be used as a point of entry for more extensive network intrusions if it is compromised. To reduce the risks associated with potential firmware assaults and maintain the router's status as a safe and dependable part of the network infrastructure, regular firmware upgrades, strong authentication procedures, and adherence to security best practices are crucial [3].

To strengthen the security framework of network devices, we conducted a thorough investigation into the nuances of router firmware analysis for our project. Using Firmadyne's capabilities [4], we carefully disassembled the router firmware and looked through its coding to find any vulnerabilities that might be security risks. The basis for selecting firmware components that required urgent attention and improvement was this in-depth analysis. As an additional component of our strategy, we created Python scripts specifically designed to carry out an exhaustive survey, which allowed us to precisely identify individual weaknesses. Our goal is to make a substantial contribution to the cybersecurity community by proactively resolving these found vulnerabilities. We intend to completely strengthen router firmware by utilizing Firmadyne and creating unique Python tools [5]. This will reduce the risks related to assaults and provide a strong defense system against future security threats.

II. Firmware Analysis

We used Static Analysis and Dynamic Analysis [6], two different but complimentary methodologies, in our Firmware Analysis project. To do Static Analysis, we used Python scripts to inspect the firmware's codebase in an unexecuted state, looking for potential weaknesses and unsafe coding techniques. However, Firmadyne [4], a program that mimicked the firmware in a safe setting, was used in Dynamic Analysis so that we could watch how it behaved during runtime. Despite their differences, these two methods gave rise to a thorough comprehension of the firmware's security environment. Whereas dynamic analysis displayed vulnerabilities related to the runtime, static analysis exposed flaws in the system itself. By using a unified technique, the goal was to provide a comprehensive assessment that would improve overall security and resistance to cyberattacks.

A. Static Analysis

In firmware analysis, static analysis means analyzing the structure and code without running it [7]. By using this strategy, potential vulnerabilities, hardcoded credentials, and insecure coding methods must be extensively examined in the source or binary code. It makes use of methods such as reverse engineering to try and decipher the firmware's logic and operation. Static analysis uses automated tools, frequently coded in languages like Python, to search for security flaws. It also finds patterns linked to known vulnerabilities and evaluates dependencies for upgrades [3]. Although it is non-intrusive and detects vulnerabilities early, runtime-specific issues might not always be visible. This approach is essential for early identification and is usually used in conjunction with dynamic analysis for a more thorough evaluation of router firmware security.

Manual firmware analysis relies on the knowledge of security experts and entails a hands-on analysis of the code, structure, and behavior of firmware [8]. The source or binary code of the firmware is carefully examined by security specialists to find flaws and unusual coding techniques. An assessment of the communication protocols, cryptographic implementations, and overall security architecture are all included in this report. When the firmware is observed in a controlled environment, possible harmful activities that are not immediately apparent from the source code can be found. Experts in security aggressively seek to take advantage of vulnerabilities that are found, offering insights into actual attack scenarios. Manual analysis entails determining the underlying causes of vulnerabilities and cross-referencing documentation with observed behavior [9]. Although laborious, manual analysis offers a more sophisticated comprehension of security threats and supplements automatic analyses for an extensive assessment of router firmware.

B. Dynamic Analysis

Firmware Analysis Tool: The open-source Firmware Analysis Toolkit is available on GitHub and is designed specifically for in-depth firmware analysis of embedded devices. FAT offers a range of capabilities to meet the needs of developers, researchers, and security experts [10]. It facilitates static analysis by giving users the ability to examine the file structure and code of the firmware, extract file systems, and find any vulnerabilities. FAT offers an emulation environment for dynamic analysis, allowing one to watch runtime behavior and identify vulnerabilities unique to the runtime. Tools for reverse engineering, helping to comprehend firmware logic and functionality, are included in the toolkit. Automated scanning techniques identify prevalent security flaws, whereas cryptographic analysis tools evaluate the use of cryptography. Because FAT is expandable, users can add more tools and scripts to meet their individual analysis needs. FAT is a flexible and cooperative tool for firmware analysis, fostering community contributions and collaboration with features for documentation and reporting [11].

Firmadyne: Strong firmware emulation and analysis capabilities are included into Firmadyne, an open-source tool that is targeted at embedded systems and Internet of Things (IoT) devices [4]. Developers, penetration testers, and security researchers who want to learn about the complexities of firmware in a controlled setting may find the tool especially helpful. Firmadyne makes it possible to emulate firmware in a virtualized environment powered by QEMU [12], so users may watch its behavior without having to install it on physical hardware.

This tool plays a crucial role in dynamic analysis by offering insights into the firmware's runtime operations. Security experts can find potential security concerns and runtime-specific vulnerabilities by simulating the firmware, which may not be visible through static analysis alone. Firmadyne is a flexible tool for analyzing firmware from a variety of embedded devices since it supports a broad range of architectures.

Firmadyne also helps with the development of customized firmware images for devices, increasing its adaptability for various analysis settings. Its prominence in the security community for firmware research and vulnerability finding can be attributed to its extensibility and interoperability with distinct types of firmware. Firmadyne's controlled environment for firmware analysis is a vital component in the advancement of embedded system security and knowledge.

III. Problem Statement

There are several convincing reasons why it is extremely important to analyze router firmware for vulnerabilities. Routers regulate and direct network traffic, acting as its gatekeepers. Any vulnerability in their firmware makes it more susceptible to data theft, illegal access, and even man-in-the-middle attacks [13]. Vulnerabilities in router firmware, which form the foundation of network security, have the potential to compromise the entire network because of their constant internet connectivity. Due to their large attack surface, routers require careful firmware analysis to find vulnerabilities in distinct parts of the system, such as the web interface, operating system, and underlying protocols.

To defend against IoT-related threats, mitigate vendor-specific vulnerabilities that may affect other devices, and guarantee regulatory compliance, it becomes essential to analyze router firmware. Furthermore, by examining encryption methods and access restrictions, firmware analysis supports proactive patch management, provides defense against zero-day attacks, and helps maintain privacy [3]. This analysis is a proactive approach critical to strengthening network security, preventing unwanted access, and reducing risks related to constantly changing cyberthreats.

In this project, we tried to simply prove that router firmware vulnerabilities still exist. We selected a set of distinct routers for dynamic analysis and the results we analyzed and understood. 55 router firmware were analyzed statically using python codes and statistics were built based on that. We also tried to fix an almost dead project that would analyze firmware (Firmware Analysis Toolkit), which would be explained later.

Static Analysis – Survey: Below are the algorithms for the codes under static analysis. Note that these results are static, and it does not mean that if the tool does not detect a vulnerability, there won't be any.

BasicDetails.py

1. Start

2. Parse command-line arguments using argparse

- Expect the path to the firmware file

3. Call find_file_system_addresses function

a. Open firmware file and read data

b. Search for CPIO and SquashFS magic values

c. Record addresses of identified CPIO and SquashFS instances

4. Call get_endianness function

a. Open firmware file and read data

b. Check for specific byte sequences to determine endianness

c. Return the endianness

5. Call get_entropy function

a. Execute binwalk tool to calculate entropy

b. Parse output to retrieve entropy value

6. Call get_first_4_bytes_xxd function

a. Execute xxd tool to display the first 4 bytes in hexadecimal

b. Return the output

7. Call identify_architecture function

a. Open firmware file and read ELF header

b. Check ELF header to determine architecture (32-bit, 64-bit, or Unknown)

c. Return the identified architecture

8. Call identify_architecture_variant function

a. Open firmware file and read first 16 bytes

b. Use capstone library to disassemble ARM instructions

c. Identify common ARM instruction mnemonics

d. Return the identified architecture variant

9. Assemble analysis results into a table

- CPIO and SquashFS addresses, endianness, entropy, first 4 bytes, architecture, and architecture variant

10. Print the table

11. End

CVE-2022-1262P1.py

1. Start

2. Check command-line arguments

a. If the number of arguments is not 2, print usage information and exit

3. Get the firmware file path from command-line argument

4. Run binwalk to extract files and redirect output to a file (binwalk_output.txt)

a. Construct binwalk command: binwalk -Me -D output_directory <firmware_file>

b. Execute binwalk command, capturing both stdout and stderr in binwalk_output.txt

5. Define vulnerability patterns for analysis

a. Patterns include command injection via pipes, semicolons, backticks, and variable assignments

6. Analyze extracted files for vulnerabilities

a. Walk through the extracted files directory

b. Read each file, decoding content to UTF-8

c. Search for vulnerability patterns in file content

d. Record vulnerabilities found with type, file path, and matching content

7. Create a set of found vulnerability types

8. Prepare and print results

a. For each vulnerability pattern

b. Check if the vulnerability type is in the set of found vulnerability types

c. Append results to a table: Vulnerability Type, Found/Not found

9. Print the table

10. End

CVE-2022-1262P2.py

1. Start

2. Parse command-line arguments using argparse

a. Expect a folder path as an argument

3. Call find_protest_bin_files function

a. Walk through the folder and its subdirectories

b. Identify files named "protest.bin"

c. Record paths of found "protest.bin" files

4. Check if protest binary files are found

a. If found, create a PrettyTable for displaying file paths

b. Print the table with file paths

c. If not found, print a message indicating no protest binary files are found

5. End

Log4jVulnerabilityDetection.py

1. Start

2. Parse command-line arguments using argparse

a. Expect a folder path as an argument

3. Call search_for_log4j_exploits function

a. Define Log4j-related patterns

b. Initialize lists for scanned files and potential exploits

c. Walk through the folder and its subdirectories

d. For each file, check if it is a .java, .xml, or. properties file

e. Open the file and read its content

f. Search for Log4j patterns in the content

g. Record scanned files and potential exploits

4. Check if scanned files exist

a. If yes, print the list of scanned files

5. Check if potential exploits exist

a. If yes, create a PrettyTable for displaying file paths and patterns

b. Print the table with potential exploits

c. If not, print a message indicating no potential exploits found

6. End

Dynamic Analysis – Firmadyne: For Dynamic Analysis, Firmadyne was used. The installation of this tool is as below. The analysis and procedure will be explained later [14].

Below are the commands typically involved in using Firmadyne for dynamic analysis.

EnvironmentSetup*:*

Download and install Firmadyne, PostgreSQl and Binwalk:

1. Install and configure dependencies and packages that support firmadyne:

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

sudo update-alternatives --install /usr/bin/python python /usr/bin/python3 10
```

2. Install binwalk

```
git clone https://github.com/ReFirmLabs/binwalk.git

cd binwalk

sudo ./deps.sh

sudo python ./setup.py install
```

3. Install PostgreSQL

```
cd ..

sudo apt-get install postgresql

sudo -u postgres createuser -P firmadyne

Give firmadyne as password.

sudo -u postgres createdb -O firmadyne firmware

sudo -u postgres psql -d firmware < ./firmadyne/database/schema
```

4. Install Firmadyne

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

cd firmadyne

./download.sh

nano firmadyne.config

Uncomment FIRMWARE_DIR and set its value to path of firmadyne folder. Use Ctrl + o to save changes and Ctrl + x to exit.
```

5. Install Qemu

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

Copy the firmware binary or zip file to the Firmadyne folder
```

Extract components of firmware:

```
sudo ./sources/extractor/extractor.py -b Netgear -sql 127.0.0.1 -np -nk "WNAP320 Firmware Version 2.0.3.zip" images
```

6. Find the ID generated during extraction, use it in next steps. Say if it is 1.

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

sudo ./scripts/makeImage.sh 1

./scripts/inferNetwork.sh 1
```

7. Interface IP (Internet Protocol) will be displayed. Here it is 192.168.0.100. Start the emulation:

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

8. Login to console using credential admin/password. Login to web UI at 192.168.0.100 using same credential. File system can be mounted using:

```
sudo ./scripts/mount.sh 1
```

9. Run Analysis: SNMP (Simple Network Management Protocol)

```
./analyses/snmpwalk.sh 192.168.0.100
less snmp.public.txt
less snmp.private.txt
```

10. Run Analysis: Web

```
./analyses/webAccess.py 1 192.168.0.100 log.txt
less log.txt
```

11. Run Analysis: Port Scan

```
sudo nmap -O -sV 192.168.0.100
```

12. Exploit: Install and run Metasploit

```
sudo apt install curl

curl https://raw.githubusercontent.com/rapid7/metasploit-omnibus/master/config/templates/metasploit-framework-
```

wrappers/msfupdate.erb > msfinstall && \

chmod 755 msfinstall && \

./msfinstall

Start Metasploit for the first time to complete initial configuration.

msfconsole

Exit from metasploit.

mkdir exploits

less exploits/exploit.metasploit.log

python ./analyses/runExploits.py -t 192.168.0.100 -o exploits/exploit -e x

These commands provide an outline of the steps involved in using Firmadyne for dynamic analysis. Do note that the actual usage might vary based on the specific firmware image, device, and analysis goals.

IV. Firmware Anyalysis - Dynamic

DLink DAP2330 V1.06: Denial of Service (DoS) Attack Vulnerability

Allows attackers to cause a Denial of Service (DoS) via uploading a crafted firmware after modifying the firmware header. Affected by this issue is an unknown part of the component *Firmware Header Handler*. Manipulation with an unknown input led to a denial-of-service vulnerability. Using CWE to declare the problem leads to CWE-404 [15]. The product does not release or incorrectly releases a resource before it is made available for re-use. Impacted is availability [16].
CVE (Common Vulnerabilities and Exposures): CVE-2022-38873.
Firmware Download Link: https://support.dlink.com/resource/PRODUCTS/DAP-2330/REVA/DAP-2330_REVA_FIRMWARE_1.06.RC020.ZIP
Framework used: Firmadyne

Procedure

```
┌──(kali㉿kali)-[~/Desktop/Project/firmadyne]
└─$ sudo ./sources/extractor/extractor.py -b Dlink -sql 127.0.0.1 -np -nk "../../INSE6120-Fall2023-Project-Group7/Rakshith/DLink DAP2330 V1
[sudo] password for kali:
>> Database Image ID: 2

/home/kali/Desktop/INSE6120-Fall2023-Project-Group7/Rakshith/DLink DAP2330 V1.06/firmware/DAP-2330_REVA_FIRMWARE_1.06.RC020.ZIP
>> MD5: 99fc0ffebab2cf5a1da1fe6fb11dbe65
>> Tag: 2
>> Temp: /tmp/tmp2vwb3_el
>> Status: Kernel: True, Rootfs: False, Do_Kernel: False,                Do_Rootfs: True
>>>> Zip archive data, at least v2.0 to extract, compressed size: 363250, uncompressed size: 468459, name: DAP-2330_REVA_RELEASENOTES_1.06.
>> Recursing into archive ...

/tmp/tmp2vwb3_el/_DAP-2330_REVA_FIRMWARE_1.06.RC020.ZIP.extracted/DAP-2330_REVA_firmware-v106-rc020.bin
        >> MD5: 1132840c5f9ac687111a5b7651523f2f
        >> Tag: 2
        >> Temp: /tmp/tmp8cb4tp4u
        >> Status: Kernel: True, Rootfs: False, Do_Kernel: False,                Do_Rootfs: True
        >> Recursing into archive ...
        >>>> Squashfs filesystem, little endian, version 4.0, compression:lzma, size: 7818490 bytes, 1054 inodes, blocksize: 131072 bytes,
        >>>> Found Linux filesystem in /tmp/tmp8cb4tp4u/_DAP-2330_REVA_firmware-v106-rc020.bin.extracted/squashfs-root!
        >> Skipping: completed!
        >> Cleaning up /tmp/tmp8cb4tp4u...
>> Skipping: completed!
>> Cleaning up /tmp/tmp2vwb3_el...
```

1. Use the extractor to recover only the filesystem, no kernel (-nk), no parallel operation (-np), populating the image table in the SQL server at 127.0.0.1 (-sql) with the Netgear brand (-b) and storing the tarball in images.

2. Identify the architecture of firmware 2 and store the result in the image table of the database ./scripts/getArch.sh ./images/2.tar.gz

3. Load the contents of the filesystem for firmware 2 into the database, populating the object and object_to_image tables. ./scripts/tar2db.py -i 2 -f ./images/2.tar.gz

4. Create the QEMU disk image for firmware 2. sudo ./scripts/makeImage.sh 2

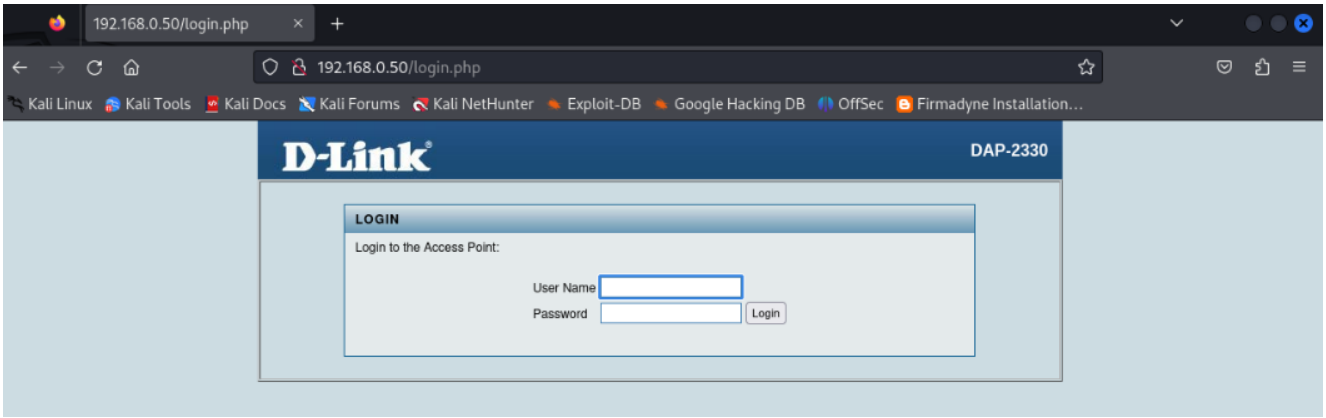
5. Infer the network configuration for firmware 2. Kernel messages are logged to ./scratch/2/qemu.initial.serial.log. ./scripts/inferNetwork.sh 2

```
┌──(kali㉿kali)-[~/Desktop/Project/firmadyne]
└─$ ./scripts/inferNetwork.sh 2
Querying database for architecture... Password for user firmadyne:
mipseb
Running firmware 2: terminating after 60 secs...
qemu-system-mips: terminating on signal 2 from pid 104331 (timeout)
Inferring network...
Interfaces: [('br0', '192.168.0.50')]
Done!
```

6. Emulate firmware 1 with the inferred network configuration. This will modify the configuration of the host system by creating a TAP device and adding a route. ./scratch/2/run.sh

```
┌──(kali㉿kali)-[~/Desktop/Project/firmadyne]
└─$ ./scratch/2/run.sh
Creating TAP device tap2_0...
Set 'tap2_0' persistent and owned by uid 1000
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 18:08:53 EDT 2020
[    0.000000] bootconsole [early0] enabled
```

7. Emulated Firmware on Web



8. Run Analysis: SNMP snmpwalk.sh: This script dumps the contents of the public and private SNMP v2c communities to disk using no credentials.

```
┌──(kali㉿kali)-[~/Desktop/Project/firmadyne]
└─$ ./analyses/snmpwalk.sh 192.168.0.50
Dumped to snmp.public.txt and snmp.private.txt

┌──(kali㉿kali)-[~/Desktop/Project/firmadyne]
└─$ less snmp.public.txt

┌──(kali㉿kali)-[~/Desktop/Project/firmadyne]
└─$ less snmp.private.txt
```

9. Run Analysis: Web - webAccess.py: This script iterates through each file within the filesystem of a firmware image that is served by a webserver, and aggregates the results based on whether they require authentication. ./analyses/webAccess.py 1 192.168.0.50 log.txt

```
Accessing: http://192.168.0.50/include/libs/plugins/modifier.cat.php...
-> HTTPError: 404
Accessing: http://192.168.0.50/login_button.html...
-> HTTPError: 404
Accessing: http://192.168.0.50/getBoardConfig.php...
-> HTTPError: 404
Accessing: http://192.168.0.50/tmpl/UnknownAPList.tpl.php...
-> HTTPError: 404
Accessing: http://192.168.0.50/help/help_RebootAP.html...
-> HTTPError: 404
Skipping: images/back_on.gif
```

10. Port Scan: The port scan was done using the command - sudo nmap -O -sV 192.168.0.50 - The result was as follows.

```
Host is up (0.0010s latency).
Not shown: 997 closed tcp ports (reset)
PORT    STATE SERVICE  VERSION
23/tcp  open  telnet   D-Link 524, DIR-300, or WBR-1310 WAP telnetd
80/tcp  open  http     D-Link WAP http ui
443/tcp open  ssl/http D-Link WAP http ui
MAC Address: 00:15:E9:2C:75:00 (D-Link)
Device type: general purpose
Running: Linux 2.6.X|3.X
OS CPE: cpe:/o:linux:linux_kernel:2.6 cpe:/o:linux:linux_kernel:3
OS details: Linux 2.6.38 - 3.0
Network Distance: 1 hop
Service Info: Device: WAP

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 14.40 seconds
```

11. Exploit: runExploits.py: This script tests for the presence of 60 known vulnerabilities using exploits from Metasploit, and 14 previously unknown vulnerabilities that we developed. These unknown vulnerabilities are tracked as follows.

```
┌──(kali㉿kali)-[~/Desktop/Project/firmadyne]
└─$ rm -rf exploits

┌──(kali㉿kali)-[~/Desktop/Project/firmadyne]
└─$ mkdir exploits

  ┌──(kali㉿kali)-[~/Desktop/Project/firmadyne]
└─$ python ./analyses/runExploits.py -t 192.168.0.50 -o exploits/e
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...

┌──(kali㉿kali)-[~/Desktop/Project/firmadyne]
└─$ head exploits/exploit.metasploit.log
[*] Processing script.rc for ERB directives.
resource (script.rc)> setg RHOST 192.168.0.50
RHOST => 192.168.0.50
resource (script.rc)> setg RHOSTS 192.168.0.50
RHOSTS => 192.168.0.50
resource (script.rc)> spool exploits/exploit.0.log
[*] Spooling to file exploits/exploit.0.log...
resource (script.rc)> use exploits/linux/http/airties_login_cgi_bo
[*] No payload configured, defaulting to linux/mipsbe/meterpreter/
resource (script.rc)> exploit -z
```

Wnap320: Cookie Leaks

Netgear wnap320 router WNAP320_V2.0.3_firmware is vulnerable to Incorrect Access Control via /recreate.php, which can leak all users' cookies [17].
Firmware Version: 2.0.3
Firmware Download Link: https://www.downloads.netgear.com/files/GDC/WNAP320/WNAP320%20Firmware%20Version%202.0.3.zip
Framework used: Firmadyne
CVE: CVE-2022-31876

Procedure

1. Use the extractor to recover only the filesystem, no kernel (-nk), no parallel operation (-np), populating the SQL server at 127.0.0.1 (-sql) with the Netgear brand (-b), and storing the tarball in images. - The tag ID is 1

2. Identify the architecture of firmware 1 and store the result in the image table of the database - ./scripts/getArch.sh ./images/1.tar.gz

3. Load the contents of the filesystem for firmware 1 into the database, populating the object and object_to_image tables. - ./scripts/tar2db.py -i 1 -f ./images/1.tar.gz

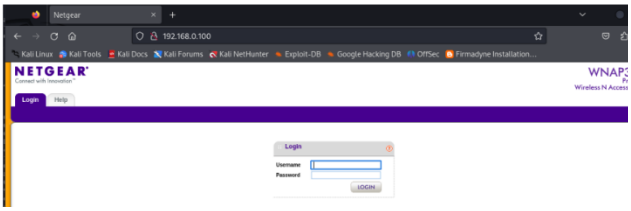
4. Create the QEMU disk image for firmware 1 - sudo ./scripts/makeImage.sh 1

5. Infer the network configuration for firmware 1. Kernel messages are logged to ./scratch/1/qemu.initial.serial.log. - . /scripts/inferNetwork.sh 1 - The interface received for emulation is [('brtrunk', '192.168.0.100')]

6. Emulate firmware 1 with the inferred network configuration. This will modify the configuration of the host system by creating a TAP device and adding a route. - ./scratch/1/run.sh

```
Welcome to SDK.

Have a lot of fun...

netgear123456 login: [   27.200000] brtrunk: port 1(eth0) entering forwarding state
```

7. Emulated Router Firmware:



8. Run Analysis: SNMP - snmpwalk.sh: This script dumps the contents of the public and private SNMP v2c communities to disk using no credentials. - ./analyses/snmpwalk.sh 192.168.0.100

9. Run Analysis: Web - webAccess.py: This script iterates through each file within the filesystem of a firmware image that is served by a webserver, and aggregates the results based on whether they appear to required authentication.

```
┌──(kali㉿kali)-[~/Desktop/Project/firmadyne]
└─$ ./analyses/webAccess.py 1 192.168.0.100 log.txt
Accessing: http://192.168.0.100/include/libs/internals/core.write_file.php..
Accessing: http://192.168.0.100/help/help_AdvancedWirelessSettings_g.html...
Accessing: http://192.168.0.100/include/libs/plugins/modifier.cat.php...
Accessing: http://192.168.0.100/login_button.html...
Accessing: http://192.168.0.100/getBoardConfig.php...
Accessing: http://192.168.0.100/tmpl/UnknownAPList.tpl.php...
Accessing: http://192.168.0.100/help/help_RebootAP.html...
Skipping: images/back_on.gif...
Accessing: http://192.168.0.100/tmpl/BasicTime.tpl.php...
Accessing: http://192.168.0.100/login.php...
Skipping: images/body_right_next_notch.gif...
Accessing: http://192.168.0.100/tmpl/siteSurvey.tpl.php...
Accessing: http://192.168.0.100/tmpl/DHCPServerSettings.tpl.php
```

10. Port Scan: sudo nmap -O -sV 192.168.0.100

```
Starting Nmap 7.94 ( https://nmap.org ) at 2023-11-01 23:11 EDT
Nmap scan report for 192.168.0.100
Host is up (0.00090s latency).
Not shown: 997 closed tcp ports (reset)
PORT    STATE SERVICE  VERSION
22/tcp  open  ssh      Dropbear sshd 0.51 (protocol 2.0)
80/tcp  open  http     lighttpd 1.4.18
443/tcp open  ssl/http lighttpd 1.4.18
MAC Address: 52:54:00:12:34:56 (QEMU virtual NIC)
Device type: general purpose
Running: Linux 2.6.X|3.X
OS CPE: cpe:/o:linux:linux_kernel:2.6 cpe:/o:linux:linux_kernel:
OS details: Linux 2.6.38 - 3.0
Network Distance: 1 hop
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

11. Exploit - runExploits.py: This script tests for the presence of 60 known vulnerabilities using exploits from Metasploit, and 14 previously unknown vulnerabilities that we developed. These unknown vulnerabilities are tracked as follows. - python ./analyses/runExploits.py -t 192.168.0.100 -o exploits/exploit -e x

DAP-1353: Username and Password Leak

D-Link DAP-1353 H/W vers. B1 3.15 and earlier, D-Link DAP-2553 H/W ver. A1 1.31 and earlier, and D-Link DAP-3520 H/W ver. A1 1.16 and earlier reveal wireless passwords and administrative usernames and passwords over SNMP [18].
CVE: CVE-2016-1559
Firmware Version: B1 v3.15
Firmware Download Link: https://ftp.dlink.ru/pub/Wireless/DAP-1353/Firmware/Rev%20B/3.15/DAP1353B-firmware-v315-rc012.bin
Framework used: Firmadyne

Procedure

1. Use the extractor to recover only the filesystem, no kernel (-nk), no parallel operation (-np), populating the SQL server at 127.0.0.1 (-sql) with the Netgear brand (-b), and storing the tarball in images. - Tag ID is 2

2. Identify the architecture of firmware 2 and store the result in the image table of the database - ./scripts/getArch.sh ./images/2.tar.gz

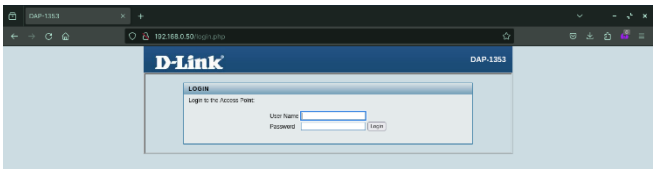
3. Load the contents of the filesystem for firmware 2 into the database, populating the object and object_to_image tables. - ./scripts/tar2db.py -i 2 -f ./images/1.tar.gz

4. Create the QEMU disk image for firmware 2 - sudo ./scripts/makeImage.sh 2

5. Infer the network configuration for firmware 2. Kernel messages are logged to ./scratch/2/qemu.initial.serial.log. - ./scripts/inferNetwork.sh 2 - The interface received for emulation is [('br0', '192.168.0.50')]

6. Emulate firmware 2 with the inferred network configuration. This will modify the configuration of the host system by creating a TAP device and adding a route. - ./scratch/2/run.sh

7. Emulated Router Firmware:



8. Run Analysis: SNMP - snmpwalk.sh: This script dumps the contents of the public and private SNMP v2c communities to disk using no credentials. - ./analyses/snmpwalk.sh 192.168.0.100 - The password set for this router firmware was “test”

```
3905 iso.3.6.1.4.1.171.10.37.38.3.16.3.1.1.17.10 = INTEGER: 0
3906 iso.3.6.1.4.1.171.10.37.38.3.16.3.1.1.17.11 = INTEGER: 0
3907 iso.3.6.1.4.1.171.10.37.38.3.16.3.1.1.17.12 = INTEGER: 0
3908 iso.3.6.1.4.1.171.10.37.38.3.16.3.1.1.17.13 = INTEGER: 0
3909 iso.3.6.1.4.1.171.10.37.38.3.16.3.1.1.17.14 = INTEGER: 0
3910 iso.3.6.1.4.1.171.10.37.38.3.16.3.1.1.17.15 = INTEGER: 0
3911 iso.3.6.1.4.1.171.10.37.38.3.16.3.1.1.17.16 = INTEGER: 0
3912 iso.3.6.1.4.1.171.10.37.38.4.1.1.1.2.1 = STRING: "admin"
3913 iso.3.6.1.4.1.171.10.37.38.4.1.1.1.3.1 = STRING: "test"
3914 iso.3.6.1.4.1.171.10.37.38.4.2.1.0 = INTEGER: 0
3915 iso.3.6.1.4.1.171.10.37.38.4.2.2.0 = INTEGER: 0
```

9. Run Analysis: Web - webAccess.py: This script iterates through each file within the filesystem of a firmware image that is served by a webserver, and

aggregates the results based on whether they appear to required authentication.

```
Accessing: http://192.168.0.100/tool_sntp.php...
-> URLError: [Errno 113] No route to host
Accessing: http://192.168.0.100/locale/en/wr_login_fail.php...
-> URLError: [Errno 113] No route to host
Accessing: http://192.168.0.100/html/Version.html...
-> URLError: [Errno 113] No route to host
Skipping: pic/plusbottom.gif...
Accessing: http://192.168.0.100/locale/en/session_full.php...
-> URLError: [Errno 113] No route to host
Skipping: pic/tool_bar_v.jpg...
Skipping: pic/plus.gif...
Accessing: http://192.168.0.100/home_sys.php...
-> URLError: [Errno 113] No route to host
Accessing: http://192.168.0.100/model/__button.php...
```

10. Port Scan: sudo nmap -O -sV 192.168.0.50

```
Starting Nmap 7.80 ( https://nmap.org ) at 2023-11-15 00:07 EST
Nmap scan report for 192.168.0.50
Host is up (0.0019s latency).
Not shown: 997 closed ports
PORT    STATE SERVICE    VERSION
23/tcp  open  telnet
80/tcp  open  http       Mathopd httpd 1.5p6
443/tcp open  ssl/https?
```

11. Exploit - runExploits.py: This script tests for the presence of 60 known vulnerabilities using exploits from Metasploit, and 14 previously unknown vulnerabilities that we developed. These unknown vulnerabilities are tracked as follows. - python ./analyses/runExploits.py -t 192.168.0.50 -o exploits/exploit -e x

Netgear D6000: Cleartext Password

The password-recovery feature on NETGEAR D3600 devices with firmware 1.0.0.49 and D6000 devices with firmware 1.0.0.49 and earlier allows remote attackers to discover the cleartext administrator password by reading the cgi-bin/passrec.asp HTML source code [19].
CVE: CVE-2015-8289
Firmware Download Link: https://www.netgear.com/support/download/?model=D6000

Framework used: Static Analysis
Procedure

1. Traverse to /boaroot/cgi-bin/passrec.asp - This is a file containing code for the password recovery page. As you can see in the snippet the code is badly written. It makes a call to the server to get the password if pressed forgot password and stores it in a variable in cleartext.

```
var if_submit_flag="<%if tcWebApi_get("Pwdrecovery_Entry","temp_flag","h") = "result" then asp_write("1") else asp_write("0") end if %>";
if(if_submit_flag == "1")
{
        username="<%tcWebApi_get("Account_Entry0","username","s")%>";
        web_passwd="<%tcWebApi_get("Account_Entry0","web_passwd","s")%>";
}
```

Here you can see that instead of directly making a call to the server and storing the value, they are instead using object-oriented concepts and accessing the value as an object and not directly storing any of the values.

2. This next snippet is from the patched firmware version 1.0.0.5.9. Here they have completely sacked the passrec.asp file and also improved their code.

```
        }

        if (cf.sysNewPasswd.value.length == 33 || cf.sysConfirmPasswd.value.length == 33)
        {
                alert(invalid_passwd_len_str);
                return false;
        }
        if(cf.sysNewPasswd.value != cf.sysConfirmPasswd.value)
        {
                alert(invalid_passwd_confirmed_str);
                return false;
        }
        //if(cf.sysOldPasswd.value != "" && cf.sysNewPasswd.value == "")
        //{
        //      alert(empty_passwd_str);
        //      return false;
        //}
        <%
        if tcWebApi_get("Account_Entry0","blank_password","h") = "1" then
                asp_Write("if(0){")
        else
                asp_Write("if(1){")
        end if
        %>
                if(cf.sysOldPasswd.value == "")
                {
                        alert(invalid_old_passwd_str);
                        return false;
                }
                cf.temp_pwd_old.value = cf.sysOldPasswd.value;
```

NETGEAR D3600 and NETGEAR D6000: Defeating Cryptographic Protection Mechanisms
NETGEAR D3600 devices with firmware 1.0.0.49 and D6000 devices with firmware 1.0.0.49 and earlier use the same hardcoded private key across different customers' installations, which allows remote attackers to defeat cryptographic protection mechanisms by leveraging knowledge of this key from another installation [20].
CVE: CVE-2015-8288
Firmware download link: https://www.netgear.com/support/download/?model=D6000

Framework used: Static Analysis

Procedure:

1. NETGEAR D3600 devices with firmware 1.0.0.49 and D6000 devices with firmware 1.0.0.49 and earlier use the same hardcoded private key across different customers' installations. First traverse to /etc/key.pem file of D6000 router and view its private key

```
# pwd
/etc
# cat key.pem
-----BEGIN RSA PRIVATE KEY-----
MIICWwIBAAKBgQCxYSK1K0gXZNbmkhq2hGmIDcTLI1oYor5YXhMCigspXu1EOWIZ
0k+RZhNClIo2Woj6JyISzKjM8132K8nbfhC3LzSN9LuqX65+m9TXM5WqoPuj7fia
Ogjzotcvbb6ZlFDBCaiNQtnWlCFNdFwg6pgdt98tnfAC39ds83lTqc9NAwIDAQAB
AoGAU1vx4rBPdecG816wPRwnODA4TrrzaWZ6C69SQbw8GJoq6QgKqVXj3NHatUx3
U6nie6UOZ0VVAtRQURMtDwDEu3mPq/LuykC7KyAhSXrWs/Evnza6R0Z24D9AfyVl
mbUTGkk1fpuEyq/6nGU1RxvrMp7UG9U/06ip1Tz3hU83+akCQQDa9r9T5wBqlOpc
eQ/BS3kN5h26DGkix4+cCnLFuX6u8sYb+9ruRXkU8U243XnuyZ2MqlL9tBttIw6J
Ysgg7F6lAkEAz2HCq64UB8G7L8Dn1kw7XfDoH4M+tT1UHc0TDTyciC7ud1juIoNo
OJ3SXgNKuCDNGqJ5lLEHVkPoqRpf5AqUhwJAPpMvgGHOu6bNBxpNx1azRKaTrjyn
uxm+z8AopprpOkQFG7AxE9Dk1nhTVtZsZzJVBQMT7o8PG3ReTiLdwJLsRQJAYgtO
Beye+6vT25fQA7i2uIsUsxFRrE36Xrx9rVxihN/rTfF1ZEvMEjhLJLdZtAON9HGs
GWg4l8gzTv9xJlb6MQJAacAptBQKjrpFSUujXhyS4bkktgzwjEf+03SPXDMfqFQ+
nfzGtp+rL9SlqWg9f1P7uSu9BZYujSIjhO0OgZUX0Q==
-----END RSA PRIVATE KEY-----
# 
```

2. As it is mentioned that devices of both D6000 and D3600 models use the same private key, we checked the private key of D3600 router as well, and found the /etc/key.pem file to have the same private key

```
# cat key.pem
-----BEGIN RSA PRIVATE KEY-----
MIICWwIBAAKBgQCxYSK1K0gXZNbmkhq2hGmIDcTLI1oYor5YXhMCigspXu1EOWIZ
0k+RZhNClIo2Woj6JyISzKjM8132K8nbfhC3LzSN9LuqX65+m9TXM5WqoPuj7fia
Ogjzotcvbb6ZlFDBCaiNQtnWlCFNdFwg6pgdt98tnfAC39ds83lTqc9NAwIDAQAB
AoGAU1vx4rBPdecG816wPRwnODA4TrrzaWZ6C69SQbw8GJoq6QgKqVXj3NHatUx3
U6nie6UOZ0VVAtRQURMtDwDEu3mPq/LuykC7KyAhSXrWs/Evnza6R0Z24D9AfyVl
mbUTGkk1fpuEyq/6nGU1RxvrMp7UG9U/06ip1Tz3hU83+akCQQDa9r9T5wBqlOpc
eQ/BS3kN5h26DGkix4+cCnLFuX6u8sYb+9ruRXkU8U243XnuyZ2MqlL9tBttIw6J
Ysgg7F6lAkEAz2HCq64UB8G7L8Dn1kw7XfDoH4M+tT1UHc0TDTyciC7ud1juIoNo
OJ3SXgNKuCDNGqJ5lLEHVkPoqRpf5AqUhwJAPpMvgGHOu6bNBxpNx1azRKaTrjyn
uxm+z8AopprpOkQFG7AxE9Dk1nhTVtZsZzJVBQMT7o8PG3ReTiLdwJLsRQJAYgtO
Beye+6vT25fQA7i2uIsUsxFRrE36Xrx9rVxihN/rTfF1ZEvMEjhLJLdZtAON9HGs
GWg4l8gzTv9xJlb6MQJAacAptBQKjrpFSUujXhyS4bkktgzwjEf+03SPXDMfqFQ+
nfzGtp+rL9SlqWg9f1P7uSu9BZYujSIjhO0OgZUX0Q==
-----END RSA PRIVATE KEY-----
# 
```

D-Link DIR-865L: Cleartext Password and Misconfiguration of encryption

The tools_admin.php page stores the admin password in clear text. For an attacker to get the password, they would require physical access to a machine that is logged on. Physical access is necessary because the credentials are not sent in clear text over the wire. With physical access, they can see the password by viewing the HTML source of the page.

Like most standard practices the router stores the hash of the password. It uses the MD5 HMAC hashing algorithm. It should be noted that this vulnerability was discovered in 2020, and MD5 is still being used in spite of its security issues. The following screenshots are taken from burpsuite [21].

CVE: CVE-2020-13783

Framework used: Firmadyne, Static Analysis using Burpsuite

Procedure:

1. Use the extractor to recover only the filesystem, no kernel (-nk), no parallel operation (-np), populating the SQL server at 127.0.0.1 (-sql) with the Netgear brand (-b), and storing the tarball in images. - Tag ID is 3

2. Identify the architecture of firmware 3 and store the result in the image table of the database - ./scripts/getArch.sh ./images/3.tar.gz

3. Load the contents of the filesystem for firmware 3 into the database, populating the object and object_to_image tables. - . /scripts/tar2db.py -i 3 -f ./images/3.tar.gz

4. Create the QEMU disk image for firmware 3 - sudo ./scripts/makeImage.sh 3

5. Infer the network configuration for firmware 3. Kernel messages are logged to ./scratch/3/qemu.initial.serial.log. - . /scripts/inferNetwork.sh 3 - The interface received for emulation is [('br0', '192.168.0.1'), ('br0', '192.168.7.1')]

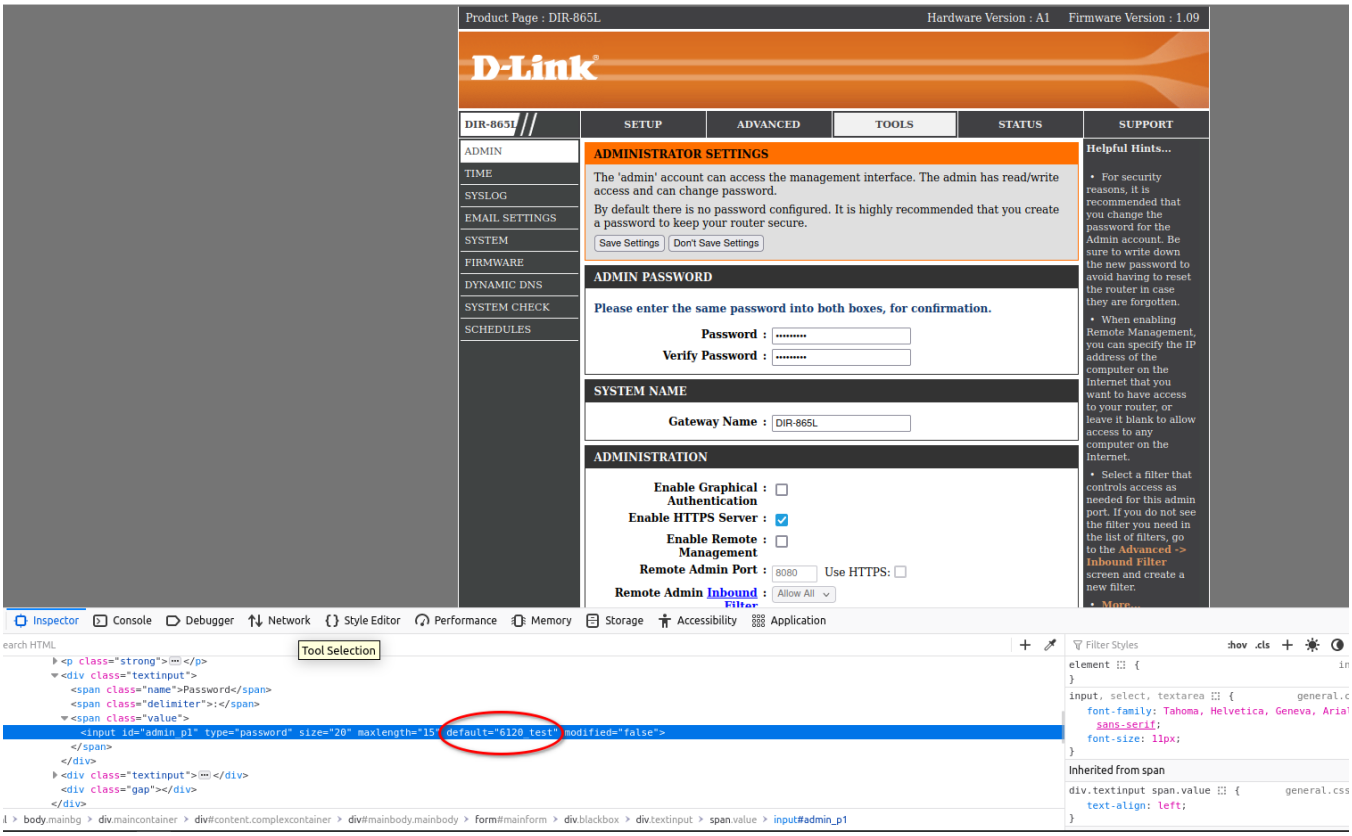
6. Emulate firmware 3 with the inferred network configuration. This will modify the configuration of the host system by creating a TAP device and adding a route. - ./scratch/3/run.sh

7. Emulation: This is the web portal that is available to us at http://192.168.0.1, does not load at https://192.168.0.1

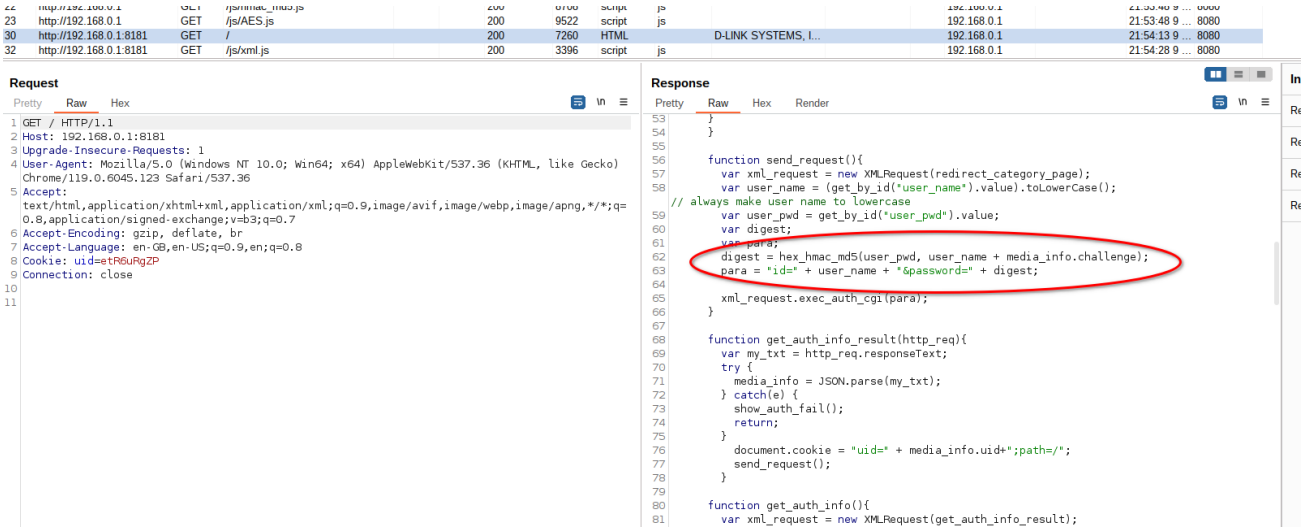
8. Nmap result: Running nmap over that ip gives us the following result. Nothing out of the ordinary. We'll be concentrating on port 8181 for one of the vulnerabilities.

```
mukul@mukul-HP-Pavilion-Gaming-Laptop-15-ec0xxx:~/Downloads/UntitledFolder/firmadyne$ sudo nmap -O -sV 192.168.0.1
[sudo] password for mukul:
Starting Nmap 7.80 ( https://nmap.org ) at 2023-11-09 18:59 EST
Nmap scan report for puma7-atom.hitronhub.home (192.168.0.1)
Host is up (0.00071s latency).
Not shown: 995 closed ports
PORT      STATE SERVICE    VERSION
53/tcp    open  domain     dnsmasq 2.78
80/tcp    open  http       D-Link DIR-865L WAP http config 1.09
443/tcp   open  ssl/https?
8181/tcp  open  http       D-Link SharePort web access 1.0 (model DIR-865L, version 1.09)
49152/tcp open  upnp       D-Link DIR-865L WAP UPnP 1.09 (UPnP 1.0)
MAC Address: 00:DE:FA:1A:01:00 (Unknown)
No exact OS matches for host (If you know what OS is running on it, see https://nmap.org/submit/ ).
TCP/IP fingerprint:
OS:SCAN(V=7.80%E=4%D=11/9%OT=53%CT=1%CU=40331%PV=Y%DS=1%DC=D%G=Y%M=00DEFA%T
OS:M=654D728D%P=x86_64-pc-linux-gnu)SEQ(SP=CB%GCD=1%ISR=CC%TI=Z%CI=Z%II=I%T
OS:S=U)OPS(O1=M5B4NNSNW4%O2=M5B4NNSNW4%O3=M5B4NW4%O4=M5B4NNSNW4%O5=M5B4NNSN
OS:W4%O6=M5B4NNS)WIN(W1=3908%W2=3908%W3=3908%W4=3908%W5=3908%W6=3908)ECN(R=
OS:Y%DF=Y%T=40%W=3908%O=M5B4NNSNW4%CC=Y%Q=)T1(R=Y%DF=Y%T=40%S=O%A=S+%F=AS%R
OS:D=0%Q=)T2(R=N)T3(R=Y%DF=Y%T=40%W=3908%S=O%A=S+%F=AS%O=M5B4NNSNW4%RD=0%Q=
OS:)T4(R=Y%DF=Y%T=40%W=0%S=A%A=Z%F=R%O=%RD=0%Q=)T5(R=Y%DF=Y%T=40%W=0%S=Z%A=
OS:S+%F=AR%O=%RD=0%Q=)T6(R=Y%DF=Y%T=40%W=0%S=A%A=Z%F=R%O=%RD=0%Q=)T7(R=Y%DF
OS:=Y%T=40%W=0%S=Z%A=S+%F=AR%O=%RD=0%Q=)U1(R=Y%DF=N%T=40%IPL=164%UN=0%RIPL=
OS:G%RID=G%RIPCK=G%RUCK=G%RUD=G)IE(R=Y%DFI=N%T=40%CD=S)

Network Distance: 1 hop
Service Info: OS: Linux; Devices: storage-misc, WAP; CPE: cpe:/h:dlink:dir-865l:1.09, cpe:/o:linux:linux_kernel, cpe:/h:d-link:dir-865l

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 31.17 seconds
```

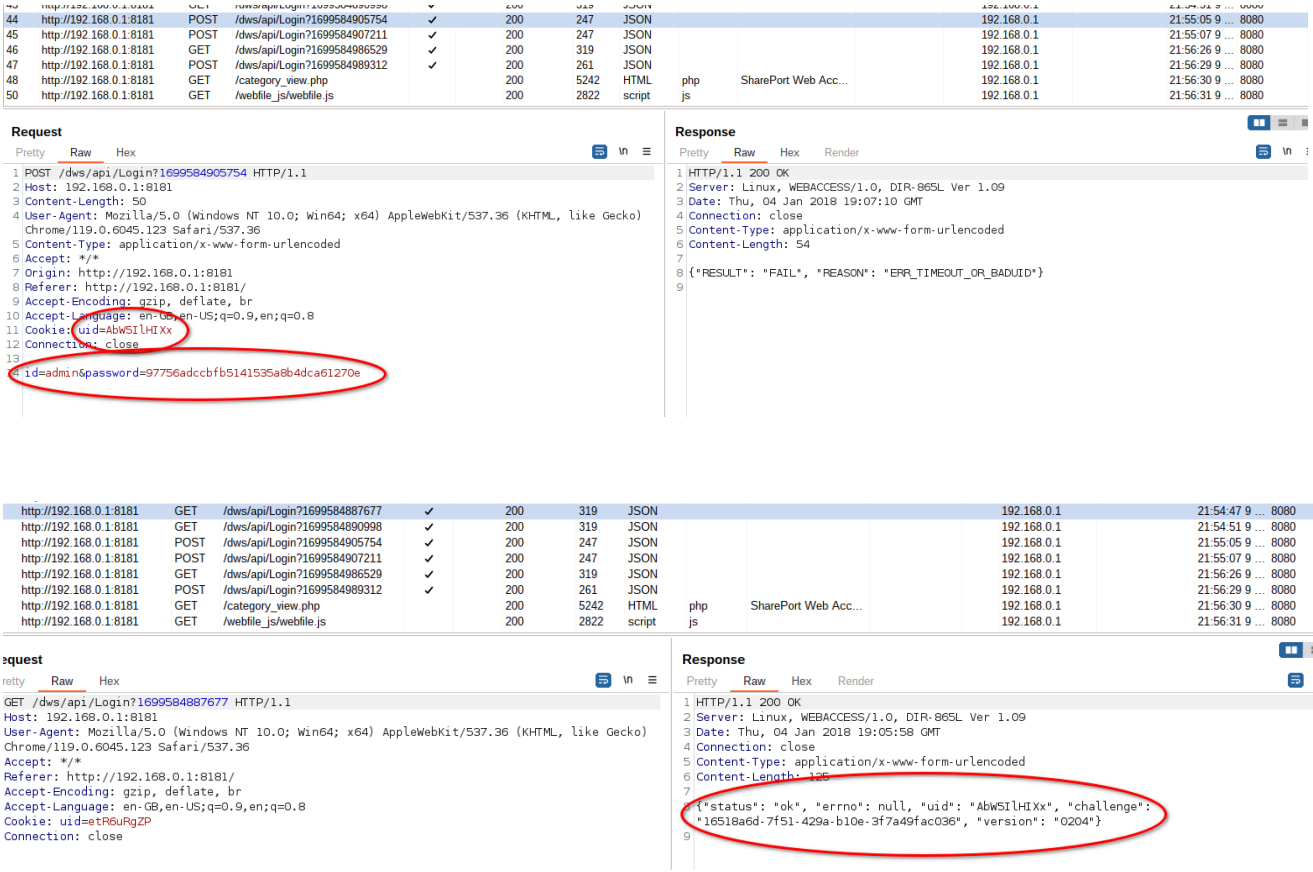
9. Under Tools tab, Admin section we can reset the default password that comes out of the box. Once we add the new password and save it, the password is stored in clear text in the tools_admin.php page. To test it out I change the password to 6120_test. The password can be seen as marked in the image. The vulnerability is impactful if the attacker has access the the system or the file system.



10. Vulnerability 2: Screenshots: We can see that the hashing code is marked in the image above. It is digest = hex_hmac_md5(user_pwd, user_name + media_info.challenge). The formula takes in the user password as the input and concatenation of the user id and the challenge as the "secret key". So, if the attacker somehow gets access to the stored credentials, he/she would get the user_id and the hashed password which isn't enough. The id and hashed password are as marked in the image.



11. Now the issue here is that the challenge, that's part of the "secret key" is available is plain sight for the attacker. Note the uid in the above image and the challenge and uid in the bottom image. Same uid suggests the same session. So now the attacker has the user id, challenge and the final hash. It's possible to guess the password by brute force, decreasing the effectiveness of the entire Hmac algorithm.Do not label axes with a ratio of quantities and units. For example, write "Temperature (K)", not "Temperature/K".



DAP-3662 v1.0.1: Buffer overflow attack and MD5 Hash

CVE-2016-1558 expresses a vulnerability in DAP-3662 with a buffer overflow attack. The D-Link router firmware contains a server vulnerability with the 'dlink_uid' cookie. The buffer is not variable; therefore, it overflows as there is no check in the code to ensure no overflow [22].

The firmware image uses an MD5 hash which is known to be unsafe. An attacker can use the exploit to make users think that they are installing firmware, when they are installing malware onto their computer.

CVE: CVE-2016-1558

Firmware Download Link: https://support.dlink.com/resource/PRODUCTS/DAP-3662/REVA/DAP-3662_REVA_FIRMWARE_1.01RC022.ZIP

Framework Used: Firmadyne, Static Analysis

1. Use the extractor to recover only the filesystem, no kernel (-nk), no parallel operation (-np), populating the SQL server at 127.0.0.1 (-sql) with the Netgear brand (-b), and storing the tarball in images. - Tag ID is 2

2. Identify the architecture of firmware 2 and store the result in the image table of the database - ./scripts/getArch.sh ./images/2.tar.gz

3. Load the contents of the filesystem for firmware 2 into the database, populating the object and object_to_image tables. - . /scripts/tar2db.py -i 2 -f ./images/1.tar.gz

4. Create the QEMU disk image for firmware 2 - sudo ./scripts/makeImage.sh 2

5. Infer the network configuration for firmware 2. Kernel messages are logged to ./scratch/2/qemu.initial.serial.log. - . /scripts/inferNetwork.sh 2 - The interface received for emulation is [('br0', '192.168.0.1')]

6. Emulate firmware 2 with the inferred network configuration. This will modify the configuration of the host system by creating a TAP device and adding a route. - ./scratch/2/run.sh

```
Extraction:
Command: sudo ./sources/extractor/extractor.py -b DLINK -sql 127.0.0.1 -np -nk "DAP-3662_REVA_FIRMWARE_1.01RC022.ZIP" images

Output:
>> Database Image ID: 2

/home/nicholas/Desktop/Analysis/firmadyne/DAP-3662_REVA_FIRMWARE_1.01RC022.ZIP
>> MD5: 0c414bea6f4c133d7b087c975a18a2f7
>> Tag: 2
>> Temp: /tmp/tmp07y4a6kc
>> Status: Kernel: True, Rootfs: False, Do_Kernel: False,                Do_Rootfs: True
>>>> Zip archive data, at least v2.0 to extract, compressed size: 365216, uncompressed size: 458109, name: DAP-3662_REVA_RELEASE
>> Recursing into archive ...
```

7. Port Scan: sudo nmap -O -sV 192.168.0.1

```
Starting Nmap 7.80 ( https://nmap.org ) at 2023-11-11 13:33 AST
Nmap scan report for 192.168.0.50
Host is up (0.0023s latency).
Not shown: 997 closed ports
PORT    STATE SERVICE    VERSION
23/tcp  open  telnet     D-Link 524, DIR-300, or WBR-1310 WAP telnetd
80/tcp  open  http       D-Link WAP http ui
443/tcp open  ssl/https?
MAC Address: 00:15:E9:2C:75:00 (D-Link)
Device type: general purpose
Running: Linux 2.6.X|3.X
OS CPE: cpe:/o:linux:linux_kernel:2.6 cpe:/o:linux:linux_kernel:3
OS details: Linux 2.6.38 - 3.0
Network Distance: 1 hop
Service Info: Device: WAP

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 22.13 seconds
```

WNDAP360 v2.0.0: Command Injection, Passwords exposed, MD5 Hash

CVE-2016-1555 expresses vulnerabilities located in BoardData.NP and BoardData.WW: These files inject commands directly into the command line interface. Exploiters can use this defect to execute arbitrary code and run commands without knowing the username or password of the router [23].

CVE-2016-1557 also explains that wireless passwords, administrative usernames, and passwords when exploiting through an SNMP analysis [24]. These values are stored in plaintext and are easy for an attacker to access, thereby giving control to the attacker [25].

Firmware also hashed using MD5.

CVE: CVE-2016-1555, CVE-2016-1556, CVE-2016-1557

Firmware Download Link: https://www.downloads.netgear.com/files/GDC/WNDAP360/WNDAP360_V3.7.11.4.zip

Framework used: Firmadyne, Static Analysis

Procedure:

1. The path to these files is in: NETGEAR_WNDAP360/_WNDAP360_V2.0.0_firmware.tar.extracted/_rootfs.squashfs.extracted/squashfs-root/home/www/ . The command line injection vulnerability can be seen in the beginning lines of code:

```php
<?php
        $flag=false;
        $msg='';
        if (!empty($_REQUEST['writeData'])) {
                if (!empty($_REQUEST['macAddress']) && array_search($_REQUEST['reginfo'],Array('WW'=>'0','NA'=>'1'))!==false && ereg("[0-
                        //echo "test ".$_REQUEST['macAddress']." ".$_REQUEST['reginfo'];
                        //exec("wr_mfg_data ".$_REQUEST['macAddress']." ".$_REQUEST['reginfo'],$dummy,$res);
                        exec("wr_mfg_data -m ".$_REQUEST['macAddress']." -c ".$_REQUEST['reginfo'],$dummy,$res);
                        if ($res==0) {
                                conf_set_buffer("system:basicSettings:apName netgear".substr($_REQUEST['macAddress'], -6)."\n");
                                conf_save();
                                $msg = 'Update Success!';
                                $flag = true;
                        }
                }
                else
                        $flag = true;
        }

?>
```

2. The exec command, shown on line 8, runs directly on the command line. There is no checking of the variables to ensure integrity before the execution.

3. CVE-2016-1557 also explains that wireless passwords, administrative usernames, and passwords when exploiting through an SNMP analysis. These values are stored in plaintext and are easy for an attacker to access, thereby giving control to the attacker. Example output over SNMP protocol is in the repository at: NETGEAR_WNDAP360/snmp.private_NETGEAR_WNDAP360.txt, NETGEAR_WNDAP360/snmp.public_NETGEAR_WNDAP360.txt

4. Firmware also hashed using MD5. Use the extractor to recover only the filesystem, no kernel (-nk), no parallel operation (-np), populating the SQL server at 127.0.0.1 (-sql) with the Netgear brand (-b), and storing the tarball in images. - Tag ID is 3

```
command: udo ./sources/extractor/extractor.py -b Netgear -sql 127.0.0.1 -np -nk "WNDAP360 Firmware Version 2.0.0.zip" images

Output:
>> Database Image ID: 3

/home/nicholas/Desktop/Analysis/firmadyne/WNDAP360 Firmware Version 2.0.0.zip
>> MD5: ea94e716634abdea2d34f1a908087e14
>> Tag: 3
>> Temp: /tmp/tmpu44njw5n
>> Status: Kernel: True, Rootfs: False, Do_Kernel: False,                Do_Rootfs: True
>>>> Zip archive data, at least v2.0 to extract, compressed size: 1912, uncompressed size: 7101, name: ReleaseNotes_WNDAP360_fw_2
>> Recursing into archive ...
```

5. Nmap Command:

```
RUNNING NMAP COMMAND:
Starting Nmap 7.80 ( https://nmap.org ) at 2023-11-11 14:37 AST
Nmap scan report for 192.168.0.100
Host is up (0.0026s latency).
Not shown: 997 closed ports
PORT    STATE SERVICE    VERSION
22/tcp  open  ssh        Dropbear sshd 0.51 (protocol 2.0)
80/tcp  open  http       lighttpd 1.4.18
443/tcp open  ssl/https?
MAC Address: 52:54:00:12:34:56 (QEMU virtual NIC)
Device type: general purpose
Running: Linux 2.6.X|3.X
OS CPE: cpe:/o:linux:linux_kernel:2.6 cpe:/o:linux:linux_kernel:3
OS details: Linux 2.6.38 - 3.0
Network Distance: 1 hop
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 24.30 seconds
```

TP-LINK AX1800 v1.2: Command Injection

CVE-2023-1389 affected AX1800 firmware versions up to v1.1.4. v1.2 fixed a vulnerability with a command injection which was in the country form of the Luci endpoint [26]. As with many command injection attacks, the command being sent through the interface was not being checked for any malicious intent, which was an oversight. In v1.2, Luci now removes the field entirely:

CVE: CVE-2023-1389

Firmware Download Link: https://www.tp-link.com/us/support/download/archer-ax1800/

Framework Used: Firmadyne, Static Analysis

Procedure:

1. Code can be found in repository at: TPLINK_AX1800/_ax1800v1.20-up-ver1-3-2-P1[20210901-rel69355]_sign_2021-09-01_19.18.14.bin.extracted/ubifs-root/1972537161/rootfs_ubifs/www/cgi-bin/luci

```
#!/usr/bin/lua
require "luci.cacheloader"
require "luci.sgi.cgi"
luci.dispatcher.indexcache = "/tmp/luci-indexcache"
luci.sgi.cgi.run()
```

2. Archer1800 uses MD5 hashing for their firmware. Firmware also hashed using MD5. Use the extractor to recover only the filesystem, no kernel (-nk), no parallel operation (-np), populating the SQL server at 127.0.0.1 (-sql) with the Netgear brand (-b), and storing the tarball in images. - Tag ID is 4

```
>> Database Image ID: 4

/home/nicholas/Desktop/Analysis/firmadyne/Archer AX1800_V1.2_210901.zip
>> MD5: 7544bdf8e02f330137fbd77562990b30
>> Tag: 4
>> Temp: /tmp/tmp51qrqv_o
>> Status: Kernel: True, Rootfs: False, Do_Kernel: False,                Do_Rootfs: True
>>>> Zip archive data, at least v1.0 to extract, name: Archer AX1800_V1.2_210901/
>> Recursing into archive ...

/tmp/tmp51qrqv_o/_Archer AX1800_V1.2_210901.zip.extracted/Archer AX1800_V1.2_210901/GPL License Terms.pdf
        >> MD5: 360ae19273142e70e23825c3920507f8
        >> Skipping: application/pdf...

/tmp/tmp51qrqv_o/_Archer AX1800_V1.2_210901.zip.extracted/Archer AX1800_V1.2_210901/How to upgrade TP-LINK Wireless Router
        >> MD5: 263625e91f946aedb3c56225e2e47611
        >> Skipping: application/pdf...
```

V. Firmware Anyalysis - Dynamic

This survey proves the contrary of what this paper is supposed to prove. Over time, firmware files have evolved drastically, and the file structures have changed. While it is ideally possible to detect vulnerability through regular expressions, this Survey will prove that over time, this is no longer possible. The algorithm to all our codes were provided above, and those codes were used to gather information. The conclusion of this will be provided later.

Parameters Surveyed:

1. CPIO Root Folder Addresses: Linux and Unix systems frequently use the archive file format known as CPIO (Copy In, Copy Out). The "root folder address" in relation to CPIO archives denotes the base directory or starting point of the archive, where files and directories are kept [27]. During the extraction procedure, this address is crucial for finding and extracting files.

2. SquashFS Root Folder Addresses: SquashFS is a Linux compressed read-only file system that is frequently used for effective data storage in embedded systems. When referring to the SquashFS file system, the "root folder address" denotes the root directory or starting point [28]. It is necessary in order to access and navigate files in the SquashFS archive.

3. Firmware Endianness: Proper data interpretation is necessary for firmware, a sort of low-level software that is kept in non-volatile memory. In computer memory, the term "endianness" describes the byte order. Big-endian, which places the most significant byte at the lowest memory address, and little-endian, which places the least significant byte at the lowest memory address, are the two varieties [29]. The term "firmware endianness" refers to the byte order that the firmware binary uses to make sure it is compatible with the hardware for proper operation.

4. Entropy: Gives a hint whether the firmware files are encrypted or not [30]. If they are, the entropy is 1.

5. First 4 bytes of xxd: First 4 bytes of xxd output of the firmware binary file. Tells if the firmware binary file is encrypted. DLink files are ideally encrypted and the first 4 bytes in xxd are ideally SHRS, of which there exists a decryptor.

6. Firmware Architecture: Whether the firmware supports 32-bit or 64-bit indexing.

7. Architecture Variant: Whether the firmware is ARM based or unknown

Firmware list

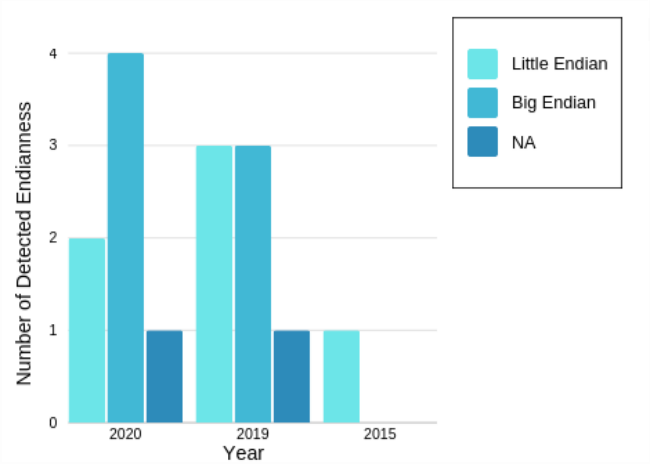
DLink: DLINK Router Firmware DAP-3662 v1.0.1, Dlink DIR 1960 v1.11, Dlink DIR 1960 v1.02, Dlink DIR 3040 v1.13, Dlink DIR 2660 v1.04, Dlink DIR 1760 v1.01, Dlink DIR 882 v1.30, Dlink DIR 867 v1.10, Dlink DIR 867 v1.30, Dlink DIR 2660 v1.02, Dlink DIR 867 v1.20, Dlink DIR 2640 v1.11, Dlink DIR 3040 v1.13, Dlink DAP 2553 v3.06, Dlink DIR 1360 v1.02, Dlink DIR 882 v1.20

Netgear: NETGEAR WNDAP360 v2.0.0, NETGEAR JWNR2000 v3.0.0.32, Netgear R6200 v1.0.1.58, Netgear R6220 v1.1.0.114, Netgear R7000 V1.v.11.136, Netgear R7350 V1.2.0.92_1.0.1, Netgear R9000 V1.0.5.42, Netgear wgr614v1_1400, Netgear WNDR3400v3-V1.0.0.38_1.0.40, Netgear WNDR3400v3-V1.0.0.38_1.0.40

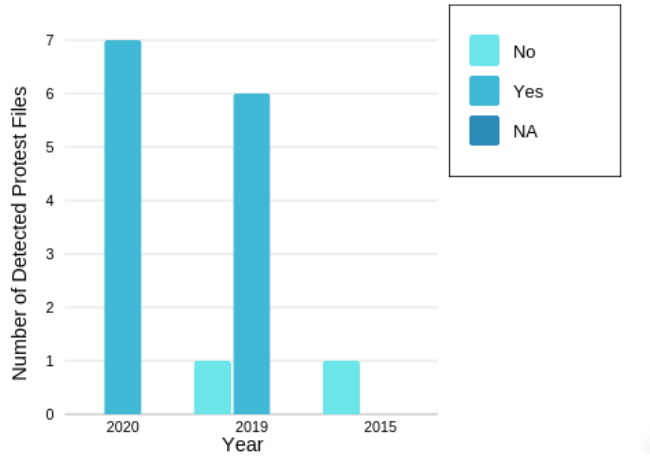
TPLink: TP-LINK AX1800 v1.2, TPLINK TL-940N v1.0, TPLINK TL-WR902AC v1.0, TPLink Archer C6 v2, TPLINK Archer BE550 v1, TPLINK Archer AX11000 v1, TPLINK Archer C7 v1, TPLINK Deco X55 v1, Deco X5000_V1_1.2.4, Deco_M3W_V1_1.0.7

Tenda: AP5 Firmware V1.0.0.2, AP4 Firmware V1.0.0.11, i9V2.0 Firmware V1.0.0.6, i29v1.0 Firmware V1.0.0.5, i27v1.1 Firmware V1.0.0.3, i25 CE, i24 Firmware V2.0.0.7, W9 Firmware V1.0.0.7, W6-S Firmware V1.0.0.4, W15-Pro CE,

Asus: ASUS RT-AX55, ASUS GS-AX5400, ASUS GS-AXE11000, ASUS RT-AC66U, ASUS RT-AC52U, ASUS RT-AC1200HP, ASUS RT-AX53U, ASUS RT-N66U, ASUS TUF-AX3000, ASUS RT-AC68U

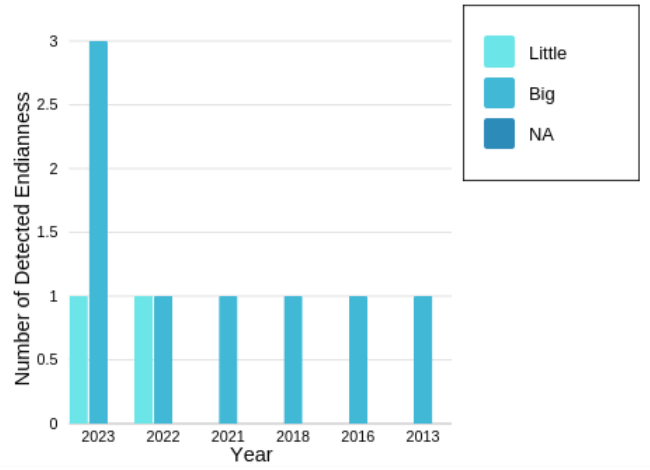


Graph 1: DLink Endianness Detection Trend



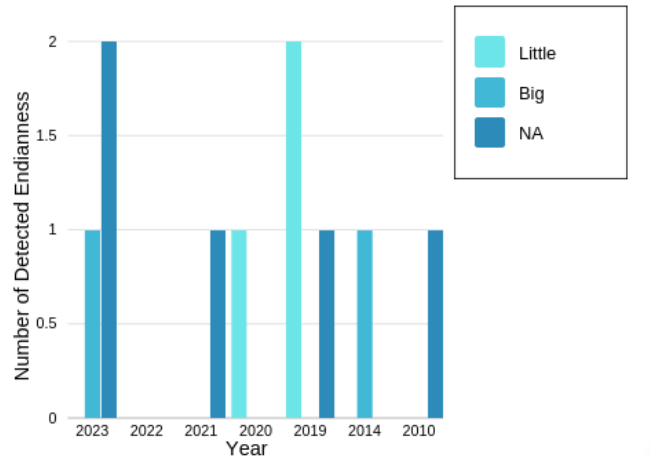
Graph 2: DLink Protest File Detection Trend

In DLink, the Endianness of framework files have trended from Little endian to big endian over the span of 2019 to 2020 (Graph 1). Getting samples of earlier framework files is tough since they no longer get posted on the download sites. Though, the one place where the algorithm works is detecting "Protest" Files (Graph 2) [31]. Protest files contribute to command injection attacks.



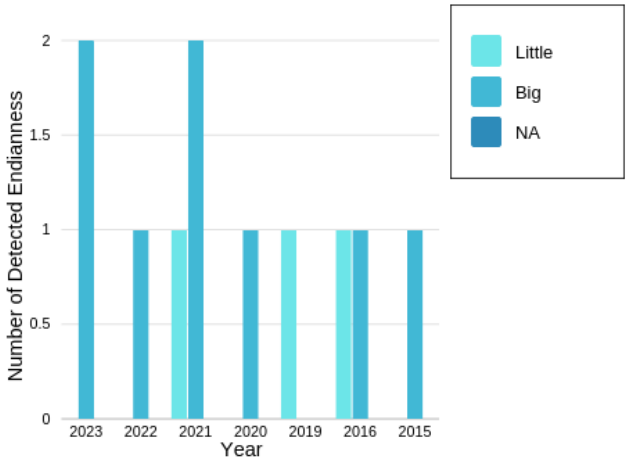
Graph 3: TPLink Endianness Detection Trend

When it comes to TPLink, the algorithm really cannot detect anything apart from Endianness, this is where this solution miserably fails. TPLink prefers Big Endian framework for its files and squashroot-fs file system.



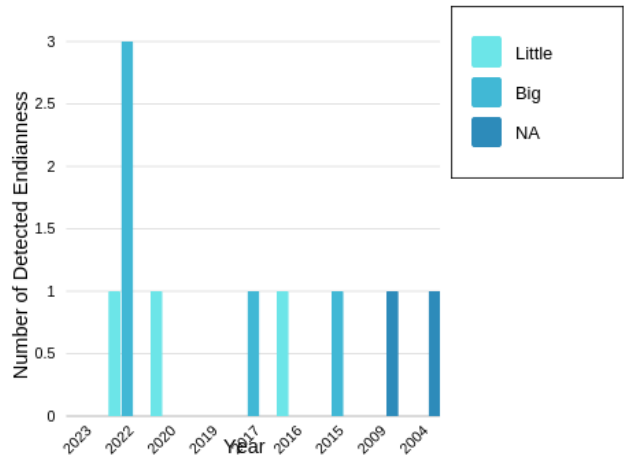
Graph 4: Tenda Endianness Detection Trend

Same goes for Tenda firmware files. We did include a few firmware files that were known to have command injection vulnerabilities, but the algorithm cannot detect it whatsoever. Increased firmware files under tenda are moving away from MIPS architecture over time.



Graph 5: Asus Endianness Detection Trend

When it comes to Asus firmware files, it does detect Endianness and whether the file supports 32-bit or 64-bit processing. The challenge with Asus firmware is with the fact that extracting these firmware files is very tedious, they tend to take a lot of time or sometimes they do not get extracted at all. There is no real endianness trend.



Graph 6: Netgear Endianness Detection Trend

Lastly, netgear. Netgear has one of the most concealed sets of framework files. Not only can we not download the framework files easily, but even if we scan it through our codes, apart from Endianness, no other information gets leaked whatsoever. You could really try any file of your choice. There is not really a clear trend in endianness.

The conclusion here would be that you really cannot get any firm data with mere codes. Even if your firmware is extracted and you can find certain strings using grep, the manner with which codes detect maliciousness of a firmware file can never really help detect any issue with the file anymore. So, this method of detection can be clearly striked out.

VI. Challenges with Analysis

One of the major problems that we encountered as a team while doing this project was the fact that, as soon as there was a vulnerability discovered and a CVE assigned against a given firmware, the firmware was either removed from the company downloads or a non-working patch was uploaded for public use. This would mean that every time we analyze such firmware say using firmadyne – the emulation would not be successful – you would either not get the interfaces to emulate, or in case you do, the emulation would fail with ioctl errors. Some such screenshots are provided below. There could be a scope for adaptability. We could find a different firmware which would closely resemble the vulnerable firmware to work on, but this would also mean that, this could be a firmware that was not enlisted under the said CVE – This may not have that vulnerability, or if it does, that could be a new discovery – which happens not as frequent, nor would anyone allow that to happen.

One other challenge that we faced is with respect to the firmware analysis tools – especially firmware analysis toolkit. FAT has not been maintained very well. The tool no longer has firmadyne that comes with it, nor does it have binwalk. The author assumes that you already have it or can figure it out. Additionally, it also tries to employ ghidra. Ghidra has a major problem – it is a jython code – Half Java, Half Python. To call this functionality, ghidra must be called – and python had once constructed a library that would interface with this tool – called pyhidra, but this library is not known to be able to function effectively. It works best on systems such as Ubuntu 18.04 or older – this would also mean that it was constructed with python 2.7 in mind. At this point, even Python 2.7 has been well maintained and constantly updated, and technicians mostly use python3. Pyhidra [32] was never maintained, hence this library flops heavily.

VII. Fixing Firmware Analysis Toolkit For Future Use

We did try to fix F.A.T. However, unfortunately we were not able to fix the pyhidra library and its co-relationship with ghidra tool. Using ghidra tool directly defeats the purpose. One possible fix was to reconstruct the manner in which ghidra was used, but no matter what, you would still need to call pyhidra. So, we focused on fixing the emulation part of the tool. One benefit of fixing this is that it emulates firmware without much of headache – just one command and you are set. We were able to fix that. One other benefit (and a problem) is that F.A.T. does not employ postgres SQL. Hence you do not have to worry about the overhead.

The fix was to write a script that directly downloads firmadyne and install it in the F.A.T. folder. Post this, the fat.py was modified to hold the firmadyne path and the system password – yes – it is a major flaw with this tool – hence this tool is best to be used on an empty VM. After which, the fat.config file was also modified to contain the same information. The fat.py file was copied to the firmadyne folder. After which, binwalk was installed on the system. The fat.py file was executed after this, and this time it executed and emulated with success.

VIII. Conclusion

The intent of doing this project was met – to prove that vulnerabilities still exist in router firmware, and they can be exploited any day. It is alarming that these vulnerabilities still

sit even after so many years, which means that one of the fixes would simply be to take firmware updates very seriously. One other conclusion would be that the firmware analysis tools need to be updated and maintained well, without which, analysis of firmware is horribly challenging. Our attempt here in that regard was to try to fix Firmware Analysis Toolkit, and we were partially successful at it. The partial miss was entirely because the necessary libraries and software were poorly maintained for re-useability and versatility.

When it comes to static analysis, according to our data, static analysis does not provide enough information for firmware analysis. Specifically, the companies we analyzed only offer downloads for firmware files which have fixed known older exploits. Therefore, static analysis provides minimal information. Companies also ensure the encryption of their firmware, making use of third-party tools like binwalk and firmadyne necessary. For most of our firmware files, the entropy of the firmware was 1, meaning the firmware is encrypted.

APPENDIX

Project GitHub Link: https://github.com/Simoary/INSE6120-Fall2023-Project-Group7

REFERENCES

[1] Router Firmware https://www.techopedia.com/definition/3177/router-firmware

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

[3] Unveiling Vulnerabilities: A Deep Dive into Firmware Penetration Testing- Part 1 https://ravi73079.medium.com/unveiling-vulnerabilities-a-deep-dive-into-firmware-penetration-testing-part-1-904599cd79be

[4] Firmadyne https://github.com/firmadyne/firmadyne

[5] Using Python for Firmware Test Development https://www.youtube.com/watch?v=J5ge4Wdghp4

[6] Automated Dynamic Firmware Analysis at Scale: A Case Study on Embedded Web Interfaces https://www.s3.eurecom.fr/docs/asiaccs16_costin.pdf

[7] [AttackDefense] Router Firmware Analysis + Reverse - Youtube

[8] Identifying Bugs in Router Firmware at Scale with Taint Analysis https://starlabs.sg/blog/2021/08-identifying-bugs-in-router-firmware-at-scale-with-taint-analysis/

[9] Firmaster: Analysis Tool for Home Router Firmware https://www.semanticscholar.org/paper/Firmaster%3A-Analysis-Tool-for-Home-Router-Firmware-Visoottiviseth-Jutadhammakorn/37c6a03ce66bf33d4ffeea8d2c4f098df0e91e2a

[10] Firmware Analysis Toolkit https://github.com/attify/firmware-analysis-toolkit

[11] Firmware Analysis https://book.hacktricks.xyz/hardware-physical-access/firmware-analysis

[12] Testing in QEMU https://www.qemu.org/docs/master/devel/testing.html

[13] Best Tools for Testing Wi-Fi MITM Attacks https://www.securew2.com/blog/best-tools-mitm-attacks

[14] Firmadyne Installation & Emulation of Firmware https://fwanalysis.blogspot.com/2022/05/firmadyne.html

[15] CWE-404: Improper Resource Shutdown or Release https://cwe.mitre.org/data/definitions/404.html

[16] CVE-2022-38873 Detail https://nvd.nist.gov/vuln/detail/CVE-2022-38873

[17] CVE-2022-31876 Detail https://nvd.nist.gov/vuln/detail/CVE-2022-31876

[18] CVE-2016-1559 Detail https://nvd.nist.gov/vuln/detail/CVE-2016-1559

[19] CVE-2015-8289 Detail https://nvd.nist.gov/vuln/detail/CVE-2015-8289

[20] CVE-2015-8288 Detail https://nvd.nist.gov/vuln/detail/CVE-2015-8288

[21] CVE-2020-13783 Detail https://nvd.nist.gov/vuln/detail/CVE-2020-13783

[22] CVE-2016-1558 Detail https://nvd.nist.gov/vuln/detail/CVE-2016-1558

[23] CVE-2016-1555 Detail https://nvd.nist.gov/vuln/detail/CVE-2016-1555

[24] CVE-2016-1556 Detail https://nvd.nist.gov/vuln/detail/CVE-2016-1556

[25] CVE-2016-1557 Detail https://nvd.nist.gov/vuln/detail/CVE-2016-1557

[26] CVE-2023-1389 Detail https://nvd.nist.gov/vuln/detail/CVE-2023-1389

[27] CPIO https://www.linuxjournal.com/article/1213

[28] Extract A Squashfs To An Existing Directory https://askubuntu.com/questions/437880/extract-a-squashfs-to-an-existing-directory

[29] Endianess https://wiki.raptorcs.com/wiki/Endianness

[30] A Look At Entropy Analysis https://fsec404.github.io/blog/Shanon-entropy/

[31] Command Injection Vulnerability in /bin/protest Binary on Multiple D-Link Routers https://www.tenable.com/security/research/tra-2022-09

[32] Callgraphs with Ghidra, Pyhidra, and Jpype https://clearbluejar.github.io/posts/callgraphs-with-ghidra-pyhidra-and-jpype/