IOSC^2: Identification & Analysis of Outdated Software Components and corresponding CVEs in IoT firmware

Asmita, Seema Upadhya

Abstract — Third-party software components (TPCs) are a crucial component of modern software systems. While being relatively small, a single vulnerable component can have an impact on a variety of businesses, applications, and individuals. Software component analysis (SCA) tools are already available for software-based products to detect such vulnerabilities, but there aren't many studies on SCA for embedded firmware, nor are there many supported tools. To fill in this gap, we present a static analysis of 107 firmwares where we identify the TPCs (hereafter referred to as packages) used in IoT devices (routers & cameras), their versions and the existing CVEs associated with these packages. We use the results from our analysis to evaluate the existing security scenario associated with embedded firmware and their corresponding packages. We discovered an unsettling situation where the majority of the devices still use out-of-date third-party packages which means that these devices include vulnerabilities related to the out-of-date versions of the packages. These third-party packages are updated at a faster pace than the firmware versions, the latter failing to keep up with the former. Furthermore, despite updates to the firmware versions, not all third-party packages are updated, putting these devices' security at serious danger. We also identify CWEs(Common Weakness Enumeration) associated with the corresponding CVEs identified in these third-party packages.

Introduction

Embedded systems are omnipresent in our everyday life. Eg. printers, mobile phones, home routers, and computer components. We are surrounded by IoT gadgets that contain embedded systems at their core in this technology-driven society. The software logic that runs on these resource-constraint embedded devices is firmware. As with any generic software system, embedded software (firmware) also comprises several third-party software components (packages). Firmware enables the necessary functionality using third-party software packages such as BusyBox, Openssl, etc. According to the findings of the study in [2], many of the evaluated apps were out-of-date and used third-party libraries, leaving openings for malicious actors. Similarly, some of these packages already have existing vulnerabilities that can be found in the CVE (Common Vulnerability Enumeration) database. In the case of embedded firmware, it is challenging to find these flaws because there is no source code available for analysis. Moreover, as in the case of generic software, there are SCA tools, but these tools are not equally useful in the case of the firmware.

Inorder to bridge the gap, we present IOSC^2 , an in-depth static analysis of 107 firmware versions of cameras and routers across different vendors to identify outdated software components and their corresponding CVEs. We begin our study by collecting firmwares from vendors based on Wayback[1] machine, search engine and Wikipedia results. We identify those

[1] https://archive.org/web/

firmwares that can successfully be extracted using our methodology. Once we have the third-party components used in the binary files, we gather the vulnerabilities associated with it and perform detailed analysis comparing different packages and their CVEs and different firmware versions of the same vendor for the same model.
We formulate the below research questions:
RQ1: What are the third-party software components in IoT devices' firmware that are commonly found?
RQ2: How likely is it to find outdated software components in such firmware?
RQ3: What are the corresponding existing vulnerabilities with respect to the outdated software components?
Our main research contributions are as follows :

1. We conducted a static analysis of real-world firmware[2] versions. We targeted outdated third-party software components in the firmware by analyzing the popular packages and finding their associated release dates and their corresponding CVEs.
2. We identified TPCs based on the number of occurrences in the firmwares across different vendors as well as the severity of CVEs associated with them to identify if certain components are more likely to be outdated (RQ1).
3. We analyzed the timeline of firmware releases as well as the release date of the version of the third-party packages present in the firmware (RQ2).
4. We compared the rate of updation of third-party packages against the updation of the firmware versions (RQ2).
5. We identified the weaknesses and severity associated with the CVEs corresponding to the third-party packages (RQ3).
6. We explored reasons as to why firmware developers can't keep up with the update in the third-party packages and associated dependencies.

We provide background and further motivation in Background and Motivation. We discuss previous works related to our study in Related Work, provide details of our implementation approach in Methodology, analysis details for the respective research questions in the Result & Analysis section, followed by Limitations section, and overall analysis details, future work has been discussed in Discussion and Conclusion section.

Background and Motivation

The majority of connected devices have firmware with vulnerable third-party packages, which poses numerous security threats. According to a recent Claroty[11] analysis, in the first half (1H) of 2022, vulnerability disclosures affecting IoT devices surged by 57% when compared to the prior six months. IoT devices were the source of 15% of these vulnerabilities, a significant rise from the 9% revealed in Team82's last analysis for the second half (2H) of 2021. They discovered a significant increase from the 2H 2021 report, which showed a nearly 2:1 difference between software (62%) and firmware (37%). The difficulty in patching firmware and the infrequent maintenance windows of these firmware are cited as reasons for this.

[2] https://github.com/asmitaj08/IOSC2

The Open Web Application Security Project, or OWASP[9] lists vulnerable and outdated components as sixth in its list of top ten security risks used in web applications. This includes software package versions which are outdated, end-of-life and nested dependencies. A popular example is the Heartbleed(CVE-2014-0160) bug found in “openssl”, a popular open-source package. Heartbleed allowed anyone on the Internet to view the memory of the systems that had the affected version of openssl in their firmwares. Affected companies included almost all the popular ones such as Google, Yahoo, Dropbox. OWASP lists that vulnerable and outdated components is the only category not to have CVEs mapped to the included CWEs which serves as an important motivation for our work. Inorder to prevent such attacks , OWASP[9] suggests continuous monitoring of Common Vulnerability and Exposures (CVE) from sources such as National Vulnerability Database (NVD) [4] for vulnerabilities in these components.
Ripple20[12] in the year 2020 presented 19 vulnerabilities in IoT devices which included a wide range of devices from major corporations such as HP,Schneider Electric and Intel. One of the vulnerabilities in this list could allow unauthorized access to the network-bound system. They analyzed how a tiny flawed component can be amplified by the supply chain factor and cause a ripple effect. The reason for these vulnerabilities to spread far and wide is due to the dissemination of these third-party component packages in multiple firmwares across different IoT devices.
There are existing software analysis tools which are not very useful in case of firmware. Firmware analysis is challenging since firmwares have hardware dependencies where neither the source code nor the hardware is available. Generally, static and dynamic techniques are used to perform the firmware analysis. Though there are several vulnerabilities that can be more efficiently identified by dynamic analysis, for this project, we have performed static analysis to identify the outdated third-party packages present in the firmware using the version number, and fetch associated vulnerabilities from CVE database NVD[6].

Related Work

With the rise of IoT devices and its omnipresence, much of the research has been focused on analyzing the firmware used on these devices. The comparison with respect to prior work is provided in Table A3.
Previous research work [1] has been performed on large-scale firmware analysis of security of embedded firmware. [1] presents a methodology for analyzing the security of embedded firmware, which involves extracting the firmware from the device and conducting a range of tests, including vulnerability scanning, reverse engineering, and binary analysis. The authors conducted this analysis on a dataset of over 3,000 firmware images obtained from various IoT devices.The paper analyzed vulnerabilities in the firmwares themselves such as hardcoded passwords, buffer overflows, and other types of memory corruption issues but it does not target outdated software components.
FIRMADYNE[3] is another related work that aligns closely to our research except that it performs dynamic analysis by booting an emulator with the filesystems extracted from the firmwares. The authors argue that traditional methods of analyzing firmware, which rely on static analysis, are not sufficient to fully understand the firmware's behavior. They show that FIRMADYNE can accurately identify the firmware's behavior and detect potential vulnerabilities

or malicious activities. However dynamic analysis was out of scope of this project because of time constraints.
Outdated and third-party libraries pose a risk as seen in [2]. Apps from Chinese app markets were more likely to contain vulnerabilities than apps from Google Play. The authors found that over 17% of apps from Chinese app markets had at least one vulnerability, compared to less than 10% of apps from Google Play. Inspired by this work, their ideas of finding outdated apps and usage of third-party libraries in Android apps has been applied to the domain of firmware.
The Open Web Application Security Project (OWASP) [9] provides a comprehensive guide on "Component Analysis" that outlines the importance of analyzing the software components used in web applications and the risks of using third-party open-source software and hardware components in modern software systems.
This project ties together the different ideas presented in these works and conducts a detailed analysis to identify outdated and third-party components and their vulnerabilities by performing a static analysis of different firmware versions.

Methodology

This section explains the different stages and approaches taken to find the answer to the proposed research questions. The overall implementation approach is shown in Figure A1.

Firmware binary dataset acquisition

The first stage of our dataset acquisition was selection of real-world firmware binaries. Initially, three products including routers, cameras, and printers were selected as these were the most common ones found in homes. The list of vendors for these products was obtained from search engines, Wikipedia, Wayback machine. Different vendor's websites were crawled for respective products and the firmware binaries were downloaded from the vendors' website.
The binaries that were unencrypted and were extracted by the selected extraction tool Binwalk [5] were collected for this project. Fetching firmware for printers could not be performed as none of them were able to meet this criterion. We obtained a total of 107 firmware binaries comprising routers (58 firmwares) and cameras (49 firmwares). The distribution is shown in Figure A3 and Figure A4. For routers, we were able to download the maximum number of firmware from the TP-Link, and for cameras, we got the highest firmware from the Reolink. Moreover, TP-Link was the only vendor that provided different versions of firmware for the same product model, but in the case of other vendors, for a single product model, only one version of the firmware was provided on the website. Apart from that, most of the firmwares downloaded from vendors such as DLink, Trendnet, etc. were either compressed or encrypted.

Challenges faced during firmware acquisition

Below were some of the challenges faced during dataset acquisition:

1. Certain firmware were not available on some of the vendors' websites.
2. Vendors like HP had their firmware encrypted which we could not extract using our methodology.

3. Some firmwares had a compression and encryption algorithm such that Binwalk could not extract the firmware binary.
4. Firmware found on untrusted websites weren't explored because of potential risk involved
5. The approach to fetch printer firmware by dumping the flash memory present in the device's hardware was unexplored. This fell outside the scope of our work.

Binary extraction/unpacking

The firmware inside camera and router devices has a Linux-based operating system. After performing the acquisition, we had a dataset of binary files. The next stage was to identify the third-party packages inside the binary by extracting it. Binwalk [5] was used to perform the binary extraction. It's an open-source tool most commonly used for extracting OS-based firmware. It has a bunch of signatures corresponding to the different file systems. It conducts signature analysis, identifies file systems, and performs dd extraction to extract the binary. After extraction, the file system and the corresponding package information were obtained. The next step was to identify the CVEs associated with the identified packages.

Tool selection and corresponding details

As the main aim of the project was to perform the analysis on the downloaded firmware, not the tool development, different existing tools were explored based on the below criteria :

1. Open source
2. Support binary (some SCA tools support source code and are language-specific).
3. Able to scan & identify the CVEs associated with the vulnerable packages.
4. Tool should be under active maintenance.
5. Easy to set up and support customized changes.

CVE binary (cve-bin) tool [6] was chosen as the tool for further implementation. It is open source, and maintained actively by the Intel git community. It scans binaries (doesn't need the source code of the packages) and identifies the CVEs associated with packages. It supports customized checker implementation, i.e making it feasible to include even the packages that are not currently supported by this tool. Internally it fetches NVD data in json form and stores locally, formats CVE in sqlite db for further analysis. For checkers, the regex pattern of the target packages is provided. For identifying the pattern, the package is executed using qemu [10] user mode. The reason behind using qemu user mode is that these binaries are either ARM or MIPS-based architecture, and so are the compiled packages within them. So, in order to execute them on x-86 based machines, qemu is used to provide the cross-architectural emulation. This tool then scans for files using the checker and identifies CVEs based on the version of the packages.

Third-party software component (packages) and corresponding CVE retrieval (main structure)

The overall implementation approach is shown in Figure A1. In stage 2, a python script was written to combine binwalk and cve-bin tool to automate the process of binary extraction, and corresponding retrieval of CVEs with respect to different packages present in the binary. Hence, the input to stage 2 were the firmware binaries, and the output were json files corresponding to each binary, and each of these json files has a list of packages and corresponding CVEs.
In stage 3, the bunch of json files obtained in stage 2 was processed to build a structure combining all these data such that all the binaries could be analyzed by traversing through one structure which is leveraged to find the answers to the proposed research questions.. The overall structure which is a nested dictionary is shown in Figure A2. The main structure has two keys for camera and router. Each of these camera and router keys have keys for different vendors from where the firmware was fetched. Each vendor has multiple keys for different firmware that were downloaded from their website for different product models/versions. Further, each of these firmware has multiple keys for respective packages present inside the firmware (like busybox, openssl, glibc, etc). Each of these packages has multiple keys for different versions of the packages found inside the firmware. Further, each of these versions of respective packages has multiple keys associated with different CVEs. Finally, each of these CVE keys has corresponding CVE details(values).

Method to answer RQ1

The main structure (Figure A2) was traversed and was leveraged to answer the respective research questions. The json form of the main structure is on the project's Github repo[3]. For RQ1, i.e the identification of commonly found third-party software components (packages) in devices' firmware, the number of occurrences of each of the packages was calculated, followed by the corresponding number of CVEs associated with each of these packages. Figure B1 shows the corresponding distribution graph of different software components (packages) with respect to the number of occurrences and the number of CVEs. Out of the 53 identified packages, the packages with a balanced combination of relatively higher occurrences and a higher number of associated CVEs were given closer attention.

Method to answer RQ2

To address RQ2, i.e., to identify the likelihood of finding the outdated versions of these identified third-party packages in firmware. Two approaches were taken to find the answer for this research question. In approach 1, the respective versions of each package were analyzed across all the collected firmware to find the difference between the release date of the package vs the release date of the firmware in which that package was used and the existing latest version of the firmware. Along with that, the update trend of the packages by respective third-party vendors was also analyzed. In approach 2, the respective product model of TP Link vendor with different firmware versions was chosen and analysis was done to observe the update of third-party packages in the newer versions of the firmware and to analyze if the

[3] https://github.com/asmitaj08/ecs289m-project

firmware vendor updates the third-party packages at the same rate as these packages are updated by their respective vendors.

Method to answer RQ3

To address RQ3, i.e., to identify the corresponding vulnerabilities and weaknesses details with respect to the identified third-party packages, the CVE numbers corresponding to different versions of the selected packages (as identified in RQ1) were fetched from the leaf node(value of dictionary) of the main structure. Further, NVD website was queried using NVD APIs to fetch the detail of vulnerabilities and weaknesses with respect to some of the identified CVE numbers. The query from the NVD website provided the json file corresponding to the CVE (vulnerabilities) from which the CWE [13] (weaknesses that lead to vulnerabilities) number was extracted. To perform the next stage, i.e. extract info about the CWE, the python package 'cwe'[4] was used which fetches the info from Mitre CWE [13] and provides the information corresponding to respective CWE number. Thus, the NVD and Mitre API were combined in a script to automate the process of fetching CWE info corresponding to a given CVE number. The analysis was done to identify the commonly found weaknesses that lead to such vulnerabilities.

Finally, an interactive UI was created (Figure A6, Figure A7, Figure A8) using cve-bin tool to fetch the CVE details corresponding to single firmware of corresponding product model of respective vendors.

Results & Analysis

This section discusses the observations after performing the detailed analysis of the collected firmware with respect to the proposed research questions.

RQ1

A total of 53 third-party software components (packages) were found across all the collected firmware (Figure B1, Table B1). Among the 53 identified packages, 10-11 were chosen for further analysis based on the criteria as mentioned in the Method to answer RQ1 section. Table B2 shows the list of these chosen packages. The full list of the component is provided in the google sheet shared in the github link. It was observed that busybox, openssl, glibc, dnsmasq, wpa_supplicant, gcc, curl, libcurl, sqlite, hostapd, and libtiff were among the most commonly found packages with corresponding higher number of CVEs amongst all the collected firmware across all the vendors.

RQ2

The results for RQ2 were quite alarming. It was found that older packages are still being used in the latest versions of the firmware. In case of Busybox, the latest stable version is 1.34.1 that was released in September 2021, but older versions of this package are still being used in some of the latest firmware versions. Busybox package 1.19.4 that was released in 2012 is still being

[4] https://pypi.org/project/cwe/

used in the latest firmware that was released in 2022. Multiple TPLink router products including router models - a2300, a7v5, archerC8, gx90, ax59, and c90 were using this package version. Similarly, in case of Openssl, though the version 0.9.7 was discontinued in 2007, it is still being used TP Link router Archer C3000 that was released in 2016, but vendors have neither updated it nor discontinued it. More details of different packages and corresponding version usage in the latest firmware has been provided in Appendix C. Figure C1, Figure C2, Figure C3, Figure C4, Figure C5 show that even in the firmware that were released in 2021-22 , older versions of the packages were more frequently used than the updated versions of these packages. For example, busybox 1.24.1 that was released in 2015 was found more than 1.31.1 that was released in 2019 including for firmwares that were released in 2022.
In approach 2 as mentioned in the Method to answer RQ2 section, it was observed that out of the multiple identified outdated third-party packages, very few of them were fixed in the latest versions of the newer firmwares. Observing the trend of 3 different releases of TPLink router GX90 firmware (Table C2), it was found that the firmware versions that were released in 2020, had 34 outdated third-party packages, even though the next version was released in 2021, but none of the packages were updated. In the latest version that was released in 2022. Out of 34 packages only 4 packages (openssl, dnsmasq, openvpn, and hostapd) were updated, rest of them were still the older package versions. Similar trend was observed for the router AX11000 (Table C1). Observations and graphs with respect to different product models with different firmware versions have been provided in more detail in Appendix C.
Overall quite a mixed pattern was observed, some of the firmware developers took initiative, discontinued the affected firmware, updated the packages, but some vendors kept using the older version. Some had multiple versions of the same package, some deprecated to older package versions in newer firmware, probably because they don't want to disrupt existing functionalities by changing the code base. They seem to build on the top of the existing packages without removing/updating the vulnerable packages as it is the easier option. There maybe certain dependencies involved in the product that are based on the older version of the packages. But there were some positive actions by vendors, like DLink, Trendnet , though the older version of the firmware was available to download on their website, their websites mentioned regarding its discontinuity to make customers aware about it. The more possible reasons behind this trend have been discussed in the Discussion section.

RQ3

As discussed in Method to answer RQ3 section, the observations across different CVE numbers gave insights about the kind of vulnerabilities that were found in the identified third-party packages, and the common weaknesses that lead to these vulnerabilities. It has been mentioned in detail in Appendix D. Most of the common weaknesses included 'use after free', 'buffer overflow', 'command injection, 'sensitive information leakage', 'infinite loop', 'improper certificate validation, etc. that lead to vulnerabilities like arbitrary code execution, denial of service attack, heap buffer overflow, side channel attacks based on cache access pattern, etc. Figure D1 and Figure D2 map the CVEs of gcc and busybox packages to corresponding CWEs; it's observed in Figure D2 that some of the weaknesses (CWEs) have occurred in multiple CVEs. For example, multiple CVEs including CVE-2021-42378, CVE-2021-42379, CVE-2021-42384, CVE-2021-42385, CVE-2021-42386 occurred because of one weakness,

CWE-416, i.e Use after free. Thus, there are good possibilities that many new vulnerabilities (CVEs) could occur because of the standard known weaknesses (CWEs). Further, the name of these CWE numbers is provided in Table D1 and Table D2. Table D3, Table D4, Table D5 provide more descriptive information about the vulnerabilities for busybox, wpa_supplicant and openssl packages respectively.

Limitations

This work was an initial step towards analyzing third-party packages in embedded firmware using existing and feasible tools. It has a lot of room for improvement by addressing certain limitations mentioned in this section. In RQ2, we compare different versions of firmware releases for just one vendor, TPLink. This is because we were unable to fetch different versions of the firmware for the same model for other vendors. In the case of other vendors, for a single product model, only one version of the firmware was provided on the website. We exclude analysis of other common devices such as printers due to failure of extraction of these firmware. The firmware that we tried to obtain were either encrypted or the filesystem could not be extracted by the tools that we used. If we had used an alternative approach of dumping the firmware directly from the device's flash memory, or working on tools that could support extraction of such binaries, it could have helped in collecting more diverse datasets across different vendors and products. We have analyzed a dataset of 107 firmware which is small and hence our results have a narrow scope.
Apart from dataset limitations, there are some limitations with our approach as well, due to the limited time we didn't try other potential and more efficient methods like use of dynamic analysis and machine learning that are currently being explored by many researchers. As these IoT device firmwares have hardware dependencies, performing dynamic analysis (as there are several vulnerabilities that cannot be correctly identified using static analysis) is a bit challenging considering the non-availability of hardware and the source code. In that case, the binary first needs to be emulated before performing the dynamic analysis like fuzzing. There are other research works using machine learning to perform binary similarity that can be leveraged to perform the component analysis but it was out of the scope of this project.

Discussion and Conclusion

The analysis of the third-party packages among the collected firmware binaries showed that these packages are widely used in IoT devices. Though, some vendors are indeed taking initiative to secure their firmware by encrypting them, making them available only to authorized users rather than making it directly available on the website, frequent package updates, notifying customers about the discontinuity of the vulnerable versions of the firmware. These are positive steps that make it harder for attackers to fetch and exploit the vulnerabilities in the firmware. But it does not provide the transparency regarding the security of the third-party packages used in these firmwares. It was observed that these third-party packages get updated quite frequently because of the continuing identification of CVEs associated with these packages. The package vendors mention on their website regarding the details of vulnerabilities, corresponding patches and updated package version. Unfortunately, on the other hand, these firmware are updated in a year or two time range, hence they are not updated with

the latest update of the third-party packages used in these firmwares. Thus the rate at which these third-party packages are updated is not the same as the rate at which these firmware are updated.

Another alarming observation was that even though the firmware is updated after a year or so, the developers still use quite an older version of third-party packages despite known vulnerabilities in them. It was found that many firmwares had multiple versions of the same packages, i.e they don't remove the older vulnerable packages. Some kept using the older versions despite the availability of newer versions of the packages.

Having these observations, we started exploring the reasons or the challenges that are stopping firmware vendors from updating these third-party packages. As the development for these products start early, so even if there is an update for the package, the base code remains the same because of time and resource overhead. As a result, even the newer versions are dependent on the older versions of the packages. For example, certain utilities of firmware need specific versions of a particular package. In these cases, it becomes very difficult for developers to completely get rid of older packages, though we didn't find any specific way to identify the exact dependencies of these packages. Many times developers are not aware of all the vulnerable third-party packages being used in their firmware. Moreover, when it comes to embedded firmware there are not feasible tools that can help developers to identify such components and make them aware regarding the existing vulnerabilities and associated risks and mitigation approaches.

Thus, there is a lot of work that can be done as part of future work, including automated inclusion of third-party packages, CVE and CWE identification in the firmware development tools to inform developers beforehand. There is future scope of expanding this research for a wider range of products and more sophisticated methods as discussed in the Limitations section.

References

[1] Costin, A., Zaddach, J., Francillon, A., & Balzarotti, D. (2014). A large-scale analysis of the security of embedded firmwares. In *23rd {USENIX} Security Symposium ({USENIX} Security 14)* (pp. 95-110).

[2] Wang, H., Liu, Z., Liang, J., Vallina-Rodriguez, N., Guo, Y., Li, L., ... & Xu, G. (2018, October). Beyond google play: A large-scale comparative study of chinese android app markets. In *Proceedings of the Internet Measurement Conference 2018* (pp. 293-307).

[3] Chen, D. D., Woo, M., Brumley, D., & Egele, M. (2016, February). Towards automated dynamic analysis for linux-based embedded firmware. In *NDSS* (Vol. 1, pp. 1-1).

[4] "NVD", https://nvd.nist.gov/ (Accessed March 2023)

[5] "Binwalk", https://github.com/ReFirmLabs/binwalk (Accessed March 2023)

[6] "CVE-bin-tool", https://github.com/intel/cve-bin-tool (Accessed March 2023)

[7] "RedHat",https://access.redhat.com/security/security-updates (Accessed March 2023)

[8] "Existing SCA Tools", https://owasp.org/www-community/Component_Analysis (Accessed March 2023)

[9] "OWASP", https://owasp.org/Top10/A06_2021-Vulnerable_and_Outdated_Components/ (Accessed March 2023)

[10] “Qemu”, https://www.qemu.org/ (Accessed March 2023)
[11] “Claroty”, https://www.helpnetsecurity.com/2022/08/29/vulnerability-disclosures-iot-devices/ (Accessed March 2023)
[12] “Ripple20”, https://www.jsof-tech.com/disclosures/ripple20/ (Accessed March 2023)
[13] “CWE” , https://cwe.mitre.org/ (Accessed March 2023)

Appendix A - Overall

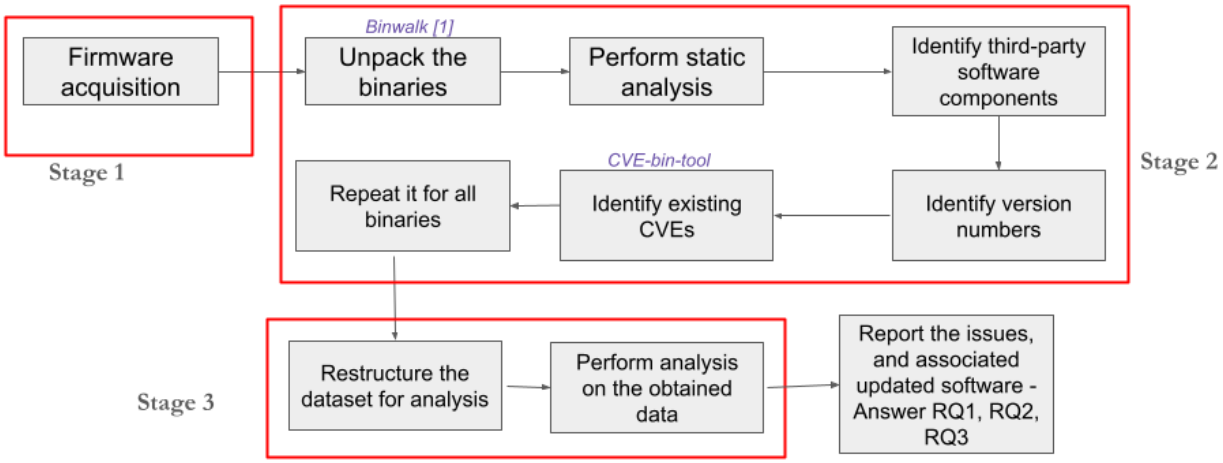
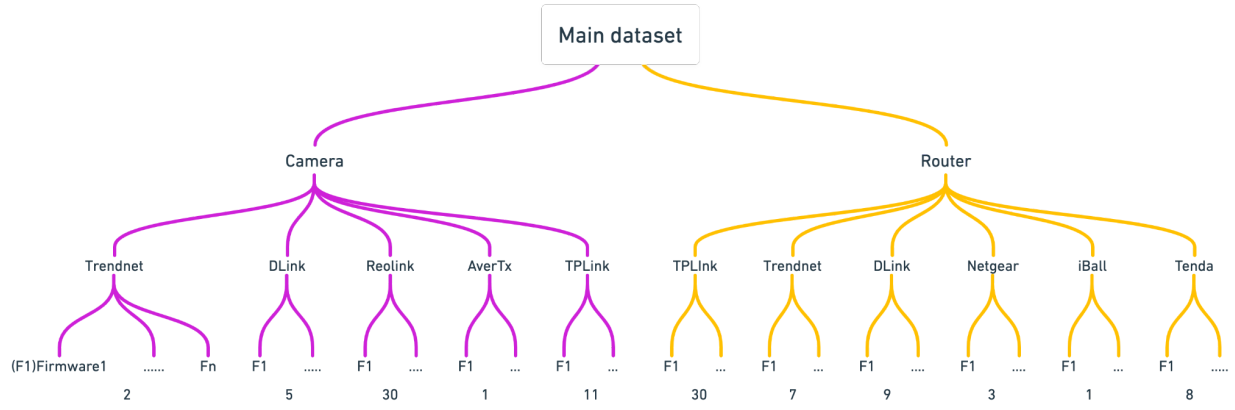


Figure A1: Overall implementation stages

The main structure -



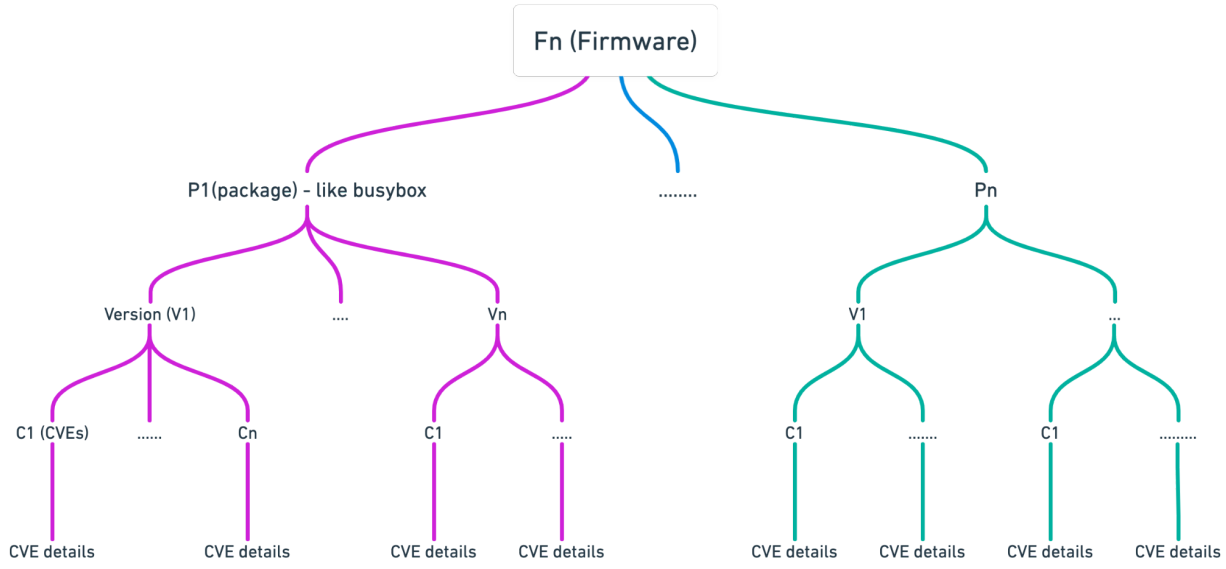


Figure A2: Overall main data structure

Dataset collection : Router

Table A1: Vendors for routers' firmware.

Router Vendors	No. of firmware downloaded
TP Link	30
Trendnet	7
DLink	9
Netgear	3
Tenda	8
iBall	1

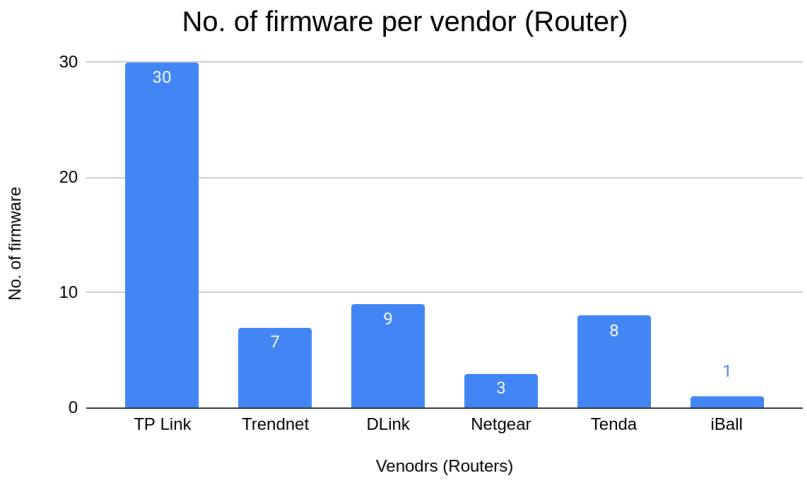


Figure A3: Distribution of downloaded router firmware across different vendors.

Dataset collection : Camera

Table A2: Vendors for cameras' firmware

Camera Vendors	No. of firmware downloaded
TP Link	11
Trendnet	2
DLink	5
Reolink	30
AverTx	1

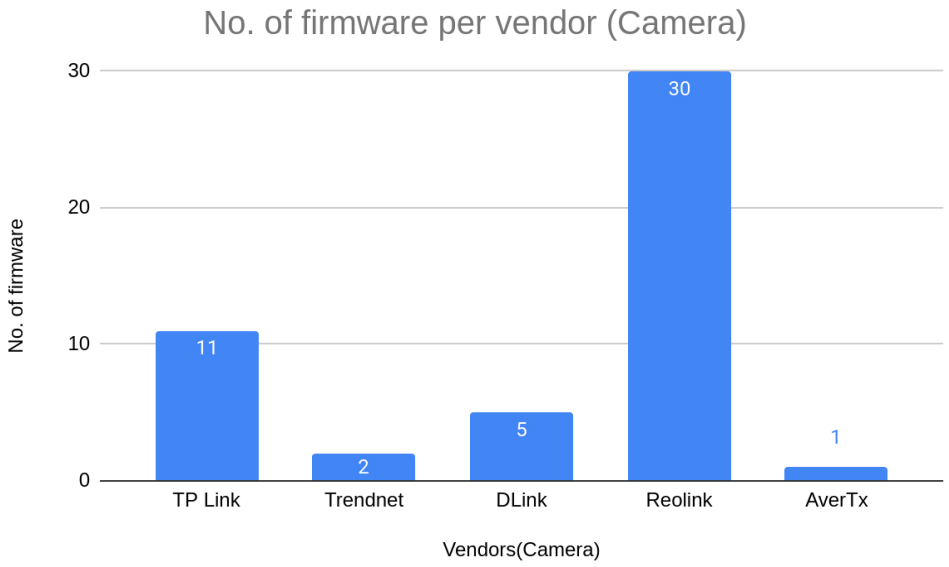


Figure A4: Distribution of downloaded camera firmware across different vendors.

```
>>> extracting.sample_dataset['camera']['dlink'].keys()
dict_keys(['DCS-935L_A1_FW_1.11.01_20171107_r4467.bin', 'DCS-6915_A1_V1.11.00', 'DCS-6915_B1_V2.0
1.00', 'DCS-960L_A1_FW_1.09.02_20191128_r4588.bin', 'DCS-8200LH_A1_FW_1.02.03_20161103_r4092.bin'
])
>>> extracting.sample_dataset['camera']['dlink']['DCS-935L_A1_FW_1.11.01_20171107_r4467.bin'].key
s()
dict_keys(['gcc', 'curl', 'libcurl', 'busybox', 'iptables', 'openssl', 'stunnel', 'point-to-point
_protocol', 'libjpeg', 'lua'])
>>> extracting.sample_dataset['camera']['dlink']['DCS-935L_A1_FW_1.11.01_20171107_r4467.bin']['gc
c'].keys()
```

Figure A5: Different third-party packages (like ‘gcc’, ‘curl’, ‘busybox’, etc) used in Dlink camera’s firmware DCS-935L_A1_FW version.

Comparison with prior work:

Table A3: Comparison with prior work.

Related works	Does it support firmware binary?	Does it target outdated software components (packages)?	Does it give existing CVE details based on older packages?	Does it involve dynamic analysis?	Does it give other statically analyzed info apart from CVE?	Is the research focused more on analysis than tool development?

[1] : Analysis of firmware	Yes	No	No	No	Yes	Yes
[2] : Beyond Google play	No	Yes	No	No	No	Yes
[3] : Dynamic Analysis	Yes	No	No	Yes	Yes	No
[8] :SCA Tools	No	Yes	Yes	No	No	NA
Proposed work	**Yes**	**Yes**	**Yes**	**No**	**No**	**Yes**

The CVE exploration UI :

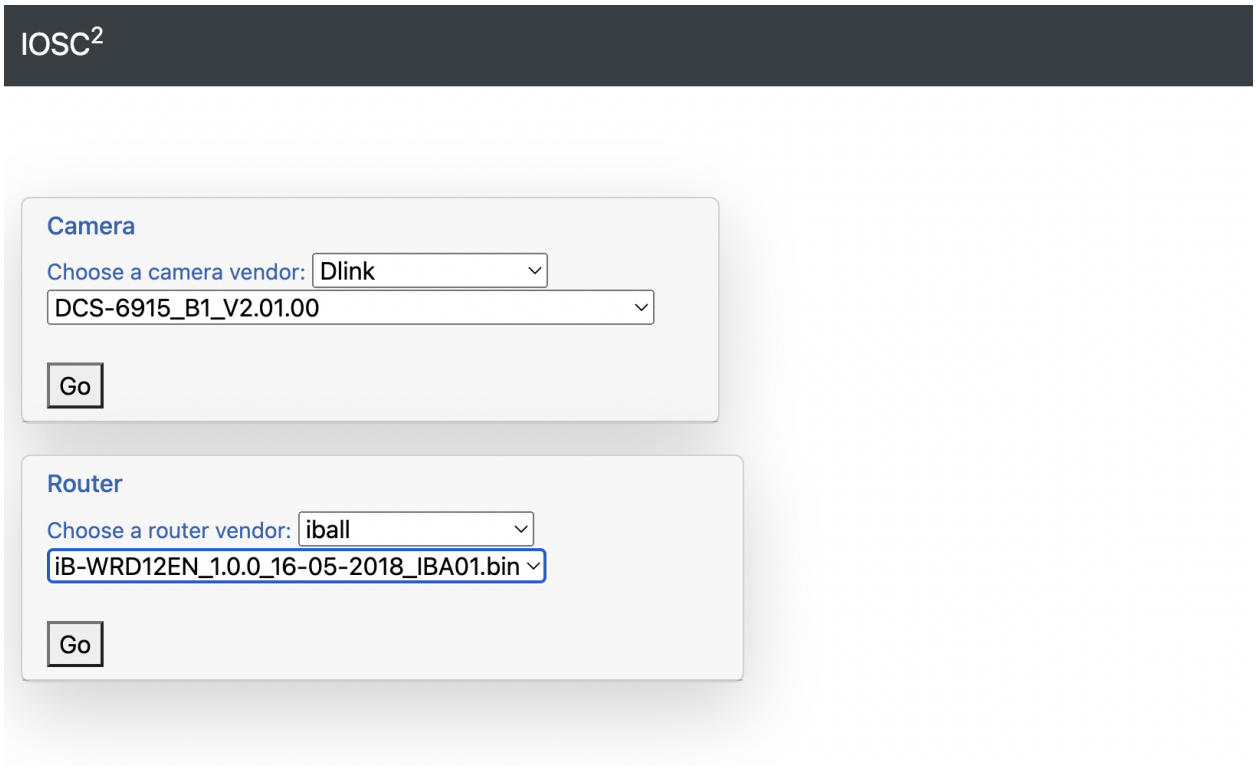


Figure A6: UI to select respective vendors and corresponding firmware model to explore third-party packages and corresponding CVE details.

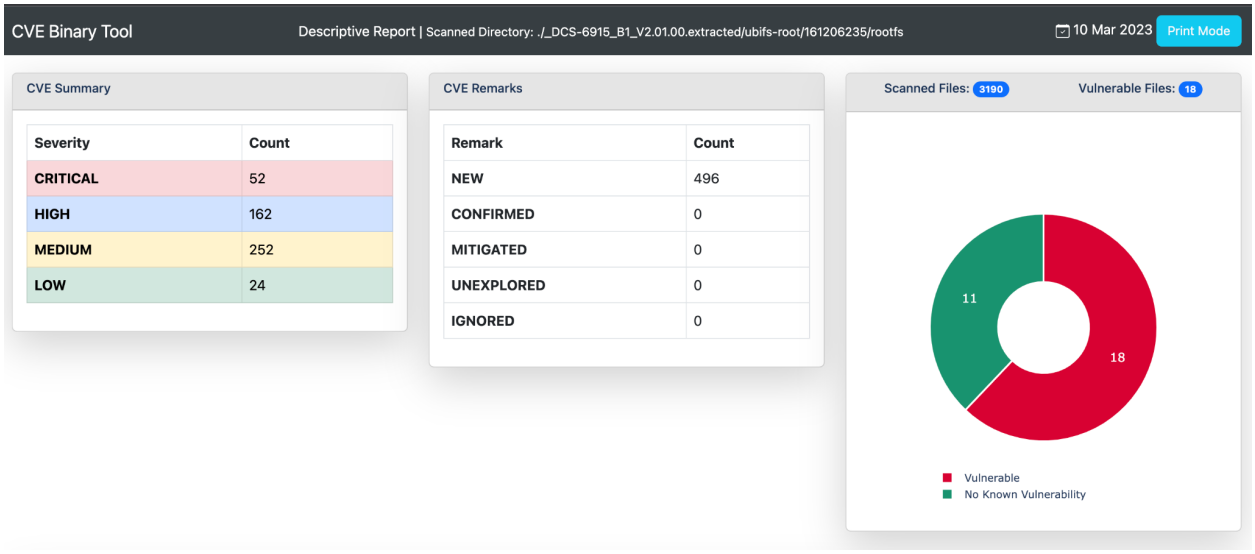


Figure A7: UI showing CVE severity present in the selected firmware.

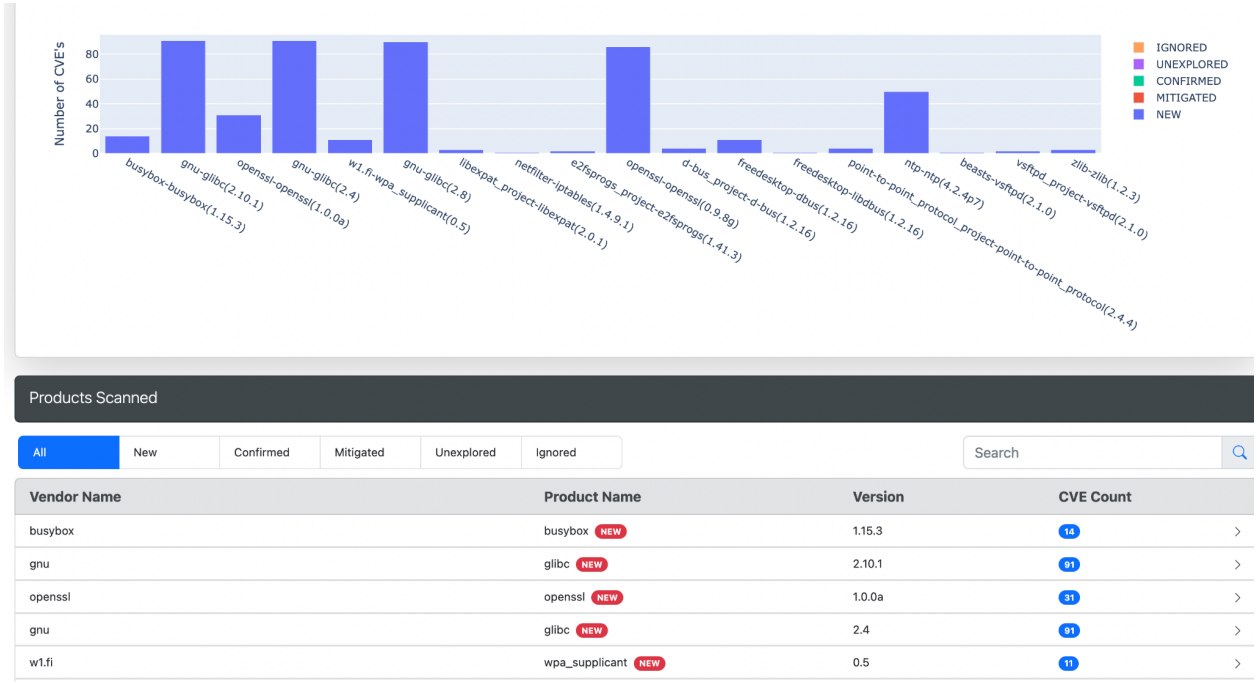


Figure A8 - UI showing distribution of different third-party packages in the selected firmware(as seen in Figure A4).

Appendix B - RQ1

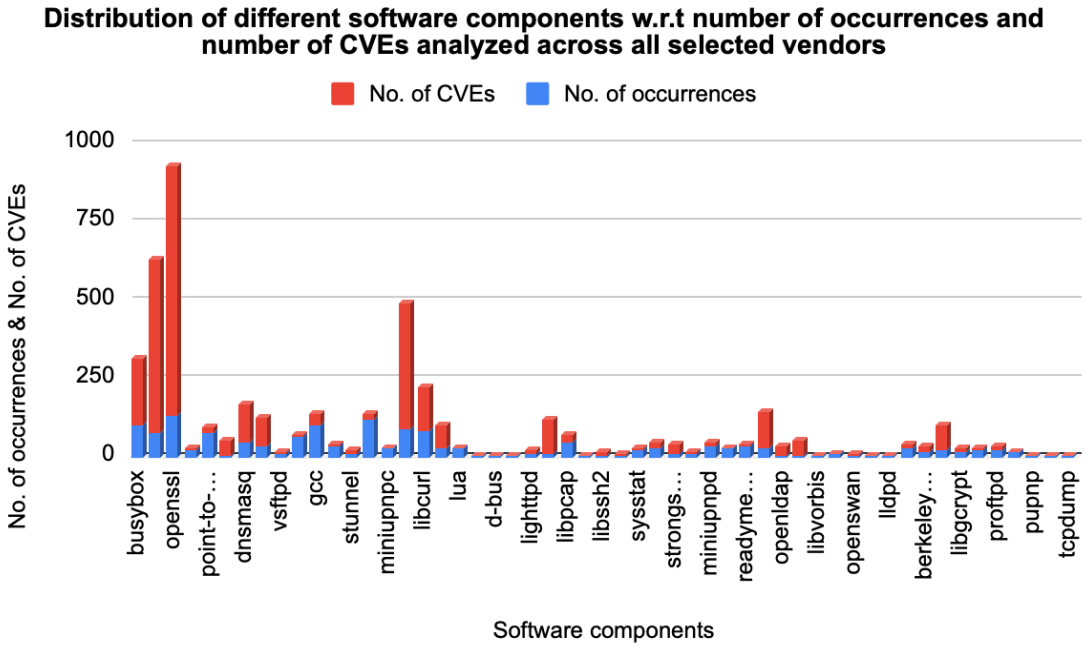


Figure B1: Distribution of different software components *w.r.t* number of occurrences and CVEs analyzed across all selected vendors.

Table B1: All identified software components w.r.t number of occurrences and CVEs analyzed within all downloaded firmware across all selected vendors.

S. No.	Software components	Total no. of occurrences of this component throughout the vendors	No. of CVEs
1	busybox	99	218
2	glibc	76	552
3	openssl	129	802
4	libexpat	25	4
5	point-to-point_protocol	75	20
6	ntp	3	50
7	dnsmasq	47	122
8	wpa_supplicant	35	91
9	vsftpd	8	8

10	iptables	64	9
11	gcc	99	39
12	libjpeg	34	5
13	stunnel	12	11
14	zlib	118	18
15	miniupnpc	26	4
16	curl	90	403
17	libcurl	82	140
18	sqlite	31	73
19	lua	27	4
20	e2fsprogs	2	2
21	d-bus	2	4
22	libdbus	2	1
23	lighttpd	11	11
24	ffmpeg	8	110
25	libpcap	45	25
26	quagga	3	1
27	libssh2	3	11
28	upx	3	10
29	sysstat	25	6
30	openvpn	28	16
31	strongswan	11	29
32	json-c	12	4
33	miniupnpd	32	13
34	bzip2	27	2
35	readymedia	33	6
36	hostapd	31	111
37	openldap	7	25
38	openssh	7	43
39	libvorbis	2	1
40	gmp	8	2
41	openswan	4	6
42	minidlna	2	1
43	lldpd	2	2
44	router_advertisement_daemon	31	10
45	berkeley_db	19	16
46	libtiff	21	78
47	libgcrypt	19	9

48	ncurses	25	6
49	proftpd	25	12
50	netatalk	14	2
51	pupnp	2	2
52	ipsec-tools	2	1
53	tcpdump	2	1

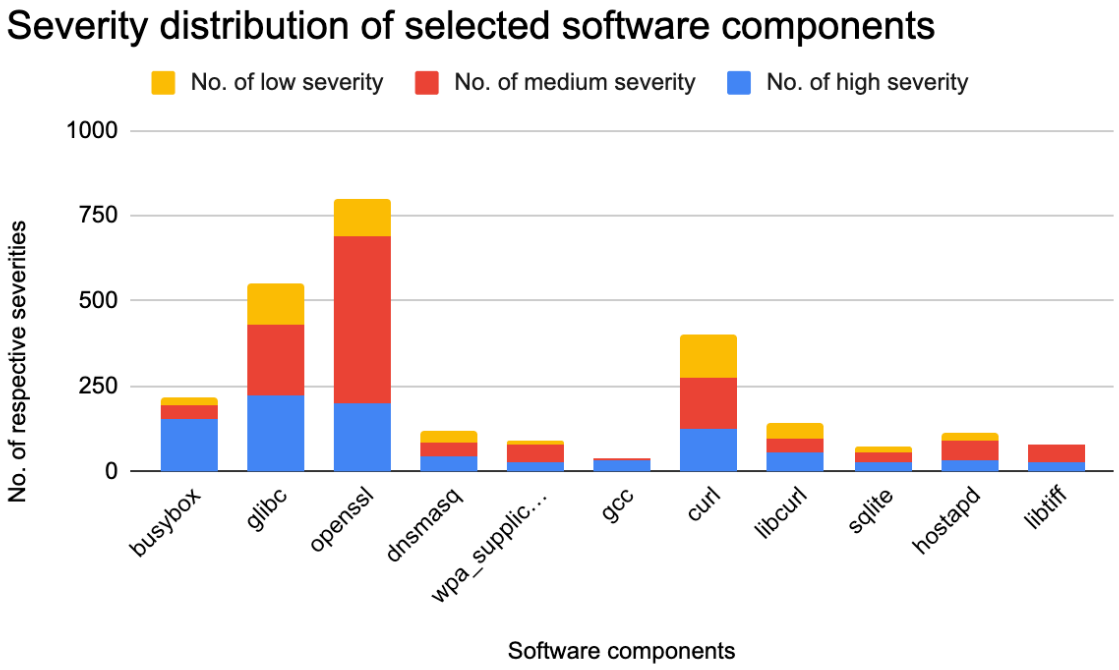


Figure B2: Severity Distribution graph for selected software components.

Table B2: Chosen software components and corresponding CVEs

S. No.	Software components	Package Vendors	No. of high severity	No. of medium severity	No. of low severity	Total number of CVEs
1	busybox	Busybox	154	40	24	218
2	glibc	gnu	221	212	119	552
3	openssl	openssl	199	491	112	802
4	dnsmasq	thekelleys	47	40	35	122
5	wpa_supplicant	w1.fi	27	49	15	91
6	gcc	gnu	31	8	0	39
7	curl	haxx	124	150	129	403
8	libcurl	haxx	57	37	46	140
9	sqlite	sqlite	26	29	18	73
10	hostapd	w1.fi	31	61	19	111
11	libtiff	libtiff	24	53	1	78

Appendix C - RQ2

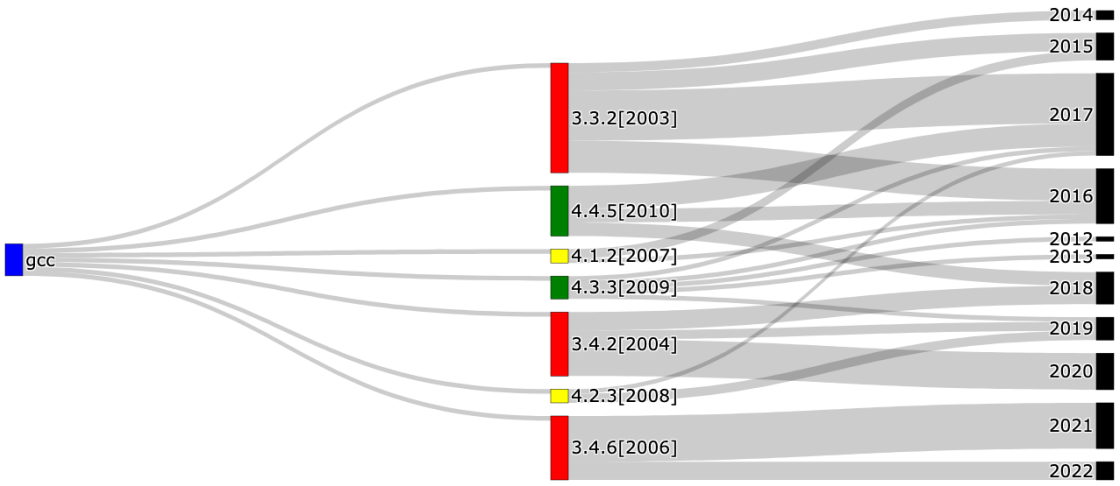


Figure C1: Mapping of release date of different versions of gcc package to the release date of firmware in which those packages were used.

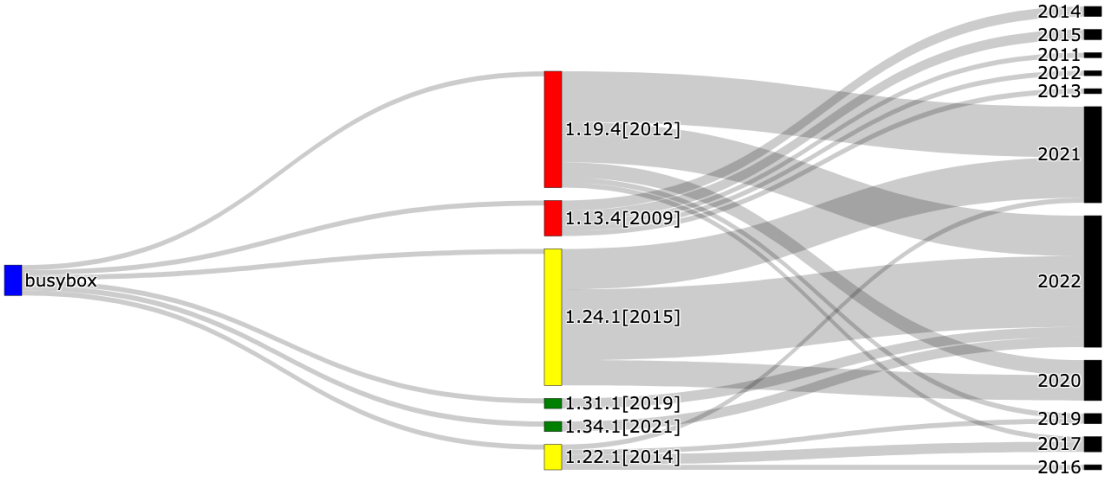


Figure C2 : Mapping of release date of different versions of busybox package to the release date of firmware in which those packages were used.

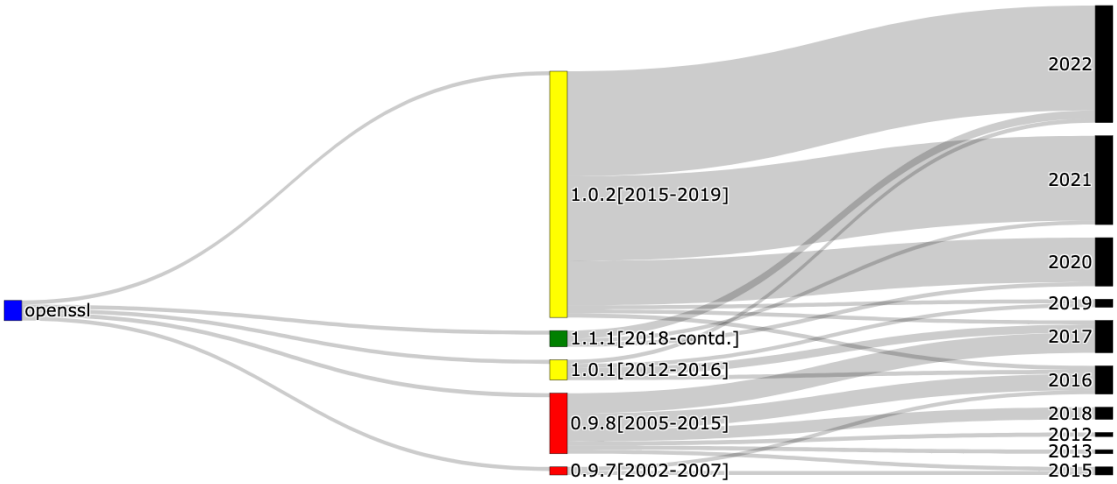


Figure C3 : Mapping of release date of different versions of openssl package to the release date of firmware in which those packages were used.

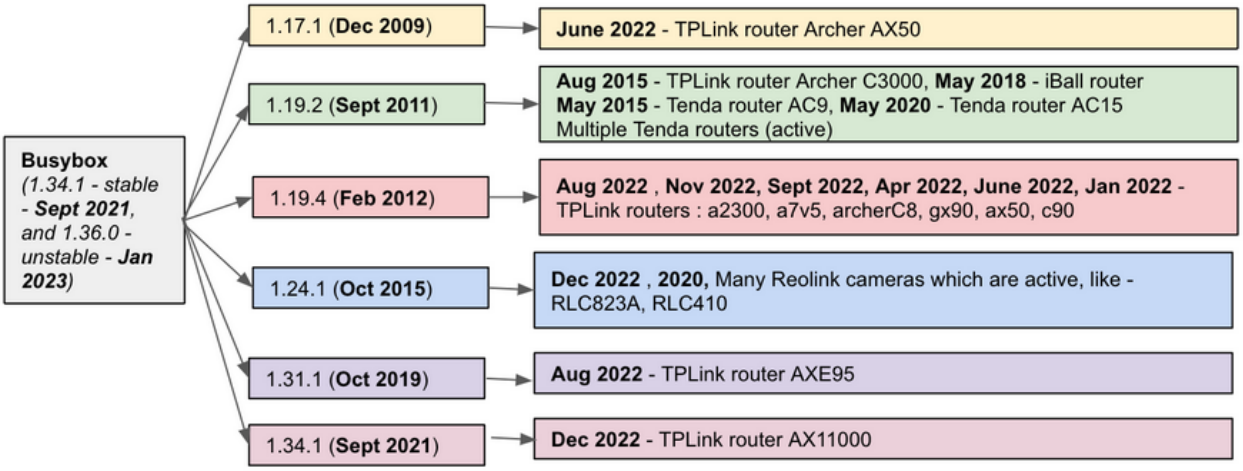


Figure C4 : Zoom into the mapping of busybox package.

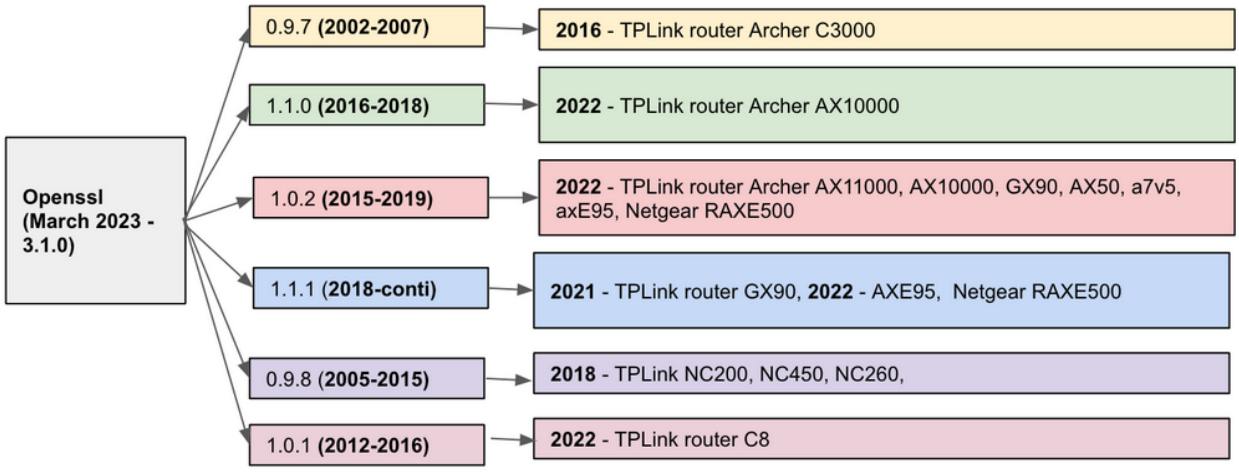


Figure C5 : Zoom into the mapping of busybox package.

Table C1 : Three versions of TPLink router AX11000 firmware (No. of vulnerable packages ~ 35 packages) showing the packages that were updated in the latest version of the firmware.

Version	Version release date	openssl version (Latest - 3.1.0)	busybox version (Latest - 1.34.1)	curl version (Latest-7.88.1)	dnsmasq version (Latest - 2.89)
ver1-2-3-P1	**2022-01-21**	1.0.2d	1.19.4	7.29.0	2.62
ver1-3-0-P1	**2022-05-27**	1.0.2d	1.19.4	7.29.0	2.62
ver1-3-2-P1	**2022-12-27**	**1.0.2u (2019)**	1.34.1 **(2021)**	7.79.1 **(2021)**	2.83 **(2021)**

Table C2 : Three versions of TPLink router GX90 firmware (No. of vulnerable packages ~ 34 packages) showing the packages that were updated in the latest version of the firmware.

Firmware Version	Firmware Version release date	openssl version (Latest -	dnsmasq version (Latest -	openvpn version (Latest -	hostapd version (Latest -

		3.1.0)	**2.89)**	**2.6.0)**	**2.10)**
ver1-0-0-P3-201111	**2020-11-11**	1.1.1b, 1.0.2d	2.62	2.3.8	2.8
ver1-0-2-P1-211124	**2021-11-24**	1.1.1b,1.0.2d	2.62 (2012)	2.3.8 (2015)	2.8
ver1-1-0-P1-220426	**2022-04-26**	**1.0.2u (2019)**	**2.83 (2021)**	**2.4.11 (2021)**	**2.9 (2019)**

Appendix D - RQ3

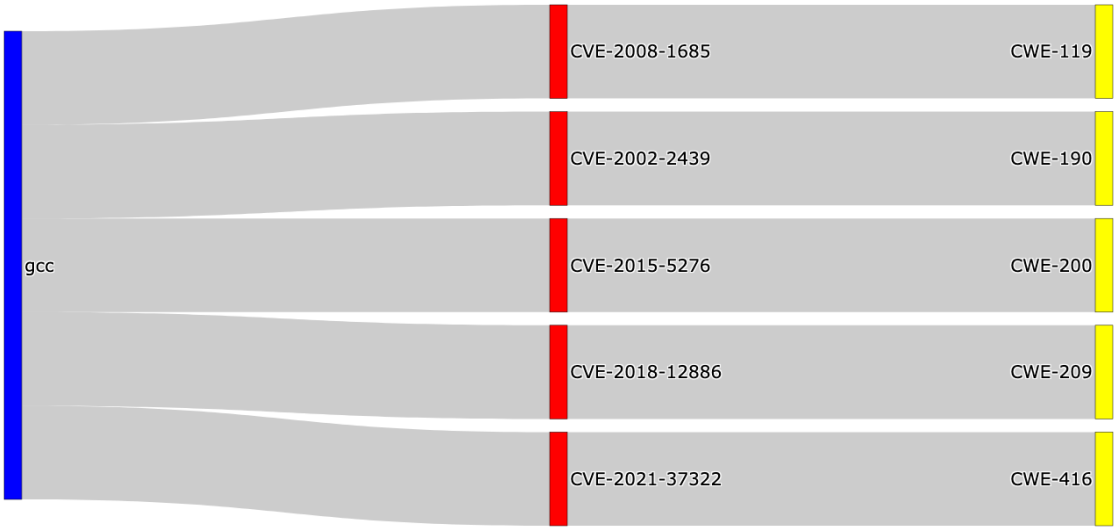


Figure D1 : Mapping of different CVEs found in different versions of gcc package to corresponding CWEs.

Table D1 : Description of CWEs found in gcc package

CWEs	Description
CWE-119	Improper Restriction of Operations within the Bounds of a Memory Buffer
CWE-190	Integer Overflow or Wraparound
CWE-200	Exposure of Sensitive Information to an Unauthorized Actor
CWE-209	Generation of Error Message Containing Sensitive Information
CWE-416	Use after Free

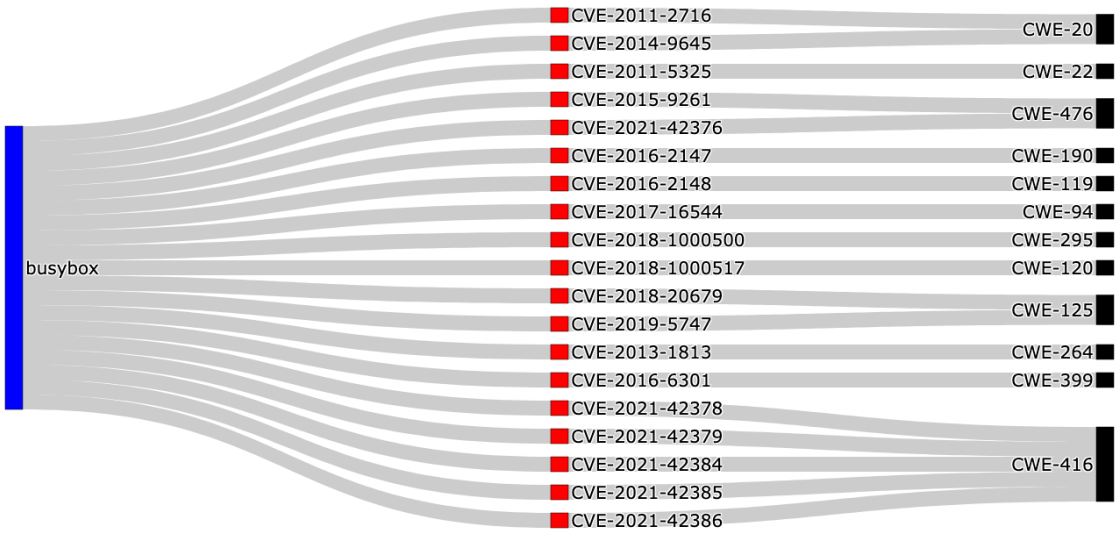


Figure D2 : Mapping of different CVEs found in different versions of busybox package to corresponding CWEs.

Table D2 : Description of CWEs found in busybox package.

CWEs	Description
CWE-20	Improper Input Validation
CWE-22	Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')
CWE-476	NULL Pointer Dereference
CWE-190	Integer Overflow or Wraparound
CWE-119	Improper Restriction of Operations within the Bounds of a Memory Buffer
CWE-94	Improper Control of Generation of Code ('Code Injection')
CWE-295	Improper Certificate Validation
CWE-120	Buffer Copy without Checking Size of Input ('Classic Buffer Overflow')
CWE-125	Out-of-bounds Read
CWE-264	Permissions, Privileges, and Access Controls
CWE-399	Resource Management Errors
CWE-416	Use After Free

Table D3 : Zoom into the CVE details for busybox package.

CVE number	Severity	Weaknesses	Vulnerability
CVE-2018-1000500	High	Improper certificate validation (CWE-295)	Missing SSL certificate validation vulnerability in The "busybox wget" applet that can result in arbitrary code execution
CVE-2021-42381	High	Use after free (CWE-416)	A use-after-free in Busybox's awk applet leads to denial of service and possibly code execution when processing a crafted awk pattern in the hash_init function
CVE-2018-1000517	High	Buffer Copy without Checking Size of Input ('Classic Buffer Overflow') (CWE-120)	Buffer Overflow vulnerability in Busybox wget that can result in heap buffer overflow
CVE-2022-28391	High	Improper Neutralization of Special Elements used in a Command ('Command Injection') (CWE-77)	BusyBox through 1.35.0 allows remote attackers to execute arbitrary code if netstat is used to print a DNS PTR record's value to a VT compatible terminal.

Table D4 : Zoom into the CVE details for wpa_supplicant package

CVE number	Severity	Weaknesses	Vulnerability
CVE-2022-23304	Critical	Observable Discrepancy (CWE-203)	wpa_supplicant before 2.10 are vulnerable to side-channel attacks as a result of cache access patterns. NOTE: this issue exists because of an incomplete fix for CVE-2019-9495
CVE-2019-9495	Low	Use of Cache Containing	The ability to install and execute applications is necessary for a

		Sensitive Information (CWE-524)	successful attack. Memory access patterns are visible in a shared cache. Weak passwords may be cracked
CVE-2021-27803	High	Use after free (CWE-416)	A flaw was found in the wpa_supplicant, in the way it processes P2P (Wi-Fi Direct) provision discovery requests. This flaw allows an attacker who is within radio range of the device running P2P discovery to cause termination of the wpa_supplicant process or potentially cause code execution.

Table D5 : Zoom into the CVE details for openssl package

CVE number	Severity	Weaknesses	Vulnerability
CVE-2022-0778	High	Loop with Unreachable Exit Condition ('Infinite Loop') (CWE 835)	It is possible to trigger an infinite loop by crafting a certificate that has invalid elliptic curve parameters. Since certificate parsing happens before verification of the certificate signature, any process that parses an externally supplied certificate may be subject to a denial of service attack.
CVE-2019-1543	High	Reusing a Nonce, Key Pair in Encryption (CWE-323)	OpenSSL allows a variable nonce length and front pads the nonce with 0 bytes if it is less than 12 bytes. However it also incorrectly allows a nonce to be set of up to 16 bytes. In this case only the last 12 bytes are significant and any additional leading bytes are ignored. It is a requirement of using this cipher that nonce values are unique.