Advances in Artificial Intelligence, Big Data and Algorithms
G. Grigoras and P. Lorenz (Eds.)

doi:10.3233/FAIA230854

Accurate Detection of Third-Party Library Version of IoT Firmware Based on GNN

Jingdong GUO[a,b], Ying WANG[c1], Zhuo WANG[a,b]
[a] *Henan International Joint Laboratory of Theories and Key Technologies on Intelligence Networks, Henan University, Kaifeng 475004, China;*
[b] *Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China*
[c] *Software Engineering Intelligent Information Processing Innovation Base-Subject Innovation Base of Henan Higher Universities, Henan University, Kaifeng 475004, China;*

Abstract. Third-party libraries are widely used in IoT firmware, and different versions of libraries have different vulnerabilities. Therefore, extracting versions of third-party libraries is of great significance for discovering known vulnerabilities of IoT devices. However, identifying the version of third-party library in IoT firmware is very challenging due to cross-architecture, cross-compiler, and cross-optimization options issues. To address this challenge, we present FirmAd, a GNN-based system that accurately detects third-party libraries' versions in IoT firmware. The system leverages a two-stage approach that calculates the similarity of different granularity features to obtain the final TPL version. We evaluate FirmAd on a large-scale dataset comprising 10,699 TPLs and achieve a version information identification accuracy of 92.68%, which is 8% higher than existing methods.

Keywords. GNN, Firmware, TPL, Version Identification

1. Introduction

The evolution of communication technology, exemplified by 5G, coupled with advancements in equipment hardware, has facilitated the extensive deployment and adoption of IoT devices, including routers, switches, network surveillance cameras, smart home appliances, and smart cars. These devices have found applications across diverse domains, such as smart homes, intelligent transportation systems, and smart cities. Firmware is a crucial component of IoT, and to expedite firmware development, Third-Party Libraries (TPLs) are extensively employed. Synopsys' 2023 Open-Source Security and Risk Analysis (OSSRA) Report[7]reveals that up to 92% of the source codes reviewed in the IoT domain comprise opensource codes. The extensive reuse of TPLs exposes firmware to potential security risks. For instance, the Heartbleed vulnerability in OpenSSL[2] affected millions of IoT devices, and this vulnerability only affected specific versions of OpenSSL, such as OpenSSL 1.0.1 - 1.0.1f. Therefore, identifying the versions of TPLs in IoT firmware is of paramount importance.

[1] Corresponding author: Ying WANG, Software Engineering Intelligent Information Processing Innovation Base-Subject Innovation Base of Henan Higher Universities, Henan University; e-mail: wangying@henu.edu.cn

To address the potential risks posed by unreliable TPLs, many researchers have focused on effective TPL version detection methods. Currently, there are several ways to obtain the versions of TPLs in firmware. One such method is dynamic testing[16], which involves running TPL in a simulated environment and using version-related commands to obtain version-related information. However, given that IoT devices have different architectures, this method requires a large and complex simulation environment, which is not conducive to large-scale analysis. Another method entails constructing a candidate library of a known TPL version and extracting binary constant features, such as function names and strings, for similarity comparison through static analysis. [8][9][10][12][15][17] Alternatively, structural features of functions, such as control flow graph (CFG), can be extracted to compare graph similarity and obtain the version of the target TPL. However, due to the significant impact of TPL cross-architecture cross-optimization compilation options and the number of functions to be compared, the accuracy and efficiency of this method in large-scale version detection need to be improved.

In this paper, we design and implement a system for accurately identifying the TPL version of IoT firmware, FirmAd. The system extracts coarse-grained and fine-grained features from TPL and utilizes Embedded-GNN in combination with function-level statistical features to calculate function similarity. TPL version detection is performed in two stages, which effectively enhances the accuracy and efficiency of TPL version detection. Given the absence of a public firmware library and TPL library, we invested significant time and effort in collecting 230 commonly used opensource projects from GitHub and SourceForge, manually compiling these projects into 19,700 binary files. We also collected 167 firmwares from 10 different vendors, spanning various device classes such as network cameras, routers, and switches. To evaluate FirmAd, we compared it with state-of-the-art version identification methods on widely used benchmarks. Experimental results demonstrate that FirmAd outperforms the state-of-the-art related work OSSPolice,[18] with an average accuracy of 92.68% for version detection, which is superior to OSSPolice's 83.7%.

2. Methodology

FirmAd is an automated system designed to identify the TPL version used in firmware. As illustrated in **Figure 1**, FirmAd comprises three main components: firmware collection and preprocessing, candidate TPL feature library construction, and target TPL version identification. The firmware collection and preprocessing module is responsible for collecting firmware files from various sources and preprocessing them in two steps: 1) identifying the necessary firmware information; 2) unpacking the firmware and extracting TPLs. The feature library construction module extracts the features of candidate TPLs in three steps: 1) collecting commonly used TPLs from opensource repositories such as GitHub and SourceForge; 2) automatically compiling them into binary files of different versions; 3) extracting different granularity features and storing them. The final version identification module extracts the coarse-grained and fine-grained features of the target binary and performs TPL version-level detection in two stages. The implementation of each module is discussed in the following subsections.

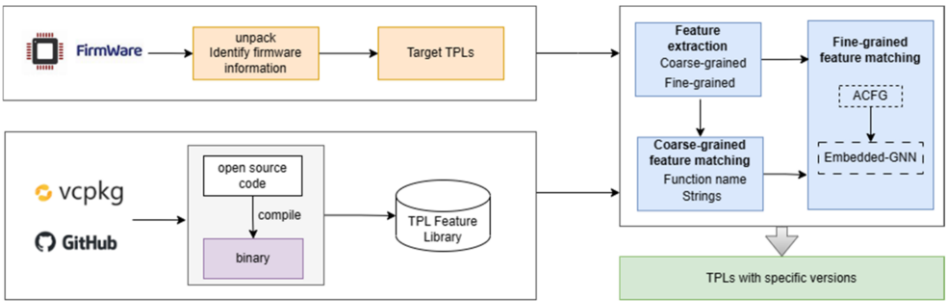

Figure 1. The workflow of FirmAd

2.1. Firmware Collection and Preprocessing

The firmware collection module is primarily responsible for gathering firmware from various manufacturers to construct a comprehensive firmware dataset. We collect different types of firmware from official websites of manufacturers such as Xiaomi, Huawei, ASUS, and others[1], as well as firmware files from open-source websites such as GitHub. Once the firmware is collected, it undergoes preprocessing. Metadata files and automated scripts are utilized to identify necessary information such as the firmware manufacturer, operating system, and architecture. Subsequently, Binwalk[3] is employed to unpack the firmware file, analyze the unpacked file, and extract the TPL file as the target TPL to be detected.

2.2. Construction of TPL Feature Library

The candidate TPL feature library construction module involves three main steps. Firstly, we collect the git repository of commonly used TPLs in firmware from open-source websites such as GitHub and SourceForge.[5] Next, we utilize the cmake tool, combined with automation scripts, to compile different TPLs into binary files of varying versions as candidate TPLs. Finally, using idapro [4]and feature extraction scripts, we extract coarse-grained features such as function names and strings, as well as fine-grained features such as attribute control flow graph (ACFG)[11] and function call graph (FCG), from candidate TPLs. Based on the features of different granularities of TPLs of different versions, we construct a TPL feature library.

2.3. Version Identification

The version identification module will be divided into three steps, 1) Extract different granularity features of the target TPL, which are the same as those used in the construction of the above-mentioned feature library; 2) Perform coarse-grained feature matching; 3) Use Embedded-GNN to calculate the similarity of the graph structure, use the local sensitive hash to calculate the similarity of the function-level statistical features, and combine the two similarity scores to obtain the TPL version information.

2.3.1. Feature Extraction

First, we implemented a script to extract coarse-grained features from TPL binary files. Coarse-grained features include function information (such as function name, function

comment, and function parameters) and string information. For each TPL, we summarize its common function names, such as sqlite3_column, which can be clearly distinguished from different types of TPLs to improve the efficiency of identification at the TPL library level.

Second, using idapro and script, extract ACFG and FCG from TPL. Attributes in ACFG include functional basic block-level features (such as the number of call instructions, jump instructions, and offset instructions) and function-level features (such as the number of functional basic blocks, CFG edges). According to Asteria[13] and Asteria-pro [14], the number of Callee and Caller in FCG can improve the accuracy of function similarity, so it is used as a statistical feature to calculate similarity

2.3.2. Coarse-Grained Feature Matching

For two sets of function names with different TPLs, the Jaccard similarity is used to calculate the similarity. When it exceeds a certain threshold, it is considered similar. The calculation formula is shown in Formula 1. For TPL's string features, the similarity of two strings is calculated using edit distance[6]. Combining the two similarities, a list of eligible candidate TPLs is preliminarily filtered out.

$$J(A,B) = \frac{A \cap B}{A \cup B} \tag{1}$$

2.3.3. Fine-Grained Feature Matching

We feed ACFGs of the target binary and the TPL binary into the GNN. After T iterations, the output vector is obtained, and the vector similarity is calculated using the cosine similarity. The parameters of the two GNNs are shared to form a Siamese network. The network structure is shown in **Figure 2**.

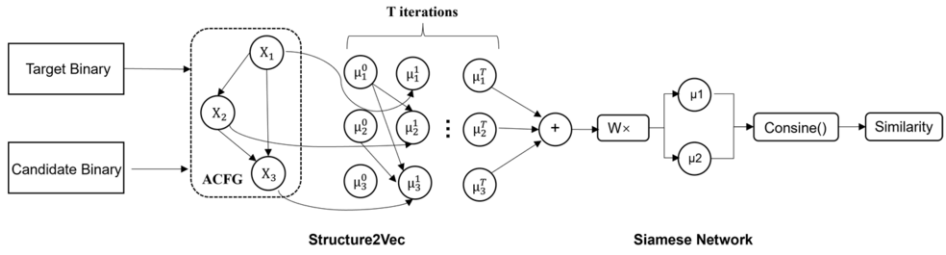

Figure 2.Network structure

3. System Evaluation

In this section, we firstly introduce the composition of the dataset. Next, we evaluate the performance of FirmAd and compare it with the two most outstanding works OSSPolice, BinaryAI [19].

3.1. Datasets

Ground Truth: In order to develop a dataset with cross-architecture and cross-

optimization options to train our Embedded-GNN network, we invested a lot of time and effort in crawling 230 commonly used open-source projects from Github and SourceForge. These projects were manually compiled into 19700 binaries covering three architectures (ARM, x86, x64) and four optimization options (O0, O1, O2, O3). We split it into training set, validation set and test set with a ratio of 8:1:1.

Real World dataset: To evaluate the ability to handle large-scale firmware TPL detection tasks, we collected 167 firmwares from 10 different vendors. The firmware extracted 10,699 binaries across various device categories such as IP cameras, routers, and switches. The composition of the dataset is shown in **Table 1**.

Table 1. Dataset scale

Dataset	Binaries	Content
Ground Truth	19700	cross arch and opti
Real World dataset	12699	cross arch and opti

3.2. Accuracy Evaluation

In this experiment, we use our method to identify TPL versions in real-world datasets and compare it with two other methods. The experimental results are shown in **Table 2**. The experimental results show that the recognition accuracy of our method in the TPL version is 92.68%, which is higher than OSSPolice's 83.7% and BinaryAI's 84.6%.

Table 2. The accuracy of TPL version identification

Experiment	Cross optimization	Cross architecture	Cross arch and opti
OSSPolice	**97.9%**	83.7%	83.7%
BinaryAI	96.9%	85.3%	84.6%
FirmAd	96.7%	**92.68%**	**92.68%**

3.3. Efficient Evaluation

The time consumption of component analysis in the experiment is shown in **Table 3**. In terms of detection efficiency, our method averages each TPL detection time is 8.36s. Although it is not the fastest detection speed, our method has achieved a good balance between accuracy and efficiency.

Table 3. Efficient evaluation

Experiment	Time-cost
OSSPolice	**0.161s**
BinaryAI	17.93s
FirmAd	8.36s

4. Conclusion

In this paper, we design and implement a GNN-based accurate detection system for the TPL version of IoT firmware. We conduct experiments on real-world datasets to evaluate our method, and the results show that in our dataset, the accuracy of TPL version detection can reach 92.68%, and the average detection time is 8.36s.

Acknowledgements

This work is supported in part by the Key Technology Research and Development Project of Henan Province under Grant 222102210055; in part by Henan Higher Education Teaching Reform Research and Practice Project(Graduate Education) under Grant 2019SJGLX080Y; in part by Postgraduate Education Reform and Quality Improvement Project of Henan Province under Grant YJS2022JD26; in part by Research and Practice Project of Postgraduate Education and Teaching Reform of Henan University under Grant YJSJG2022XJ039; and in part by Postgraduate Training Innovation and Quality Improvement Action Plan Project of Henan University under Grant (Talent Plan) SYLYC2022148, Grant (Education Innovation Training Base) SYLJD2022008, SYLKC2022028.

References

[1] 2023.Dlink firmwares website. http://files.dlink.com.au/Products/
[2] 2023.OpenSSL. https://www.openssl.org/
[3] 2023.Binwalk. https://github.com/ReFirmLabs/binwalk
[4] 2023.Idapro. https://hex-rays.com/IDA-pro/
[5] 2023. Sourceforge. https://sourceforge.net/
[6] 2022. Edit Distance. https://en.wikipedia.org/wiki/Edit_distance.
[7] 2023.https://www.synopsys.com/zh-cn/software-integrity/resources/analyst-reports/open-source-security-risk-analysis.html
[8] Xulun Hu, Weidong Zhang, Hong Li, Yan Hu, Zhaoteng Yan, Xiyue Wang, and Limin Sun. 2020. VES: A Component Version Extracting System for Large-Scale IoT Firmwares. In WASA, Dongxiao Yu, Falko Dressler, and Jiguo Yu (Eds.),Vol. 12385. 39–48
[9] Zhao B, Ji S, Xu J, et al. A large-scale empirical analysis of the vulnerabilities introduced by third-party components in IoT firmware[C]//Proceedings of the 31st ACM SIGSOFT International Symposium on Software Testing and Analysis. 2022: 442-454.
[10] Li S, Wang Y, Dong C, et al. LibAM: An Area Matching Framework for Detecting Third-party Libraries in Binaries[J]. arXiv preprint arXiv:2305.04026, 2023.
[11] Xu X, Liu C, Feng Q, et al. Neural network-based graph embedding for cross-platform binary code similarity detection[C]//Proceedings of the 2017 ACM SIGSAC conference on computer and communications security. 2017: 363-376.
[12] Gao J, Yang X, Fu Y, et al. VulSeeker: A semantic learning based vulnerability seeker for cross-platform binary[C]//Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering. 2018: 896-899.
[13] Yang S, Cheng L, Zeng Y, et al. Asteria: Deep learning-based AST-encoding for cross-platform binary code similarity detection[C]//2021 51st Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN). IEEE, 2021: 224-236.
[14] Yang S, Dong C, Xiao Y, et al. Asteria-Pro: Enhancing Deep-Learning Based Binary Code Similarity Detection by Incorporating Domain Knowledge[J]. arXiv preprint arXiv:2301.00511, 2023.
[15] David Y, Partush N, Yahav E. Firmup: Precise static detection of common vulnerabilities in firmware[J]. ACM SIGPLAN Notices, 2018, 53(2): 392-404.
[16] Qasem A, Shirani P, Debbabi M, et al. Automatic vulnerability detection in embedded devices and firmware: Survey and layered taxonomies[J]. ACM Computing Surveys (CSUR), 2021, 54(2): 1-42.
[17] Zhan X, Fan L, Chen S, et al. Atvhunter: Reliable version detection of third-party libraries for vulnerability identification in android applications[C]//2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE). IEEE, 2021: 1695-1707.
[18] Duan R, Bijlani A, Xu M, et al. Identifying open-source license violation and 1-day security risk at large scale[C]//Proceedings of the 2017 ACM SIGSAC Conference on computer and communications security. 2017: 2169-2185.
[19] Tencent Security Keen Lab. 2022. BinaryAI. https://www.binaryai.cn/.