Vulnerability Detection in IoT Firmware: A Survey

Wei Xie, Yikun Jiang, Yong Tang, Yuanming Gao
College of Computer
National University of Defense Technology
Changsha, China
xiewei@nudt.edu.cn

Ning Ding
Technician Department
Shandong Labor Vocational and Technical College
Jinan, China

***Abstract*—With the development of Internet of Things(IoT), more and more smart devices are connected into the Internet. The security and privacy issues of IoT devices have received increasingly academic and industrial attentions. Vulnerability detection is the key technology to protect IoT devices from zero-day attacks. However, traditional methods and tools of vulnerability detection cannot be directly used in analyzing IoT firmware.**

This paper firstly reviews related works on vulnerability detection in IoT firmware, previous researches are classified into four types i.e. static analysis, symbolic execution, fuzzing on emulators and comprehensive testing. Then, this paper points out that the specificity of vulnerability detection in IoT firmware is to detect logical flaws in embedded binaries which are built on the MIPS architecture. Finally, this paper proposes a method based on fuzzing and static analysis to detect authentication bypass flaws in IoT embedded binary servers. The proposed method is proved to be effective by verifying known CVEs as well as discovering unknown ones.

Keywords—vulnerability detection; logical flaws; IoT firmware; authentication-bypass

I. INTRODUCTION

IoT devices have suffered from security vulnerabilities so far. In 2010, Ang Cui et al.[1] develop an embedded device default credential scanner based-on nmap, which support 73 embedded device types. After scanning 3 billion IPs, about 4 million were recognized as embedded system devices, 540 thousand were vulnerable devices, the vulnerability rate reaches 13.81%. The authors retest 102,896 vulnerable embedded devices, approximately 96.75% of accessible vulnerable devices are still vulnerable after a 4-month period. In 2014, Mark Patton et al. [2] develop a framework to do a similar scanning work. They use python scripts utilizing the Shodan [3] API to select Supervisory Control And Data Acquisition (SCADA) devices into database, and then try to login with default password to identify vulnerable systems. Result shows that the vulnerability rate of HP printers running the JetDirect print server is the highest, reaches 41%. Rates are lower in web cameras and SCADA devices.

There have been some surveys addressing vulnerabilities in IoT firmware. For instance, Costin et al. [4] provide a systematic review of existing threats and vulnerabilities in video surveillance, closed-circuit TV and IP-camera systems based on publicly available data, and choose 7 criteria to describe attacks and mitigations at various levels. However, the survey doesn't focus on vulnerability detection, and only addresses monitoring devices rather than varieties of IoT devices. Recently, Hou et al.[5] provide a brief survey on vulnerability detection in embedded system firmware. They classify existing methods of firmware vulnerability detection as fuzzy testing, behavioral analysis, homology analysis and symbolic execution. However, there are still some important works beyond their classifications.

The main contents and contributions of this paper are as follows. The section 2 reviews related works on vulnerability detection in IoT firmware. The section 3 discuses difficulty and specificity of vulnerability detection in IoT firmware. The section 4 proposes an effective method based on fuzzing and static analysis to detect authentication bypass flaws in IoT embedded binary servers. Finally in section 5, conclusions and future works are summarized.

II. LITERATURE REVIEW

Classical software analysis methodology has been utilized in firmware vulnerability detection in recent years. Most of related works are published on top academic conferences like *Network and Distributed System Security Symposium* and *Usenix Conference on Security*. However, classical methods cannot be perfectly suitable for IoT firmware vulnerability detection, and there are still a lot of unsolved problems derived from the specificity of IoT firmware. Related works are classified as follows:

A. Static Analysis

Static analysis is a method to detect vulnerabilities in a program without executing the program on a real device or an emulator. This method has been utilized to compare homology of vulnerabilities among firmware images of different devices and vendors.

For instance, Costin et al. [6] propose a framework to perform firmware collection, filtering, unpacking and static analysis in a large scale. Their work implements several efficient static analysis techniques, and introduces a correlation technique which allows to propagate vulnerability information to similar firmware images. The authors gather about 32,356 firmware images from the Internet, discover that 693 firmware images affected by at least one vulnerability, and report 38 new CVEs. The framework shows a broader view on firmware and helps researchers to see how known vulnerabilities propagate.

978-1-5386-2129-5/17/31.00 ©2017 IEEE
DOI 10.1109/ICPADS.2017.00104

B. Symbolic Execution

Symbolic execution is a method of analyzing a program by computing symbolic states rather than implementing concrete values. It is able to determine what inputs cause each part of a program to execute, and theoretically possible to discover all vulnerabilities in a program.

FIE [7] is an open-source tool based on KLEE[8] which is a symbolic execution engine. It is designed for automatically analyzing firmware vulnerabilities of widely used MSP430 microcontrollers. It utilizes both state pruning and memory smudging to significantly improve code coverage and to support the ability of analyzing all possible paths in simple firmware programs. FIE supports finding two types of bugs: memory safety violations and peripheral-misuse errors. As for the limitations, source codes of the target firmware are required due to using KLEE, and users are required to manually verify all reported bugs. Imprecision arises due to misconfiguration and discrepancies between symbolic execution and native deployment. Besides, some bugs cannot be discovered until more techniques such as loop elision and improved state selection heuristics are introduced into FIE.

Firmalice [9] is the first firmware symbolic analysis system working at the binary level without requirements of instrumentation on target devices. It builds on top of Angr [10] which is a famous symbolic execution engine. It utilizes a novel model to detect authentication bypass flaws, such as intentionally hardcoded credentials, intentionally hidden authentication interfaces, and unintended access points without authentication protections. As for the limitations, manual work is needed when providing devices with security policy, and as a result, it cannot be applied in large-scale analysis. Besides, malicious firmware may use obfuscation to defeat static program slicing, or use sophisticated operations to confuse symbolic execution.

C. Fuzzing on Emulators

Fuzzing is considered to be the most effective method to detect vulnerabilities in traditional software. It usually requires dynamic execution contexts to challenge the tested targets. Unfortunately, there are millions of different IoT devices, some of which may not provide debugging interfaces. Therefore, a general emulator, which is capable to simulate heterogeneous firmware, is quite essential for fuzzing IoT firmware.

At Black Hat USA 2009, H Bojinov et al. [11] put forward a scanning work on embedded web interfaces of IoT devices. They overall scan a total of 21 devices spanning 8 device categories and 16 brands. They report more than 40 new vulnerabilities and find a new type of web vulnerability named XCS(Cross Channel Scripting). Their work greatly draws researchers' attentions on security of web interfaces in IoT devices. However, their scanning is not scalable because of using real devices rather than emulators.

Avatar [12] is an event-based arbitration framework that orchestrates the communication between an emulator and a physical device. By injecting a special software proxy in the embedded device, Avatar can execute the firmware instructions inside the emulator while channeling the I/O operations to the physical hardware. After the emulation, dynamic analysis like fuzzing can be implemented on the target firmware. In experiment, it is proved to be capable of doing reverse engineering and vulnerability detection as well as finding hard-coded backdoors. However, the emulated execution is much slower than execution on a real device, and it is hard to automatically emulate peripherals.

Firmadyne [13] is an open source framework to analyze Linux-based firmware in a large scale. It crawls and downloads firmware images from vendors' websites on the Internet. After unpacking firmware images and extracting embedded file-systems, software-based full system emulation is enabled to automate dynamic analyzing embedded binaries in a large scale. The authors evaluate the framework on 23,035 firmware images across 42 device vendors. Using a sample of 74 known exploits on the 9,486 firmware images that were successfully extracted, they discover that 887 firmware images spanning at least 89 distinct products are vulnerable to one or more exploits which include 14 new vulnerabilities that were discovered by the framework. However, the framework is invalid to discover vulnerabilities in kernels or kernel modules. Because it is based on a general emulator which only has three standard kernels of ARM, MIPS and MIPSEL architectures.

D. Comprehensive Testing

There are also some works using several different methods together to discover vulnerabilities in IoT firmware. Both dynamic and static solutions are introduced in these works to provide comprehensive results.

Costin et al.[14] present a new methodology to perform large scale security analysis of embedded web interfaces within IoT devices. They design a framework that achieves scalable and automated analysis of firmware. The framework was precisely developed to discover vulnerabilities in embedded devices using the software-only approaches and off-the-shelf static and dynamic analysis tools. From static analysis, 9,271 issues were found in 185 firmware images. From dynamic analysis, 225 high impact vulnerabilities were found in web interfaces.

Dai et al.[15] present a method for locating and exploiting firmware vulnerabilities by using dynamic fuzzing on emulators as well as static taint tracing. As a complement to fuzzing, static taint tracking improves code coverage and the granularity of dynamic analysis, facilitates to generate exploits efficiently. Their method is proved to be effective for vulnerability detection of routers, IP cameras and set top boxes. As for the limitations, some firmware images that need hardware supports cannot be accurately simulated by the emulators.

III. Specificity of Vulnerability Detection in IoT Firmware

We survey ratios of different vulnerabilities in different targets by searching keywords in the CVE list. According to the Table.1, Web applications(e.g. wordpress, phpMyAdmin) and web servers(e.g. Apache) mainly suffer from logical flaws(e.g. XSS, injection, authentication-bypass) which cannot cause system crashes. While memory vulnerabilities(e.g.

TABLE I. RATIOS OF DIFFERENT VULNERABILITIES IN DIFFERENT TARGETS

Targets	XSS	Inject	Authentication -bypass	Overflow
Wordpress	51.4%	39.2%	1.3%	0%
phpMyAdmin	38.5%	36.9%	1.2%	0%
Apache	12.9%	9.6%	4.8%	4.8%
Router	7.9%	6.7%	6.5%	6.5%
Camera	9.3%	12.6%	3.7%	10.2%
Firmware	9.8%	10.6%	5.5%	7.2%
Excel	1.4%	0.7%	0%	19.4%
Linux	1.2%	1.3%	0.7%	15.6%
Android	0.6%	0.7%	0.1%	6.8%

buffer-overflows) that often cause system crashes, are more likely to be discovered in binary programs and systems(e.g. Excel, Linux, Android).

It is worth noting that firmware of IoT devices(e.g. routers, cameras) have considerable percentage of logical flaws as well as memory vulnerabilities. It is largely because most IoT devices have web interfaces which are prone to be compromised by logical flaws, meanwhile these interfaces are commonly implemented by binary CGIs rather than scripts(e.g. PHP, ASP, JSP) that usually used in websites.

According to Figure.1, vulnerability detection in IoT firmware belongs to a different quadrant from vulnerability detection in other targets. In most cases, vulnerability detection in IoT firmware is aimed at discovering logical flaws in binaries. However, there is currently few research on this aim. On the one hand, most of logical flaw audit tools(e.g. RIPS[16]) require source codes or scripts of the targets, but CGIs, which perform actual actions in IoT firmware, can be obtained only in the form of binaries. On the other hand, although some fuzzing(e.g. AFL[17]) and symbolic execution (e.g. S2E[18]) tools support binaries, they are available only to detect memory vulnerabilities by monitoring system crashes that cannot be caused by logical flaws. Moreover, existing tools give little help even for detecting memory vulnerabilities in IoT firmware, because lots of embedded binaries are based on the MIPS architecture rather the x86 architecture which is commonly used in traditional software. Actually, few of existing tools supports for the MIPS architecture.

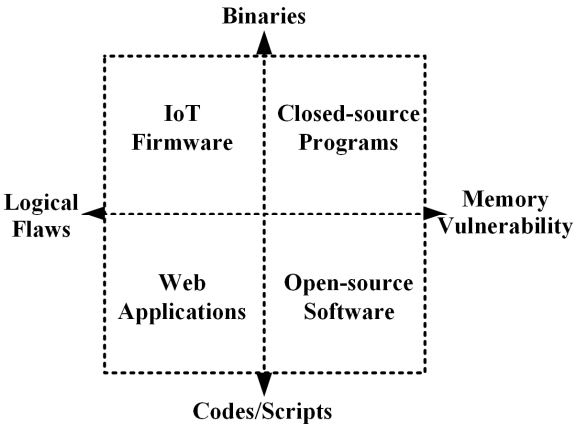

Fig. 1. Comparisons of vulnerability detection in different targets

IV. FUZZING AUTHENTICATION BYPASS FLAWS IN IOT EMBEDDED BINARY SERVERS

Authentication bypass flaws have obviously higher percentage in IoT firmware than that in other targets as shown in Table.1. And detecting authentication bypass flaws in IoT firmware is a typical example of discovering logical flaws in embedded binaries.

Firmalice[9] has been proposed for automatic detection of authentication bypass flaws in binary firmware. However, it is based on symbolic execution which is complex for non-expert users. And it cannot finally confirms a suspected flaw without the help of actual execution on a real device or at lest on a virtual emulator.

This paper proposes a new method, which combines fuzzing with static analysis, to detect and verify authentication bypass flaws in IoT embedded binary servers. The proposed method is shown in Figure. 2. Firstly, the static analyzer extracts all strings from the target binary web server, and outputs a list of keywords with parameter positions. Then, the dynamic fuzzer uses the keywords and the corresponding parameter positions to create and send a series of packets as test cases. Finally, the fuzzer checks the corresponding responses of the test requests to verify if an authentication bypass flaw is triggered. Its philosophy is quite simple: all keywords to trigger an authentication bypass flaw are believed to be contained in the target binary, and in most cases, these keywords are in the form of plaintext strings without encryption.

The proposed method can be proved to be effective by verifying some known authentication bypass flaws which can be classified into two types as follows.

A. CGIs without authentication protections

Some CGIs in an embedded web server may be accessible even for an unauthenticated user. However, some of the vulnerable CGIs may neither have separate binary files, nor

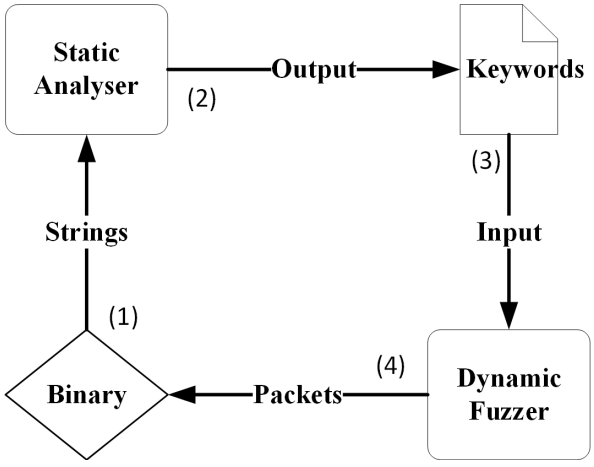

Fig. 2. The proposed method to detect authentication bypass flaws in IoT embeddedd binary servers

can be crawled by web spiders from any web pages. Therefore, these vulnerable CGIs cannot be noticed by a traditional fuzzer.

The proposed method is effective to discover CGIs without authentication protections. For instance, CVE-2017-5521 is reported that the "passwordrecovered.cgi", which outputs an administrator's password, is accessible without authentication in a series of netgear routers. Although there is not a single binary file named "passwordrecovered.cgi" in the vulnerable firmware, the proposed analyzer can find the vulnerable CGI's name as a keyword by statically analyzing the embedded binary web server "/usr/sbin/httpd". And then, it is easy for the fuzzer to discover the authentication bypass flaw by checking the status code of the CGI's response which is not "401 unauthorized" but "200 OK".

B. Sophisticated backdoors

Some IoT device have backdoors for unauthenticated users to remotely execute system commands. Some of the backdoors are created by devices' developers to simplify debugging tests, while some of the other backdoors are intentionally elaborated for malicious purpose.

Our method is also effective to discover backdoors. For instance, CVE-2013-6026 is reported that a backdoor is elaborated in a series of D-link routers. All HTTP requests without authentication are available once an attacker sets the user-agent header as "xmlset_roodkcableoj28840ybtide". It is hard for a computer fuzzer to understand the semantic string "xmlset_roodkcableoj28840ybtide" which means "edit by 04882 joel backdoor" in reverse order. However, our analyzer can extract the trigger string as a keyword from the embedded binary web server "/bin/webs". Then our fuzzer can find the authentication bypass flaw when sending a test case which has the sophisticated user-agent header.

The proposed method is also proved to be effective in discovering unknown new flaws(i.e. zero-day vulnerabilities). However, it is inappropriate to publish the details until the vendors fix the discovered flaws.

V. CONCLUSION

The main contents and contributions of this paper include: (1) Related works on vulnerability detection in IoT firmware are reviewed into 4 classes, which are static analysis, symbolic execution, fuzzing on emulators, and comprehensive testing. (2) The specificity of vulnerability detection in IoT firmware is pointed out. It is to detect logical flaws in embedded binaries which are built on the MIPS architecture. (3) A method based on fuzzing and static analyzing is proposed to detect authentication bypass flaws in IoT embedded binary servers. It is proved to be effective by verifying known CVEs.

Our future works include: (1) Improving the proposed static analyzer by using machine learning approaches to create more accurate keywords for the fuzzer. (2) Applying the proposed method to varieties of IoT firmware in a large scale.

Acknowledgment

This work is partially supported by the National Natural Science Foundation of China under Grant Nos. 61472437.

References

[1] Cui, Ang, and S. J. Stolfo. "A quantitative analysis of the insecurity of embedded network devices:results of a wide-area scan." *Twenty-Sixth Computer Security Applications Conference, ACSAC 2010, Austin, Texas, Usa, 6-10 December* DBLP, 2010:97-106.

[2] Patton, Mark, et al. "Uninvited Connections: A Study of Vulnerable Devices on the Internet of Things (IoT)." *IEEE Joint Intelligence and Security Informatics Conference* IEEE Computer Society, 2014:232-235.

[3] https://www.shodan.io/

[4] Costin, Andrei. "Security of CCTV and Video Surveillance Systems: Threats, Vulnerabilities, Attacks, and Mitigations." *International Workshop on Trustworthy Embedded Devices* ACM, 2016:45-54.

[5] Hou, Jin Bing, T. Li, and C. Chang. "Research for Vulnerability Detection of Embedded System Firmware ☆." *Procedia Computer Science* 107(2017):814-818.

[6] Costin, Andrei, J. Zaddach, and D. Balzarotti. "A large-scale analysis of the security of embedded firmwares." *Usenix Conference on Security Symposium* USENIX Association, 2014:95-110.

[7] Davidson, Drew, et al. "FIE on firmware: finding vulnerabilities in embedded systems using symbolic execution." *Usenix Conference on Security* USENIX Association, 2013:463-478.

[8] Michel, Sebastian, P. Triantafillou, and G. Weikum. "KLEE: a framework for distributed top-k query algorithms." *International Conference on Very Large Data Bases*, Trondheim, Norway, August 30 - September DBLP, 2005:637-648.

[9] Yan, Shoshitaishvili, et al. "Firmalice - Automatic Detection of Authentication Bypass Vulnerabilities in Binary Firmware." *Network and Distributed System Security Symposium* 2015.

[10] Yan, Shoshitaishvili, et al. "SOK: (State of) The Art of War: Offensive Techniques in Binary Analysis." *Security and Privacy IEEE*, 2016:138-157.

[11] Bojinov, Hristo, et al. "Embedded Management Interfaces: Emerging Massive Insecurity." *Black Hat Usa* (2009).

[12] Zaddach, Jonas, et al. "AVATAR: A Framework to Support Dynamic Security Analysis of Embedded Systems' Firmwares." *Network and Distributed System Security Symposium* 2014.

[13] Chen, Daming D., et al. "Towards Automated Dynamic Analysis for Linux-based Embedded Firmware." *Network and Distributed System Security Symposium* 2016.

[14] Costin, Andrei, and A. Zarras. "Automated Dynamic Firmware Analysis at Scale: A Case Study on Embedded Web Interfaces." *ACM on Asia Conference on Computer and Communications Security* ACM, 2016:437-448.

[15] Zhonghua Dai, et al. "Research on the localization of firmware vulnerability based on stain tracking." *JOURNAL OF SHANDONG UNIVERSITY(NATURAL SCIENCE))*51.09(2016):41-46.

[16] http://rips-scanner.sourceforge.net/

[17] http://lcamtuf.coredump.cx/afl/

[18] Chipounov, Vitaly, V. Kuznetsov, and G. Candea. "The S2E Platform: Design, Implementation, and Applications." *Acm Transactions on Computer Systems* 30.1(2012):1-49.