electronics

Article

Unauthorized Access Detection for Network Device Firmware WEB Pages

Minwei Peng, Qiang Wei, Rongkuan Ma, Yangyang Geng *, Yahui Yang, Shichao Zhang and Yali Zhang

Information Engineering University, Zhengzhou 450001, China; mwpeng2021@163.com (M.P.); funnywei@163.com (Q.W.); rongkuan307@163.com (R.M.); yongyh2020@163.com (Y.Y.); sc1222hm@163.com (S.Z.); thaodesi@126.com (Y.Z.)
* Correspondence: young9471@163.com

Abstract: WEB technology is utilized for the configuration, interaction, and management of network equipment, which has become ubiquitous in the intelligent industry and consumer electronics field. Unauthorized access on WEB allows unauthorized users to access authorized information, causing security vulnerabilities such as information leakage and command execution. However, commonly used vulnerability detection techniques for WEB unauthorized access face increasing challenges and more efficiently identify potentially sensitive pages. We propose WEBUAD, a WEB Unauthorized Access Detection framework, for the vulnerability detection of WEB service IoT network devices. WEBUAD utilizes the depth-first search algorithm to fully mine available information on device firmware and generate a potential-visit page set as well as a similarity–matching algorithm of machine learning to calculate the similarity of the responses of a web request. Finally, we evaluate WEBUAD on 9 real physical devices from four vendors and 190 device firmware from seven vendors. The result shows that compared with the state-of-the-art tool such as IoTScope, WEBUAD discovered 5007 potentially available pages, of which 658 were accessible and 9 sensitive pages existed, taking 50 s. Furthermore, WEBUAD exposed 13 security-critical vulnerabilities. Our approach can be used to automate the discovery of the WEB unauthorized access vulnerabilities of IoT devices.

Keywords: network device firmware; unauthorized access vulnerabilities detection; accessible WEB page

check for updates

Citation: Peng, M.; Wei, Q.; Ma, R.; Geng, Y.; Yang, Y.; Zhang, S.; Zhang, Y. Unauthorized Access Detection for Network Device Firmware WEB Pages. *Electronics* **2023**, *12*, 3674. https://doi.org/10.3390/electronics12173674

Academic Editor: Alessandro Gabrielli

Received: 22 June 2023
Revised: 14 July 2023
Accepted: 17 July 2023
Published: 31 August 2023

1. Introduction

With the continuous development of IoT (the Internet of Things), more and more various IoT devices are connected to the Internet. It is estimated that by 2025, there will be approximately 64 billion IoT devices worldwide [1]. Since WEB connection offers the benefits of remote configuration, easy identification, and graphical interface operation, network devices are typically equipped with web services and configuration to facilitate network configuration, operation, maintenance management, etc. [2] This not only provides convenient services but also leads to severe security vulnerabilities. As shown in Table 1, OWASP's official website lists the top 10 web application security risks in 2021 [3], and at least three of them—including A01, A04, and A07—are related to WEB unauthorized access vulnerability. Therefore, it is imperative to precisely identify unauthorized access to the web on network devices.

Various approaches have been presented to discover unauthorized access vulnerabilities on IoT devices. For example, some researchers collect accessible pages and discover a WEB unauthorized access vulnerability by using brute-force search or a dictionary. However, the brute-force search-based method is time-consuming, laborious, and has a high detection cost. The dictionary-based method requires a lot of manual prior knowledge and can only search for existing URL resources in the dictionary. As a result, searching for accessible pages more efficiently remains a significant challenge. Fortunately, there are several mainstream tools available for firmware simulation of IoT devices, which makes it feasible to test IoT devices for security issues using firmware simulation approaches [4]. Some researchers extract file systems from firmware, perform system simulations using only software, and then conduct static and dynamic analyses of the re-managed firmware, yet these approaches simulate firmware without system firmware code, making it difficult to judge the success of the simulation before the simulation process. There are also some full-system simulation tools for WEB security research of IoT devices. But they do not focus on unauthorized access, so there is still room for progress in the evaluation of functionality and efficiency.

Table 1. Top 10 WEB Application Security Risks in 2021 [3].

ID	Vulnerabilities
A01	2021-Broken Access Control
A02	2021-Cryptographic Failures
A03	2021-Injection
A04	2021-Insecure Design
A05	2021-Security Misconfiguration
A06	2021-Vulnerable and Outdated Components
A07	2021-Identification and Authentication Failures
A08	2021-Software and Data Integrity Failures
A09	2021-Security Logging and Monitoring Failures
A10	2021-Server-Side Request Forgery

To address these challenges, we propose WEBUAD, a WEB Unauthorized Access Detection framework. Instead of using brute force or dictionaries, WEBUAD utilizes effective information in firmware to generate a "potential-visit" page. Specifically, WEBUAD employs a depth-first search algorithm to search for vendors' unauthorized access "Protected" and "login-page" URL pages and a similar-match algorithm in machine learning for sensitive pages in target devices. With WEB access control and login authentication, we are able to discover and detect unauthorized access and apply it to real physical devices and firmware simulation devices.

We evaluated WEBUAD using 10 real-world devices from four vendors and 180 firmware simulation devices from seven vendors to discover and detect unauthorized access during WEB access control and login authentication. For the device adaptation of firmware emulation, the framework integrates the FirmAE emulation tool. The experimental results showed that WEBUAD demonstrated impressive performance, discovering 5007 potentially available pages, of which 658 were accessible and 9 sensitive pages existed, all within 50 s. Additionally, WEBUAD successfully discovered 13 0-day vulnerabilities on 30 devices from seven manufacturers and assigned 13 new CNVD IDs.

Our contributions are summarized as follows:

- We define a framework, WEBUAD, to detect unauthorized access for network device firmware WEB pages. This framework allows us to identify sensitive pages in network device firmware and further discover their security issues;
- We propose an automated approach to detect WEB unauthorized access. Specifically, we utilize a depth-first search algorithm to search for unauthorized access URL pages and a similar-match algorithm to identify sensitive pages in target devices;
- We design and implement the prototype system of WEBUAD, which is capable of detecting and verifying unauthorized access to WEB pages on both firmware emulation and real devices. WEBUAD discovered a total of 13 0-day vulnerabilities, of which 2 were found on two real devices from two vendors, while the remaining 11 were exposed on 190 simulated firmware devices from seven vendors;
- The source code of WEBUAD is available on Github for further research. https://github.com/mwpeng2021/WEBUAD [5].

The study is organized as follows. In Section 2, we list mainstream tools used for prior work related to firmware simulation and IoT WEB security research. In Section 3, we explain the background and challenges of our work, and our proposed system WEBUAD is explained in Section 4. Prototype implementation and experimental results are described in Section 5. Finally, conclusions are given in Section 6.

2. Related Work

As far as we know, there are several mainstream tools for firmware simulation of IoT devices. Sacrificing some code accuracy to provide faster emulation, QEMU [6] has become one of the primary tools used by academics and industry professionals because of its open-source licenses and widespread use and promotion by the community. It emulates architectures such as IA32, x86, MIPS, SPARC, ARM, SH4, PowerPC, ETRAX CRIS, and RISC-V, and it provides peripherals for many systems, making it almost the most widely used emulator. PANDA [7] is an open-source platform built on top of QEMU's entire emulator system for architecture-independent dynamic analysis. Firmadyne [8], Costin Firmware Analysis [9], and ARMX [10] extract file systems from firmware perform system simulations using only software, then perform static and dynamic analyses of the re-managed firmware. Being a fully automatic framework for simulation and vulnerability analysis developed on the basis of Firmadyne, FirmAE [11] proposes a way of arbitration simulation, from the firmware startup, network, NVRAM, kernel, and other aspects, it sums up the causes of firmware simulation failure and has a universal method, greatly improving the success rate of simulation. Pretender [12] requires hardware only during the training phase, where the peripheral model is generated by observing real hardware behavior. Avatar [13] proposes a new hybrid simulation framework, which significantly improves its forwarding performance through customized hardware agents. Avatar^2 [14] extends Avatar to allow the replay of forwarded peripheral I/O without using real devices. Prospect [15] forwards peripheral access at the system call level, but it does not exist on a bare firmware MCU device, so cached peripheral access is used to approximate the firmware state for analysis. Symbolic execution-based methods of simulating execution include µEmu [16], Laelaps [17], Jetset [18], etc., which model peripherals by simulating the software layer and taking all the values read from the hardware as symbolic values. These methods require symbolic actuators such as Angr [19] or S2E [20]. HALucinator [21] solves the challenge of providing peripherals that are not implemented in the base emulator by observing the interaction with peripherals usually performed with the hardware abstraction layer. Totally, the current methods still simulate firmware without system firmware code or Linux base, thus, it is difficult to judge the success of the simulation before the simulation process.

There are also some mainstream tools for the WEB security research of IoT devices. IoTFuzzer [22], a fuzzy automation framework for the web interface of IoT devices based on full-system simulation, uses stateful message generation (SMG) capabilities, which make a message composed of feeds that can basically cover all page actions and applications. Like IoTFuzzer, WMIFuzzer [23] tests running IoT firmware without the need for a predefined data model, applies fuzzy technology to the web management interface of Commercial off-the-shelf (COTS) Internet of Things devices for administration or user interaction, and evaluates seven popular COTS IoT devices and finds 10 vulnerabilities, 6 of which are 0-day vulnerabilities. IoTHunter [24], a keyword-based approach to IoT traffic classification, takes a labeled stream from each device type and uses DPI to extract the correct keywords from each device type. These extracted keywords serve as unique identifiers for a particular device. As a new approach that automatically exposes hidden web interfaces of IoT devices, IoTScope [25] uses firmware analysis to construct probe requests to test physical devices, narrow down the identification, and filter out unrelated requests and interfaces through variance analysis. It pinpoints hidden interfaces by attaching various interfaces, detecting parameters of device settings in requests and matches, as well as sensitive information keywords. To sum up, the current IoT WEB security research does not focus on unauthorized access, so there is still room for progress in the evaluation of functionality and efficiency.

3. Background and Challenges

3.1. Background

IoT security is a branch of cybersecurity that focuses on protecting, monitoring, and remediating IoT-related threats. It includes but is not limited to the following attack surfaces: web/mobile applications, cloud, sensors, gateways, smart border devices, and so on. Devices include not only traditional terminals such as desktops, laptops, mobile phones, and servers but also printers, cameras, routers, switches, smartphones, and navigation systems. The contemporary IoT ecosystem is exceedingly intricate with a plethora of vendors and device models, rendering a "one size fits all" solution unfeasible.

Unauthorized access pertains to the illicit utilization of network resources, encompassing the unauthorized entry of unauthorized users into the network or system as well as the unauthorized operations of legitimate users. It is a prevalent security vulnerability of configuration management in industry, household, and other fields. It is mainly manifested as the authorization or login-page defects—resulting in the direct access of users with no permission or insufficient permission to pages requiring permission or a higher level of authorization—or any other vulnerabilities, such as the leakage of databases, website directories, and so on, as well as the viewing or even the modification or deletion of important permission files. Unauthorized access detection technologies include port scanning detection, vulnerability scanning detection, WEB application scanning detection, network traffic analysis scanning detection, etc.

The interaction between the computer client and the WEB server is accomplished through the client's transmission of HTTP requests and the server's reception of HTTP responses. The client dispatches HTTP requests to the server by accessing the URL. After receiving and processing the requests, the server sends different HTTP responses according to varying response states. HTTP redirection technology redirects the user from one URL to another. Before configuration management, the WEB server typically employs HTTP redirection technology to redirect the user to the login authentication page. The user enters his or her username and password on the login-page, along with the verification code required by some devices to complete the verification process. If the authentication fails, some devices display error messages. When a user accesses any resources of the device, even if a malicious user attempts to access sensitive information such as the device configuration page and device information, a secure server should redirect any pages accessed by an unauthorized user to the login homepage of the device through HTTP redirection technology. However, throughout the entire login process, the login page may lack protection, and the access permission may not be strictly defined, thereby resulting in security vulnerabilities such as unauthorized access, information leakage, and command execution. Therefore, the failure of the login authentication process may compromise device security and network security.

3.2. Challenges

The detection of unauthorized access for network device firmware WEB pages poses the following three primary challenges.

C1: How to enhance the efficiency of firmware simulation? When studying physical devices that are hard to obtain (e.g., no longer to be sold), or that are impossible to purchase in large quantities due to their costs, sizes, etc., it is a viable option to find the firmware to emulate the device using tools like FirmAE. However, the challenge lies in the fact that it may not be clear whether the simulation is successful or not until the experiment is completed. If the simulation is successful, the information related to the simulation success will be output normally, and the firmware simulation time will be acceptable regardless of how long it takes. If the simulation fails and a long waiting time passes without any output of the prompt message of simulation failure, security researchers may be misled, and a significant amount of research time may be delayed. Determining whether the firmware has been successfully simulated as soon as possible can greatly improve work efficiency.

In response to the first challenge, extensive research has been conducted on the widely utilized FirmAE firmware simulation tools. By reading and analyzing their source code, particularly run.sh and firmae.config, it has been discovered that in the event of a failed firmware simulation during the initial attempt, the program logic error causes it to enter an infinite loop, rendering it unable to output the prompt message "simulation failure" [26]. Consequently, the firmware simulation remains stuck in an endless wait without knowledge of the firmware simulation results. To address this issue, we can modify the FirmAE source code to accurately detect and promptly output a notification when the firmware fails to simulate successfully within a specific time frame, such as 360 s, which can significantly reduce the waiting time for a simulation failure.

C2: How to obtain more accessible pages in IoT devices? Generally, the process of obtaining the list of accessible pages of the device can be divided into three steps. The first step is to obtain the device firmware from the manufacturer more quickly, which can be accomplished by downloading from the vendor's official websites, etc. The second step is to unpack the firmware, using an existing tool such as Binwalk [27]. Getting and decompressing device firmware has already been well investigated. The third is to generate a list of accessible pages for the device, which is challenging to complete with speed and quantity. By using the dictionary search method, the size of the dictionary is positively correlated with the content, the vendor, the device, and the firmware information. The larger the dictionary size, the larger the number of pages that can be accessed using the device; the lower the efficiency and the longer the running time, the more security vulnerabilities to be discovered. It is the biggest challenge to obtain the potentially accessible pages of the device to evaluate and test the efficiency and power performance.

Regarding the second challenge, it is imperative to fully leverage the information contained within the device firmware. To achieve this, a tool must be developed to identify the file system within the device firmware and identify all directories and files after decompression. By utilizing the known information within the firmware, a depth-first search algorithm can be employed to generate a comprehensive set of potential accessible page paths, which can then be systematically accessed one by one to get a consolidated list of accessible pages.

C3: How to determine whether a page is sensitive? The device boasts an extensive array of accessible pages, rendering it impractical to manually access each one and record the outcomes. However, if the crawler and other techniques are used for sequential scanning and log storage, the log can be manually checked after the access is completed, but the log-writing process is very slow. Furthermore, the log is of considerable size, which will require a significant amount of time to manually check. So, finding a fast, precise, and automatic way to determine whether a page on a device is a sensitive one or not is a significant challenge.

To address the third challenge, it is necessary to propose several definitions. Firstly, the "login-page" refers to the page where a user enters his or her username and password for authentication during the login, or the page that is accessed after HTTP redirection. Secondly, the "Protected-Page" refers to the page that cannot be viewed or configured before the login verification, or when the login fails. After a successful login, the device information and configuration modification can be viewed. Lastly, "Non-existent page" refers to pages that are almost impossible to exist in a set, such as "impossible_1a2b3c4d.html". Based on these definitions, a framework was designed and a prototype system was developed to verify the security issues related to the WEB side of the device. As a first step, the system visits the login-page of the device. If the login-page is not protected and can be accessed, read, or modified, the device does not protect it, otherwise, the similarity–matching algorithm of machine learning is further used to determine whether any accessible pages are sensitive ones and whether the device has security vulnerabilities. This is achieved by establishing the list set of "Protected" and "Non-existent" pages and calculating the similarity between page access response and "Protected" and "Non-existent" page access response.

4. System Design

As shown in Figure 1, the process of preparation entails a series of steps, the first of which is to acquire the firmware. There are various methods to obtain the firmware, such as accessing the configuration management page on a physical device, downloading the current firmware version of the device, or extracting the firmware from the device by disconnecting the hardware. Another option is to download the compressed firmware file from the device's official website, which may be in LZMA, ZIP, or Gzip format. The next step is to unpack the firmware, which involves extracting various components such as the boot loader, the kernel, and the file system. This can be achieved using firmware unpacking tools such as Binwalk. The initialization process is primarily handled in initialization, which utilizes Binwalk to extract the firmware and locate its file system. Additionally, FirmAE provides the option to emulate the device in firmware mode.

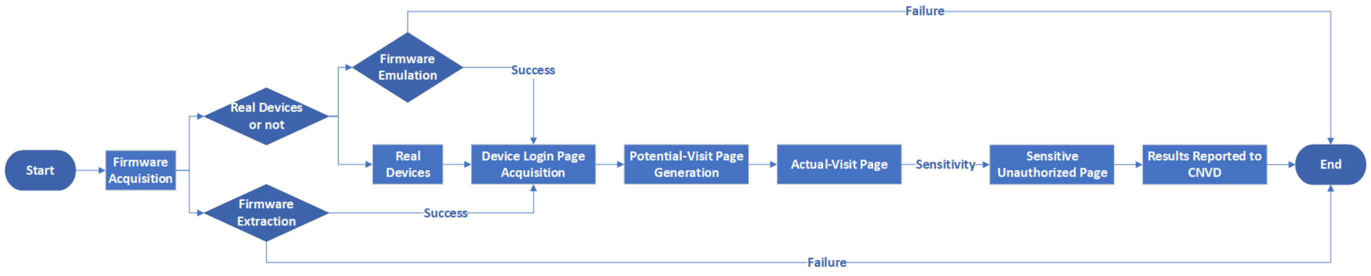

Figure 1. WEBUAD flowchart.

Unauthorized access detection and verification consist of the following parts. Firstly, it is necessary to ensure that the device is operational. If real equipment is utilized, it needs to be integrated into the network of the research environment. If the device is emulated with firmware, it must be successful with an enabled WEB service as the signal. The subsequent step is to execute the detection and verification program, which entails accessing the device's login-page and examining it for security vulnerabilities such as HTTP redirection and unauthorized access. In addition, a list of all accessible pages of the device should be generated based on the information extracted from the firmware, and a list of Protected and Non-existent pages should also be created. The existence of security vulnerabilities on each accessible page is determined by similarity–matching, and sensitive page logs are generated. Unauthorized access detection and verification includes two parts: reproducing the same firmware and different firmware versions of the device and attempting to reproduce on different devices from the same manufacturer or on different firmware versions of the same device from the same manufacturer.

During device access, HTTP redirection is often configured to redirect to the device login homepage due to security settings. HomeScope utilizes a depth-first search algorithm for redirection discovery, storing all URLs in the redirection chain and visiting them in sequence to identify potential security vulnerabilities in the redirection process. Hidden-Scope employs two algorithms. The first generates pages accessible to devices by using a depth-first search to traverse all directories and files, concatenating them to create a collection of potentially accessible pages and adding WEB suffixes such as php, asp, htm, html, etc. The second is the similarity–matching algorithm, which calculates the threshold based on the "Protected" and "Non-existent" pages of the device, attempts to visit each accessible page of the device, calculates the similarity, and compares it with the threshold to determine whether it is a sensitive page.

The compliance work mainly consists of two aspects. Firstly, the analysis logs of the framework and prototype system need to collect the following information: the collection of potentially accessible pages generated, the list of accessible pages attempted and successfully accessed using the device, the list of sensitive pages that may have unauthorized access, the program running time, and other related information. Secondly, the security vulnerabilities are exposed. All unauthorized access and related information disclosure, command execution, and other security vulnerabilities have been submitted to the CNVD. None of the experiments conducted have resulted in any actual threatening attacks on manufacturers and devices. Only the detection and verification were performed for real devices or devices simulated using the firmware, no actual related attacks.

5. Evaluation

The prototype implementation of WEBUAD is given in Section 5.1, and the experimental settings are given in Section 5.2. The research questions are listed and discussed in Sections 5.3 and 5.4. In Section 5.5, the experimental results and case studies are given.

5.1. Prototype Implementation

To evaluate WEBUAD experimentally, we have implemented a prototype of WEBUAD. As shown in Figure 2, the implementation details of its main components are described as follows. The first part involves preparation, which includes decompressing firmware and extracting firmware information. The second part is the search and security verification of the device's login-page, which involves obtaining relevant information on the login-page and detecting HTTP redirection. The third part is the hidden page search and security detection and verification, which involves using a similarity–matching algorithm to identify sensitive pages and detect unauthorized access to device WEB pages.

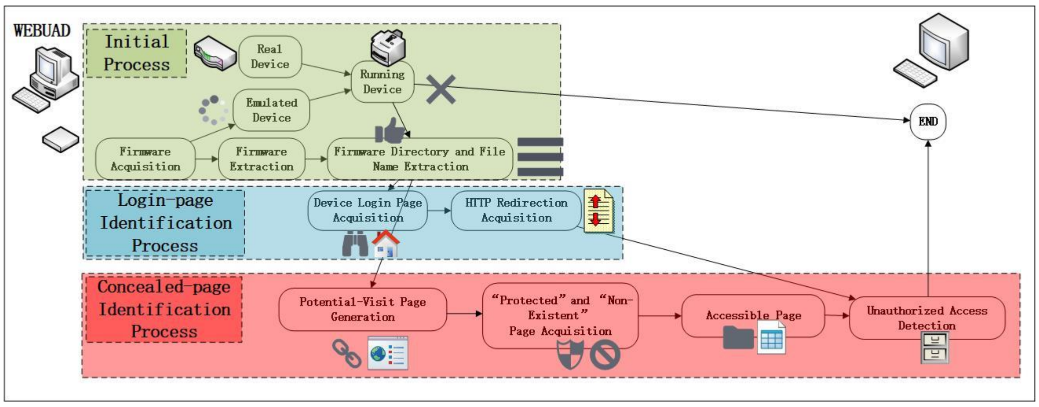
WEBUAD
Initial Process
Real Device
Running Device
Emulated Device
Firmware Acquisition
Firmware Extraction
Firmware Directory and File Name Extraction
END
Login-page Identification Process
Device Login Page Acquisition
HTTP Redirection Acquisition
Concealed-page Identification Process
Potential-Visit Page Generation
"Protected" and "Non-Existent" Page Acquisition
Accessible Page
Unauthorized Access Detection

Figure 2. WEBUAD implementation overview.

The three parts are further explained as follows:

- Firmware Directory and File Name Extraction. We search all directories, filenames, and more information in extracted firmware and combine them together, generating a "login-page" and "potential-visit" page set. Specifically, we utilize Binwalk to extract firmware information and employ FirmAE to simulate the dynamic experimental environment;
- Device Login-Page Acquisition and HTTP Redirection Acquisition. We use a depth-first algorithm to find all redirection HTTP links of a device "login-page" to detect whether unauthorized access exists or not.
- "Protected" and "Non-Existent" Page Acquisition and Unauthorized Access Detection. We use "Protected" and "Non-Existent" pages to generate the baseline, and then we use a similarity–matching algorithm to visit and calculate the similarity of each page. Then, comparing it to the baseline, we can detect unauthorized access on the device's concealed WEB page.

5.2. Experimental Settings

To assess WEBUAD's efficiency, a total of 190 device firmware from seven vendors were tested together with the firmware of an RT-AC53 physical device from an ASUS manufacturer and with a device whose vendor information is hidden. WEBUAD and IoTScope both run in Ubuntu18.04 VM with 4 GB of RAM, 2 cores, and 4 processors, and the host is configured with 16 GB of memory and an Intel(R) Core(TM) i7-10750 H 2.60 GHz CPU.

The device test set included real physical devices and firmware-simulated devices. There were two types for the former: X-Vendor and X-Device named to hide the information of the vendor device and firmware version and an RT-AC53 device from ASUS with a firmware version of XXX.bin. For the latter, there were 190 device firmware from seven manufacturers, including ASUS, NETGEAR, and D-Link, among which—except for 10 EDIMAX downloaded from the official website—the other 180 from six vendors came from FirmAE's firmware set.

5.3. Research Questions

To evaluate the functionality and efficiency of the prototype system, the following questions are proposed:

RQ1: Is WEBUAD effective in identifying sensitive unauthorized visit pages?

RQ2: Is WEBUAD effective in discovering unauthorized access vulnerabilities on tested real-world devices and firmware-simulated simulations?

RQ3: How does the time overhead of WEBUAD in discovering sensitive pages compare with mainstream tools?

5.4. Experimental Evaluation

RQ1: HomeScope is implemented in the login-page identification process, and we use the depth-first algorithm rather than any other algorithms. First, according to WEB HTTP redirection, it is natural to use the depth-first algorithm. Second, we can get the WEB source-code after we get the login-page of the device, in which there are several tags including: "javascript" or "href", therefore, using the depth-first algorithm to get all redirect tags is convenient and fast. Since HomeScope detects security vulnerabilities on the device login-page, security risks such as login verification are not guaranteed on the device login-page configuration management. In terms of details, there is a security issue on the "Protected" page of "X" and "Y". As a result, "X" manufacturer can log into the device login-page and perform unauthorized access and command execution as the administrator before authorization. Similar issues also exist in some devices of "Y" vendor and a certain printer of "Z". For real devices, security issues exist on the "Protected" and login-page of vendors "X" and "Y" detected by HomeScope, thus timeliness is unnecessary to be considered.

Table 2 proves the device login-page classification is necessary. We were surprised by the number of devices and vendors with no or incomplete protection on the login-page that caused unauthorized access security issues. HomeScope detected security vulnerabilities for a total of eight devices from a total of three vendors, including unauthorized access, information leakage, and command execution, resulting in four CNVD numbers. For the security and privacy of the manufacturers and devices, sensitive information is replaced with symbolic letters such as "X" and "Y".

Table 2. Detection results of HomeScope.

Vendor	Device	Type	CNVD Number
X	X1 X2 X3 X4 X5	unauthorized access, information leakage, command execution	CNVD-2022-73093 CNVD-2022-77987
Y	Y1 Y2	unauthorized access	CNVD-2022-73098
Z	Z1	information leakage	CNVD-2022-73410

RQ2: To investigate WEBUAD effectiveness, we set some experiments on real devices and firmware emulation. As shown in Tables 3 and 4, on real devices, WEBUAD found five 0-days, including CNVD-2022-73093, CNVD-2022-77987, CNVD-2022-73098, and CNVD-2022-73410. In Table 3, WEBUAD found eight 0-days, including CNVD-2022-89524, CNVD-2023-02802, CNVD-2022-91483, CNVD-2022-69655, CNVD-2022-69516, CNVD-2022-70391, CNVD-2022-82283, and CNVD-2023-02734 in the device firmware emulation.

Table 3. Detection results of HiddenScope.

Vendor	Device	Type	CNVD Number
	R6400v2	Unauthorized access	CNVD-2022-69489
	R8000	Unauthorized access	CNVD-2022-89524
NETGEAR	WNR2000		
	WNR1000	Unauthorized access	CNVD-2023-02802
	WN2000RPT		
	HP5101WN		
	BR6228GNS		
EDIMAX	BR6428NS	Information leakage	CNVD-2022-91483
	BR6479GN		
	EW7416APN_v2		
D-Link	DIR815	Information leakage	CNVD-2022-69655
	RT-AC53		
	DSL-N55U		
	RT-AC66U		
	RT-AC88U		CNVD-2022-69516
	RT-AC1200G+	Unauthorized access	CNVD-2022-70391
ASUS	RT-N11P	Information leakage	CNVD-2022-82283
	RT-N12+		
	RT-N12E		
	RT-N12VP		
	RT-N66U		
	RT-AC68U	Unauthorized access	CNVD-2023-02734

Table 4. Efficiency self-evaluation of WEBUAD.

Program Name	Device Name	Sensitive Pages	Time(s)	CNVD Number
WEBUAD	R6400v2	14	4	CNVD-2022-69489

As shown in Table 4, taking the NETGEAR R6400v2 device as an example, we set an experiment to evaluate the functionality of WEBUAD; 14 sensitive pages were found in 4 s, one of which was accepted by CNVD, CNVD-2022-69489.

Figure 3 showcases the results of WEBUAD's firmware simulation capabilities and the discovery of unauthorized sensitive pages. Figure 3a displays the number of successful firmware emulations. Figure 3b represents the number of potentially accessed pages. Figure 3c provides an overview of the number of actual accessed pages. Figure 3d shows the number of unauthorized sensitive pages. We adopted D-Link vendors for illustration. WEBUAD successfully simulated 29 firmware versions from D-Link vendors, generating 149,364 accessible pages. Out of these, WEBUAD was able to access 69,994 pages and identified a total of 363 sensitive pages. Figure 4 presents the time overhead of WEBUAD. The average time overhead excludes the simulation time but includes the login-page identification time and hidden page identification time. For the case of D-Link, WEBUAD's total time overhead at the D-Link provider was 1413 s. As depicted in Figures 3 and 4, WEBUAD generated 135,904 accessible pages and successfully accessed 17,862 ones, including 245 sensitive ones, with an average time of 1366 s.

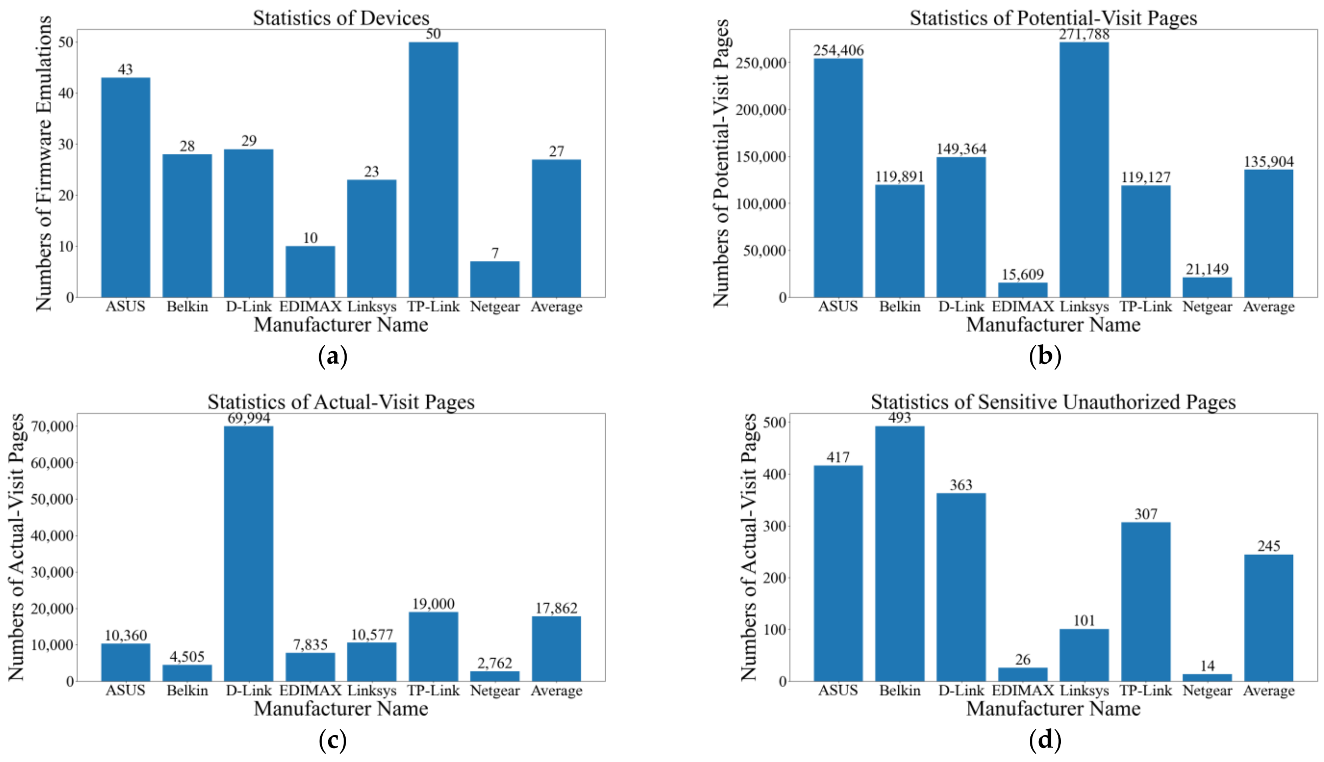

Figure 3. (**a**) Vendor device's function evaluation of WEBUAD in a firmware simulation; (**b**) potential-visit pages generated with WEBUAD; (**c**) actual-visit pages generated with WEBUAD; (**d**) sensitive unauthorized visit pages discovered with WEBUAD.

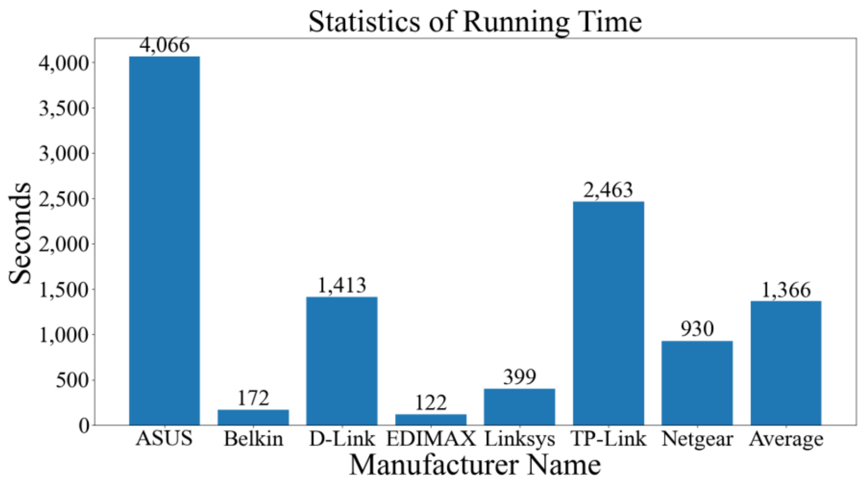

Figure 4. Time overhead of WEBUAD.

RQ3: To compare the efficiency of WEBUAD and IoTScope, we conducted experiments using both real devices and firmware emulation. Specifically, we selected the R6400v2 as the real device and the DIR-815 as the firmware emulation device to evaluate the efficiency of WEBUAD. To facilitate the comparison, we integrated the IoTScope tool into the IoTscope.py script. The primary objective was to combine the following steps: Enumerating Interfaces, Delivering Probing Requests, Identifying Unprotected Interfaces, and Identifying Hidden Interfaces. This approach aimed to minimize the time required to enter commands and reduce potential errors during the comparison process.

Table 5 shows the comparison between IoTScope and WEBUAD for the same target of the real device R6400v2. IoTScope found 177,498 potential-visit pages with 176,375 pages successfully accessed, among which 290 were sensitive, taking 22,978 s or about 6.4 h.

One of the sensitive pages that passed manual verification was CNVD-2022-69489. The WEBUAD framework generated 11,923 accessible pages and successfully visited 3301 of them with 1 sensitive page. After verification, it was the same CNVD-2022-69489. The total time spent was 32 s, which accounts for 0.14% of the running time of IoTScope in terms of time efficiency.

Table 5. Real device R6400v2: efficiency evaluation of WEBUAD.

Program Name	Potentially Accessible Pages	Device-Accessible Pages	Sensitive Pages	Time(s)
IoTScope	177,498	176,375	290	22,978
WEBUAD	11,923	3301	1	32

As shown in Table 6, the scope was the same firmware emulation DIR-815, and we made a contrast between IoTScope and WEBUAD. IoTScope found 97,125 potential-visit pages and successfully accessed 48,948 of them, including 35 sensitive ones in 5734 s or about 1.6 h. One sensitive page was manually verified and declared as CNVD-2022-69655. WEBUAD found 11,277 accessible pages, 8059 of which were successfully visited, and there were 6 sensitive pages, one of which was verified to be the same CNVD-2022-69655, spending 227 s in total and accounting for 3.96% of IoTScope's running time of in terms of time efficiency.

Table 6. Firmware emulation device DIR-815: efficiency evaluation of WEBUAD.

Program Name	Potentially Accessible Pages	Device-Accessible Pages	Sensitive Pages	Time(s)
IoTScope	97,125	48,948	35	5734
WEBUAD	11,277	8059	6	227

In general, WEBUAD costs less time in the detection of WEB unauthorized access vulnerabilities on real devices and firmware emulation devices. Based on the above discussion, compared with the IoTScope tool, the WEBUAD framework has significantly reduced the number of potentially accessible pages of devices, the number of successfully accessed pages of devices, the number of potentially sensitive pages of devices, and the time spent, and it also has a certain degree of improvement in false reports. The first reason is that WEBUAD skipped the binary file with the cgi suffix when stitching, so the number of accessible pages was reduced to a certain extent. The second is that during the interaction with real devices, the program interacts with real hardware. Hardware has a certain processing time, and the interaction speed of the firmware simulation is faster than that of real devices.

5.5. Discussions

WEBUAD successfully emulated firmware for 190 devices across seven different vendors. It generated a total of 801,970 potential-visit pages and 125,033 actual-visit pages during the process. Out of these, it identified 1721 sensitive pages. The entire emulation process took approximately 9565 s. Additionally, WEBUAD discovered five 0-day vulnerabilities in real devices and eight 0-day vulnerabilities in device firmware emulations. To provide more technical insights, we illustrated some case studies.

We have conducted many experiments with ASUS manufacturers, including a real RT-AC53 device and a firmware simulation device of RT-AC1200+. RT-AC53 is a real device, and AC1200G+ is a device simulated with firmware. Unauthorized access and information leakage exist in some firmware versions of the two devices. As a result, the firmware version, kernel version, operating system version, and network address of the devices were leaked without user login verification. We reported this vulnerability to CNVD officials and obtained CNVD-2022-69516 and CNVD-2022-70931 certificates.

We conducted many experiments for D-Link vendors, including the DIR-815 firmware simulation device. There was a common information leakage problem on some firmware versions that could reveal information such as the firmware version, operating system kernel version, and network address. We reported this vulnerability to the official CNVD and obtained the CNVD-2022-69655 certificate.

6. Conclusions

To address the security problem of WEB unauthorized access of network device firmware, we proposed the WEBAUD framework and designed a prototype system, which solves the problems of fast generation access of device pages and fast detection and verification of WEB unauthorized access. WEBUAD combines the capabilities of FirmAE and Binwalk, allowing it to analyze a device or emulated firmware within a matter of seconds to a minute. This quick analysis enables WEBUAD to identify potential unauthorized access efficiently. One of the key advantages of WEBUAD is its accessibility. It is easy to obtain and utilize, making it a convenient tool for unauthorized access detection. Additionally, WEBUAD boasts a low detection time, meaning it can swiftly identify potential security breaches. Moreover, WEBUAD has a high success rate in detecting unauthorized access attempts, making it a reliable solution for security assessments. Although WEBUAD has a certain effectiveness and efficiency in the study of WEB unauthorized access to network device firmware, future work will continue to carry out more in-depth research in the following two aspects. The main research objective of WEBUAD is to detect the absence of authorization and to continue to study other security problems in the case of authorization, such as WEB aspects and binary aspects, like stack overflow, heap overflow, etc. The second main research objective of WEBUAD is to be accessible to real physical devices and devices that can simulate firmware. The main work is to complete the detection and verification of security problems on the premise that the firmware is decompressed and that the WEB service of the device can be accessed. The firmware decompression process uses the most available Binwalk tool while the firmware simulation employs the FirmAE tool. When Binwalk and FirmAE are not efficient or when there are firmware decompression and simulation failures, WEBUAD cannot complete the detection, resulting in a limited range of target devices. Further research on firmware decompression and simulation technology is needed.

Author Contributions: Conceptualization, S.Z.; Methodology, M.P. and Q.W.; Software, M.P. and R.M.; Formal analysis, R.M. and Y.Y.; Resources, M.P.; Data curation, Y.Y.; Writing—original draft, M.P., S.Z. and Y.Z.; Writing—review & editing, M.P., Y.G. and Y.Z.; Visualization, Y.G.; Project administration, Q.W. and Y.G.; Funding acquisition, R.M. All authors have read and agreed to the published version of the manuscript.

Funding: This work is supported by the National Key R&D Program of China under Grant No. 2020YFB2010900 and the Program for Innovation Leading Scientists and Technicians of ZhongYuan under Grant No. 224200510002.

Data Availability Statement: The data used to support the findings of this study are included within the article.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Riad, K.; Huang, T.; Ke, L. A dynamic and hierarchical access control for IoT in multi-authority cloud storage. *J. Netw. Comput. Appl.* **2020**, *160*, 102633. [CrossRef]
2. Costin, A.; Zarras, A.; Francillon, A. Automated dynamic firmware analysis at scale: A case study on embedded web interfaces. In Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security, Xi'an, China, 30 May–3 June 2016; pp. 437–448.
3. Top 10 Web Application Security Risks. Available online: https://owasp.org/www-project-top-ten/ (accessed on 5 May 2023).
4. Wright, C.; Moeglein, W.A.; Bagchi, S.; Kulkarni, M.; Clements, A.A. Challenges in Firmware Re-Hosting, Emulation, and Analysis. *ACM Comput. Surv.* **2020**, *54*, 5. [CrossRef]

5. WEBUAD. WEB Unauthorized Access Detection Tool. Available online: https://github.com/mwpeng2021/WEBUAD (accessed on 5 May 2023).
6. Bellard, F. QEMU, a fast and portable dynamic translator. In Proceedings of the Annual Conference on USENIX Annual Technical Conference, Anaheim, CA, USA, 10–15 April 2005; USENIX Association: Berkeley, CA, USA, 2005; pp. 41–47.
7. Panda, P.R. Systemic: A modeling platform supporting multiple design abstractions. In Proceedings of the 14th International Symposium on Systems Synthesis, Montrél, QC, Canada, 30 September–3 October 2001.
8. Chen, D.D.; Woo, M.; Brumley, D.; Egele, M. Towards automated dynamic analysis for linux-based embedded firmware. In Proceedings of the 23rd Annual Network and Distributed System Security Symposium, San Diego, CA, USA, 7 February–3 March 2023.
9. Costin, A.; Zaddach, J.; Francillon, A.; Balzarotti, D. A Large-Scale Analysis of the Security of Embedded Firmwares. In Proceedings of the 23rd USENIX Security Symposium, San Diego, CA, USA, 20–22 August 2014; USENIX Association: San Diego, CA, USA, 2014; pp. 95–110. Available online: https://www.usenix.org/conference/usenixsecurity14/technical-sessions/presentation/costin (accessed on 18 June 2023).
10. Shah, S. The ARM-X Firmware Emulation Framework. Available online: https://github.com/therealsaumil/emux (accessed on 11 April 2023).
11. Kim, M.; Kim, D.; Kim, E.; Kim, S.; Jang, Y.; Kim, Y. FirmAE: Towards large-scale emulation of iot firmware for dynamic analysis. In Proceedings of the Annual Computer Security Applications Conference, Virtual, 7–11 September 2020; Association for Computing Machinery: New York, NY, USA, 2020; pp. 733–745.
12. Gustafson, E.; Muench, M.; Spensky, C.; Redini, N.; Machiry, A.; Fratantonio, Y.; Balzarotti, D.; Francillon, A.; Choe, Y.R.; Kruegel, C.; et al. Toward the analysis of embedded firmware through automated rehosting. In Proceedings of the 22nd International Symposium on Research in Attacks, Intrusions and Defenses, Beijing, China, 23–25 September 2019.
13. Zaddach, J.; Bruno, L.; Francillon, A.; Balzarotti, D. AVATAR: A Framework to Support Dynamic Security Analysis of Embedded Systems' Firmwares. In Proceedings of the Network and Distributed System Security Symposium, San Diego, CA, USA, 23–26 February 2014; pp. 1–16.
14. Muench, M.; Nisi, D.; Francillon, A.; Balzarotti, D. Avatar^2: A Multi-target Orchestration Platform. *Proc. Workshop Binary Anal.* **2018**, *18*, 1–11.
15. Kammerstetter, M.; Platzer, C.; Kastner, W. Prospect: Peripheral proxying supported embedded code testing. In Proceedings of the 9th ACM Symposium on Information, Computer and Communications Security, Kyoto, Japan, 4–6 June 2014; ACM: New York, NY, USA, 2014; pp. 329–340. [CrossRef]
16. Zhou, W.; Computer, N.; Intrusion, N.; Symposium, U.S. Automatic Firmware Emulation through Invalidity-guided Knowledge Inference. In Proceedings of the 30th USENIX Security Symposium (USENIX Security 21), Vancouver, BC, Canada, 11–13 August 2021.
17. Cao, C.; Guan, L.; Ming, J.; Liu, P. Device-agnostic firmware execution is possible: A concolic execution approach for peripheral emulation. In Proceedings of the Annual Computer Security Applications Conference, Virtual, 7–11 September 2020; Association for Computing Machinery: New York, NY, USA, 2020.
18. Johnson, E.; Diego, S.; Bland, M.; Zhu, Y.; Mason, J.; Champaign, U.; Checkoway, S.; College, O.; Savage, S.; Diego, S.; et al. Jetset: Targeted Firmware Rehosting for Embedded Systems. In Proceedings of the 30th USENIX Security Symposium (USENIX Security 21), Vancouver, BC, Canada, 11–13 August 2021.
19. A Binary Framework Based on Symbolic Execution and Analog Execution, Angr. Available online: https://angr.slack.com (accessed on 12 July 2023).
20. Chipounov, V.; Kuznetsov, V.; Candea, G. S2E: A platform for in-vivo multi- path analysis of software systems. *Acm Sigplan Notices* **2011**, *46*, 265–278. [CrossRef]
21. Clements, A.A.; Sandia National Laboratories; Gustafson, E.; UC Santa Barbara; Sandia National Laboratories; Scharnowski, T.; Ruhr-Universität Bochum; Grosen, P.; UC Santa Barbara; Fritz, D.; et al. HALucinator: Firmware re-hosting through abstraction layer emulation. In Proceedings of the 29th USENIX Security Symposium (USENIX Security 20), Santa Clara, CA, USA, 14–16 August 2019.
22. Chen, J.; Diao, W.; Zhao, Q.; Zuo, C.; Lin, Z.; Wang, X.; Lau, W.C.; Sun, M.; Yang, R.; Zhang, K. IoTFuzzer: Discovering memory corruptions in iot through app-based fuzzing. In Proceedings of the Network and Distributed System Security Symposium (NDSS'18), San Diego, CA, USA, 18–21 February 2018.
23. Wang, D.; Zhang, X.; Chen, T.; Li, J. Discovering vulnerabilities in COTS IoT devices through blackbox fuzzing web management interface. *Secur. Commun. Netw.* **2019**, *2019*, 1–19. [CrossRef]
24. Khandait, P.; Hubballi, N.; Mazumdar, B. IoTHunter: IoT network traffic classification using device specific keywords. *IET Netw.* **2020**, *10*, 59–75. [CrossRef]
25. Xie, W.; Chen, J.; Wang, Z.; Feng, C.; Wang, E.; Gao, Y.; Wang, B.; Lu, K. Game of Hide-and-Seek: Exposing Hidden Interfaces in Embedded Web Applications of IoT Devices. In Proceedings of the ACM Web Conference 2022 (WWW '22), Lyon, France, 25–29 April 2022; ACM: New York, NY, USA, 2022. [CrossRef]

26. FirmAE issue, Make Some Changes in Firmae.Config can Make FirmAE Faster when Facing a Firmware Image Cannot Be Emulated. Available online: https://github.com/pr0v3rbs/FirmAE/issues/56 (accessed on 9 June 2023).
27. Binwalk. Firmware Analysis Tool. Available online: https://github.com/ReFirmLabs/binwalk (accessed on 2 February 2023).