Security Analysis of SOHO Wi-Fi routers

Sandeep Romana
Centre for Development of Advanced Computing
Hyderabad, India
sandeepr@cdac.in

Jyostna Grandhi
Centre for Development of Advanced Computing
Hyderabad, India
gjyostna@cdac.in

P R Lakshmi Eswari
Centre for Development of Advanced Computing
Hyderabad, India
prleswari@cdac.in

2020 International Conference on Software Security and Assurance (ICSSA) | 978-1-6654-3246-7/20/$31.00 ©2020 IEEE | DOI: 10.1109/ICSSA51305.2020.00020

Abstract—Wi-Fi-enabled Small Office/Home Office (SOHO) routers have become ubiquitous for having internet access in many network environments. With the advent of the 'Internet of Things' (IoT) and smart devices, almost all household devices are connected to the internet via these Wi-Fi-enabled SOHO routers. There have been numerous reports of security issues in SOHO routers because of known vulnerabilities. Except for enterprise networks, these devices acting as gateways are rarely safeguarded properly. The reasons for neglecting the security of these devices are many. Reports suggest that users often deploy these devices with insecure default configurations, leaving them as easy prey in the hands of malice users. Due to the easy exploitability, these devices have become attackers' paradise. The security analysis of these devices is often overlooked and challenging to undertake because of various versions of custom Linux based firmware and bootloaders they use. The security analysis of these devices holds strategic importance as they act as a gateway to almost every household having access to the internet. In this paper, we list the software and hardware interfaces of SOHO routers which are important for security analysis. We present a network environment for carrying out the security analysis of these devices. An iterative approach using static & dynamic analysis is described. Further, the static & dynamic analysis is augmented with symbolic analysis. To validate the described procedures, we present the case study of performing security analysis on a Netis WF2411 router.

Keywords — Wi-Fi, SOHO, router, security, analysis

I. INTRODUCTION

Wi-Fi-enabled Small Office/Home Office (SOHO) routers have become ubiquitous for connecting to the internet in different network environments. They are now being used in homes, public places such as society common areas, shops, airports, clubs, shopping malls, railway stations, etc. With the advent of 'Internet of Things' (IoT) and smart devices, most of the household devices such as mobile phones, laptops, personal computers, televisions, audio, common storage, printers, video surveillance systems, air-conditioners, refrigerators, washing machines, smart meters, etc. are connected to the internet via these Wi-Fi-enabled SOHO routers. In the majority of the above-listed network environment, these low budget routers act as gateway devices. A large percentage of home users are using the device supplied by their Internet Service Provider (ISP) with the default security settings unchanged which are not secure most of the times. These devices are generally installed with firmware based on a variant of Linux and have basic firewall capabilities. Except for enterprise networks, these devices are rarely re-configured for ensuring reasonable security. In addition to this, enabling advanced threat mitigation techniques for these devices, which are otherwise available to personal computers, is a challenge [1] because of the limited processing, memory, storage, etc.

In 2019 alone, there have been several reports and research publications highlighting the security issues of routers. Cross-Router Covert Channels have been found by Adar Ovadya et al and presented in Usenix [2]. Multiple bugs have been found in 4G hotspots from ZTE and Netgear [3] and critical remote code execution (RCE) vulnerabilities have been found in TP-Link Wi-Fi range extenders, smart hub and routers [4], [5], [6]. On multiple occasions, Vodafone in Europe found backdoors in routers from Huawei [7]. Security researchers have found five undocumented backdoors in Cisco Routers in the year 2018 [8] and Kaspersky Labs have discovered a backdoor in D-Link DIR 620 router [9] in the past. Researchers at Eurocom have proved the negligence of device manufacturers by concluding that known vulnerabilities that occur in one firmware reappear in the firmware of other manufacturers and found 326 out of 30,000 instances of firmware with backdoors [10]. These are few to mention in the plethora of issues with the devices under consideration.

The reasons for neglecting the security of these devices include: (a) The users of these devices in non-enterprise networks are rarely aware of the security threats these devices can bring along with them. (b) Even if the users are aware of the security threats, limited users have the required technical skills to configure these devices for optimal security. (c) The vendors often overlook the security of these devices and sell them with default insecure settings. (d) The vendors give the least priority for releasing patches for known vulnerabilities of these devices. (e) Even if vendors release the security patches, very few devices end up getting installed with these patches. This can be due to the mode of delivery of patch or lack of technical skills at the end-user level to install these patches. (f) These devices end up getting deployed for a very long period which is way beyond their end of support cycle. Thus, vendors can't afford to release patches for these low costing devices beyond the end of the support cycle. (g) For the vendor-supplied devices, it is a common practice to disguise the backdoor in the name of TR-069 compliance [11]. These open ports and default remote login credentials can be easily discovered, misused and exploited [12]. The reasons for this can range from simply keeping the remote access for ease of support to retaining the control of the device as part of mass surveillance [12]. Though the surveillance argument listed above can't be proved easily, it is a matter of concern and discussion for the strategic agencies and governments around the world.

In addition to the above-listed reasons, it is common for vendors to intentionally leave a vulnerability or a backdoor as a side effect of security patch release [8]. As per the study [13], leaving an intentional backdoor has been and will continue to be the most common way of surveillance.

978-1-6654-3246-7/20/$31.00 ©2020 IEEE
DOI 10.1109/ICSSA51305.2020.00020

Thus, analysing the devices from the perspective of security is crucial and hence, security analysis of these devices is important. Though there are certification bodies that do functionality testing of these devices, these undocumented backdoors cannot be tested through this. In this paper, we present the procedures for security analysis of the firmware of these devices and it is assumed that the firmware is available for carrying out the analysis. Along with this, we present a case study of how we applied these procedures to Netis WF2411 routers. The analysis procedure presented here can easily be extended or customized to other embedded devices having Linux based firmware.

The remainder of this paper is organized as follows: Section 2 discusses related work and summarizes the research in the area of embedded device and firmware analysis. Section 3 lists the formulated analysis steps. A case study is presented in Section 4. The conclusion, acknowledgements, and references are given in Sections 5, 6, and 7 respectively.

II. Related Work

Security analysis of embedded device firmware is an active research area, and research organizations around the globe have different research initiatives in this direction. An interested reader can refer to [14], [15], [16], [10] the University of Wisconsin-Madison, [17] the University of Birmingham for the details of such programs.

The research on embedded firmware analysis is progressing around the usage of technologies such as symbolic execution, machine learning, web application analysis, firmware and device emulation. The static and dynamic analysis using listed technologies are being used for the identification of vulnerabilities, malware, and hidden functionality, etc. Despite being an active research area, there are only limited efforts [18] that provide easy to use steps along with further research-based exploration direction for embedded device security analysis.

Thomas et al have developed a tool called Humidify, based on a semi-automated approach to detect hidden functionality within the binaries, by classifying them, profiling and then applying static analysis using machine learning algorithms [17]. FIRMASTER is a service for static and dynamic analysis of home router firmware [19]. Researchers from the same group have developed a penetration testing tool for the Internet of Thing devices which is named as PENTOS [20]. FIRMADYNE [21] is an automated dynamic analysis system that targets Linux based firmware on network-connected COTS devices, using software-based full system emulation. It provides a framework to find the vulnerabilities in the web interface and exploit them.

Authors of [22] have presented a distributed framework for scalable security testing of embedded devices through web interfaces with automation, using firmware chroot with a generic kernel and filesystem. Zaddach et al [23] have implemented an event-based arbitration framework called Avatar for detecting vulnerabilities in the firmware binaries. Avatar enables the execution of firmware code in an emulator by forwarding memory access and peripheral access to the real device. However, it doesn't provide a feasible way to do whole-system analysis.

A combination of static analysis and dynamic symbolic execution (DSE) for achieving multi-tag analysis is being used for finding vulnerabilities in the boot loaders [24]. Firmalice is a binary analysis framework to support the analysis of firmware running on embedded devices. It is built on top of a symbolic execution engine and program slicing techniques to increase its scalability. It can determine the required inputs to perform privileged operations by using a model of authentication bypass flaws [25]. Symbolic execution is used in [26] for detecting bugs in firmware programs for the popular MSP430 family of microcontrollers. SymPLAID directly parses and interprets assembly language programs for a MIPS processor for finding vulnerabilities [27]. However, it does not support system calls.

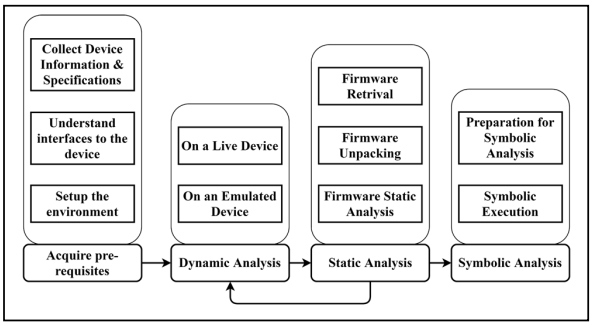

Fig. 1. Security analysis steps

The embedded device vulnerability analysis case study by Oliver et al [18] comes closest to this work. There are a few key differences between Trommel and this work. Trommel provides holistic, generic and macro-level steps for embedded device analysis, whereas this work is focused on details of SOHO routers analysis. Unlike Trommel, this work provides research-oriented analysis and does not restrict only to static analysis.

III. Steps for Security Analysis

The security analysis process is carried out in steps and/or phases as depicted in Figure 1. The initial step is to acquire the prerequisites. This can be followed by either one of the dynamic or static analysis as shown in the case (I) and (II) of Figure 3. Irrespective of the type of analysis carried out after acquiring prerequisites, analysis can iterate between static and dynamic analysis as shown by the back connector in Figure 1.

Further, the dynamic analysis is carried out at two levels a) network level and b) on extracted binaries. During the second level of dynamic analysis i.e. case (II) from Figure 3, tools such as Firmadyne will be useful as they help to emulate the file-system for further analysis. Next, symbolic analysis can be tried. In the end, the results from all the steps can be co-related for deducing the results.

A. Prerequisites

a) Collecting the device information: This step requires collecting as much information as possible about the Device Under Test (DUT) from the public domain. This may include information such as whether the firmware is downloadable from vendor website, firmware version, firmware updates/patches if any, reported security issues, hardware version, the procedure to get access to the device console, location of serial, JTAG or any other interfaces, etc.

b) Understanding the various interfaces to the device: From a security analysis perspective, the first step is to identify various network interfaces [28] of the DUT. Typical Wi-Fi-enabled SOHO router has three types of network

interfaces - WAN, LAN, and Wi-Fi. WAN interface is a wired interface for connecting to broadband internet, LAN Interface is a wired interface for connecting LAN systems to the router and a wireless Wi-Fi interface for connecting LAN devices through Wi-Fi. Typically, there is a single WAN and Wi-Fi interface, and there are multiple (2 to 4) LAN interfaces, though they are becoming less common nowadays and there is a possibility that SOHO routers may not have any LAN interface.

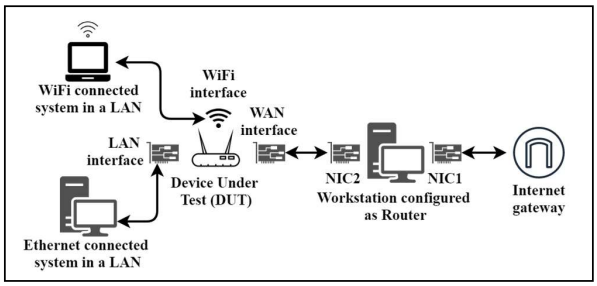

Fig. 2. Network setup

Apart from the network interfaces, there can be serial, JTAG, I2C, and SPI interfaces. These interfaces may not be available externally and would require opening the device casing. Further, these interfaces may have been concealed or just left on the track of PCB pads sans connectors, and thus identifying them may require probing. If identified, the serial interface may allow access to the device logs or console and JTAG may allow retrieval and flashing of the firmware. Rarely, if one can identify I2C and SPI, they may allow access to the device configuration and/or file system.

c) Setting up the environment: Next, the analyst has to set up an environment for security analysis of the DUT. This includes setting up the network for dynamic analysis and having the right set of software/tools to assist during analysis. Rather than directly connecting the DUT to the internet via WAN port, it is advisable to connect it via the workstation. This workstation needs to have two network interfaces, the external interface which connects to the internet directly or via internet gateway and the internal interface connecting to the WAN interface of the DUT. The traffic from the external interface of the workstation has to be forwarded to the internal interface i.e. the workstation needs to be configured as a router. This will serve a dual purpose: i) to monitor the network traffic and ii) perform Vulnerability Assessment and Penetration Testing (VAPT) through the WAN interface of the DUT. The LAN interface of the DUT can be connected to a system through an ethernet connection. This will serve the following purposes: perform VAPT through the LAN interface and optionally extract the device firmware via ethernet, subject to the possibility of doing the same. Fig 2 depicts the network setup.

B. Dynamic Analysis

a) Online analysis with a live device: Online analysis is a dynamic analysis that is carried out on a live device. One must note that either of the dynamic or static analysis can be carried out first. See Fig 3 for details. Further, dynamic & static analysis can be carried out iteratively as described in the case study below.

Dynamic analysis can help to discover open network ports, carry out fuzz testing, perform VAPT on WAN, Ethernet & Wi-Fi Interfaces, including the web management interface provided by the DUT. The network traffic between the WAN interface and the internet is to be continuously monitored via software tools installed on the workstation configured as a router/gateway. This can be a never-ending job, so the analyst has to take a call on how long monitoring has to be carried out. At a minimum, under the following cases, network monitoring may be done: during times of heavy use, when the device is idle and when device firmware is updating. The document root folder of the device can be analysed with the available tools to find the vulnerabilities. Useful tools can be Nikto, Nessus, and RouterSploit, etc. Open ports give the clue of the services running on the device. Fuzzing on the vulnerable ports helps to find unique crashes leading to new vulnerabilities. Though it is a time-consuming task, this will be the starting step to find new vulnerabilities.

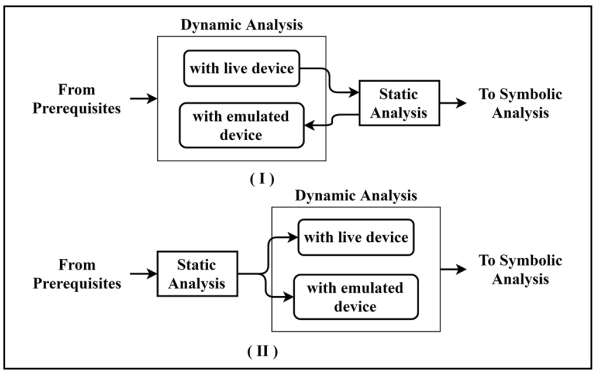

Fig. 3. Dynamic and static analysis relation

b) Online analysis with the emulated device: After the static analysis, when the firmware image is unpacked, emulating the firmware using Firmadyne [21] helps for carrying out the dynamic analysis of the web interface. Firmadyne is an automated dynamic analysis system that specifically targets Linux-based firmware. It emulates ARM and MIPS based firmware along with the NVRAM and network peripherals. This helps to emulate the document root folder of the device for carrying out the web application vulnerability testing. IP provided by the Firmadyne during emulation helps to find the vulnerabilities in other protocols such as SSH, Telnet, etc. The same set of tools used for the testing live device can be used for testing the emulated device.

C. Static Analysis

a) Firmware Retrieval: Once the device is received and its specifications are known, firmware can be retrieved. For that, identifying the medium of retrieval and its support is crucial. Generally, firmware can be retrieved through the network service/serial/JTAG port(s) interfaces. If none of the modes is successful, firmware can be downloaded from the vendor website. The hash of the retrieved firmware/files can be compared with that of the downloaded firmware from the vendor website. This helps to confirm i) if you have retrieved the firmware completely, and ii) whether the firmware inside the device is same as available on the vendor website. When it is not possible to retrieve the device firmware, and the device vendor hasn't made the firmware available for

download; a limited analysis based on intelligent fuzzing is suggested.

b) Firmware Unpacking: Unpacking the firmware is a challenge as the firmware is often packed with proprietary algorithms or can even be encrypted. Decrypting the encrypted file-system is a separate research problem. For unpacking the firmware, Binwalk [29] tool followed by GNU Coreutils command 'dd' and/or firmware modification kit [30] is suggested. Firmware modification kit contains various tools, such as unsquash_all, uncramfs_all, extract-sh, build-firmware.sh, firmware-modification-kit.sh, etc., for different types of activities. Binwalk provides an option for finding the entropy of the firmware image which helps us to know whether it is encrypted or not and also helps to find the sections of the firmware which are encrypted.

c) Firmware Static Analysis: After firmware unpacking, analysis of the extracted file system can be carried out. This analysis can reveal configuration files, scripts, cryptographic keys such as SSL and SSH, database files etc. These in turn can provide clues about the possible presence of vulnerabilities in the firmware. This phase can capture strings from binaries, which can be used during future analysis. Along with these clues, the document root folder of the device can also be analysed. Tools like VCG [31], RIPS [32] can be used for finding the vulnerabilities in the source code of the document root folder.

D. Symbolic Analysis

Symbolic Execution was introduced by James C. King [33] as a method of software testing, way back in 1976. For details of symbolic execution, the interested reader can refer to [34], [35], [36]. With the recent developments in technology, symbolic execution is used for security testing, as it can give the precise inputs which lead to the execution of specific portions of program functionality. To identify backdoors, it is required to have inputs which trigger backdoor functionality and thus symbolic execution helps in identifying backdoors. Hardcoded credential-based backdoor has been frequently reported to be found in SOHO routers [7], [8], [9]. For identification of such backdoors, symbolic execution can be used.

a) Preparation for identifying hardcoded credential-based backdoor functionality: Due to the nature of the problem at hand, the analyst need not try symbolic execution of all binaries of the firmware. In the pre-symbolic execution phase, the analyst can focus on identifying the binaries which behave as servers i.e. listening on incoming connections on a network. This can be achieved by looking for calls to APIs such as Socket, Listen, Accept, etc., while statically analysing the binary.

In parallel to this, the output of all strings from the binary should be saved for later analysis and confirmation of results. This output can be generated simply by using GNU strings command.

Binaries can't be symbolically executed directly. For symbolic execution of binary, it is required to lift the binary to an intermediate representation/format (IR). For lifting, frameworks such as MCSEMA [37], ANGR [38], etc. can be used. Further analysis will be dependent on the instruction set architecture (ISA) of the binary and the capabilities of the chosen framework.

b) Symbolic execution: There are many tools available for symbolic execution. The most prominent ones are a) using a combination of MCSEMA and REMILL [40] with KLEE and b) using ANGR.

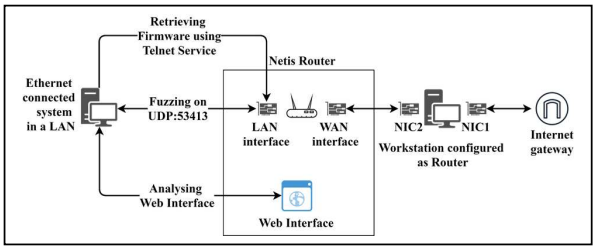

Fig. 4. Network setup for analysis of Netis WF2411

MCSEMA has support for x86, amd64 and aarch64 architectures [39]. Thus, if the firmware binary belongs to one of these architectures, one can use MCSEMA and REMILL for lifting binary to LLVM IR. After lifting the binary to IR, the target functions for symbolic execution has to be identified. For example, functions printing the output message to the console after successful login and performing the comparison of credentials can be focused on. Next, the identified functions have to be symbolically executed with KLEE. This tool will give all inputs which can help reach the specific program statement. For details on KLEE, refer [35]. Next, the inputs generated with KLEE should be searched in the output of strings command saved before. If a match is found, there is a high probability that this string is a hardcoded credential for a backdoor functionality. The results can be manually verified and confirmed.

ANGR is a multi-architecture binary analysis toolkit with the capacity to perform dynamic symbolic execution. It is implemented as a python library. It has a loader component called CLE (CLE Loads Everything). which can turn the executables and libraries into usable address spaces. This is then translated from architecture-specific native binary code into a VEX intermediate representation (IR). ANGR uses pyVEX for this purpose. For analysing the binaries, ANGR provides static analysis routines namely control flow analysis, data flow analysis and value-set analysis and symbolic execution engine. This engine is capable of even analysing a part of the binary which is of interest by leveraging CFG generation, forced execution, backward slicing and symbolic execution.

IV. Case Study of Netis WF2411 Router

To validate the above-described procedures, as a case study, a security analysis of Netis WF2411 router is presented in this section. Netcore is a popular brand for networking equipment, and routers manufactured by Netcore are sold under the brand name Netis. Following the analysis steps listed above, we could confirm the presence of an unencrypted HTTP authentication and a backdoor vulnerability. Details are given below.

A. Collecting the prerequisites

During this step, information about the network interfaces is collected. We found that the Netis WF2411 router has backdoor as per the report from TrendMicro [41]. During further analysis, we also confirmed the presence of the backdoor and a vulnerability.

B. Network Setup

The network setup used for the analysis of the Netis WF2411 router is presented in Figure 4. On the LAN side of the device, the analysis system is connected to the router. The analysis system is installed with tools namely (i) Binwalk: for unpacking the firmware, (ii) Vega scanner for analysing the web interface vulnerabilities, (iii) NMAP: to find the open ports on LAN side and (iv) peach fuzzer to fuzz the UDP port 53413.

C. Dynamic Analysis - I

After preparing the environment for dynamic analysis we scanned the network interfaces of the device. The NMAP scan of TCP and UDP ports revealed open TCP & UDP ports. The interesting ports open were TCP port 23 & UDP port 53413. TCP port 23 is a standard port for Telnet daemon and was helpful in firmware retrieval & UDP port 53413 is not a standard service port.

D. Static Analysis

a) Firmware Retrieval: Using the Telnet service, we retrieved the firmware. For firmware retrieval, the analysis computer was connected to the router at IP: 192.168.1.1 for Telnet service. After providing the default login credentials, we were able to log in via Telnet. By observing the banner of the Telnet login, it was concluded that the device has BusyBox V1.00-pre8 [42] and a Linux based firmware.

Memory Technology Device (mtd) provides the interface to raw flash devices of any Linux based system. One of the means to retrieve the firmware is to check the status of mtd under /proc/mtd. This hints of mtd0 holding "boot+cfg+linux" and mtd1 holding "root fs".

Looking at the details of /dev/mtd, we could deduce that /dev/mtdblock1 has the firmware of the device. This firmware was copied to the /tmp directory with the filename firmware.bin. Using TFTP service, we copied the firmware to the analysis PC for unpacking and further analysis.

b) Firmware Extraction: After the retrieval of firmware from the device, the extraction of firmware was done using binwalk [29]. At the first level, unpacking the firmware using binwalk resulted in two files: rootfs.bin, and boot-linux.bin. At the second level, unpacking rootfs.bin yielded a squashfs based file-system and busybox based kernel.

c) Firmware Static Analysis: In the extracted file system, configuration files, script files and binary files were listed and investigated. On applying the 'strings' command on the individual binaries/script files/configuration files, interesting strings resembling username and/or a password were pruned. These strings can play an important role in the discovery of backdoor accounts based on hardcoded credentials.

We found that the firmware has a document root folder in the extracted filesystem. In this device, the document root folder had CGI files. In this case, we used a vulnerability scanner on the web interface rather than analysing the CGI files.

E. Dynamic Analysis – II

We used a free & open-source Vega web security scanner [43] for the vulnerability assessment of the web interface. Vega found one high-risk vulnerability. It was found that communication over HTTP is unencrypted.

Further, we carried out fuzz testing on UDP port 53413 using Peach fuzzer [44]. The input test cases for fuzzing were the strings from the output of strings command as mentioned in the firmware static analysis. Following observations were made from the fuzz testing output:

a) Firmware Retrieval: Using the Telnet service, we retrieved the firmware. For firmware retrieval, the analysis computer was connected to the router at IP: 192.168.1.1 for Telnet service. After providing the default login credentials, we were able to log in via Telnet. Observing the banner of the Telnet login, it was concluded that the device has BusyBox V1.00-pre8 [42] and a Linux based firmware.

Memory Technology Device (mtd) provides the interface to the raw flash devices of any Linux based system. One of the means to retrieve the firmware is to check the status of mtd under /proc/mtd. This hints of mtd0 holding "boot+cfg+linux" and mtd1 holding "root fs".

i. For payload "AA", the device responded with prompt "Login:"

ii. For payload "AA..AAAAnetcore", device responded with prompt "Login Succeeded!"

iii. For payload "AA..AAAAls", device responded with a list of directories in root folder "bin dev etc lib media proc sbin sys tmp usr var web"

From the above output, one can conclude that open UDP port 53413 for Netis WF2411 router allows for remote login and command execution with specially crafted payload inputs, which is a backdoor vulnerability.

V. Conclusion

As of writing, there are no standards and formal procedures for the security testing of SOHO routers and other embedded devices. It is an ad hoc and man-power intensive task, where full automation is a challenge. This paper lists the procedures that can be followed for security analysis of Wi-Fi routers under the condition that the firmware is retrievable from the device and the file-system can be unpacked (i.e. extractable). In this paper, a case study on security analysis of Netis WF2411 router is presented, where the correlation of static & dynamic analysis was used. The identified strings from the firmware static analysis were given as input test cases for fuzz testing i.e. dynamic analysis. The output of fuzz testing provided us with the input, resulting in the stack-based buffer overflow and backdoor access to the device. With this, we could confirm the presence of the backdoor account on the device.

Evolving a generic process for security analysis for embedded devices is an arduous task and requires hiding many details specific to the category of device. As future work, we would like to perform the case study on the latest SOHO routers and FTTH (Fibre to the Home) devices.

Acknowledgement

We would like to thank Mr Mahesh Patil for reviewing and Mr MK Chaithanya for proofreading the paper. This work is supported by the Ministry of Electronics and Information Technology (MeitY) Govt. of India. Any views, opinions, and

findings made in this paper are only of the authors and do not reflect the views of MeitY.

REFERENCES

[1] K. Koscher, T. Kohno and D. Molnar, "SURROGATES: enabling near-real-time dynamic analyses of embedded systems," WOOT'15 Proceedings of the 9th USENIX Conference on Offensive Technologies, pp. 7-7, 2015.

[2] A. Ovadya, R. Ogen, Y. Mallah, N. Gilboa and Y. Oren, "Cross-router covert channels," in WOOT'19 Proceedings of the 13th USENIX Conference on Offensive Technologies, 2019.

[3] G Richer, "Reverse-Engineering 4g Hotspots for Fun, Bugs and Net Financial Loss," 10 August 2019. [Online]. Available: https://media.defcon.org/DEF%20CON%2027/DEF%20CON%2027%20presentations/DEFCON-27-grichter-Reverse-Engineering-4G-Hotspots-For-Fun-Bugs-Net-Financial-Loss.pdf. [Accessed 30 July 2020].

[4] Grzegorz Wypych, "Critical RCE Vulnerability in TP-Link Wi-Fi Extenders Can Grant Attackers Remote Control," IBM X-Force, 18 June 2019. [Online]. Available: https://securityintelligence.com/posts/critical-rce-vulnerability-in-tp-link-wi-fi-extenders-can-grant-attackers-remote-control/?mhsrc=ibmsearch_a&mhq=Grzegorz%20Wypych. [Accessed 30 July 2020].

[5] Matthew Garrett, "Remote code execution as root from the local network on TP-Link SR20 routers," Google, 28 March 2019. [Online]. Available: https://mjg59.dreamwidth.org/51672.html. [Accessed 30 July 2020].

[6] Tim Carrington, "A Curious Tale of Remote Code Execution, The TP-Link Story - CVE-2017-13772," Fidus' Penetration Testing & Research team, 17 October 2017. [Online]. Available: https://fidusinfosec.com/tp-link-remote-code-execution-cve-2017-13772/. [Accessed 30 July 2020].

[7] Daniele Lepido, "Vodafone Found Hidden Backdoors in Huawei Equipment," Bloomberg, 30 April 2019. [Online]. Available: https://www.bloomberg.com/news/articles/2019-04-30/vodafone-found-hidden-backdoors-in-huawei-equipment. [Accessed 30 July 2020].

[8] Lucian Armasu, "Backdoors Keep Appearing In Cisco's Routers," tomshardware, 19 July 2018. [Online]. Available: https://www.tomshardware.com/news/cisco-backdoor-hardcoded-accounts-software,37480.html. [Accessed 30 July 2020].

[9] Denis Makrushin, "Backdoors in D-Link's backyard," Kaspersky, 23 May 2018. [Online]. Available: https://securelist.com/backdoors-in-d-links-backyard/85530/. [Accessed 30 July 2020].

[10] A. Costin, J. Zaddach, A. Francillon and D. Balzarotti, "A large-scale analysis of the security of embedded firmwares," in SEC'14 Proceedings of the 23rd USENIX conference on Security Symposium, 2014.

[11] "TR-069 CPE WAN Management Protocol," Broadband Forum, 2018.

[12] G. Kambourakis, M. Anagnostopoulos, W. Meng and P. Zhou, "Botnets : Architectures, Countermeasures, and Challenges," , 2019.

[13] M. Smith and M. Green, "A Discussion of Surveillance Backdoors: Effectiveness, Collateral Damage, and Ethics," 2017, pp. 131-142.

[14] [Online]. Available: https://www.dhs.gov/science-and-technology/mobile-security-rd.

[15] [Online]. Available: https://www.darpa.mil/program/vetting-commodity-it-software-and-firmware.

[16] "Software and System Security Group," [Online]. Available: http://s3.eurecom.fr/. [Accessed 04 August 2020].

[17] S. L. Thomas, F. D. Garcia and T. Chothia, "HumIDIFy: a tool for hidden functionality detection in firmware," in International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment, 2017.

[18] M. Oliver and K. O'Meara, "Framework for Embedded Device Vulnerability Analysis," , 2017.

[19] V. Visoottiviseth, P. Jutadhammakorn, N. Pongchanchai and P. Kosolyudhthasarn, "Firmaster: Analysis Tool for Home Router Firmware," in 2018 15th International Joint Conference on Computer Science and Software Engineering (JCSSE), 2018.

[20] V. Visoottiviseth, P. Akarasiriwong, S. Chaiyasart and S. Chotivatunyu, "PENTOS: Penetration testing tool for Internet of Thing devices," in TENCON 2017-2017 IEEE Region 10 Conference, 2017.

[21] D. D. Chen, M. Woo, D. Brumley and M. Egele, "Towards Automated Dynamic Analysis for Linux-based Embedded Firmware.," in NDSS, 2016.

[22] A. Costin, A. Zarras and A. Francillon, "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, 2016.

[23] J. Zaddach, L. Bruno, A. Francillon, D. Balzarotti and others, "AVATAR: A Framework to Support Dynamic Security Analysis of Embedded Systems' Firmwares.," in NDSS, 2014.

[24] N. Redini, A. Machiry, D. Das, Y. Fratantonio, A. Bianchi, E. Gustafson, Y. Shoshitaishvili, C. Kruegel and G. Vigna, "Bootstomp: on the security of bootloaders in mobile devices," in 26th {USENIX} Security Symposium ({USENIX} Security 17), 2017.

[25] Y. Shoshitaishvili, R. Wang, C. Hauser, C. Kruegel and G. Vigna, "Firmalice-automatic detection of authentication bypass vulnerabilities in binary firmware.," in NDSS, 2015.

[26] D. Davidson, B. Moench, T. Ristenpart and S. Jha, "{FIE} on firmware: Finding vulnerabilities in embedded systems using symbolic execution," in 22nd {USENIX} Security Symposium ({USENIX} Security 13), 2013.

[27] K. Pattabiraman, N. Nakka, Z. Kalbarczyk and R. Iyer, "Discovering application-level insider attacks using symbolic execution," in IFIP International Information Security Conference, 2009.

[28] T. Weber, "Reverse Engineering Hardware of Embedded Devices: From China to the World," 2017.

[29] "Binwalk," ReFirm Labs, [Online]. Available: https://github.com/ReFirmLabs/binwalk. [Accessed 10 August 2020].

[30] "Firmware Mod Kit," [Online]. Available: https://github.com/rampageX/firmware-mod-kit. [Accessed 10 August 2020].

[31] [Online]. Available: https://github.com/nccgroup/VCG.

[32] [Online]. Available: https://www.ripstech.com/.

[33] KingJamesC, "Symbolic execution and program testing," Communications of the ACM, vol. 19, p. 385–394, 1976.

[34] C. Cadar and K. Sen, "Symbolic execution for software testing: three decades later," Communications of The ACM, vol. 56, no. 2, pp. 82-90, 2013.

[35] C. Cadar, D. Dunbar and D. Engler, "KLEE: unassisted and automatic generation of high-coverage tests for complex systems programs," in OSDI'08 Proceedings of the 8th USENIX conference on Operating systems design and implementation, 2008.

[36] C. S. Păsăreanu and W. Visser, "A survey of new trends in symbolic execution for software testing and analysis," International Journal on Software Tools for Technology Transfer, vol. 11, no. 4, pp. 339-353, 2009.

[37] A. Dinaburg and A. Ruef, "Mcsema: Static translation of x86 instructions to llvm," in ReCon 2014 Conference, Montreal, Canada, 2014.

[38] Y. Shoshitaishvili, R. Wang, C. Salls, N. Stephens, M. Polino, A. Dutcher, J. Grosen, S. Feng, C. Hauser, C. Kruegel and others, "Sok:(state of) the art of war: Offensive techniques in binary analysis," in 2016 IEEE Symposium on Security and Privacy (SP), 2016.

[39] "McSema," Trail of Bits, Inc., [Online]. Available: https://github.com/lifting-bits/mcsema. [Accessed 04 August 2020].

[40] "Remill," Trail of Bits, Inc., [Online]. Available: https://github.com/lifting-bits/remill. [Accessed 04 August 2020].

[41] Tim Yeh, "Netis Routers Leave Wide Open Backdoor," TrendMicro, 25 August 2014. [Online]. Available: https://blog.trendmicro.com/trendlabs-security-intelligence/netis-routers-leave-wide-open-backdoor/. [Accessed 10 August 2020].

[42] EWEN MACASKILL, EWEN MACASKILL, "NSA Files decoded," The Guardian, 01 November 2013. [Online]. Available: https://www.theguardian.com/world/interactive/2013/nov/01/snowden-nsa-files-surveillance-revelations-decoded#section/1. [Accessed 04 August 2020].

[43] "Vega Vulnerability Scanner," Subgraph, [Online]. Available: https://subgraph.com/vega/. [Accessed 11 August 2020].

[44] "Peach Fuzzer Community Edition," PeachTech, [Online]. Available: https://www.peach.tech/resources/peachcommunity/. [Accessed 10 August 2020].