Detecting and Mitigating Rootkits in Embedded Systems

Jeremy Porter and Adam Bryant
Wright State University, USA
porter.149@wright.edu
adam.bryant@email.com

Abstract: Networking is pervasive in our daily lives and we often assume the network is secure. With actors ranging from organizations, manufacturers, and nation-states, launching sophisticated attacks the need to ensure the integrity of network equipment is critical. However, attacks on router firmware are numerous and it is easy to demonstrate firmware attacks. In particular, rootkits are an especially troublesome type of attack because they involve malicious code that attackers intentionally make difficult to detect. Much of the recent research in securing network devices focuses on finding vulnerabilities in firmware and web-based management interfaces. Those methods emulate firmware using standard emulation tools such as QEMU, but they do not fully emulate the kernel which is where most rootkit code is stored and from where it runs. Instead, they involve building a custom kernel and emulating the file system, but they cannot readily emulate the hardware portion of an embedded system. In this paper, we develop a framework for detecting and mitigating rootkits in embedded devices using static and dynamic analysis. Our emphasis is on improving the security of commodity routers and networking equipment. Our goal is to develop a system capable of emulating embedded system hardware, to reverse engineer firmware and detect the hardware requirements of embedded systems, and to create a wedge or modify QEMU to emulate the hardware. We will use these emulation techniques to develop progressively sophisticated detection (and eventually protection) technologies to combat rootkits in network devices. We will start with simple methods such as comparing the hash values of trusted firmware and suspect firmware images, perform static analysis, fully emulate the kernel, and use that emulation to perform dynamic analysis paired with network traffic analysis. Since network equipment runs Linux-based firmware, we limit our scope to the investigation of rootkit techniques on Linux-based firmware.

Keywords: rootkits, embedded systems, emulation, QEMU, Linux firmware

1. Background and motivation

Currently there are three potential sources for malicious firmware: malware developers, manufacture developed, or state sponsored attackers. “Malware developers are working on long term attacks, which will give hackers an ongoing and virtually undetectable access to the target system” (Korkin & Nesterov, 2014).

Manufacturers have created rootkits like Lenovo Service Engine (Gibson & Laporte, 2015). The National Security Agency (NSA) developed malware called EquationDrug and GrayFish that obtains a payload code from the Internet to flash the existing hard drive firmware (Zetter, 2015).

Currently, there are two systems for analyzing embedded firmware (Chen, et al., 2016) and (Costin, et al., 2015). These two systems use dynamic and static analysis on embedded firmware to detect vulnerabilities. Neither system emulates the kernel. Instead, they build a custom kernel and emulate the file system.

2. Existing research

2.1 Dynamic analysis of embedded firmware

In (Costin, et al., 2015) the authors build a cloud-based framework to validate the firmware of embedded devices. They find approximately 10% of tested firmware has some form of vulnerability. (Costin, et al., 2015) perform both static and dynamic analysis of embedded web interfaces within the firmware. From static analysis, they find cross-site scripting and file manipulation are the majority of vulnerabilities. From dynamic analysis, they find a variety of vulnerabilities including command injection, cross-site scripting, and cross-site request forgery (Costin, et al., 2015, p. 8). While the authors do not explicitly state these vulnerabilities lead to firmware-based rootkits, it is likely that at least some of the vulnerabilities found lead to firmware compromise.

The framework (Costin, et al., 2015) build is certainly remarkable, but it does not fully emulate the firmware kernel. Instead, it uses a custom kernel and emulates the filesystem of the firmware (Costin, et al., 2015, pp. 4-5). The problem with this approach in detecting rootkits is that they may reside directly in the kernel and therefore this method does not appear to detect rootkits.

2.2 Firmadyne

In (Chen, et al., 2016) the authors describe Firmadyne. Firmadyne has four primary components: 1) it downloads firmware images from vendors, 2) it extracts the firmware file system, 3) it uses a custom kernel that matches the requirements of the firmware to emulate in QEMU, and 4) it performs dynamic analysis on the firmware. Firmadyne uses QEMU with a custom Linux kernel to attach to firmware file system and run various tools on the system (Chen, et al., 2016).

The problem here is that rootkits are not just in the file system. Rootkits can be part of the Linux kernel itself. For example, the SucKIT rootkit patches the kernel and does not require loadable kernel module support (sd & devik, 2001). Thus, it is necessary to emulate an existing embedded system with minimal modification to the firmware. This means emulating both the kernel and the filesystem.

2.3 Router Post Exploitation Framework (RPEF)

RPEF (Coppola, 2012) is a framework to inject malicious code directly into router firmware. This tool first unpacks the firmware. It breaks the firmware into its constituent pieces using standard I/O then fully expands the file system with appropriate tools like unsquashfs. Once the firmware is unpacked, it adds malicious code by creating a script with the same name as a chosen binary (httpd) while replacing the chosen binary with malicious code. RPEF repacks the file system and the firmware to its original state with the modifications.

3. Methodology

We plan to rectify the problem of emulation in our framework while leveraging the work of (Chen, et al., 2016) and (Costin, et al., 2015). We plan to use RPEF (Coppola, 2012) to inject malicious code and emulate that along with emulating unmodified firmware. Using this information as a baseline, we propose that detecting firmware rootkits is possible.

We plan to build a framework to detect and mitigate firmware rootkits. We expect to detect rootkits unknown rootkits without prior knowledge of the rootkit. The system will perform static and dynamic analysis of the firmware and monitor network behavior to indicate behavior that suspect.

As a simplified approach, we plan to take the following steps to detecting firmware rootkits:

- Start with unmodified firmware
- Inject a rootkit into unmodified firmware using RPEF
- Use Firmadyne with kernel emulation in QEMU perform static and dynamic analysis
- Examine the results and examine the rootkit behavior

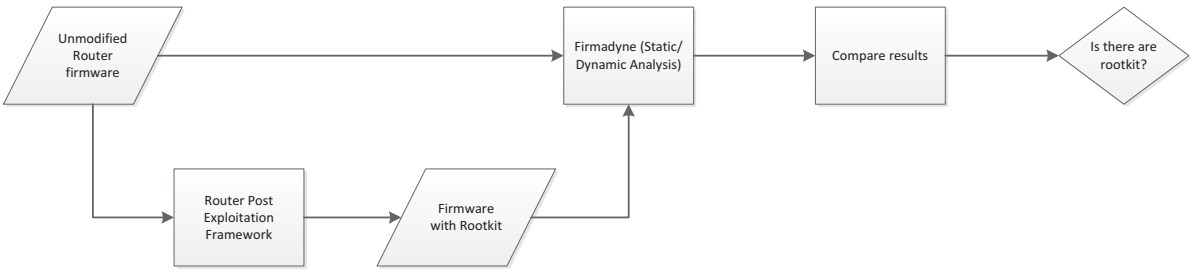

Figure 1: Framework to detect rootkits

3.1 Understanding emulation

As first step to building the above framework, we have done work to understand emulation and examine the technical problems of emulating the firmware including its kernel. We have found that emulating firmware including the kernel is non-trivial. We have followed the work of (Holt & Huang, 2014) in building custom embedded systems and emulating in User Mode Linux. We extended this work to emulate OpenWrt in QEMU.

User Mode Linux is essentially a sandbox that allows a piece of code to execute in a well-defined space in an existing operating system. We used User Mode Linux to emulate a simple system from source. We took the following steps for a simple system:

- Create a root filesystem
- Build a kernel
- Build a root filesystem
- Run the system in User Mode Linux

We continued the process building a more complex embedded system from source including GLIBC, busybox, NCurses, and SysVinit. Otherwise, we followed the steps outlined above.

We continued our work on emulation by building OpenWrt from source and emulating in QEMU instead of User Mode Linux. User Mode Linux is limited in that it relies on the host architecture and cannot emulate stems of a different architecture from the host. QEMU does not have this restriction. Building OpenWrt for QEMU is nearly identical to the above steps.

Through our work, we found that it is important to understand how hardware drivers work in embedded systems. We suspect that by learning how to emulate (or work around) drivers, full kernel emulation will be possible. We believe we can leverage QEMU to emulate firmware including the kernel. We plan to build custom drivers for QEMU or modify QEMU to emulate drivers for existing firmware. We will then use the modified system to emulate unmodified firmware.

4. Current status

We are currently building a kernel emulation system that will allow for rootkit detection as previously described. Emulating the kernel is one of the major hurdles in creating a full firmware rootkit detection and mitigation framework. Firmware often relies on proprietary hardware and it is not trivial to identify hardware requirements. The firmware must have some indication about what the hardware requirements are. The first step is to understand where this information is stored and how detailed it may be.

We built firmware using OpenWRT and emulated the resultant system in QEMU. We plan to reverse engineer the firmware and to understand how firmware uses drivers for hardware. Disassembling OpenWrt should prove instructive in the process of reverse engineering embedded firmware. Reverse engineering firmware should provide understanding into the executable format of OpenWrt and more specifically how device drivers are implemented (loadable kernel modules vs. kernel drivers). Once an understanding of how drivers are managed in the firmware can be ascertained from, the disassembled (not source) code then this same methodology can be applied to a vendor provided piece of firmware.

We plan to build a better understanding of how hardware drivers and the kernel interact in embedded systems. Using this knowledge, we plan to either modify QEMU or build appropriate driver wedges. We emulate unmodified firmware including the kernel. We will leverage Firmadyne to perform static and dynamic analysis and compare the results to detect firmware rootkits.

5. Conclusions

The need for resolving the security concerns in firmware is an important part of securing networks in the modern landscape. While there are some existing tools to analyze firmware, these systems do not provide an easy mechanism to detect rootkits.

We are working to mitigate the risk of rootkits in firmware. The rootkit detection and mitigation framework described should provide a mechanism for emulating the kernel and file system. The goal of this system is to provide a full suite of static, dynamic, and network analysis.

Currently, the focus is on emulating the unmodified kernel of the firmware. The kernel is the most likely place for a rootkit. We are focused is on understanding how the kernel interacts with drivers and how to emulate that hardware driver. Reverse engineering known systems will provide insight on how to identify necessary drivers. This will provide a base set of knowledge on how to emulate kernels in QEMU.

References

Chen, D. D., Egele, M., Woo, M. & Brumley, D., 2016. *Towards Automated Dynamic Analysis for Linux-based Embedded Firmware.* s.l., s.n.

Coppola, M., 2012. *Def Con 20 Archive.* [Online] Available at: https://www.defcon.org/html/links/dc-archives/dc-20-archive.html [Accessed 27 January 2016].

Costin, A., Zarras, A. & Francillon, A., 2015. *Automated Dynamic Firmware Analysis at Scale: A Case Study on Embedded Web Interfaces.* [Online] Available at: http://arxiv.org/pdf/1511.03609v1.pdf

Gibson, S. & Laporte, L., 2015. *Security Now! 521: Security is Difficult.* [Online] Available at: https://www.grc.com/sn/sn-521.htm

Holt, A. & Huang, C.-Y., 2014. *Embedded Operating Systems: A Practical Approach.* London: Springer-Verlag.

Korkin, I. & Nesterov, I., 2014. *Applying Memory Forensics to Rootkit Detection.* Richmond, VA, Association of Digital Forensics, Security and Law, pp. 115-142.

Lenovo, n.d. *SuperFish Uninstall Instructions.* [Online]

Available at: https://support.lenovo.com/us/en/product_security/superfish_uninstallsd & devik, 2001. Linux On-the-fly Kernel Patching without LKM. *Phrack*, 12 December.

Zetter, K., 2015. *Wired.* [Online] Available at: http://www.wired.com/2015/02/nsa-firmware-hacking/

Alan Lin (PhD) received a B.S. in Computer Engineering from Rutgers University in 2004, an M.S. and PhD in Computer Science from the Air Force Institute of Technology (AFIT) in 2008 and 2015, respectively. He is currently an assistant professor at AFIT, teaching networking and secure programming courses.

Nicholas Masceri is a junior Criminal Justice major at Temple University. He has interests in pursuing a career in ethical hacking or working for criminal justice agencies. Also he participated in research with engineering students at Temple University and Idaho National Labs. He is currently working with Dr. Rege on her NSF CPS project

Svitlana Matviyenko is a Research Associate in the Faculty of Information and Media Studies at the University of Western Ontario, co-editor of The Imaginary App (2014), and author of "Liquid Categories for Augmented Revolutions", available at the website The Exceptional and the Everyday: 144 Hours in Kiev.

Paul Maxwell (PhD) is an associate professor in the department of Electrical Engineering and Computer Science. His current position is the Cyber Fellow of Computer Engineering at the Army Cyber Institute at West Point. He has a BSEE, MSEE, and PhD in electrical engineering. His research interests include programmable logic, computer architecture, robotics, and robustness.

Eric McKinion received a B.S. in Software Engineering from Mississippi State University in 2015 and is currently finishing a M.S. in Cyber Operations at the Air Force Institute of Technology while stationed at Wright-Patterson Air Force Base, OH.

Barry E. Mullins (PhD) is Professor of Computer Engineering at the Air Force Institute of Technology, Wright-Patterson AFB, OH, USA. He served 21 years in the Air Force teaching at the U.S. Air Force Academy for seven of those years. He is a registered Professional Engineer in Colorado and a member of IEEE, ASEE, Tau Beta Pi, Eta Kappa Nu, Phi Beta Chi (Science), and Kappa Mu Epsilon (Mathematics). His research interests include cyber operations, critical infrastructure protection, computer/network/embedded systems security, and reverse code engineering.

Wynand Nel is a lecturer in the Department of Computer Science and Informatics at the University of the Free State, South Africa. His research focus is Human Computer Interaction and he is lecturing modules in Information Security and Digital Forensics.

Sean P. O'Neill (**Captain)** received his of Science in Electrical Engineering from California State Polytechnic University Pomona in 2011. Soon after graduation, Sean commissioned in the United States Air Force and is currently attending the Air Force Institute of Technology for his graduate degree in Electrical Engineering with emphasis in Software Engineering.

Luke Osterritter is a Cyber Security Engineer and Member of the Technical Staff at the CERT division of Carnegie Mellon University's Software Engineering Institute. He has professional experience relating to cyber security, workforce development, data center operations, and system and network engineering. His interest areas include social and behavioral factors in cyber security, public key infrastructure, network and systems architecture. Mr. Osterritter holds a B.S. in Information Sciences and Technology from The Pennsylvania State University and a Master of Science in Information Science from the University of Pittsburgh.

Seeley Pentecost (2Lt) is a master's student at the Air Force Institute of Technology majoring in electrical engineering. He is from New Braunfels, TX and received his B.S. in electrical engineering from Texas A&M University in 2016. His current work focuses on advanced signal concepts for GPS. Seeley's research interests include digital systems, cryptography and cyber security

Anthony Portante is a 2nd Lieutenant in the U.S. Air Force pursuing his Masters of Science degree in Cyber Operations at the Air Force Institute of Technology. His areas of research are in network security and network attack with his thesis focusing on Software Defined Networks.

Jeremy Porter is a graduate student at Wright State University studying Cyber Security. His research interests include embedded systems, rootkits, and Internet of Things (IoT). He holds several industry certifications including CISSP, Certified Ethical Hacker, and CCNP. He also teaches Information Security and Cisco Network Academy courses.