Don't Trust Your Router: Detecting Compromised Routers

Ahmad Darki, Alexander Duff, Zhiyun Qian, Gaurav Naik
Spiros Mancoridis, Michalis Faloutsos
Computer Science and Engineering
University of California, Riverside, Drexel University
adark001@ucr.edu

ABSTRACT

Safeguarding one's router has received very little attention despite a plethora of router-specific malware, which has emerged recently. Here, we propose a systematic approach to distinguish a router infected by malware from a healthy router. Our key novelty is that we analyze the behavior of the router, thus not relying on binary signatures (like anti-virus software for computers). Our contribution is two fold. First, we develop a non-trivial emulation capability, to observe the behavior of a router. This capability allows to instantiate a virtual router with or without malware and feed it a pre-recorded data trace. This setup monitors the behavior at multiple layers including: OS system calls, process information, and the network layer. Second, using the emulated environment, we provide initial evidence that a behavior-based method can distinguish between infected and healthy routers. We have collected 820 router-specific malware binaries and an initial set of real data traces. We find that infected routers exhibit: (a) an initial spike and an overall 50% increase in the number of system calls, (b) an initial spike and a modest increase in the number of active processes. Our preliminary work is a promising step towards understanding and securing routers against malware infections.

1. PROBLEM DEFINITION

On October 2016, there has been a series of DDoS attack on *Dyn, Inc.* servers using IoT devices including routers[1]. It is clear that the compromising embedded systems is becoming a serious threat, whether it is for stealing private information or carrying out DDoS. [2]. The lack of mature protection technologies makes this a fertile ground for attacks. With a compromised router, the attacker can gain access to the network packets being forwarded by it and steal sensitive information as well as cookies and session IDs[3]. In addition, DNS hijacking can be done by means of which an adversary can subvert DNS requests and redirect all requests to a rogue DNS server controlled by the attacker. Finally, an infected router can be used as a zombie to carry out DDoS attacks. Such attacks, targeting routers and other embedded systems, introduce a new class of threats in the computer security community.

Attacking and protecting embedded systems has its own challenges. Such systems have limited resources in terms of CPU, and memory. This raises the challenge for the malware developers to utilize the limited resources to perform their act. However, the same challenges apply to the security solutions as well. When designing embedded systems in production, a default setup configuration is used for an automatic first time installation and any further changes on the devices needs to be done manually by users. Such configurations include default username and password, as well as an initial network setup. While a default setup configuration may result in less work for users, it introduces a new threat in which an adversary can exploit such devices by knowing the default configurations.

CoNEXT '16 December 12–15, 2016, Irvine, CA, USA

ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235

2. CONTRIBUTION

In this work, we have developed an environment to emulate a realistic setup for a router and monitor its behavior by collecting raw data. We emulate the hardware architecture of the router (be it `ARM/MIPS` both `Big Endian` and `Little Endian`) using `QEMU`[4].

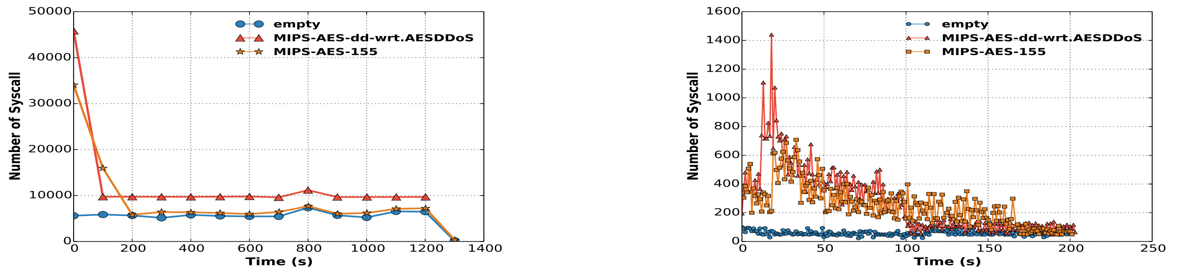
A key capability of our system is that it enables the replaying of any given network trace. To achieve this, we create two subnetworks of `LAN` and `WAN` for QEMU. Using this we can connect the router to the Internet and servers using the WAN subnet, and also the clients can be connected through LAN. In order to feed the traffics of clients and servers we used `tcpreplay`[5].

Another key property is that we can test different routing `firmware` on our emulated hardware. We analyzed the firmware binary files commonly used by SOHO routers, such as Linksys and TP-Link. These firmwares are based on an open source project called `openwrt` [6]. The differences between the vendor specific firmware and the openwrt are in modules, embedded web interfaces, and more importantly the `rc` files that initialize the router. The `rc`, or sometimes `rc.S`, file holds in the initial configuration of a router, namely the default username and password as well as the network configuration.

We use the 820 router specific malware binaries from a non-profit organization, that requested to stay anonymous, and we monitor the infected routers. We use `tcpreplay` to feed pre-recorded network traffic collected by `MAWI`[7] to the router. This is a powerful capability for our framework since researchers can study the dynamic behavior of the router under variable data traces. This has been a challenging endeavor that required careful instrumentation of our infrastructure.

We created an analysis module to process and profile the collected data. The analysis module is responsible for analyzing the following collected behavioral data:

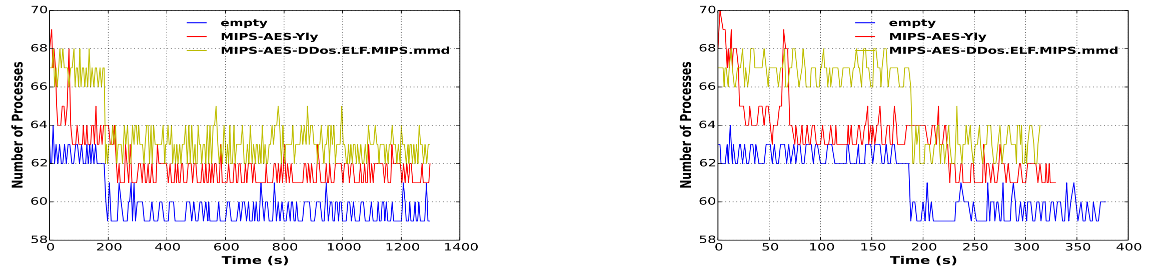
- Network Traffic: Traces on both the `lan` and `wan` will be analyzed and flows will be detected in order to distinguish the



(a) Intervals of 100 seconds

(b) Intervals of 1 seconds

Figure 1: The number of system calls of a benign and two infected routers over 1300 seconds. (a) The number of system calls in infected routers is roughly 50% more than a benign one. (b) Focusing on the first 200 seconds: the infected routers exhibit a spike in the number of system calls.



(a) 1300 seconds of execution

(b) First 400 seconds of execution

Figure 2: The number of active processes of a benign and two infected routers. (a) The number of processes of the experiment. (b) The first 400 seconds shows that malware spawns child processes and kills some of the existing ones.

different flows that do not match the ones that are being fed to the router. The unmatched flows and packets will be mapped to the corresponding processes that generated the packets.

- System Calls: Sequence of system calls will be analyzed in different aspects.
  - Timeseries: Identifying behavioral patterns in the time series data.
  - Network: Analyzing the behavior of the system based on network level interactions.
  - System calls: Dynamic analysis of malware based on the system calls.
- Processes: Identify the default processes in the router and determine the effects of the malware on them. Some malware samples kill processes in the router in order to gain more resources as well as to exterminate any anti-virus or firewall processes.
- Memory Usage: Memory usage and related operations can also reveal anomalous activity.

3. EXPERIMENTS AND RESULTS

One of the notable aspects of our work is the ability to easily repeat experiments with different configurations and data traces. Furthermore, we have an extensive number of malware samples available, which can help us identify common and persistent malware behaviors.

Assessing the "correctness" of our setup. We created multiple virtual `Debian` machines using `qemu` to act as clients inside the host machine, and have them connect to the router's `LAN`. We had the clients to connect to make `ping`, `ssh`, `HTTP(s)`, and `DNS` requests to a selected number of websites and servers. The results show that all the requests went through the router. We were ale to confirm that the router operation is correct in terms of processing packets.

We conducted the following types of experiments as an initial proof of the usefulness of this type of analysis.

- Live Internet connection: The router can have the Internet connection enabled or disabled. Using this option we can analyze how differently a router functions when it is connected to the Internet. Using this technique we can look into the behavior of the malware and see their dependencies to their `Command and Control` server.
- Replayed Traffic: As mentioned earlier, with this setup we can feed the router any network traces that have been recorded previously. On one side we feed client's traffic, and on the other side we feed the server's traffic.
- Presence of malware: We have 820 malware samples that are compatible with different hardware architectures. Using this system we can infect an instance of a router and collect raw data on the router's behaviour.

Figure 1 and 2 show an example of three experiments with one benign and two infected routers. We examine the number of system calls and processes and it shows that the malware introduces significant deviations into the router's behavior.

4. FUTURE WORK

Following the promising results here, we intend to further expand on our profiling techniques and to emulate different vendor firmwares. We also plan to create multiple honeypot farms to detect and collect attacks and malware for routers. The overarching goal is to protect routers with the same comprehensive approaches that we protect desktops and laptops.

5. PREVIOUS WORK

Recently there has been a work done by Chen et al. related to SOHO routers [8]. In this work a dynamic analysis of different vendors' firmwares has been done, by using which a list of known vulnerabilities has been detected in their web interface.

Another more general work has been done by Costin et al. [9], in which an Internet crawler has been developed to find and download firmware images of every embedded systems available. Using static analysis on the collected firmware images, they identified backdoors in their embedded web interfaces as well as shared credentials between multiple devices from the same vendor.

Acknowledgement

This work was supported by DHS ST Cyber Security (DDoSD) HSHQDC-14-R-B00017 grant.

6. REFERENCES

[1] Wikipedia. October 2016 dyn cyberattack: https://en.wikipedia.org/wiki/october_2016_dyn_cyberattack, 2016.

[2] Dorottya Papp, Zhendong Ma, and Levente Buttyan. Embedded systems security: Threats, vulnerabilities, and attack taxonomy. In *Privacy, Security and Trust (PST), 2015 13th Annual Conference on*, pages 145–152. IEEE, 2015.

[3] Olivier Bilodeau and Thomas Dupuy. Dissecting Linux/Moose The Analysis of a Linux Router-based Worm Hungry for Social Networks. (May), 2015.

[4] Fabrice Bellard. Qemu, a fast and portable dynamic translator. In *USENIX Annual Technical Conference, FREENIX Track*, pages 41–46, 2005.

[5] Appneta. Tcpreplay: http://tcpreplay.appneta.com/, 2016.

[6] Openwrt.org, linux distribution for embedded devices.

[7] K Cho, K Mitsuya, and A Kato. Traffic data repository maintained by the mawi working group of the wide project. *URL http://mawi. wide. ad. jp/mawi*.

[8] Daming D Chen, Manuel Egele, Maverick Woo, and David Brumley. Towards automated dynamic analysis for linux-based embedded firmware. In *ISOC Network and Distributed System Security Symposium (NDSS)*, 2016.

[9] Andrei Costin, Jonas Zaddach, Aurélien Francillon, and Davide Balzarotti. A large-scale analysis of the security of embedded firmwares. In *23rd USENIX Security Symposium (USENIX Security 14)*, pages 95–110, San Diego, CA, August 2014. USENIX Association.