RARE: A Systematic Augmented Router Emulation for Malware Analysis

Ahmad Darki, Chun-Yu Chuang, Michalis Faloutsos, Zhiyun Qian, Heng Yin

University of California, Riverside,
Riverside, CA, USA

Abstract. How can we analyze and profile the behavior of a router malware? This is the motivating question behind our work focusing on router. Router-specific malware has emerged as a new vector for hackers, but has received relatively little attention compared to malware on other devices. A key challenge in analyzing router malware is getting it to activate, which is hampered by the diversity of firmware of various vendors and a plethora of different platforms. We propose, RARE, a systematic approach to analyze router malware and profile its behavior focusing on home-office routers. The key novelty is the intelligent augmented operation of our emulation that manages to fool malware binaries to activate irrespective of their target platform. This is achieved by leveraging two key capabilities: (a) a static level analysis that informs the dynamic execution, and (b) an iterative feedback loop across a series of dynamic executions, whose output informs the subsequent executions. From a practical point of view, RARE has the ability to: (a) instantiate an emulated router with or without malware, (b) replay arbitrary network traffic, (c) monitor and interact with the malware in a semi-automated way. We evaluate our approach using 221 router-specific malware binaries. First, we show that our method works: we get 94% of the binaries to activate, including obfuscated ones, which is a nine-fold increase compared to the 10% success ratio of the baseline method. Second, we show that our method can extract useful information towards understanding and profiling the botnet behavior: (a) we identify 203 unique IP addresses of C&C servers, and (b) we observe an initial spike and an overall 50% increase in the number of system calls on infected routers.

1 Introduction

Compromising routers is emerging as a new type of threat with potentially devastating effects [24]. For example, on October 2016, in Mirai botnet attack there has been a series of DDoS attacks on *Dyn, Inc.* servers using IoT devices including routers [2]. The lack of mature protection technologies makes this a fertile ground for attacks. We argue that a compromised router provides significant

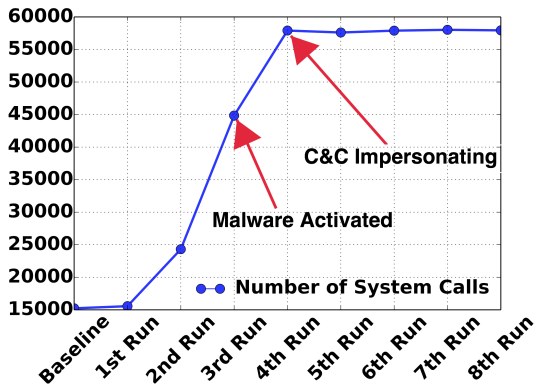

Fig. 1: RARE gets the router malware to activate and communicate with the C&C server in the 3^{rd} run of its iterative operation. We plot the number of system calls for each 1300 seconds of a run. The baseline approach is shown first. The 1^{st} run of RARE is an emulation informed by the static analysis only. Each subsequent run is informed by the previous run.

new capabilities to an attacker, beyond those of a compromised end-device. By compromising a router, the attacker can: (a) access or block the network packets going through it, (b) steal cookies and session IDs[25] to impersonate the user or compromise her privacy, and (c) hijack and redirect communication via a DNS redirection to rogue DNS server.

Attacking and protecting routers is significantly different compared to laptops and desktops, therefore new methods and tools are needed. First, routers have limited resources in terms of CPU, and memory. Therefore, malware developers have less resources in their disposition. However, the same challenges apply to the security solutions as well. Second, routers have variable device configurations [16] that decreases the applicability of both malware, and system analysis tools. The former is a challenge for malware authors, while the latter is a challenge for any emulation capability, which needs to pretend to be many different configurations in order to get firmware and malware to run. Finally, many, if not most, router firmware are proprietary, and thus difficult to emulate [6].

Our goal in this paper is to fully understand router malware binaries and their operation focusing on off-the-self home-office routers.The desired output of this work is two-fold. First, we want to analyze the malware in order to create an environment that will “fool” it to activate and reveal its behavior. Second, we want to profile and distinguish the behavior of an infected router from that of a benign one. The overarching challenge is the plethora of proprietary firmware and hardware router configurations, as we mentioned above. In addition, there is a scarcity of tools for static analysis for MIPS and ARM architectures, which are the most common platforms for routers. Such tools could have helped inform the emulation environment. As we will see below, a straightforward emulation attempt could have a very low success rate in fooling the malware to activate.

Router malware has received relatively little attention compared to malware on other devices. We can distinguish the following areas of research: (a) developing emulator capabilities [4, 17, 13, 27, 13, 27], (b) vulnerability analysis for embedded devices, [10, 6, 11, 14], and (c) malware analysis focusing on PC and smartphone based malware [9, 22, 20, 21]. We discuss related work in section 4.

We propose, RARE (Riverside's Augmented Router Emulator), a systematic approach to analyze router malware and understand its behavior in depth. The key novelty is the augmented operation of our approach, which fools malware binaries to activate irrespective of their preferred target platform. In other words, instead of trying to guess the right router platform for each malware, we start with a generic one and we carefully and iteratively "adapt" it to fool the malware. This is achieved by leveraging two key capabilities: (a) a static level analysis that informs the dynamic execution, and (b) an iterative feedback loop across a series of dynamic runs, the output of which informs the subsequent run. We provide an overview of our approach in Figure 2.

From a practical point of view, RARE has the ability to run the malware on an emulated router and consists of the following modules, whose goal is to: a) extract information for malware execution by analyzing the binary statically, b) create an emulated router enhanced with the appropriate configurations that a malware needs for its execution, c) inject malware into the router and fool it to activate, d) replay pre-recorded network traffic with crafted C&C responses to malware requests, e) enhance the emulation using information derived from the previous runs. This process works in an iterative fashion to enhance the emulation using information from previous runs in order to have the malware to activate itself.

We evaluate our approach using 221 router-specific malware binaries from a community-based project, which requested to be anonymous. Our results can be summarized in the following points.

a. Achieving 94% malware activation success ratio, and 88.8% for obfuscated malware. We show that our system is successful in fooling malware to activate, as **94%** of our binaries become active. We say that a malware binary activates, when the malware attempts to communicate with the C&C server. By contrast, an emulation without any of our augmented functions, which we refer to as **baseline**, can activate only 10% of the binaries. Furthermore, our approach manages to activate 88.8% of our obfuscated binaries, which are the types of binaries that are especially crafted to outplay static analysis techniques. We show the iterative operation of our approach in Figure 1, where we plot the number of system calls for each run. Note that the malware is activated within a sandboxed environment and thus does not pose any threat to real devices

b. Extracting useful information: IP addresses, and domains. Having this powerful capability, we are able to extract useful information for the malware. We find **203 malicious IP addresses and domains**, and we naturally consider these addresses to be malicious, since they are or have been used for botnet communications. Note that using just static analysis, we find less than 25% of these C&C communication addresses and domain names.

c. Developing infected router profiles. We identify features for detecting infected routers. Specifically, we observe an initial spike and a 50% increase in the number of system calls in infected routers. We also observe an initial large spike and a subsequent slight increase in the number of active processes.

Our emulation capability is a powerful building block towards understanding router malware. As a preview of the capabilities provided by RARE, we discuss how we assumed the botmaster role for two malware binaries at the end of section 3. Finally, we intend to make our tool, the malware binaries, and the data traces available upon request to researchers.

2 System Design and Implementation

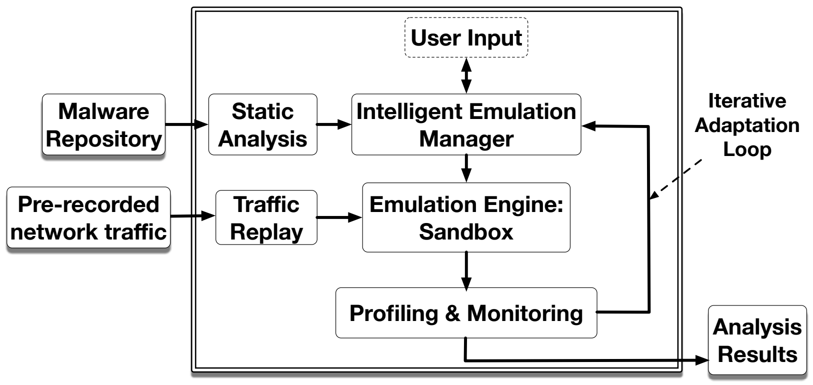
User Input
Malware Repository
Static Analysis
Intelligent Emulation Manager
Iterative Adaptation Loop
Pre-recorded network traffic
Traffic Replay
Emulation Engine: Sandbox
Profiling & Monitoring
Analysis Results

Fig. 2: Overview of the key components of our approach. A key novelty is the feedback loop that enables previous runs of the malware to inform the subsequent run in order to fool the malware to activate.

We present an overview and highlight the novel capabilities of our approach.

Philosophy. We adopt the following approach: we start with a general purpose platform and learn what the malware needs to activate through a sequence of executions.

A visual depiction of RARE is shown in Figure 2. The key capabilities of the system are listed below: a) it can perform static analysis on the malware to extract information for its execution, b) it can instantiate an emulated router with hints on what configurations the malware wants to "see", c) it can replay arbitrary network traffic and response to malware requests, and d) it can monitor the malware and provide information to subsequent runs of the same malware. If the malware fails to activate, we repeat the process, and the last step provides information that guides the new execution.

Defining success: malware activation. We set as our goal for the emulation the activation of malware: we want the malware to feel "comfortable" within the emulator, so as to attempt to contact its bot-master. After reaching this point, we enter a new stage: if the C&C server responds, the malware will enter a stand by mode waiting for C&C commands. We define this stage as "activated" stage of the malware. A subsequent goal is to turn one's self into the bot-master, by reverse engineering the communication protocol with the bot, which we achieve for some of our binaries (see the discussion in section 3).

We present the key functionality, novelty and challenges of each module.

a. Static Analysis Module. The first step is to analyze a given malware binary statically to extract as much information as possible. The purpose is dual: (a) we want to inform the dynamic execution, and (b) we want to understand as much about the malware, which could have independent interest. Specifically, this module provides the following information to the dynamic execution: a) IP addresses and domains that malware will possibly contact, and b) files and resources that the malware will attempt to access. This information is collected by the Intelligent Execution module and is provided to the Emulation Engine module. In more detail, we currently use IDA-Pro [18] for the static analysis, but one could envision using other tools with similar capabilities. In addition, we developed an initial capability to extract the high level structure of the code by lifting the binaries to Intermediate Representation [26], which could provide data dependencies. The IDA-Pro tool is an excellent foundation, but so some of the desired functionality was missing. Therefore, we developed several non-trivial plug-ins, specifically for MIPS and ARM architectures, such as extensions to extract communication tokens, and control flow information which are the main components that assists this module. Naturally, we will make our plugins available to the community.

b. Emulation Engine Module. This module provides the basic capabilities for emulating the router with the provided augmenting information and records execution traces. This module can: (a) instantiate an emulated router, (b) inject malware into the router, (c) replay pre-recorded network traffic or crafted C&C responses for the malware requests, (d) receive commands and information from the Intelligent Emulation malware in order to convince the malware to activate. The emulation is hosted in an Ubuntu server and `QEMU` [5] (an open source and widely used tool) is used to emulate the router hardware of interest, which here is `ARM` and `MIPS`.

We have made significant extensions and engineering in order to enable all the functionality. For example, we modify QEMU to recognize two subnets to represent the enterprise network and the rest of the Internet. We also added the ability to interact with the router through the network interfaces connecting to subnets, which is further discussed in the Traffic Replay module. For the firmware of the instatiated router, we use `OpenWrt` [1], which is fairly widely used codebase and it is considered as a reference firmware for routers. OpenWrt has the essential basic modules of a home router such as `DHCP` server, packet forwarding and routing, and a web interface. We added the monitoring capabilities to the instantiated router to collect execution traces for the router for further behavioral analysis, which we discuss in the Profiling and Monitoring module.

c. Intelligent Emulation Manager. The key novelty of RARE is represented by the Intelligent Emulation Manager module. This module drives the emulation in a way that achieves malware activation. First, it uses information from static and dynamic analysis in the previous two modules. The intelligence of the emulation is based on a feedback loop that gets execution information from each run. Every run is a clean start of the router augmented with information about the malwares requirement to fully execute itself. This loop represents the learning process of our approach. Second, the module facilitates the manual interaction with the malware, such as crafting C&C responses.

d. Traffic Replay Module. To observe a router in its natural element, we developed the Traffic Replay module, for replaying arbitrary network traffic to the router. Our module can get a real trace of two directions, incoming and outgoing, and replay it through the emulated router. This module is built on top of `tcpreplay` [3], but significantly we added new functionality. For example, the concept of incoming and outgoing traffic needed significant engineering, as it was not fully provided by tcpreplay. We also needed to take care of implementation issues, such as timing the replay traffic through the router. The network traffic we use is real traffic data from project *MAWI* [7], ensuring that we capture the beginning of each TCP flow across different days and different hours. Another key feature in this module is its ability to inject traffic at the command of the Intelligent Emulation Manager. This allows us to impersonate the C&C server by providing server responses and commands. In other words, our approach combines the knowledge of who the malware attempts to talk to, and what the malware expects to hear, through the deep profiling in the Static Analysis and Profiling and Monitoring modules, and we expect to be able to fully explore the intention and capability of the malware.

e. Profiling and Monitoring Module. This module synthesizes information from the static analysis and dynamic executions with the following goals. We want to: (a) understand the malware in order to create an environment that will make it to activate, and (b) profile the behavior of the infected router in order to distinguish it from that of a benign one. We list the types of data that this module collects and analyzes.

– Network Traffic: The module collects the network traces for both router interfaces (think incoming and outgoing) and analyzes them. The goal is to observe packets and flows generated by the malware, so that we can respond to them as if we are the C&C server. We can determine if a packet is generated by the malware by comparing the log files of the execution with and without the malware.
– OS System Calls: The module also collects the system calls at the operating system level. The goal is to extract information for augmenting the next iteration of the emulation. For example, the malware usually checks for existence of different files in the system as a means to infer which platform it is operating on. Another family of malware uses the */proc/sysinfo* file to infer the CPU architecture, while in another family they use `banner` file.

– System Processes: The module also monitors the processes in the router, which provides a complementary view on the behavior and intentions of the malware. For example, several malware binaries kill: (a) processes in the router in order to free up resources for themselves, (b) security processes, such as the `iptable` firewall process, and (c) user access processes, such as the `http server` process.

This module initiates the feedback-loop by providing the information to the Intelligent Emulation Manager to ensure that it can "fool" the malware to reveal itself and even believe that it is communicating with its C&C server.

Finally, this high-level description of our approach can be seen as a blueprint for a router malware analysis tool. Although in RARE, we have made specific implementation decisions for each module, functions and methodologies can be modified and replaced easily due to its modular design.

3 Evaluation

We evaluate our method to assess the effectiveness of RARE in activating the malware and profiling the behavior of malware.

The malware binaries. In our evaluation, we use malware binaries that were collected from a community-based project, that requested to stay anonymous. These sources use honeypot and manual effort to collect these malware binaries. We have a total of 221 unique malware binaries targeting MIPS Big Endian (BE), MIPS Little Endian (LE), and ARM architectures. We focus on these architectures given that they are the most common ones in routers.

Network traffic. A key feature of our emulation is that we can do experiments using arbitrary network traffic. We report results with real network traffic from `MAWI` [8] with a duration of 1300 seconds. Note that we have experimented with other data traces, and also observed the router with no traffic at all. We obtained qualitative similar results, which we do not report here due to space limitations.

The baseline approach. We define the baseline approach, as an emulation using the RARE infrastructure (OpenWrt over QEMU) but without any of the intelligence of our approach. We inject the malware in such an instance of an emulated router, and we monitor its behavior as the traffic passes through.

Evaluation. For each malware and data trace, we compare the following approaches in executions with and without the malware: (a) Baseline run: which is the approach described above. (b) RARE run: The emulation gets augmented with the information extracted from the malware in each run. We define each execution as k^{th} run, being 1^{st} emulation informed by *Static Analysis* module and later runs informed by the behavior observed in the previous run.

Defining failure: no malware activation by the 6^{th} run. Setting a high standard for our approach, we require that we achieve malware activation by the sixth run. If this does not happen, we consider this a failure.

A. Evaluating RARE: 94% malware activation success. Using our tool we have been able to reach activation for 208 malware binaries out of 221. Table 1

Table 1: The success ratio of our solution RARE compared to a baseline method.

		Baseline		RARE		
	Binaries	Raw No.	Percentage	Raw No.	Percentage	Avg Runs
All	221	**23**	**10.4%**	**208**	**94.1%**	3
Obfuscated	45	**5**	**11.1%**	**40**	**88.8%**	3
ARM	101	11	10.9%	91	90.1%	3
MIPS BE	77	6	7.8%	75	97.4%	2
MIPS LE	43	6	14%	42	97.6%	3

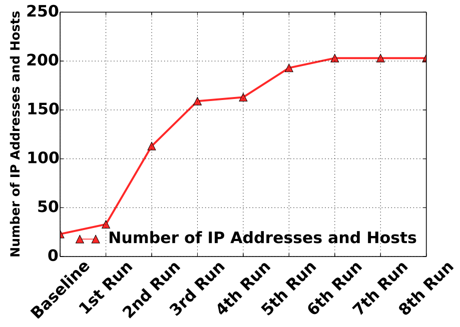

Fig. 3: The number of extracted IP addresses and hosts rises with every run.

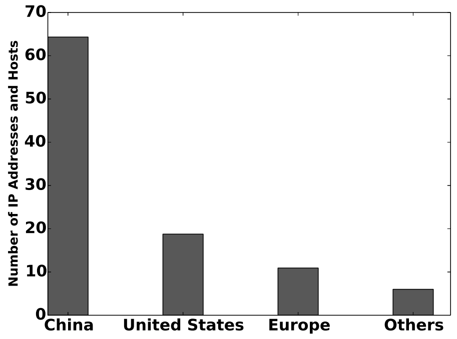

Fig. 4: Distribution of the geolocation of the detected IP addresses.

details this success by comparing RARE with baseline approach. We report the average number of runs that RARE needed in order to reach activation for the binaries in that group that reach activation. We see that RARE requires a relatively low number of repetitions on average, namely three. We also separate the malware based on the target architecture: ARM, MIPS BE, and MIPS LE.

RARE exhibits great performance even for obfuscated malware. We compare the two approaches on 45 obfuscated malware binaries, as, for these binaries, static analysis cannot be used. Although there is a drop, our approach maintains a success ratio of 88.8%.

What does the malware need to activate? We observe an interesting and systematic progression in the types of requests that the malware generates. In the first few runs, the malware binaries typically attempt to change or check the existence of files, such as configuration files such as `/etc/ISP_name` or web interface files from *Linksys* and `TP-Link` routers, which are found specifically in home routers routers or specific brands of routers. Subsequently, some binaries attempt to tamper with different routing services, such as `libnss` and DNS, or change the routing table to subvert traffic. Typically, in the last run before the activation stage, the malware tries to resolve the hostname to locate its C&C server. In response, we inject DNS lookup responses using our Traffic Replay module. Note that, in our system, we retrieve the request that led to the failure using the Profiling and Monitoring module, we prepare for the subsequent run accordingly. As an example, when running one of the binaries an HTTP request to `userRpm/SoftwareUpgradeRpm.htm` (part of the `TP-Link` web interface) is observed which tampers with the installed firmware on the router. Failure to respond to this request will interrupt malware's execution. Using the Intelligent Emulation Manager module the appropriate web interface is installed for the next clean run of the router emulation.

Why do some malware fail to activate? This is an open question, which we will continue to investigate in our future work. In many of the malware requests listed above, we are able to provide a fake answer or a fake file. In most of the cases where we fail to do so, the malware tries to dynamically link to a custom library, which are not publicly available. Furthermore, these malware binaries are usually obfuscated, so faking the libraries is not straightforward.

B. Extracting useful information: 203 botnet IP addresses/hosts. We highlight initial elements of the information that we can extract with our approach. Overall, we find that RARE finds 203 malicious entities, IP addresses and host names, which are used used by the malware for botnet communications.

Static analysis: 48 addresses and hosts. Using static analysis, we traversed the CFG for all the paths from binary's *Entry Point* to system call `connect` and traced the input arguments values. We were able to extract 22 unique IP addresses and 26 host names.

RARE dynamic analysis: 155 addresses and hosts. Using RARE's consecutive runs, we extracted an additional 155 IP addresses and hosts. As shown in Figure 3 with every run the number of extracted IP addresses and hosts rises until the 6^{th} run after which no new IP or hosts are detected.

The dynamic analysis finds 75% of the malicious entities. The corollary of the observations above is that the dynamic analysis is essential in detecting malicious entities of botnet communication. Using just static analysis, we find less than 25% of these C&C communication addresses and domain names. The issue is that the malware often obfuscates the addresses and the domain names, by using hexadecimal numbers and using number transformation and encryption techniques. For example, one binary has the following address generation approach: a hardcoded hexadecimal base value, and a function that adds decimals values to this base to obtain a series of IP addresses.

What is the geographical distribution of these malicious Internet entities? In Figure 4, we do a reverse look up to identify the geolocation of the IP addresses, and we observe that China (64.3%) and United States (18.7%) are the top destinations for hosting C&C servers. Many times these botnet entities could be compromised machines. In figure 3, we plot the number of malicious IP addresses extracted at each run of RARE. Initially, more information is extracted with each run, but this stops by the sixth or seventh execution for most binaries. In future work we intend to study the uniqueness and timeliness of the IP addresses that we find compare to well known blacklists.

C. Profiling infected router behavior. Our approach gives us the ability to compare the behavior of non-infected and infected router with a rich set of information at both the network and OS layers.

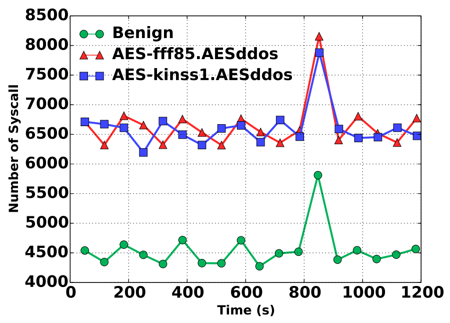

Fig. 5: The number of system calls of a benign and two infected routers for the last 1200 seconds of the execution. The infected routers have consistently 1.5 times more system calls.

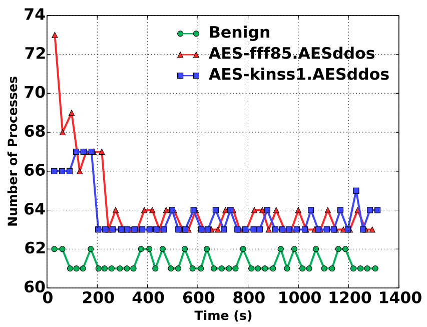

Fig. 6: The number of active processes of a benign and two infected routers for our 1300 sec experiment. Initially, the malware spawns child processes to make itself persistent.

In all malware, we observe an increase in the number of system calls of almost 50% or more in an infected router compared to a benign one. Figure 1 shows a comparison of the number of system calls between infected router using the baseline, and the different RARE executions (numbers 1 to 8). This particular malware binary, reaches the activation stage at the third run. On the fourth run, we impersonate the C&C and we start issuing commands to the malware.

To better understand the malware, we show the number of system calls over time for two infected routers compared to a benign one in Figure 5. For visual clarity, we show only 1200 seconds of the execution to avoid the huge initial spike, which corresponds typically to the reconnaissance of each malware. However, we do show the initial spike in the number of processes of an infected router in Figure 6: the malware makes itself persistent by spawning child processes.

Discussion: Becoming the botmaster. Using RARE we identify the functions available on two of the malware binaries from MIPS LE and MIPS BE. This was achieved by combining the static analysis and profiling information after the execution. We were able to convince the malware that we are the botmaster, and we were able to have it do: 1) HTTP flooding, 2) reverse shell, and 3) kill processes based on their process ID.

4 Related Work

We briefly review related work due to space limitations.

Emulation techniques: Several emulation techniques and tools exist, but they mostly focus on PC and Android platforms: `Anubis` [4], `PANDA` [12], `DECAF` [17]. The approach is to simulate the target platform and apply monitoring tools to record the execution traces of the malware. PC and Android platforms and malware differ significantly from router-specific ones, which is our focus here.

Vulnerability detection in embedded systems: Several recent studies focus on detecting vulnerabilities in the firmware of embedded devices [10, 6].

Chen et al. [6] argue that emulating platforms for specific firmware is not a trivial task since it includes emulating hardware components designed by vendors who do not necessarily practice a global Hardware/Software design standard. Other approaches [28] use real hardware to overcome this difficulty, but introduce the high cost and overhead. In our case, with thousands of router configurations, this approach would be very expensive and time consuming.

PC and smartphone malware studies: Many studies propose malware analysis tools using static or dynamic analysis. In static analysis, several studies focus on Control Flow Graphs characteristics [9, 19]. Static analysis on binary code requires platform specific tools, so PC-based or smartphone bases tools do not work for ARM and MIPs platforms. A limitation of the static analysis is that it does not work for obfuscated malware [23]. Several studies use dynamic analysis to classify and distinguish between different families of malware by studying the operation at the OS level [20, 22, 15]. In the PC and smartphone space, getting the malware to activate is an easier task given the more limited diversity in these platforms.

5 Conclusion

We propose, RARE, a comprehensive approach to analyze router malware. The key novelty is the augmented operation of our approach: instead of trying to guess the right router platform for each malware, we start with a generic one and we iteratively "adapt" it to fool the malware.

Our system provide the following key capabilities: a) perform static analysis on the malware, b) instantiate an emulated router, c) inject malware into the router and fool it to activate, d) replay arbitrary network traffic, and e) profile the malware behavior. Using real router malware, we are able to show that: (a) our system works effectively and manages to activate 94% of all our binaries, and (b) we can extract useful and insightful information from the execution of the malware. First, we find that we can identify malicious IP addresses and domain names, which subsequently could be investigated and blocked in firewall filters. Second, we identify tell-tale signs of an infected router operation, such as 50% increase of the system calls.

Our approach is a solid first step towards developing a key capability for an under-served segment of devices. Although the results are already promising, we plan on expanding the capabilities significantly in two different dimensions. First, we will develop a more extensive static analysis capability, where we could infer the structure and key operations of the malware code. Second, we will further explore how to fully interact, and ultimately control both a bot, but ultimately a C&C server.

References

1. Openwrt embedded devices linux. `https://openwrt.org/`, accessed: 2017-09-22

2. Antonakakis, M., et al.: Understanding the mirai botnet. In: 26th USENIX Security Symposium (USENIX Security 17) (2017)
3. Appneta: Tcpreplay (2016), `http://tcpreplay.appneta.com/`
4. Bayer, U., et al.: Dynamic analysis of malicious code. Journal in Computer Virology 2(1), 67–77 (2006)
5. Bellard, F.: Qemu, a fast and portable dynamic translator. In: USENIX, FREENIX Track (2005)
6. Chen, D.D., Woo, M., Brumley, D., Egele, M.: Towards automated dynamic analysis for linux-based embedded firmware. In: NDSS (2016)
7. Cho, K., Mitsuya, K., Kato, A.: Traffic data repository maintained by the mawi working group of the wide project. URL http://mawi. wide. ad. jp/mawi (2005)
8. Cho, K., et al.: Traffic data repository at the wide project. In: USENIX, ATEC
9. Christodorescu, M., Jha, S.: Static analysis of executables to detect malicious patterns. Tech. rep., Wisconsin Univ-Madison Dept of Computer Sciences (2006)
10. Costin, A., et al.: Automated dynamic firmware analysis at scale: a case study on embedded web interfaces. In: Asia CCS (2016)
11. Costin, A., Zaddach, J., Francillon, A., Balzarotti, D., Antipolis, S.: A large-scale analysis of the security of embedded firmwares. In: USENIX Security (2014)
12. Dolan-Gavitt, B., et al.: Tappan zee (north) bridge: mining memory accesses for introspection. In: ACM SIGSAC CCS. ACM (2013)
13. Enck, W., et al.: Taintdroid: an information-flow tracking system for realtime privacy monitoring on smartphones. ACM TOCS (2014)
14. Feng, Q., et al.: Scalable graph-based bug search for firmware images. In: ACM SIGSAC CCS (2016)
15. Gasparis, I., Qian, Z., Song, C., Krishnamurthy, S.V.: Detecting android root exploits by learning from root providers. In: USENIX Security 2017)
16. Hampton, N., et al.: A survey and method for analysing soho router firmware currency. Australian Information Security Management Conference (2015)
17. Henderson, A., et al.: Make it work, make it right, make it fast: Building a platform-neutral whole-system dynamic binary analysis platform. In: ACM STA (2014)
18. Hex-Rays, S.: Ida pro disassembler (2008)
19. Kinable, J., Kostakis, O.: Malware classification based on call graph clustering. Journal in computer virology (2011)
20. Kolbitsch, C., et al.: Effective and efficient malware detection at the end host. In: USENIX security symposium (2009)
21. Kolbitsch, C., Holz, T., Kruegel, C., Kirda, E.: Inspector gadget: Automated extraction of proprietary gadgets from malware binaries. In: IEEE S&P (2010)
22. Lanzi, A., et al.: Accessminer: using system-centric models for malware protection. In: ACM CCS (2010)
23. Moser, A., et al.: Limits of static analysis for malware detection. In: IEEE ACSAC
24. Papp, D., et al.: Embedded systems security: Threats, vulnerabilities, and attack taxonomy. In: IEEE PST (2015)
25. Paquet-Clouston, M., et al.: Can we trust social media data?: Social network manipulation by an iot botnet. In: ACM Conference on Social Media & Society (2017)
26. Song, D., et al.: Bitblaze: A new approach to computer security via binary analysis. Springer Information systems security (2008)
27. Tam, K., et al.: Copperdroid: Automatic reconstruction of android malware behaviors. In: NDSS (2015)
28. Zaddach, J., Bruno, L., Francillon, A., Balzarotti, D.: Avatar: A framework to support dynamic security analysis of embedded systems' firmwares. In: NDSS (2014)