# ELF Analyzer Demo: Online Identification for IoT Malwares with Multiple Hardware Architectures

Shin-Ming Cheng*†, Tao Ban‡, Jr-Wei Huang*, Bing-Kai Hong*, and Daisuke Inoue‡
*Department of Computer Science and Information Engineering,
National Taiwan University of Science and Technology, Taipei, Taiwan
†Research Center for Information Technology Innovation, Academia Sinica, Taipei, Taiwan
‡Cybersecurity Laboratory, National Institute of Information and Communications Technology, Tokyo, Japan
Email:{smcheng, m10815007, d10815003}@mail.ntust.edu.tw, {bantao, dai}@nict.go.jp

***Abstract*—This demonstration presents an automatic IoT runtime platform with a web interface, ELF Analyzer, where suspicious ELF files uploaded by users could be executed and dynamically analyzed for malicious behavior identification. The key component of our platform is a crafted IoT sandbox, where multiple hardware architectures are emulated using QEMU. With the introduction of strace functionality, we demonstrate that system call and traffic logs of an uploaded ELF file with different hardware architectures can be generated successfully. After proper analysis, malicious ELF files can be identified.**

***Index Terms*—architecture emulation, dynamic analysis, IoT malware, QEMU, strace**

## I. Introduction

In order to understand the behavior of tremendous kinds of IoT malwares, static code analysis or dynamic runtime analysis are necessary. However, with multiple architectures, such as ARM, MIPS, X86, PPC, or SPARC, the execution of malware binaries in different hardware architectures are tough, thereby making dynamic analysis challenging [1]. We developed an automatic IoT runtime platform with a web interface, ELF Analyzer, where different hardware architectures are emulated using QEMU so that an ELF file can be executed in a customized virtualized system. By including `strace` functionality, our platform could retrieve system call as well as network traffic logs of the examined ELF file, and consequently malicious one can be identified.

## II. System Description of ELF Analyzer

As shown in Fig. 1, various kinds of IoT malwares downloaded from malware databases (such as Virustotal) are exploited as the training dataset for malicious behavior identification. Users could upload any suspicious ELF file to the platform via the web interface to verify its legality. The parser located in the platform scans the file header to determine the corresponding hardware architecture.

The core part of the platform is a crafted sandbox, where eight main IoT hardware architectures, including ARM, MIPS, X86, X86-64, PPC, SPARC, SH4, and m68k, are emulated using QEMU. According to the information in the file header, the file is delivered to the virtualized system with corresponding architecture. Then we build a `strace`-enabled OS using OpenWRT or lightweight Buildroot on the system and execute the file to get system call logs and network traffic pcap files.

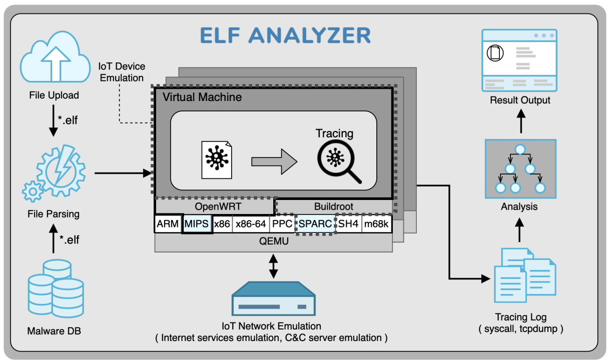
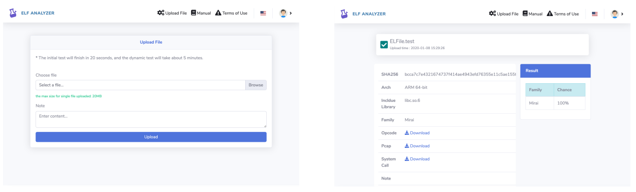


Fig. 1. Operation flow of ELF Analyzer



* The initial test will finish in 20 seconds, and the dynamic test will take about 5 minutes.

Choose file: Select a file...

the max size for single file uploaded: 20MB

Note: Enter content...

ELFile.test
Upload time : 2020-01-08 15:29:26

| | |
|---|---|
| SHA256 | bcca7c7e4321674737f414ae4943efd76355e11c5ae155 |
| Arch | ARM 64-bit |
| Incldue Library | libc.so.6 |
| Family | Mirai |
| Opcode | Download |
| Pcap | Download |
| System Call | Download |
| Note | |

Result

| Family | Chance |
|---|---|
| Mirai | 100% |

(a) ELF file uploading interface (b) Identification result

Fig. 2. ELF Analyzer Intefaces

For example, the solid and dot lines respectively represent virtualized systems with MIPS architecture and OpenWRT OS as well as SPARC architecture and Buildroot OS. To ensure that the file could be executed and communicated with outside servers normally, we simulate various Internet services using `Inetsim` and build a Mirai-based C&C server. By using a simple classifier such as KNN, our platform returns identification results to the users via the web page or an email.

## III. Demonstration

Fig. 2 shows the web interfaces of ELF Analyzer where user could upload ELF files for testing and get the identification results.

## References

[1] Y. Minn, P. Pa, S. Suzuki, K. Yoshioka, T. Matsumoto, T. Kasama, and C. Rossow, "IoTPOT: Analysing the rise of IoT compromises," in *Proc. USENIX Workshop 2015*, Aug. 2015.

DOI 10.1109/SPW50608.2020.00036