**IJDI-ERET**

**INTERNATIONAL JOURNAL OF DARSHAN INSTITUTE ON ENGINEERING RESEARCH AND EMERGING TECHNOLOGIES**
Vol. 10, No. 2, 2021
www.ijdieret.in

(Review Article)

# Diving into Linux Malware

**Vikas Sharma[1*]**

[1*]*Department of Computer Science and Engineering, Ganpat University, Ahmedabad, Gujarat, INDIA*

***Abstract***

*The security community all over the world has been confronting malicious programs for Windows- Based operating systems for the last few decades. However, as technological expansion took place, the adoption of embedded devices and IoT has exponentially increased leading to rapidly changing the malware landscape. Embedded devices are quite different than personal computers as they run on different architecture whereas personal computers are still dominating on x86 or x64 flavored architectures. The surge in the utilization of Linux or its variant has forced malicious actors to introduce "Linux Malwares". There is presently no systematic study attempting to classify, evaluate, and understand Linux malware that we are aware of. The majority of resources on the topic are sparse reports often published as blog posts, while the few systematic studies focused on the analysis of specific malware families (e.g., the Mirai botnet) primarily through network-level behavior, leaving the main challenges of analyzing Linux malware unaddressed.*

*Keywords: Linux Malware, Malicious programs, Malware Analysis, Security*

## 1. Introduction

For more than two decades, the security community has been fighting malware. Despite the fact that both academic and industrial communities have put a lot of effort into this subject, automated analysis and detection of malicious software is still a work in progress. Because of its massive market share, the great majority of malware has historically been designed to target nearly solely personal computers running Microsoft's Windows operating system (currently estimated at 83 percent [1] for desktop computers).

The malware environment is rapidly changing as a result of the recent exponential surge in the popularity of embedded devices. Embedded devices have long been used in industrial settings, but it is only recently that they have begun to pervade every area of our society, mostly (but not exclusively) as a result of the so-called "Internet of Things" (IoT) revolution. Companies that make these gadgets are constantly competing for market share, focusing primarily on quick time-to-market and creative features to attract new users. This frequently leads to the postponement (if not outright dismissal) of any security or privacy concerns. With these assumptions in mind, it's no surprise that the vast majority of these newly interconnected devices are routinely discovered to be vulnerable to critical security issues, ranging from Internet-facing insecure logins (e.g., easy-to-guess hardcoded passwords, exposed telnet services, or accessible debug interfaces) to unsafe default configurations and unpatched software containing well-known vulnerabilities [1].

Traditional personal computers are vastly different from embedded systems. While personal computers mostly use x86 and x64 architectures, embedded devices use a number of alternative CPU architectures—and are frequently constructed on hardware with restricted resources. Developers frequently use Unix-like operating systems to support these new technologies, with various varieties of Linux swiftly gaining favor in this field.

## 2. Challenges

A number of challenges and obstacles were encountered while analyzing the dangerous and generic Linux apps/programs.

The further part examines the issues faced while understanding the Linux malware.

*2.1. Target Diversity:* The foremost issue is that it has a wide range of target scenarios. The conventional inclination to diversity is because of various supporting architectures such as ARM or MIPS, yet this is entirely a part of an entirely larger problem. In the case of other operating systems, the malware analysis system can utilize the benefit of information about the execution environment.

[*]*Corresponding Author: e-mail: vikassharma.dpseast@gmail.com, Tel-+91 98791 13429*
ISSN 2320-7590

Since the Linux OS is used in a wide range of devices, thus Linux-based malware may attack different devices and act differently with different scenarios.

Without the proper execution environment for the analysis of Linux-based malware, it becomes quite difficult than any other analysis.

*2.1.1. Computer Architecture:* Linux-based OS are utterly compatible with a plethora of architectures. Thus different architecture has different execution environments making it difficult to create a generic-analysis platform for the analysis of Linux Based Malware.

*2.1.2. Libraries and Loaders:* ELF file format is commonly used to define an arbitrary loader in a linux application, which is accountable for loading and preparing the executable in the memory. In the analysis environment, the same copy of the loader can not be present preventing the sample malware from executing. In addition to the same, in case of dynamically linked libraries will require every single of the libraries to be present in the analysis environment, if not so will prevent it from running. Not only do these alternative (libraries) need to be present, but so do the loaders that are compatible with them.

*2.1.3. Operating System:* Along with the different architectures, it also has the wide range of the OS such as Linux, Android, Debain , FreeBSD. It becomes difficult to biffargate ELF codes created for Linux from codes which are compiled for the other ELF compatible operating system. The syscalls , arguments, execution environments and other pivotal factors vary from system to system when we go for Linux Based Malware

*2.2. Static Linking:* While in the compilation process, all the library dependencies are attached to the resulting binary when a binary is statically linked. It offers various merits including the portability of the resulting binary(as all the binary's library dependencies will be attached, will execute in the target environment even if doesn't contain any dependencies), increasing the difficulty of reverse analysis (since it is difficult to map the library used by the binary)

Moreover, due to the portability of binary, the resulting application does not rely on any external wrapper to execute the system calls. Normal applications invoke higher levels of API functions of the operating system which wrap the communication with the kernel. Since communication between the application and kernel (syscalls) is not direct, it may crash if the API and kernel are not compatible with the operating system

*2.3. Analysis Environment:* The hypothetical execution environment along with the analysis sandbox should work as closely as possible the system in which the malware was tended to execute. We have already discussed the challenges of getting a perfect analysis environment with the required architectures, libraries, loaders, and the last operation system for the malware analysis. Additionally, the privileges play an important role in the analysis environment to measure the impact of the malware under full administrative rights and as normal users. Linux malwares are generally written in manner with assumption that it has full administration rights(root) while executing the malware or the sample

*2.4. Lack of information:* To the best of my knowledge, there does not exist much detailed work on the Linux Malware. Hardly, one can get few of the resources for better understanding of linux malware. Then comes the language used in the paper, either it gets too technical or too laymen.

*2.5. Off the Radar:* It is very common for linux malware (along with the sophisticated malwares)to remain fully undetected by security vendors. One of the examples can be Mirai - the one which we have discussed in the introduction. It is a DDOS botnet whose source code got leaked. Now we have different variants on the basis of the working of Mirai's code

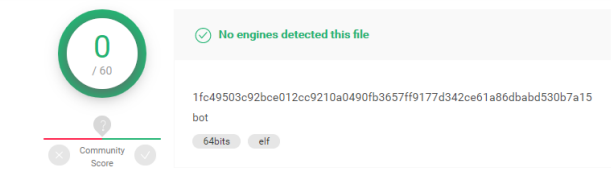


**Figure 1.** Static Analysis of Mirai's code

The sample was tested by one of the popular security vendor-Virus Total back then and was diagnosed with zero detection.

## 3. Threat Landscape

Use of the linux in the technological world is rising at an exponential rate, so does the threat landscape for the operating system. Traditionally, the malwares were different but in the present scenario, the threat landscape for Linux is enormously concentrated either with miners (crypto-mining) or DDoS botnets.

Following are some of the illustrations of Linux-based malware threats:

- Cloud Snopper - RAT (Remote Access Trojan) founded by Sophos' researchers
- Evil Gnome - Backdoor with connection to Russia's Gamaredon Group.

- MessageTAP - FireEye discovered an information stealer on Linux servers

These are a few listed from much-existing malware. The larger the platform for the malware, the more the number of malware. In fact, apart from the Linux desktop and servers, there exist innumerable malwares of Android which is also part of the Linux operating system.

## 4. Working (Infecting the System)

To infect a system, traditional methods like phishing won't work in the case of Linux because of zero interaction between browsers and email accounts, especially for a non-desktop Linux system, there are a few of the attack vectors are:

*4.1 Vulnerability Exploit:* Attackers will detect vulnerability and unpatched publicly faced components in order to gain access to the system. Misconfiguration can also be an entry point for attackers for infecting the system.

For instance, The Asnarok trojan was initiated and exploited a zero-day (SQLi RCE i.e is SQL Injection Remote Code Execution) in Sophos XG Firewalls. Kinsing malware was expanded after the attackers took advantage of misconfigured open Docker Daemon API ports [2].

*4.2 Use of valid credentials:* Default software credentials or credentials that have been compromised. Passwords can be stolen via a variety of methods, including password spraying, local discovery and credential stuffing. Researchers suspect the Cloud Snooper infection was started by an attacker who gained access to the servers using SSH, which is password-protected [2].

*4.3 Trusted relationships abuse:* Attackers can manage to get entry with the help of 3rd Party software which has direct access to the victim's system. Exploit in the software may lead risk of exploiting the entire system. For example, vulnerable software like Browsers, PDF Reader may cause risk to the victim system as it can be easily exploited.

## 5. Solutions

Due to its low prevalence, anti-malware solutions for Linux Malware are scarce. A few of the options are listed:

- Use of Linux Based AV and Malware Detection Engines
- Use of scripts such as Rootkit Hunter, chkrootkit to detect rootkits and other local exploits
- Control user access and implement a Zero Trust Security Strategy
- Configured Firewall, monitoring inbound and outbound traffic, if any anomaly detected.

## 6. Conclusions

Since Linux is so widely used, the threat is real and growing. Two of many APT organizations, Winnti and Lazarus, have recently been revealed as employing ELF in their malware toolkits.

There are many unrevealed Linux threats waiting to be discovered, due to the poor detection mechanisms of security vendors, absence of ELF malware visibility, and a scarcity of resources available publically [2].

Thus, the paper presents the practical concepts and challenges behind the Linux malware. We have discussed the empirical study on how Linux malwares implements its malicious behavior, how the complexity of malware is defined, the way Linux malwares can evade the AVs or Threat Intelligence softwares.

## References

1. Cozzi, Emanuele, et al. "Understanding linux malware." 2018 IEEE symposium on security and privacy (SP). IEEE, 2018.
2. ELF Malware Analysis 101: Linux Threats No Longer an Afterthought, 2021 - https://www.intezer.com/blog/malware-analysis/elf-malware-analysis-101-linux-threats-no-longer-an-afterthought
3. Carrillo-Mondéjar, J., José Luis Martínez, and Guillermo Suarez-Tangil. "Characterizing Linux-based malware: Findings and recent trends." Future Generation Computer Systems 110 (2020): 267-281
4. Malin, Cameron H., Eoghan Casey, and James M. Aquilina. Malware forensics field guide for Linux systems: digital forensics field guides. Newnes, 2013.

## Biographical notes

**Vikas Sharma** is pursuing B. Tech in Computer Science and Engineering with a specialization in Cyber Security from Ganpat University. He is CEH (Certified Ethical Hacker) by EC-Council. His research interest includes Cyber Forensics, Malware Analysis, Network Security, and System Engineering.