TALLINN UNIVERSITY OF TECHNOLOGY

# TED - The ELF Doctor

A container based tool to perform security risk assessment for ELF binaries


## D. Mucci, B. Blumbergs


Feb 23, 2019

# Introduction

# Abstract

**Attacks against binaries** are still very common.
This paper presents TED, an **auditing tool** which acts from the defense perspective and verifies whether proper defenses are in place for the GNU/Linux system and for each ELF binary in it.
TED aims to integrate several tools and techniques by the use of **software containers**.
The containerization approach allows to reduce complexity and gain flexibility at the cost of a negligible performance loss, while significantly reducing the dependencies needed.

# Outline

- [+] Motivation and problem statement
- [+] Methodology used
- [+] Project, Contributions and Results
- [+] Conclusion and Future work

# Motivation and Problem statement

**Why TED?**

- [+] (stack|heap|buffer) overflows still common.
- [+] New hardware vulnerabilities attacks
- [+] The majority of the servers run on Linux
- [+] Compliance

How do we protect from such attacks?
Can containerization help us?
What about cloud environments?

**Who is TED for?**

- [+] System administrators. Proactive approach.
- [+] CSIRTs. Proactive and Analytical approach.
- [+] Forensic investigators. Analytical approach.

# Methodology

To carry on this research

- [+] Investigate and select container platforms
- [+] Investigate and select defense measures
- [+] Build scoring system
- [+] Implement

# Results

# Background

**The attacks**

- [+] Stack, heap overflow
- [+] Format string bugs
- [+] Reverse engineering
- [+] Hardware attacks (Spectre, Meltdown)

Many more attacks: ROP, ret2*, (Row|Net|Throw)hammer, etc.

**The defenses**

- [+] SSP/canaries
- [+] ASLR
- [+] NX/W^X
- [+] KPTI/KAISER
- [+] Stripping

Some are NOT included

- [+] Libsafe & co., AAAS, StackGuard, StackShield, RAF-SSP, Secure patrol, JIT-ASLR, DieHarder, many more..

# Container engine Selection

### Container engines

Several containers platforms have been evaluated with respect to availability (A), functionality (F) and performances (P).

| Container Engine | A | F | P | Total |
|---|---|---|---|---|
| Docker | 4 | 5 | 5 | 47 |
| LXC | 3 | 3 | 4 | 32.5 |
| Rkt | 4 | 4 | 5 | 42.5 |

### Docker

- [+] Extensive set of functionality
- [+] The most widespread engine
- [+] Images in a public repository

### LXC

Virtualization system for ’full machines’

- [+] Configuration overhead
- [+] Poor image distribution
- [+] Very narrow use case

### Rkt

- [+] Almost like Docker
- [+] Worse API
- [+] Less usage
- [+] Benefits of *rkt* not relevant here

# Defense Measures Selection

### How to choose

Quite a problem, there are plenty of defenses, no objective criteria to measure them.

### Usage

The defense technique needs to be used in the real world, in production environments. If it is a default, even better.

### Effectivness

- [+] What attacks the technique protects from?
- [+] How common that attack is?
- [+] How bad would be the lack of this measure?

### Observations

- [+] Big gap between the academic and the industrial world
- [+] Novel tools designed are not available or used
- [+] No standard protection from RE for ELFs

# Scoring system

[+] **Two categories** scored separately: **system** and **ELF**.

[+] **System score** evaluates the vulnerabilities of the **environment** where binaries run.

[+] **ELF score** evaluates the vulnerabilities of a **single ELF** binary (per binary score).

[+] The scoring system is a primitive version of standard impact/likelihood score.

# Implementation

## Container images

| Image | Desription |
| --- | --- |
| kernelpop | Image containing an installation of Kernelpop tool. |
| radare2 | Image containing an installation of radare2 framework |
| aslr_check | Image containing a custom binary to verify the ASLR configuration. |
| spectre-meltdown | Image containing the spectre-meltdown-checker.sh script. |

## Programming language

- [+] Python
- [+] Native Docker Python lib
- [+] Supporting programs in C and bash

## General flow

- [+] Every check in a separate container
- [+] From every check the output/result is collected
- [+] Scoring is computed
- [+] Report written to file

# Testing & Validation

# 4 Benchmarks

**Performance test**

Execute the code that performs the check inside Docker and on the native machine.

Result: the only performance loss comes from bootstraping the container.

**Functionality test**

Execute TED on 3 different OS/VMs. Tamper the machines. Execute again TED and compare the results.

Result: TED caught the differences (tampers) in all cases.

**Real machine test**

Execute TED on an actual server and examine the result.

Result: TED allowed to detect possible vulnerabilities and to establish an action plan.

**Complex environment**

Execute TED on 4 machines (2 undefended, 2 defended) used for the Locked Shields 2018.

Result: TED managed to put in evidence few possible vulnerabilities, the test allowed to understand where TED lacks some functionalities but also showed that TED can be executed offline.

# Conclusions and Future Work

# Conclusions and Future Work

**Conclusions**

[+] Container usage in this context extremely helpful and profitable

[+] Tool produced effectively helps protecting machines from binary exploitation and/or establishing action plan

**Future work**

[+] Several features (ClamAV, unsafe function usage, remote host scanning, *hammer attacks)

[+] Development for running TED on a Kubernetes cluster

[+] RE protections techniques as research topic ?

# Thank you!

**Questions?**