Project Report: SQL Injection Detection Using Zeek

1. Introduction to the Project

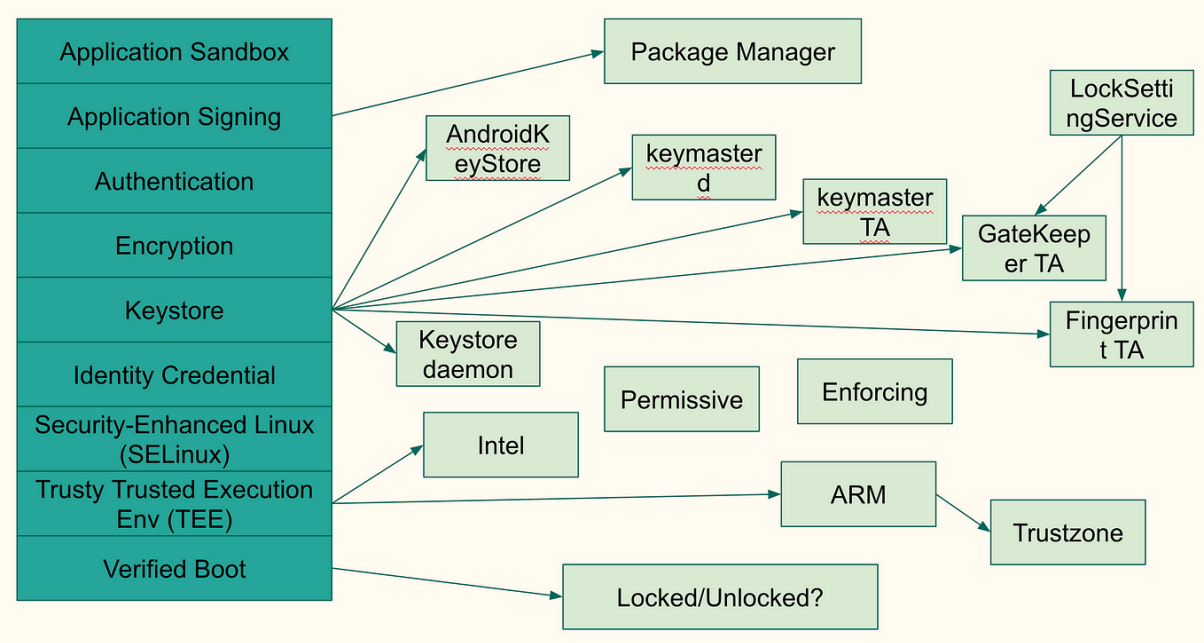
This study investigates the detection of SQL injection attempts directed towards web applications using Zeek, a network security monitoring tool. SQL injection is a common and serious attack vector that allows malicious SQL queries to be injected into user input fields, hence jeopardising the security of web applications. The aim of this research is to successfully identify and neutralise these assaults by utilising Zeek's capabilities.

2. Prerequisite of the Project

In order for this project to be completed successfully, the following conditions must be met:

- A basic understanding of the effects of SQL injection attacks.
- Knowledge of Zeek's (formerly called Bro) scripting features.
- Permission to test a web application that is susceptible to SQL injection.
- A monitoring system for Zeek installation and configuration.

3. Network Diagram



4. Project Methodology (Steps)

4.1 First Task: Set Up Zeek on the Watching Computer

- **Description:** Zeek needs to be set up on a specific monitoring computer in order to record and examine network data.

- **Steps:**

1. Go to the official website and download the Zeek installation package.
2. Observe the operating system-specific installation guidelines.
3. Check Zeek's status and run it to confirm the installation.

4.2 Task 2: Set Up Zeek to Record and Examine Data

- **Description:** To monitor traffic between the web server and the internet, configure Zeek.

- **Steps:**

1. Add the network interfaces that the web server uses to the Zeek configuration files.
2. Configure the right alerting and logging settings.
3. To implement the configuration changes, restart Zeek.

4.3 Task 3: Create Zeek Scripts to Identify SQL Injection

- **Description:** Create and modify Zeek scripts to find patterns of SQL injection in HTTP requests.

- **Steps:**

1. Create or alter Zeek scripts to find particular SQL injection patterns, like functions, keywords, and encoding schemes.
2. Make sure these scripts correctly detect SQL injection attempts by testing them.
3. For improved detection, make use of already-written scripts like detect-sqli.zeek.

4.4 Task 4: Create Attacks via SQL Injection

- **Description:** To mimic SQL injection attacks on a web site that is vulnerable, use a testing tool.

- **Steps:**

1. Find or develop a testing tool that can replicate SQL injection attacks.
2. Conduct deliberate assaults on the web application to potentially produce SQL injection traffic.
3. Keep an eye out for these fake attacks in Zeek's logs and alerts.

4.5 Task 5: Review and edit the Zeek logs

- **Description:** Examine Zeek's logs and warnings to spot erratic traffic patterns and improve routines.

- **Steps:**

1. Look for any patterns of SQL injection in the logs.
2. To increase detection accuracy, Zeek scripts should be adjusted and improved in light of the research.
3. Keep an eye on the scripts and update them frequently to accommodate new SQL injection methods.

5. Difficulties the Project Faces

Challenge 1: The first challenge is setting Zeek up to correctly record and examine pertinent traffic.

Challenge 2: Creating efficient Zeek scripts to identify different SQL injection patterns is the second challenge.

Challenge 3: Reducing the number of false positives while identifying SQL injection attempts.

Challenge 4: To ensure effective testing, simulate realistic attacks on the web application.

6. Conclusion: Project-Related Lessons Learnt

Lesson 1: Comprehending the subtleties of SQL injection attacks and the different types of assaults.

Lesson 2: Learning how to analyse and spot SQL injection attempts using Zeek scripting.

Lesson 3: Understanding how crucial it is to continuously improve scripts in order to stay ahead of emerging attack strategies.

Lesson 4: Zeek implementation as a useful part of an all-encompassing web application security plan.

7. Project Management: Enumerating the Tasks Completed by Every Team Member

Kunj Modi and Dhairya Soni: Setting up and configuring Zeek initially.

Jay Parekh: Writing Zeek scripts and modifying them to detect SQL Injection.

Deep Patel: Zeek log analysis and SQL injection attack simulation.

Mohitkumar Patel: Scripts are improved, and findings are documented.

REFERENCES:

1. https://github.com/zeek/zeek/blob/master/scripts/base/protocols/http/detect-sqli.zeek

2. https://zeek.org/