1. ABSTRACT

The global rise in smartphone usage has significantly increased attacks targeting these devices, particularly on the Android platform. Despite advancements in Android malware detection methods, many systems fail to deliver early detection, leaving devices vulnerable to malicious attacks. This underscores the critical need for mechanisms capable of identifying malware before it accesses sensitive data, along with improving detection accuracy and efficiency. To address these challenges, this study proposes a novel deep learning framework that leverages network traffic features for malware detection. Unlike traditional machine learning methods, which require extensive and time-consuming data preprocessing, deep learning automates feature extraction, significantly improving performance. The framework integrates Convolutional Neural Networks (CNN) to extract relevant features and Long Short-Term Memory (LSTM) networks to analyze sequential patterns in network traffic, achieving robust detection accuracy. Using the CICAndMal2017 dataset, the proposed model demonstrates exceptional effectiveness in malware family classification, paving the way for more secure mobile environments.

2. TCP Connect / Full Open Scan

TCP Connect Scan is a reliable method of TCP scanning. It uses the operating system's connect() system call to attempt connections to each port on the target machine.

- **How it works:**
  - If a port is listening, the connect() call successfully establishes a connection; otherwise, it returns an error indicating the port is unreachable.
  - The scan completes using the three-way handshake (3-way handshake):
    1. A SYN packet is sent to the target.
    2. The target responds with a SYN+ACK packet.
    3. An ACK packet is sent back to complete the connection.
  - After the handshake, a RST packet is sent to terminate the connection.
- **Accelerating the scan:**
  - Multiple sockets can be used in parallel.
  - Non-blocking I/O with short timeout periods allows monitoring multiple sockets simultaneously.
- **Drawbacks:**
  - Easily detectable and filterable as the target's system logs will record the connections.
  - Does not require superuser privileges to execute.

3. Stealth Scan (Half-open Scan)

Stealth Scan, also known as **SYN Scan**, involves partially opening a connection by abruptly resetting the TCP handshake before it completes. This method creates a "half-open" connection.

- **How it works:**
  - Sends a single SYN packet to a target port.
  - **Responses indicate the port state:**
    - If open, the server replies with a SYN/ACK packet.
    - If closed, the server sends an RST packet.
  - The client sends an RST packet after receiving a response, terminating the process before a full connection is established.
- **Key Characteristics:**
  - Uses only the SYN packet, preventing the service from logging the connection attempt.
  - Examines packets entering the interface to identify port states.
  - Avoids completing the full three-way handshake.
- **Purpose:**
  - Stealth scanning techniques are designed to bypass firewalls and logging systems, allowing attackers to mask their activities within normal network traffic.

4. SQL Injection là gì?

- SQL injection is a technique used to take advantage of un-sanitized input vulnerabilities to pass SQL commands through a web application for execution by a backend database.
- SQL injection is a basic attack used to either gain unauthorized access to a database or retrieve information directly from the database.
- It is a flaw in web applications and not a database or web server issue.

5. SQL Injection Concepts, Definition, Examples?

Understanding HTTP POST Request

HTTP POST request is one of the methods to carry the requested data to the web server. Unlike the HTTP GET method, HTTP POST Request carries the requested data as a part of the message body. Thus, it is considered more secure than HTTP GET. HTTP POST requests can also pass large amounts of data to the server. They are ideal in communicating with an XML web service. These methods submit and retrieve data from the webserver.

When a user provides information and clicks **Submit**, the browser submits a string to the web server that contains the user's credentials. This string is visible in the body of the HTTP or HTTPS POST request as

```
select * from Users where (username = 'smith' and password = 'simpson');
```

Understanding Normal SQL Query

A query is an SQL command. Programmers write and execute SQL code in the form of query statements. SQL queries include data selection, data retrieval, inserting/updating data, and creating data objects like databases and tables. Query statements begin with a command such as SELECT, UPDATE, CREATE, or DELETE. Queries are used in server-side technologies to communicate with an application's database. A user request supplies parameters to replace placeholders that may be used in the server-side language. From this, a query is constructed and then executed to fetch data or perform other tasks on the database. The above diagram depicts a typical SQL query. It is constructed with user-supplied values, and upon execution, it displays results from the database.

Understanding SQL Injection Queries

SQL injection is an attack that exploits vulnerabilities in how applications handle data. Attackers submit malicious values through input fields, taking advantage of the application's failure to properly validate or filter the data before processing.

A common example involves an HTML form that transmits information, like a username and password, to an ASP script running on an IIS server. Attackers might input:

- **Username:** Blah' or 1=1 --
- **Password:** Springfield

```
SELECT Count(*) FROM Users WHERE UserName='Blah' or 1=1 -- ' AND Password='Springfield';
```

Upon analysis, the condition 1=1 is always true, allowing the query to execute successfully. Since there are no syntax errors, the database processes the query normally, potentially exposing sensitive data. This illustrates the mechanics of an SQL injection attack.

Understanding an SQL Injection Query—Code Analysis

Code analysis or code review is the most effective technique in identifying vulnerabilities or flaws in the code. An attacker exploits the vulnerabilities found in the code to get an access to the database. The process by which an attacker logs into an account is as follows:

1. A user enters a username and password that matches a record in the user's table
2. A dynamically generated SQL query is used to retrieve the number of matching rows
3. The user is then authenticated and redirected to the requested page
4. When the attacker enters **blah' or 1=1 --** then the SQL query will look like:
   ```
   SELECT Count(*) FROM Users WHERE UserName='blah' Or 1=1 --' AND Password=''
   ```
5. A pair of hyphens designate the beginning of a comment in SQL; therefore, the query simply becomes
   ```
   SELECT Count(*) FROM Users WHERE UserName='blah' Or 1=1
   string strQry = "SELECT Count(*) FROM Users WHERE UserName='" + txtUser.Text + "' AND Password='" + txtPassword.Text + "'";
   ```

6. How to countermeasure sql injection

To defend against SQL injection attacks, a system should follow the countermeasures described in the previous section and use type-safe SQL parameters as well. To protect the web server, use WAF firewall/IDS and filter packets. Regularly update the software using patches to keep the server up-to-date to protect it from attackers. Sanitize and filter user input, analyze the source code for SQL injection, and minimize the use of third-party applications to protect the web applications. Use stored procedures and parameter queries to retrieve data and disable verbose error messages, which can guide an attacker with useful information, and use custom error pages to protect the web applications. To avoid SQL Injection into the database, connect using nonprivileged accounts and grant the least possible privileges to the database, tables, and columns. Disable commands such as xp_cmdshell, which can affect the OS of the system.

- Make no assumptions about the size, type, or content of the data that is received by your application
- Test the size and data type of input and enforce appropriate limits to prevent buffer overruns
- Test the content of string variables and accept only expected values
- Reject entries that contain binary data, escape sequences, and comment characters
- Never build Transact-SQL statements directly from user input and use stored procedures to validate user input
- Implement multiple layers of validation and never concatenate user input that is not validated
- Avoid constructing dynamic SQL with concatenated input values
- Ensure that the Web config files for each application do not contain sensitive information
- Use most restrictive SQL account types for applications

7. Qúa trình Scanning Pen Testing

- **Step 1: Perform host discovery**

  The first step of network penetration testing is to detect live hosts on the target network. You can attempt to detect the live hosts (i.e., accessible hosts in the target network), using network scanning tools such as Nmap, Angry IP Scanner, SolarWinds Engineer's toolset, and NetScanTools Pro. It is difficult to detect live hosts behind a firewall.
- **Step 2: Perform port scanning**

  Perform port scanning using tools such as Nmap, NetScanTools Pro, Hping3, PRTG Network Monitor, and SuperScan. These tools help to probe a server or host on the target network for open ports. Open ports are the doorways through which an attacker installs malware on a system. Therefore, you should always check for open ports and close them if they are not necessary.
- **Step 3: Scan beyond IDS and firewall**

  Scan beyond IDS and firewall; this helps you to understand the organization's security limitations. Use IDS/firewall evasion techniques such as packet fragmentation, source routing, IP address spoofing, etc. to bypass IDS and firewall rules.

  Use proxy tools such as Proxy Switcher, Proxy Workbench, CyberGhost, Tor, and Burp Suite to hide yourself from detection.
- **Step 4: Perform banner grabbing or OS fingerprinting**

  Perform banner grabbing/OS fingerprinting by sending specially crafted packets to the target machine and then comparing the responses with the database. This determines the operating system running on the target host of a network and its version. Once you know the version and the operating system running on the target system, find and exploit the vulnerabilities related to that OS. Try to gain control over the system and compromise the whole network.
- **Step 5: Draw network diagrams**

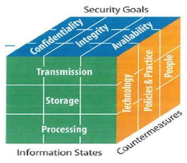
  Draw a network diagram of the vulnerable hosts that helps you to understand the logical connection and path to them in the network. You can draw the network diagram with the help of tools such as Network Topology Mapper, OpManager, The Dude, NetSurveyor, and NetBrain. The network diagrams provide valuable information about the network and its architecture.
- **Step 6: Document all the findings**

  The last but the most important step in penetration testing is to preserve all the outcomes of tests conducted in previous steps in a document. This document will assist in finding potential vulnerabilities in the network which you can use to suggest countermeasures. Thus, penetration testing helps in assessing the security posture of the network and fixing any security loopholes before they can cause trouble and result in severe organizational loss.

8. McCumber Cube

McCumber cube

- The McCumber Cube is a comprehensive cybersecurity framework developed by John McCumber in 1991 to address the complexities of information security.
- It is represented as a three-dimensional cube that incorporates three key dimensions: **information states**, **security goals**, and **countermeasures**.
- The **information states** include transmission, storage, and processing, while the **security goals** focus on confidentiality, integrity, and availability (CIA Triad).
- The third dimension, **countermeasures**, emphasizes technology, policies, and human factors as essential strategies to protect information.
- By integrating these dimensions, the McCumber Cube provides a holistic approach to designing and evaluating effective information security systems.



1. Information States

This dimension identifies the phases in which information exists:

- **Transmission**: When information is being transferred (e.g., sending emails, data transfer over networks).
- **Storage**: When information is at rest (e.g., saved in databases, cloud storage, or physical devices).
- **Processing**: When information is being actively used or manipulated (e.g., calculations, data analysis).

---

2. Security Goals

Based on the **CIA Triad**, this dimension defines the core objectives of cybersecurity:

- **Confidentiality**: Ensures information is accessible only to authorized individuals (e.g., encryption, access control).
- **Integrity**: Ensures data remains accurate, consistent, and unaltered (e.g., digital signatures, hash verification).
- **Availability**: Ensures data and systems are accessible when needed (e.g., redundancy, disaster recovery plans).

---

3. Countermeasures

This dimension focuses on strategies to protect information:

- **Technology**: Tools and systems to secure data (e.g., firewalls, intrusion detection systems, encryption).
- **Policies & Practices**: Organizational rules and procedures to guide information handling (e.g., access policies, incident response plans).
- **Human Factors**: Training and awareness programs to mitigate risks caused by human error (e.g., security awareness training, phishing prevention).

9. 5 phases of hacking

1. Reconnaissance

Also known as **information gathering or footprinting**, this phase involves collecting as much information as possible about the target.

- **Objective**: Identify vulnerabilities, network details, employee information, and other data.
- **Techniques**:
  - Passive: Public sources (e.g., websites, social media, WHOIS lookups).
  - Active: Probing networks with tools (e.g., Nmap, Shodan).
- **Tools**: Maltego, Google Dorking, Nmap, Recon-ng.

---

2. Scanning

In this phase, the attacker probes the target to identify open ports, services, and potential vulnerabilities.

- **Objective**: Determine the network architecture, live systems, and exploitable weaknesses.
- **Types**:
  - Network scanning (e.g., finding live hosts).
  - Vulnerability scanning (e.g., identifying weak spots in software).
  - Port scanning (e.g., detecting open ports).
- **Tools**: Nessus, OpenVAS, Nikto, Nmap.

---

3. Gaining Access

The attacker exploits identified vulnerabilities to gain unauthorized access to the target system.

- **Objective**: Compromise the system and establish control.
- **Techniques**:
  - Exploiting software vulnerabilities.
  - Phishing attacks.
  - Password cracking.
- **Tools**: Metasploit, SQLmap, Hydra, Burp Suite.

---

4. Maintaining Access

After gaining access, the attacker ensures they can retain control over the compromised system.

- **Objective**: Create backdoors, escalate privileges, or persist within the system undetected.
- **Techniques**:
  - Installing malware (e.g., rootkits, Trojans).
  - Exploiting misconfigurations.
- **Tools**: Netcat, Persistence modules in Metasploit, keyloggers.

---

5. Covering Tracks

This phase involves erasing evidence of the attack to avoid detection and ensure the compromised system remains usable for further exploitation.

- **Objective**: Remove logs, disable security systems, and obscure activities.
- **Techniques**:
  - Deleting logs.
  - Modifying timestamps.
  - Using anti-forensic tools.
- **Tools**: CCleaner, log manipulation scripts, Steganography tools.

The process of attacking a **target machine** typically involves two key frameworks: the **hacking phases** and the **penetration testing (pentesting) process**. Below is an outline of each:

Hacking Phases

1. **Reconnaissance (Information Gathering)**
   - Collect information about the target, such as IP addresses, domain names, operating systems, and services running on the system.
   - Tools: Nmap, WHOIS, Shodan, OSINT (Open Source Intelligence) platforms.
2. **Scanning (Vulnerability Scanning)**
   - Identify open ports, active services, and potential vulnerabilities.
   - Tools: Nmap, Nikto, OpenVAS.
3. **Gaining Access (Exploitation)**
   - Exploit identified vulnerabilities to gain unauthorized access to the target.
   - Tools: Metasploit, SQLmap, or custom malware.
4. **Maintaining Access**
   - Install backdoors or persistence mechanisms to retain control over the target system.
   - Tools: Netcat, Empire, Cobalt Strike.
5. **Covering Tracks**
   - Erase system logs and hide activities to avoid detection.
   - Techniques include log wiping, encryption, and forging log entries.

Pentesting Process

1. **Planning and Reconnaissance**
   - Define the scope, goals, and limitations of the test.
   - Gather information about the target (similar to reconnaissance in hacking).
2. **Scanning**
   - Perform active or passive scans to identify system vulnerabilities.
3. **Exploitation**
   - Exploit vulnerabilities to test access and control mechanisms.
4. **Post-Exploitation**
   - Assess the level of control gained, identify sensitive data, and evaluate potential impact.
5. **Reporting**
   - Provide a detailed report of identified vulnerabilities, attack methods, and remediation steps.

10. OWASP Top 10 Mobile

The OWASP Mobile Top 10 outlines the most critical security risks for mobile applications, offering developers and security professionals a reference to secure mobile environments. Here's a brief breakdown:

1. **Improper Credential Usage (M1)**: This involves mishandling of sensitive credentials, such as hardcoded secrets in source code or insecure storage practices.
2. **Inadequate Supply Chain Security (M2)**: Vulnerabilities may be introduced through third-party libraries or dependencies, which can potentially compromise the app.
3. **Insecure Authentication/Authorization (M3)**: Weak authentication systems or flawed access control mechanisms can allow unauthorized users to gain access.
4. **Insufficient Input/Output Validation (M4)**: Failing to properly validate and sanitize data can lead to attacks such as SQL injections or data corruption.
5. **Insecure Communication (M5)**: Communication vulnerabilities arise when data is transmitted without proper encryption or security protocols.
6. **Inadequate Privacy Controls (M6)**: This involves failing to protect user data adequately, which could result in privacy violations.
7. **Insufficient Binary Protections (M7)**: Weak protections make it easier for attackers to reverse-engineer, tamper with, or exploit the app's binary code.
8. **Security Misconfiguration (M8)**: Security misconfigurations, such as leaving default credentials or debug mode active, can leave an app vulnerable to attacks.
9. **Insecure Data Storage (M9)**: Storing sensitive data insecurely on the device (e.g., in plain text) can lead to leakage if the device is compromised.
10. **Insufficient Cryptography (M10)**: The use of outdated or weak cryptographic algorithms can jeopardize the security of the app and its data.