UNIVERSITI SAINS MALAYSIA

CST334 : NETWORK MONITORING AND SECURITY
CST338 : NETWORK AND COMMUNICATION SECURITY

Port Scanning Detection System Testing Plan

Daniel Hergast
Simon Alix

Date: 16.01.2022

Contents

1 Introduction

We acknowledge that no network or company is completely immune to cyber attacks. While we cannot completely prevent an attack, we can strengthen our defense by implementing various security tools. Some of these tools actively block attempts while others passively detect intrusions. The PSDS belongs to the latter category, it is a passive security tool designed to detect and log port scanning attempts. Our assumption is that cyber attacks usually begin with a reconnaissance phase, which often involves port scanning. The primary goal of our PSDS tool is to accurately record the date, time, and other details of these attempts, providing a starting point for further analysis of the attack. The PSDS can significantly reduce the time and resources required for packet analysis by focusing on the specific time frame of the attack.

Our tool, the PSDS, has three main components: the Network Capturing System, the Pattern-based Port Scan Detection, and the logging and alerting component.:

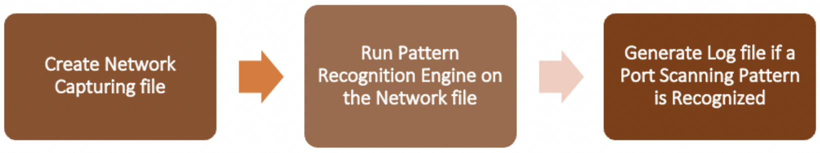
Create Network Capturing file
Run Pattern Recognition Engine on the Network file
Generate Log file if a Port Scanning Pattern is Recognized

Figure 1: workflow of the port scanning detection system

- The buffer component of our tool is responsible for capturing network traffic on a specific interface and temporarily storing the packets. This buffer allows us to analyze the packet flow at a later time, as we are unable to track it in real-time. The buffer size can be configured using a configuration file, which includes parameters such as buffer size, number of files, and number of packets. These specifications can be adjusted based on the specific needs and resources of the company. For example, in our use case, the operator is required to take a log every day, so the buffer size should be sufficient to store one day's worth of data.

- The second component of our tool is a filter that helps us identify potential scan attacks by analyzing IP packets, particularly TCP packets, and paying close attention to the TCP handshake. We use certain clues, such as a conversation that is only a half-handshake, or a header and windows size that is abnormally short, to identify potential scan attacks. However, this filter cannot be applied to real-time traffic, as it requires tracing back all the conversations and identifying redundant IP addresses. So far, our detection algorithm can detect two types of scan: stealth scan and TCP scan.

- The final component of our tool is the output, which is a log file that collects information about suspicious packets. This includes details such as the name of the .pcap file, the packet number, the type of scan, the IP address of the attacker and the target, and the number of scanned ports. This log file can provide a detailed starting point for further analysis of the attack.

2 Test Items

The following functions are integral to the main functionality of the PSDS tool, and should be thoroughly tested to ensure their proper operation:

1. Function for selecting the desired network interface: As the PSDS tool may be connected to multiple interfaces, it is crucial to have a function that allows users to choose the specific interface they want to monitor.
2. Traffic capturing function: Using the dumpcap function from Wireshark, this function captures network traffic and saves it to a .pcap file.
3. Scan detection function: This function analyzes the packets in the input .pcap file to identify any potential port scan attempts.
4. Directory management function: This function manages the process of running the filter on new .pcap files and controls the buffer ring.
5. Configuration management function: This function allows users to adjust various parameters such as buffer size, filter sensitivity, and log frequency.
6. Log and alert function: This function records the output of the scan detection function in a text file.

3 Features to be tested and not

The PSDS project and tool are nearing completion, our initial plan divided the work into 6 different functions. We were able to successfully complete and test 3 of them, which are the main items B, C, D, and F. However, due to limited time, we have not been able to fully test items A and E. The tool has potential for further improvement, specifically in the filter component where more types of port scan attacks such as UDP, SYN, or FIN scan can be recognized.

4 Software Risk Issues

As a software tool, PSDS has the potential to impact its surrounding environment. The software risks associated with PSDS refer to the negative consequences that may result from the use or implementation of the PSDS software system. These risks may include security vulnerabilities, data loss, system failures, and compliance violations. It is crucial to identify and manage these risks during the development and implementation of the PSDS software. Methods for managing these risks include risk assessment, risk mitigation, and ongoing risk monitoring. The traffic capturing function requires administrator level access to execute the Wireshark command, which is the only potential side effect of the project. Overall, there are no significant risk issues associated with our project and no conflicts between the internal functions.

5 Approach

Testing a software system such as PSDS is crucial to ensure that it functions correctly and meets the necessary requirements. A test approach for PSDS involves the careful planning and execution of various tests that are tailored to the specific needs and goals of the system and organization.

5.1 Testing Levels

Testing levels refer to the different stages at which testing is performed during the software development process.

1. Unit testing: This level of testing is focused on testing individual units or components of the system, such as functions or methods. Unit testing is typically performed by developers and is focused on verifying that each unit of the code is working correctly and independently.

2. Integration testing: This level of testing is focused on testing the interactions and integration between different components or systems. Integration testing is typically performed after unit testing and is focused on verifying that the components work together as intended, and there is no conflict between them.

3. System testing: This level of testing is focused on testing the entire system as a whole. System testing is typically performed after integration testing and is focused on verifying that the system meets the requirements and behaves as expected.

4. Acceptance testing: This level of testing is focused on testing the system from the perspective of the end-user. Acceptance testing is typically performed by the client or customer and is focused on verifying that the system meets their needs, is user-friendly, and is ready for release.

5. Performance testing: This level of testing is focused on testing the system's performance under different load and stress conditions. Performance testing is typically performed after system testing and is focused on verifying that the system can handle the expected load and perform well under different conditions.

6. Security testing: This level of testing is focused on testing the system's security measures and identifying any potential vulnerabilities. Security testing is typically performed after system testing and is focused on verifying that the system is secure and protected against external threats.

5.2 Test Tools

To effectively test and evaluate our code, we rely on specific tools. To test the port scan detection function, we use the nmap tool to simulate various port scan attacks. Additionally, we use Wireshark for its visual interface and filtering capabilities to inspect and analyze the packets in more detail:

- Nmap (Network Mapper) is a widely-used and open-source tool for network exploration and security auditing. It can be used to discover hosts and services on a computer network, and to gather information about their features and vulnerabilities. Nmap can scan a network for open ports, operating systems, and other information. It can also be used for advanced tasks such as version detection, OS detection, and scriptable interaction with the target. Additionally, Nmap can be used to assess the security of a network and identify potential vulnerabilities that could be exploited by an attacker. In our case, we use Nmap to simulate port scan attacks on a machine that we own and is connected to the same network as the traffic capturing machine.
- Wireshark is a widely-used and open-source packet analyzer tool. It is used for network troubleshooting, analysis, software and protocol development, and education. With Wireshark, you can capture network traffic in real-time and analyze the packets to understand the workings of a network and troubleshoot issues. It enables to see all network traffic, including broadcast and multicast packets, and can be used to filter and analyze the data. It also supports a wide range of protocols such as TCP, UDP, HTTP, DNS, among others. Wireshark has a rich set of features, including the ability to save captured packets to a file and export the packet data to various file formats, including XML, CSV, and the PCAP format we use.

5.3 Meetings

This project is being developed by two students from USM. The work distribution is as follows: one student is responsible for focusing on the main architecture of the code, from the network capturing to the alert log file, and the other student is responsible for working on the filter to detect port scan attempts. To ensure that the project is on track and objectives are met, regular meetings are held. The average meeting frequency is twice a week to check on the progress of the main structure and address any issues that arise.

5.4 Measures and Metrics

There are various measures and metrics that can be used to evaluate the performance of a port scanner detector during testing, such as:

- True Positive Rate (TPR) or Sensitivity: This metric measures the proportion of actual port scans that were correctly identified by the detector.
- False Positive Rate (FPR) or Fall-out: This metric measures the proportion of normal network traffic that was incorrectly identified as port scans by the detector.
- Precision: This metric measures the proportion of positive cases identified by the detector that are actually true positives.
- Detection Rate (DR): This metric measures the proportion of all port scans that were correctly identified by the detector.
- Detection Accuracy (DA): This metric measures the proportion of all network traffic that was correctly identified by the detector (both true positives and true negatives).

- False Alarm Rate (FAR): This metric measures the proportion of all negative cases that were incorrectly identified as positive port scans by the detector.
- Missed Detection Rate (MDR): This metric measures the proportion of all port scans that were not identified by the detector.

These metrics can be used to assess the performance of the port scanning detection system, identify areas for improvement, and compare different detector implementations.

6 Item Pass/Fail Criteria

The failure criteria for a testing plan for the port scanning detection system could include:

- False negatives: The detector fails to detect actual instances of port scanning.
- False positives: The detector incorrectly flags legitimate network traffic as port scanning.
- Missed scan types: The detector fails to detect specific types of port scanning techniques.
- Limited coverage: The detector only detects a limited number of ports or IP addresses.
- Inability to handle high traffic: The detector is not able to handle high volumes of network traffic and becomes overwhelmed.
- Lack of adaptability: The detector is not able to adapt to new or evolving port scanning techniques.
- Difficulty of use: The detector is difficult to set up, configure, or use.
- Lack of robustness: The detector is not robust enough and fails under certain conditions.
- Integration challenges: The detector does not integrate well with other security systems.

7 Remaining Test Tasks

The PSDS project has the potential for real-world application and can be further developed and expanded. However, before proceeding with further developments, the following tasks need to be completed:

- Code and test items A and E, which are the interface choice function and the configuration and management file.
- Implement a graphical user interface (GUI) that centralizes the management file, buffer parameters, and log alerts.

- Create a visually appealing display for the scan detection logs, currently it is a text file.
- Add more filters to detect UDP scans or FIN scans.
- Establish a permanent connection to a firewall.

8 Environment Needs

To support the testing efforts throughout the project, the following tools and libraries are required:

- Wireshark: a packet analysis tool
- Python 3: a programming language
- Scapy library: a packet manipulation library for Python
- os library: a library for interacting with the operating system in Python
- Personal library of filters based on Scapy: a set of custom filters for the detection of port scans