ARP Cache Poisoning Simulation

Project Description

The project is focused on implementing a simulation of ARP cache poisoning, a common technique used in man-in-the-middle (MITM) attacks. It will involve creating a tool that can intercept, redirect, or modify network traffic between two devices by manipulating ARP tables. The tool will be developed in C, and the primary goal is to gain a deeper understanding of ARP protocol vulnerabilities and how MITM attacks are executed. We will create a controlled environment for the simulation, focusing on educational purposes and defensive measures.

Learning Goals

- **ARP Protocol**: Understand the Address Resolution Protocol, how ARP tables work, and the role of ARP requests and responses in network communication.
- **ARP Spoofing Techniques**: Learn about different methods attackers use to perform ARP cache poisoning and how they intercept or manipulate data in a local network.
- **C Programming for Networking**: Develop a more profound knowledge of socket programming, raw packet creation, and low-level network manipulation using C.
- **Defensive Measures**: Understand the detection and prevention techniques, such as packet inspection and static ARP tables, for mitigating ARP spoofing attacks.

Development Goals

Core Features

- **Network Discovery**: A feature that identifies all devices in the local network using ARP requests.
- **ARP Spoofing**: Implement the core ARP cache poisoning technique to target specific devices.
- **Packet Interception**: Capture and optionally modify packets between the victim devices.
- **Packet Redirection**: Redirect packets to other devices or relay them to maintain regular network operation.
- **Visualization**: Create a simple command-line output or log file showing ARP tables before, during, and after the attack.

Stretch Goals

- **GUI Visualization**: A basic GUI that shows a real-time network map with spoofed connections highlighted.
- **Defensive Mode**: Implement a defensive mode that detects ARP spoofing and alerts the user.

- **Custom MITM Attacks**: Enable custom scripts to automate actions once the attacker is in the middle, such as injecting packets or performing DNS spoofing.
- **ARP Spoofing Defense Simulation**: Demonstrate different methods of defending against ARP spoofing, such as static ARP tables and packet inspection.

Testing and Benchmarking

Testing Plan

- **Unit Testing**: Develop unit tests for individual modules such as network discovery, ARP spoofing, and packet interception.
- **Controlled Environment Testing**: Use a virtual network environment with multiple VMs to simulate the local network. Test the tool's ability to perform ARP poisoning and intercept traffic successfully.
- **Correctness Testing**: Verify that ARP tables on the victim machines are modified as expected and that packet interception works without causing network disruptions.
- **Defensive Test Cases**: Test the tool's behavior against standard defensive techniques, such as static ARP tables and intrusion detection systems.

Benchmarking Plan

- **Performance Analysis**: Measure the time to send ARP requests and modify ARP tables.
- **Scalability Testing**: Test the tool's performance with varying numbers of devices in the network.
- **Resource Usage**: Monitor CPU and memory usage to ensure the tool is lightweight and does not significantly degrade network performance.

Development Schedule

Week 3

- **Research**: Gather information on ARP protocol/spoofing, socket programming, and techniques.

Week 4

- **Environment Setup**: Set up a virtual network environment using VMs.
- **Initial Planning**: Finalize the project structure and critical modules.

Week 5

- **Network Discovery Module**: Implement network scanning and device identification.
- **ARP Spoofing Module**: Implement a basic version of ARP spoofing to alter ARP tables.

Week 6

- **Packet Capture and Analysis**: Implement packet sniffing using libpcap or similar libraries.
- **Packet Modification**: Add functionality to modify intercepted packets and test this on the virtual

network.

Week 7

- **Unit Testing**: Develop and run unit tests for all major modules.
- **Controlled Testing**: Simulate attacks on the virtual network to identify bugs or stability issues.

Week 8

- **Stretch Goals:** Try

Week 9

- **Performance Testing**: Measure the tool's efficiency and resource usage.
- **Documentation**: Complete detailed documentation for each module and write a user guide.
- **Project Presentation**: Prepare a demonstration for the class, showcasing key features and findings.