SSH Cracker Using Python

By Inlighn Tech

Objective:

The objective of this project is to develop an SSH brute-force tool that tests username-password combinations to gain unauthorized access to an SSH server. The project helps students understand SSH security vulnerabilities, brute-force techniques, multi-threading, and password list attacks.

Project Overview:

This project consists of two Python scripts:

1. `advance_ssh_brute.py` – An advanced SSH brute-forcing script that supports username lists, password lists, password generation, multi-threading, and retry mechanisms.
2. `ssh_brute.py` – A simple SSH brute-force script that accepts a single username and a password list.

The tool attempts to connect to an SSH server by iterating through different username-password pairs. If successful, it saves the credentials to a file.

How the Project Works:

1. **User Input: The script accepts inputs such as hostname, username(s), password(s), and optional parameters like password generation settings.**
2. **Connection Attempt: The tool uses the `paramiko` library to connect to the SSH server with the given credentials.**
3. **Authentication Handling: If authentication fails, it moves to the next combination. If successful, it stores the credentials.**
4. **Retry Mechanism: In case of SSH errors or rate limits, the tool implements retry logic with delays.**
5. **Multi-threading (Advanced Version): The `advance_ssh_brute.py` script uses multiple threads to speed up brute-force attempts.**

Key Concepts Covered:

- **Understanding SSH authentication and security vulnerabilities**
- **Implementing brute-force and dictionary attacks**
- **Using Python libraries (`paramiko`, `argparse`, `queue`, `threading`)**
- **Handling network timeouts and authentication errors**
- **Optimizing performance with multi-threading**

Step-by-Step Implementation:

1. **`advance_ssh_brute.py` - Advanced SSH Cracker:**
    a. Supports username lists and password lists.
    b. Can generate passwords dynamically.
    c. Uses multi-threading for faster brute-force attempts.
    d. Implements a retry mechanism for failed SSH connections.
2. **`ssh_brute.py` - Basic SSH Cracker:**
    a. Accepts a single username and password list.
    b. Tests passwords sequentially without multi-threading.
    c. Implements a retry mechanism in case of SSH errors.

Expected Outcomes:

By completing this project, students will:

- Understand SSH authentication mechanisms and security flaws.
- Learn brute-force attack methodologies and countermeasures.
- Gain experience with multi-threading for performance optimization.
- Develop a functional tool for penetration testing and ethical hacking.

Next Steps:

Students should implement their own version of the SSH cracker using the outlined concepts. A video lecture will be provided later to demonstrate the correct implementation and solution. This project serves as a foundational step for cybersecurity and ethical hacking tasks in Python.

For further enhancements, students can:

- **Add Proxy Support: Implement proxy rotation to bypass IP blocking mechanisms.**
- **Improve Multi-threading: Optimize thread management for better performance and resource utilization.**
- **Use Machine Learning: Train a model to predict commonly used passwords for targeted brute-force attacks.**
- **Implement Key-based Authentication Cracking: Extend functionality to test SSH private key authentication.**
- **Develop a GUI: Create a graphical interface for better usability and visualization of attack progress.**
- **Enhance Logging & Reporting: Generate detailed reports of successful and failed login attempts for analysis.**