Homework 4
COSI 141A: System & Network Security

Due: December 8, 2025 at 11:59 PM. No late submissions

Designing and deploying an SSH-based Honeypot

A honeypot is a decoy system used to detect and study attackers on a network. It simulates a vulnerable service so that attackers interact with it instead of a real system. When it detects suspicious activity, it responds as if it were a real system, while recording the attacker's behavior. In this homework, you will create a honeypot server that exposes an `ssh` service. Your honeypot must:

- detect brute-force login attempts
- provide the attacker with an interactive shell, and
- emulate a small, fake file system.

You are required to implement your honeypot server so that it should initiate operation using the following inline argument:

```
>> python honeypot.py -p [port].
```

The argument defines the TCP port that your honeypot `ssh` server will bind to. While the default SSH port is `22`, your server must work on *any* port number.

1.Detecting a brute force attack

Your honeypot server should be able to keep track of `ssh` login attempts for a specific user.If a given username has more than 5 failed login attempts, the honeypot should grant access to a shell for that user. An SSH connection can be initiated using the following command:

```
>> ssh -p port USER@honeypot_ip
```

For this homework, you may assume:

- The honeypot runs on `localhost`.
- The only valid usernames are `victim55`, `bob`, and `garth22`.
- The attacker only brute-forces one username at a time.
- The password itself is *irrelevant*: you do not need to verify it.
- You may use any open-source Python library to implement the SSH server functionality (e.g., paramiko)
- SSH servers require a host key to run. You do *not* need to ship or submit any private keys. Instead, generate a temporary in-memory RSA key inside your code whenever your server starts.

2. Implementing SSH functionality

Once the attacker (client) has been granted access and a connection has been established, your honeypot should communicate with the client which replicates a normal `ssh` connection. For instance, on a successful attempt by an attacker for the username `bob`, attacker should be able to see a shell prompt on the client side which resembles the following:

```
bob@honeypot:/$
```

The client should be able to type commands, which are sent to your honeypot when they press Enter. Your server must:

- Maintain an interactive session with the client over SSH (using a real `ssh` client on the attacker side).
- Disconnect the client if no input is received for 60 seconds (idle timeout).
- Handle backspace correctly so that the user can edit their current line.

3. Implementing a basic file system

Once you have `ssh` implemented, the final task is to create a **lightweight, in-memory** file system and a set of related commands used exclusively by your honeypot. Specifically, your honeypot must support the following commands:

- **`pwd`**: prints the absolute path of the current working directory.
- **`ls`**:Lists all files in the current directory. Specifically:
  - Files appear without a slash.
  - You must pre-populate the following file at the start of each session: `secrets.txt` a fake "interesting" honeyfile with sensitive-looking content.
- **`echo "content" > destination.txt`**:
  - Creates or overwrites a text file named `destination.txt` in the current directory with content `''content''`.
  - If the destination does not end in `.txt`, you must print `Unknown file extension` and not create a file.
- **`cat source.txt`**:Prints the contents of the specified file in the current directory.
  - If the file does not exist, print `File X not found`, where `X` is the filename given by the user.
  - If the filename does not end in `.txt`, print `Unknown file extension`.
  - Assume that the attacker will use this command to read your honeyfile too.
- **`cp source.txt destination.txt`**:
  - You may assume both `source.txt` and `destination.txt` are simple filenames (no path separators).
  - If the source file does not exist, print `File X not found`, where `X` is the source filename.
  - If either filename does not end in `.txt`, print `Unknown file extension`.
- You do not need to support absolute paths in commands. All paths may be treated as *relative* to the current directory.

Note: If the source file does not exist, `cat` and `cp` must print `File X not found`. Only filenames ending in `.txt` are valid. Otherwise print `Unknown file extension`. You can find more about the bash commands and their syntax here.

Testing.

Testing for the honeypot system should be simple. You can test everything on your local machine, by running the honeypot server in one terminal window and then running an `ssh` client in another one, which attempts to make connections with your honeypot server.

Session Logging.

Your honeypot must maintain a log of all authenticated sessions in a file named `logs.csv`. Each *session* must produce one line in the log.

Each log entry must contain the following comma-separated fields:

- Session identifier (for example, an incremental integer)
- Client IP address
- Username used to log in
- Timestamp of session start
- Timestamp of session end
- The full list of commands typed by the client during the session

Example shell interaction.

Commands typed by the client are in blue.

```
bob@honeypot:/$ ls
secrets.txt
bob@honeypot:/$ echo "this is a message" > msg.txt
bob@honeypot:/$ cat msg.txt
this is a message
bob@honeypot:/$ cp msg.txt msg2.txt
bob@honeypot:/$ ls
secrets.txt msg.txt msg2.txt
bob@honeypot:/$ cat foo.txt
File foo.txt not found
bob@honeypot:/$ cat file
Unknown file extension
```

Example logs file

```
1,192.0.2.10,bob,2025-11-19T10:15:23,2025-11-19T10:18:47,"ls"
2,198.51.100.23,victim55,2025-11-19T11:02:05,2025-11-19T11:04:12,"ls; cat secrets.txt"
3,203.0.113.5,garth22,2025-11-19T12:30:41,2025-11-19T12:31:09,"ls; cat foo.txt; cat unknown"
```

Submission

For this assignment you are required to upload the following files in an archive named as `hw4.zip`.

1. **`honeypot.py`:** Your main honeypot program.
2. **`honeypot.log`:** A sample log file generated by your honeypot.
3. **`explanation.txt`:** Describe your general approach for this homework and the list of online resources that you referred to.

Good Luck!