Computer Security Capstone

Project III: Ransomware Propagation and Payload

Chi-Yu Li (2025 Spring)
Computer Science Department
National Yang Ming Chiao Tung University

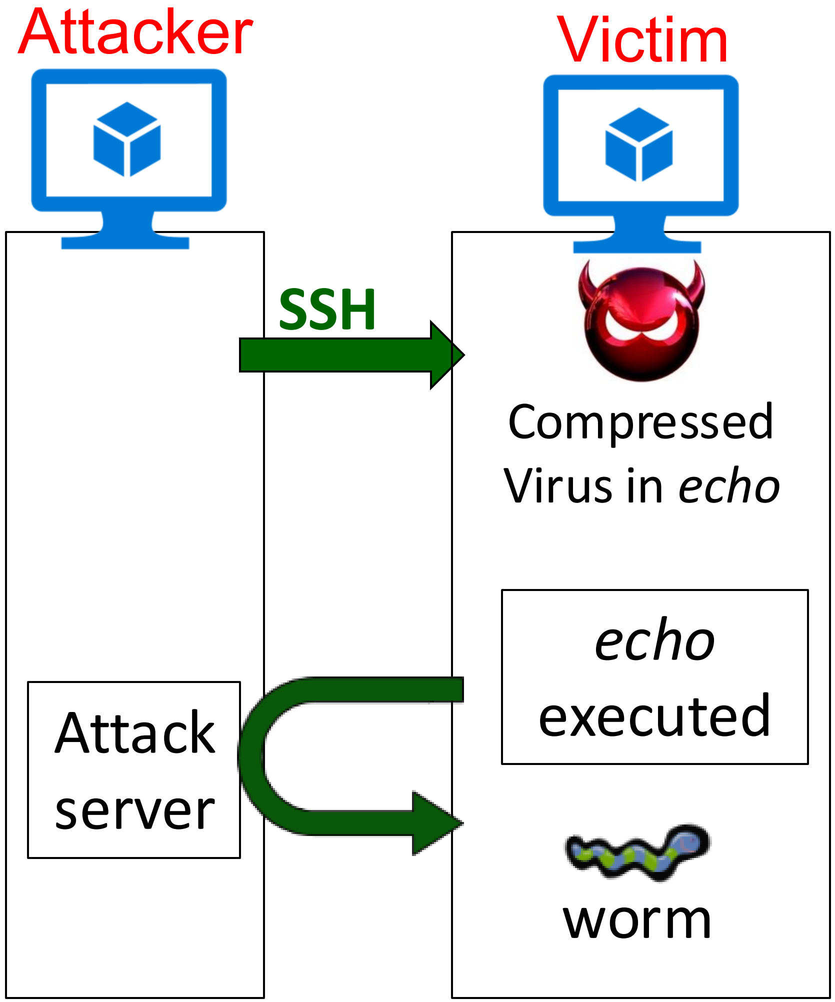
Goals

- Understand how a ransomware propagates and executes

- You will learn about the operation of
  - Dictionary attacks
  - Ciphering and deciphering
  - Digital signature
  - Compressed viruses
  - Worm propagation
  - Ransomware

Attack Scenario

- You are going to play the role of an attacker
- Assume that you know the IP of the victim and the username of his/her SSH account, you are asked to
  - Crack the victim's SSH password
  - Install a compressed virus in an affected program with attacker's signature generated by quantum safe algorithm
  - (Virus payload) Download and trigger a ransomware worm
- Consider the affected program: /app/echo in victim.
  - When it is running, the virus payload is executed



Three Tasks

- Task I: Crack SSH password (30%)
- Task II: Create a compression virus with the propagation of the ransomware worm (40%)
- Task III: Prepare the ransomware payload (30%)

Task I: Crack SSH password

- Cracking the victim's password by launching a dictionary attack
  - Assume that the victim's username is known as csc2025
  - Assume that the password is created based on the victim's personal information
    - A file including the victim's personal information: /app/victim.dat
      - One row contains an information entry
    - The password is composed of one or few information entries

- Hints
  - Try strings combination in Python: **itertools**
  - Automatic SSH and SFTP operation in Python: **paramiko**

Task II: Compression Virus with Ransomware Propagation

- Infect /app/echo in victim by embedding your compression virus
- Infected ‘echo’ shall
  - Keep the same size as the original ‘echo’
    - The original ‘echo’ shall be compressed
  - Contain the virus payload and the functionality of the original ‘echo’
  - Finish the execution of the payload before the end of the ‘echo’ execution
- The virus payload shall
  - Fetch a ransomware worm from the attack server
  - Execute the ransomware worm

Task II: Compression Virus with Ransomware Propagation

- Requirements
  - The infection cannot leave any files except the infected ‘echo’ on the victim container
  - The last 512 bytes of the infected ‘echo’ should be the signature generated by Dilithium3 algorithm
  - The size of infected ‘echo’ should be identical to the original one
- Hints
  - Compress ‘echo’ using a compression algorithm
  - Minimize the virus size with various methods
    - e.g., using /dev/tcp/host/port to build tcp connections, gcc flags and strip
  - Execute a program using the exec() family
  - Use ‘openssl dgst’ to create & verify your signature

Task III: Ransomware Payload

- Two major actions
  - Encrypting all picture files with jpg subtype in /app/Pictures at the victim using AES
  - Show a graph indicating a message requesting ransom (check the `banner` file for the source)

- Requirements
  - Using a key to do AES encryption & decryption
- Hints
  - Sample codes for AES encryption & decryption

Requirements

- You need to develop/run your program in the given Dockerfile
  - Resource is provided in /app
    - username/password: csc2025/csc2025
    - Note: the victim's password will be changed based on a new victim.dat file in the demo
  - Please complete your project under the path: /home/csc2025/${yourstudentID}
- You are allowed to use C/C++, Shell Script or/and Python
- You are allowed to team up; each team has at most 2 students
  - Teams: discussions are allowed, but no collaboration
- Please submit your source codes to E3
- Please email your questions to csc2025@nemslab.tw

Important: How to set up environment?

- Build the project3 image
  - Linux: “ docker compose build ”
- Create the project3 containers
  - Linux: “ docker compose up -d ”
- Enter the attacker container
  - Linux: “ docker exec -it attacker bash ”
- Enter the victim container
  - Linux: “ docker exec -it victim bash ”

Important: How to Prepare Your Attack Programs?

- Must provide a **Makefile** which compiles your source codes into at least two executable files: **crack_attack** and **attack_server**

- Test requirements for your program
  - Must be run in the given Dockerfile without any additional tools or libraries
  - Must work for the following two test commands
    - ./crack_attack <Victim IP> <Attacker IP> <Attacker port>
    - ./attack_server <Attacker port>

Important: Major Demo Steps (Not Exactly the Same)

- Attacker container
  - Run “make“ to compile your source codes
  - Run “./attacker_server <Attacker port>” to set up the attacker server
  - Run “./crack_attack <Victim IP> <Attacker IP> <Attacker port>” to crack the victim’s password and infect ‘echo’ in victim
- Victim container
  - Check the size of ‘echo’ and any additional files generated
  - Run 2 or 3 commands of ‘echo’
    - ‘echo’ shall perform its original function
    - Only the jpg files in /app/Pictures are encrypted with the given security context
    - A ransom graph shall show up
  - Check whether the encrypted files can be decrypted
- Note: no Internet access for both attacker and victim container

Project Submission

- Due date: 5/14 11:55 PM (Late submissions will not be accepted)
- Makeup submission (75 points at most): After the final
- Submission rules
  - Put all your files into a directory and name it using your student ID(s)
    - If your team has two members, please concatenate your IDs separated by "-"
  - Zip the directory and upload the zip file to E3
  - A sample of the zip file: 1234567-7654321.zip

```
1234567-7654321.zip
└── 1234567-7654321 (dir)
    ├── Makefile
    ├── crack_attack.c
    ├── attack_server.c
    └── ...
```

- If your files are not in a directory after unzip, 10 points will be deducted

Appendix: Quantum-Safe Cryptos (QSC)

- Conventional cryptographic algorithm bases on some mathematic problems that is difficult to be cracked via modern computers
- Quantum computers can use Shor's algorithm to solve these kinds of problems efficiently, thus threatens several crypto algorithms (e.g., RSA, ECDSA, ECDHE, …)
- New algorithms that can resist this kind of threats are called quantum-safe cryptos

Appendix: QSC in Practice

- OpenSSL introduces provider mechanism to embrace algorithms from third-parties
- Open Quantum Safe Project implements algorithms and adapt them with the provider interface into OpenSSL framework
- Check oqs-provider / USAGE.md to know its usage
- You can check the availability of QSC in your container with the command:

```
$ openssl list -providers
Providers:
oqsprovider
    name: OpenSSL OQS Provider
    status: active
```

Questions?