Computer Security Capstone

Project III: Ransomware Propagation and Payload

Chi-Yu Li (2022 Spring)
Computer Science Department
National Yang Ming Chiao Tung University

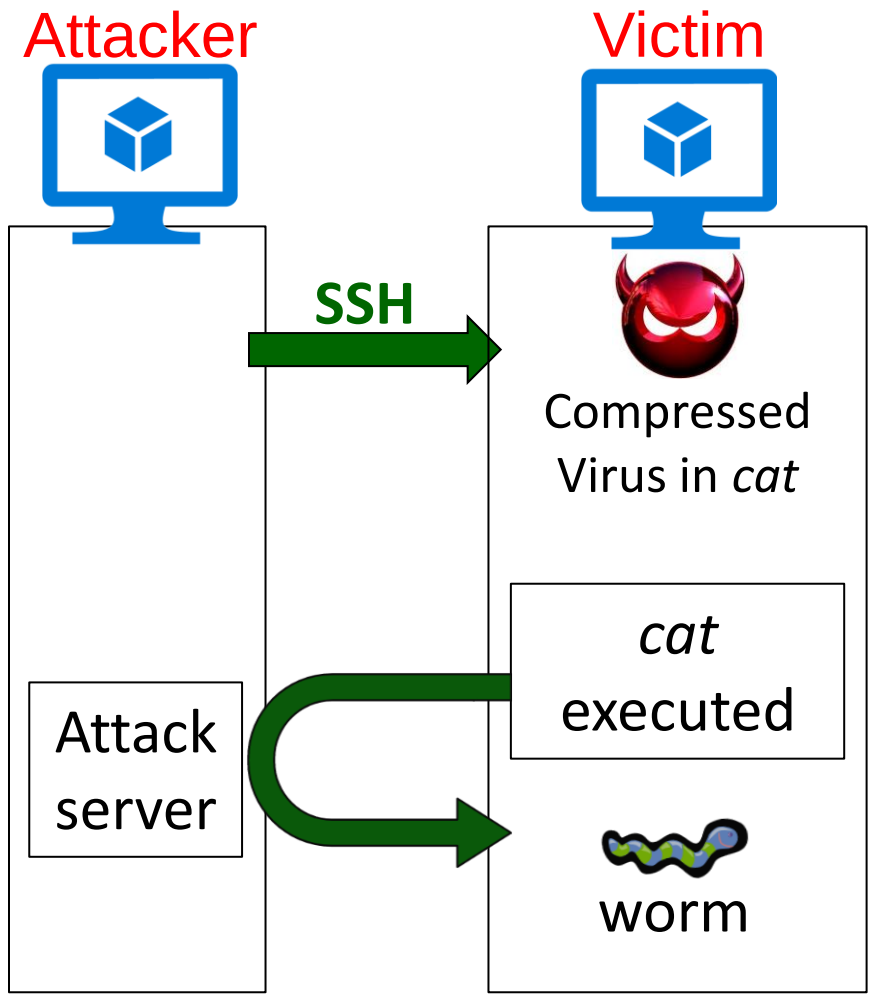
Goals

- Understand how a ransomware propagates and executes

- You will learn about the operation of
  - dictionary attacks
  - ciphering and deciphering
  - 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
  - (virus payload) download and trigger a ransomware worm
- Consider the affected program: /home/csc2022/cat
  - When it is run, 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 csc2022
  - Assume that the password is created based on the victim's personal information
    - A file including the victim's personal information is given: /home/csc2022/materials/victim.dat
    - Note: the password is composed of one or few information entries

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

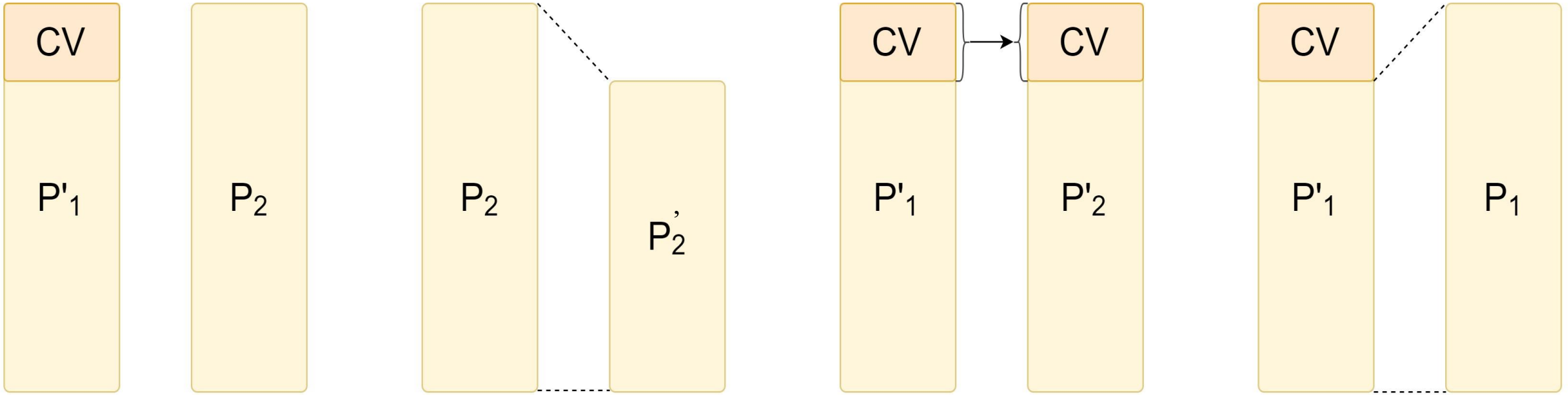
A Simple Virus

- This virus code, V, is prepended to infected programs
  - Assume that the entry point to the program is the main action block

- However, it is easily detected. Why?

```
program V
1234567;

procedure attach-to-program;
begin
        repeat
                file := get-random-program;
        until first-program-line ≠ 1234567;
        prepend V to file;
end;

procedure execute-payload;
begin
        (* perform payload actions *)
end;

procedure trigger-condition;
begin
        (* return true if trigger condition is true *)
end;

begin (* main action block *)
        attach-to-program;
        if trigger-condition then execute-payload;
        goto original program code;
end;
```

A Compression Virus



t_0: P'_1 is infected version of P_1; P_2 is clean

t_1: P_2 is compressed into P'_2

t_2: CV attaches itself to P'_2

t_3: P'_1 is decompressed into the original program P_1

Task II: Compression Virus with Ransomware Propagation

- Infect /home/csc2022/cat by embedding your compression virus
- Infected ‘cat’ shall
  - keep the same size as the original ‘cat’
    - The original ‘cat’ shall be compressed
  - contain the virus payload and the functionality of the original ‘cat’
  - finish the execution of the payload before the end of the ‘cat’ 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 ‘cat’ on the victim machine
  - Including 0xdeadbeaf in the last 4 bytes of the infected ‘cat’ as your signature
  - You can check the last bytes of a file with xxd

```
$ xxd cat | tail -n 1
000088f0: 2024 0000 afbe adde
```

- Hints
  - Compressing ‘cat’ using pack or zip
  - Minimizing the virus size with various methods
    - e.g., using /dev/tcp/host/port to build tcp connections, gcc flags and strip
  - Executing a program using the exec() family

Task III: Ransomware Payload

- Two major actions
  - Encrypting all picture files in jpg in /home/csc2022/Pictures using RSA
  - Popping up a window showing a message requesting ransom

- Requirements
  - Using a public key (*n* & *e*) for the RSA encryption:
    (22291846172619859445381409012451 & 65535)
- Hints
  - Sample codes for RSA encryption/decryption in /home/csc2022/materials/

Requirements

- You need to develop/run your program in given VMs
  - VM image: Press Me
    - username/password: csc2022/csc2022
    - Note: Virtualbox network setting: Host-only Adapter (through File -> Host Network Manager)
    - Note: the victim's password will be changed based on a new victim.dat in the demo
- 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 csc2022@nems.cs.nctu.edu.tw

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 VM 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 VM
  - 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 ‘cat’ in /home/csc2022/
- Victim VM
  - Check the size of ‘cat’ and any additional files generated
  - Run 2 or 3 commands of ‘cat’
    - ‘cat’ shall perform its original function
    - Only the jpg files in /home/csc2022/Pictures are encrypted with the given security context
    - A ransom window shall pop up
  - Check whether the encrypted files can be decrypted
- Note: no Internet access for both attacker and victim VMs

Project Submission

- Due date: 5/17 11:55pm
- Makeup submission (75 points at most): TBA (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 New e3
  - A sample of the zip file: 1234567-7654321.zip
    - 1234567-7654321
      - Makefile
      - crack_attack.c
      - attack_server.c
      - …

Questions?