Ransomware Simulation Program

By Derell Facey

Project Overview:

Ransomware is a type of malicious software that is designed to block access to data by either encrypting critical data or locking an account/devices, the actor then demands a payment in exchange for restoring access. This project is a java program that was programmed to mimic the behavior of ransomware. Additionally, the program is meant to simulate 2 different scenarios to educate the user on how to respond in the event of a ransomware attack. ***BE ADVISED, THE PROGRAM IS TO BE EXECUTED IN AN ISOLATED NETWORK ENVIRONMENT***.

Tools:

- Eclipse
- Java
- VMware
- 1x Windows 11 VM

---

1) About the Program

The program primarily utilizes the 2 methods, `encryptFile()` and `restoreFromBackup()`. The `TARGET_FOLDER` is specified to contain the "ransomware attack" to a predefined directory containing synthetic data. Upon executing, `RANSOM_NOTE` is created inside the directory:

```
private static final String TARGET_FOLDER = "C:\\Users\\VICTIM\\Desktop\\test";
private static final String RANSOM_NOTE = "README_RANSOM NOTE.txt";
```

When `encryptFile()` is executed, it first creates a backup of the directory with the `backupFile()` method. The `encryptFile()` method reads the contents of the file and stores it. Next, it encodes the file data into Base64 format. The method then writes the encoded data into a file, then replaces the original file with the encoded file:

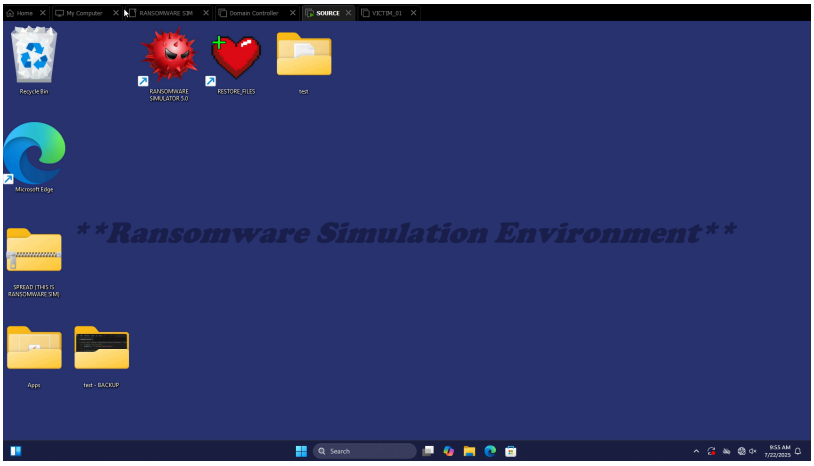
```
private static void encryptFile(File file)
{
    try
        {

        backupFile(file);
        //Reads the data of the file and stores it
        byte[] content = Files.readAllBytes( file.toPath() );

        //Encodes stored file data and saves in a string
        String encoded = Base64.getEncoder().encodeToString( content );

        //writes the encoded data into a file
        Files.write( file.toPath(), encoded.getBytes());

        //renames the file type to ".locked"
        Path encryptedPath = Paths.get(file.getParent(), file.getName() + ".locked");

        //Moves the encrypted file to the path of the original, replacing it
        Files.move(file.toPath(), encryptedPath);

        } catch (IOException e) {
```

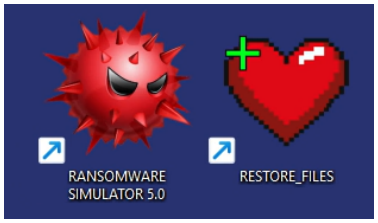
The other main method is `restoreFromBackup()`. When executed, the method identifies the original directory of `TARGET_FOLDER` and the backup folder created upon executing the program. The backup folder is then copied into the original directory, replacing corrupted or damaged files:

```
public static void restoreFromBackup()
{
File mainDir = new File(TARGET_FOLDER);
File backupDir = new File(TARGET_FOLDER, "backup");

for (File backupFile : backupDir.listFiles())
    {
    try
        {
        Path restorePath = Paths.get( mainDir.getAbsolutePath(), backupFile.getName() );
        Files.copy( backupFile.toPath(), restorePath,  StandardCopyOption.REPLACE_EXISTING );

        } catch (IOException e) {

        JOptionPane.showMessageDialog( null, "Failed to restore backup." );
        }
```

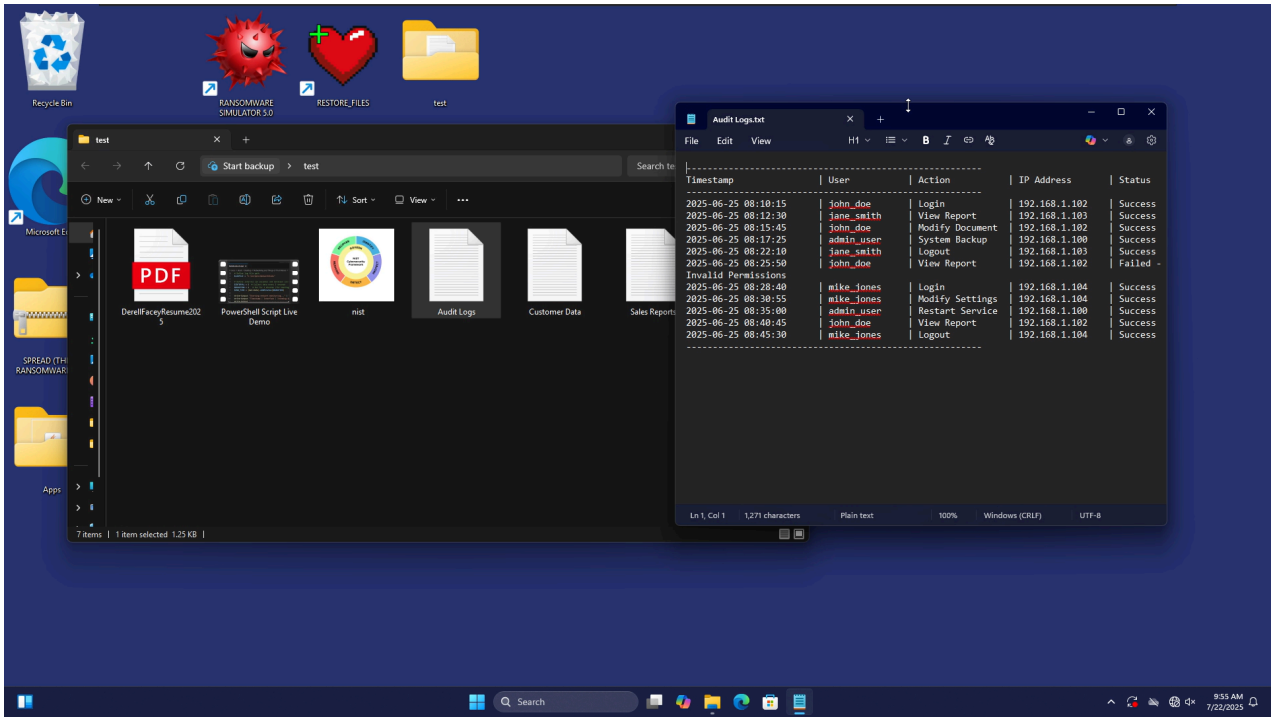
2) Setting up Simulation Environment

The Ransomware Simulation program was imported onto a virtual machine in a private VLAN. The VM connected to a host-only network, isolating the simulation environment from external networks and/or devices. The Ransomware Simulation program is accompanied by a File Restoration program in the event of a crash:





Inside the test folder are files containing synthetic data, meant to emulate critical data found in a bank environment. Additionally, there are non-bank files that serve to demonstrate the effects of the program with different file extensions:

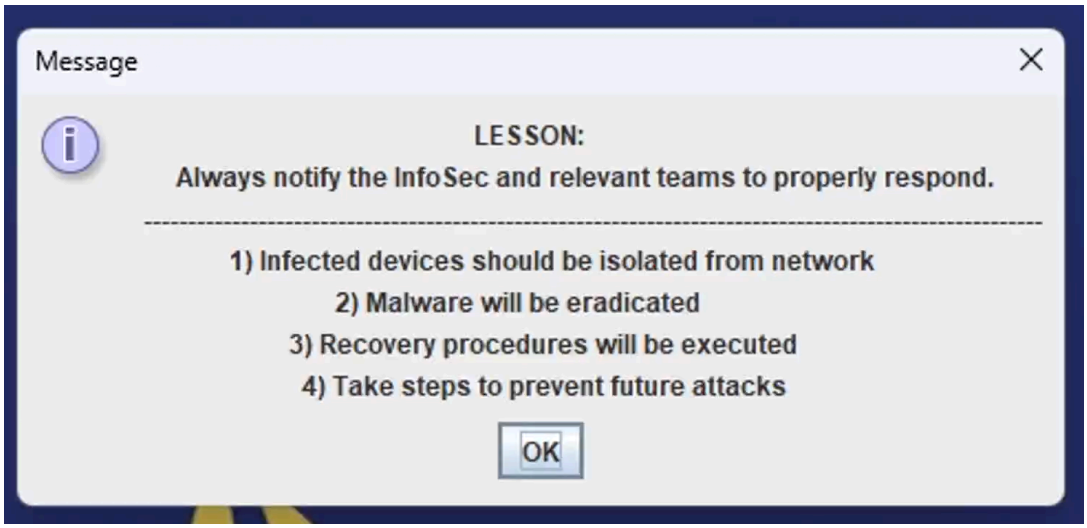


3) Simulation - (*Demonstration video: https://youtu.be/am6liW0fF9w)*

The program is designed to educate users on how to properly respond in the event of a ransomware attack with 2 main scenarios. *Scenario 1* simulates when a user decides to make the ransomware payment. *Scenario 2* simulates when a user instead decides to enter the decryption code, "INTERN", to emulate the deployment of proper restoration procedures. The demonstration video is linked above.

Scenario 1 Lesson:

Scenario 2 Lesson:



Potential Improvements:

- Implement a timer to simulate the urgent atmosphere of a ransomware attack.
- Implement a way for users to solve the decryption code, instead of giving it to them.
- Improve GUI