Read_Me.docx

by Damith Rathnayaka

Submission date: 29-Oct-2023 02:40PM (UTC-0400)
Submission ID: 2210717746
File name: 19243_Damith_Rathnayaka_Read_Me_880130_1268757835.docx (577.9K)
Word count: 545
Character count: 3431

Project's Title

The objective of this project is to implement the simulator for Smart Home based on the design proposal from Unit 3 of the Secure Systems Architecture module. This README file will guide you through the features and functionalities of the application.

Question

How do the mitigation strategies proposed in the document ensure the continuous availability of smart home services and functionalities in the face of various cyber-attacks?

Argument -

- Feasibility - Testing availability involves simulating cyber-attacks and observing whether the system remains operational.
- Measurability - The success of the mitigation strategies can be measured by the time period of the system services and the user's ability to access and use them during the simulation of attacks.
- Practical Relevance - Ensuring that smart home services remain available during cyber-attacks is crucial for user experience and the practical utility of smart home technologies.

Category	Description
Components	**Smart Devices (Client Devices):** Simulate various Smart home Devices such as thermostats, lights, and security cameras. **Controller (Hub):** A central controller that manages and communicates with all Smart devices.
Communication	**Protocol:** A secure communication protocol to interact between devices and the controller. **Encryption:** Implement encryption to secure the messages exchanged between devices and the controller.
Mitigation Strategies	**Authentication:** Ensure that the devices and controller authenticate each other. **Rate Limiting**: Implement rate limiting to prevent DDoS attacks.
Failure and Recovery	**Redundancy:** Implement redundancy to ensure system availability in case of a component failure. **Auto-Recovery:** Implement mechanisms for automatic recovery in case of network failures or device malfunctions.
Experiments	**Network Failure:** Simulate network failures and observe the system's resilience and recovery. **Cyber-Attack:** Simulate different cyber-attacks like DDoS to test the system's robustness and availability.

Figure 1: Model Specification

Project Description and Program functions

- Classes: Three main classes are defined: Vulnerability, Attack, and Mitigation.

  1) Vulnerability represents individual security vulnerabilities.
  2) Attack represents attacks that consist of multiple vulnerabilities.
  3) Mitigation represents mitigation strategies applied to attacks to reduce vulnerability risks.
  4) Objects creation: Instances of vulnerabilities, attacks, and mitigations are created with specific attributes.

- Risk calculation: The remaining overall risk of the attack is calculated after applying all mitigations, and the result is printed.

- SmartDevice and controller Classes: These classes simulate the operation of a smart home environment, including device registration, authentication, and encrypted communication.

```python
class SmartDevice:
    def __init__(self, device_id, controller):
        self.device_id = device_id
        self.controller = controller
        self.is_authenticated = False

    def authenticate(self, secret_key):
        hashed_key = hashlib.sha256(secret_key.encode()).hexdigest()
        if hashed_key == "expected_hashed_key":
            self.is_authenticated = True
            print(f"Device {self.device_id} authenticated.")

    def send_status(self, status):
        if self.is_authenticated:
            encrypted_status = hashlib.md5(status.encode()).hexdigest()
            success = random.random() > 0.1  # 10% chance of message loss
            if success:
                time.sleep(random.uniform(0, 2))  # Simulate latency
                self.controller.receive_status(self.device_id, encrypted_status)
            else:
                print(f"Device {self.device_id} status message lost.")

    def receive_command(self, encrypted_command):
        if self.is_authenticated:
            decrypted_command = encrypted_command  # Add decryption logic here
            print(f"Device {self.device_id} received command: {decrypted_command}")

class Controller:
    def __init__(self):
        self.devices = {}

    def register_device(self, device, secret_key):
        self.devices[device.device_id] = device
        device.authenticate(secret_key)

    def receive_status(self, device_id, encrypted_status):
        decrypted_status = encrypted_status  # Add decryption logic here
        print(f"Controller received status from Device {device_id}: {decrypted_status}")

    def send_command(self, device_id, command):
        device = self.devices.get(device_id)
        if device:
            encrypted_command = hashlib.md5(command.encode()).hexdigest()
            device.receive_command(encrypted_command)
```

Figure 2: Class file for Client side

- Latency and message Loss: These aspects are simulated within the send_status method of the SmartDevice class, adding realism to the communication between devices and the controller.

- Main simulation code: This part of the code creates instances of the classes, simulates the operation of the smart home environment, applies mitigation strategies, and evaluates the remaining risks.

Detailed Design:

Category	Description
Smart Devices (Python)	Attributes: Device_ID, Status, Last_Communication_Time Methods: Send_Status, Receive_Command
Controller (Python)	Attributes: Registered_Devices, Command_Queue. Methods: Send_Command, Receive_Status, Authenticate_Device
Communication	Implement a secure messaging protocol using encryption for message exchanges.
Mitigation Strategies	Implement authentication mechanisms and rate-limiting in the communication protocol.
Failure and Recovery	Implement redundancy and auto-recovery mechanisms to handle network failures and device malfunctions.

Prerequisites

This project has been developed in PyCharm.

1. PyCharm Professional is needed to run the project.

2. Go to [PyCharm](https://www.jetbrains.com/pycharm/download/) (JetBrains, 2023b) to install PyCharm Professional; select "Free 30-day trial" or "Pay for subscription".

3. PyCharm website will automatically detect your Operating System type and provide a suitable link to download; alternatively, choose between Windows, macOS and Linux.

4. Before the download, select the location folder to save the file.

5. Once downloaded, click on "Install" to install PyCharm program.

How to install libraries

- Go to File > Settings > Project: main.py > Python Interpreter > + > Install package > to install the following packages:

- **datetime:** A library supplying classes to work with time and date - an extract for time, date and time intervals is provided, and date/time object is created (PSF, 2023c).

- **hashlib:** A library providing a common interface to hash and message algorithms, e.g., SHA256, SHA512, MD5, etc. (PSF, 2023d).

How to run the "main" program

1. Click on the “**Run**” in the dropdown navigation and click on “**Run Main**” as seen below:

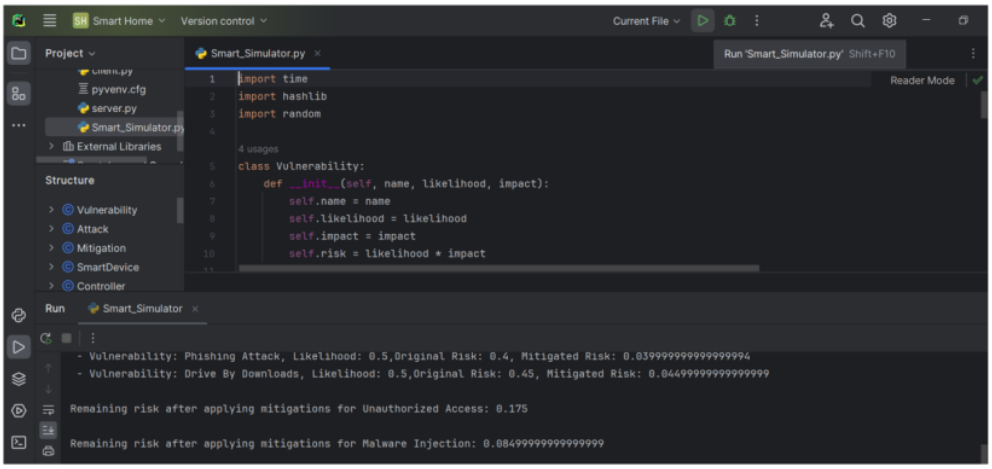

Figure 3: Program Window

Output from demonstrations

Impact score is generated based on the assumptions.

```
SIMULATION for Unauthorized Access

Applying Mitigation: Strong Password Policy to Attack: Unauthorized Access
 - Vulnerability: Brute Force, Likelihood: 0.25,Original Risk: 0.2, Mitigated Risk: 0.039999999999999994
 - Vulnerability: Credential Theft, Likelihood: 0.75,Original Risk: 0.675, Mitigated Risk: 0.13499999999999998

SIMULATION for Malware Injection

Applying Mitigation: Using an Intrusion Detection System to Attack: Malware Injection
 - Vulnerability: Phishing Attack, Likelihood: 0.5,Original Risk: 0.4, Mitigated Risk: 0.039999999999999994
 - Vulnerability: Drive By Downloads, Likelihood: 0.5,Original Risk: 0.45, Mitigated Risk: 0.04499999999999999

Remaining risk after applying mitigations for Unauthorized Access: 0.175

Remaining risk after applying mitigations for Malware Injection: 0.08499999999999999
```

Figure 4: Test Result

Read_Me.docx

ORIGINALITY REPORT

0%	0%	0%	0%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	Off		