Custom Threat Emulation Platform in Python Report

Project:
Submitted to: Ms.Selin Tor
Submitted by: Ojas Gaur
Date: 27-05-2025

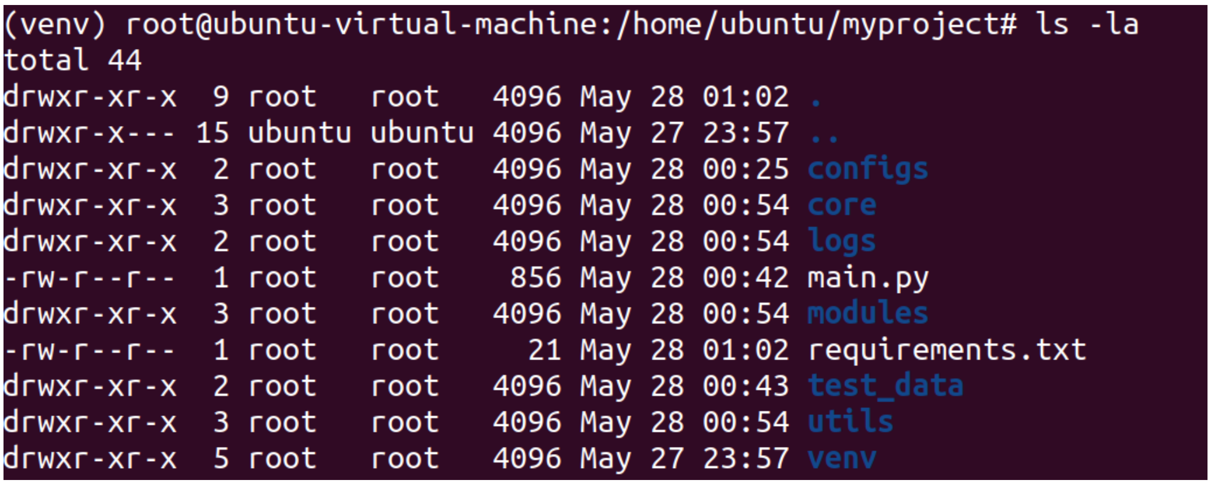
1. Introduction

This report outlines the design and implementation of a modular red team emulation tool developed in Python. The platform simulates attacker techniques such as data exfiltration, credential dumping, and local privilege escalation. It leverages a YAML-based configuration system to enable dynamic selection of techniques at runtime, offering flexibility and control for repeated security testing scenarios.

2. System Architecture

The emulation platform was built with a modular architecture consisting of the following components:

- **main.py**: The core execution controller.
- **/configs/**: Holds YAML configuration templates.
- **/modules/**: Contains individual simulation techniques.
- **/core/**: Loads and executes technique modules.
- **/utils/**: Manages centralized logging.
- **/logs/**: Stores generated red and blue team activity logs.
- **/test_data/**: Contains sample files used for exfiltration scenarios.



```
(venv) root@ubuntu-virtual-machine:/home/ubuntu/myproject# ls -la
total 44
drwxr-xr-x  9 root   root   4096 May 28 01:02 .
drwxr-x--- 15 ubuntu ubuntu 4096 May 27 23:57 ..
drwxr-xr-x  2 root   root   4096 May 28 00:25 configs
drwxr-xr-x  3 root   root   4096 May 28 00:54 core
drwxr-xr-x  2 root   root   4096 May 28 00:54 logs
-rw-r--r--  1 root   root    856 May 28 00:42 main.py
drwxr-xr-x  3 root   root   4096 May 28 00:54 modules
-rw-r--r--  1 root   root     21 May 28 01:02 requirements.txt
drwxr-xr-x  2 root   root   4096 May 28 00:43 test_data
drwxr-xr-x  3 root   root   4096 May 28 00:54 utils
drwxr-xr-x  5 root   root   4096 May 27 23:57 venv
```

Each module operates independently and logs actions in two files:

1. Attack.log (offensive activity)
2. Defender.log (simulated detection).

3. Objective

- Implement a red team simulation framework in Python.
- Support dynamic technique selection through YAML templates.
- Simulate realistic attacker behaviors for testing detection capabilities.
- Generate structured logs for correlation analysis by blue teams.

4. Tools Used

Tool	**Purpose**
Python 3.10+	Core development language
PyYAML	Parsing YAML configuration files
Logging (lib)	Log attacker actions and detection alerts
Bash/Linux	Development environment and test runs

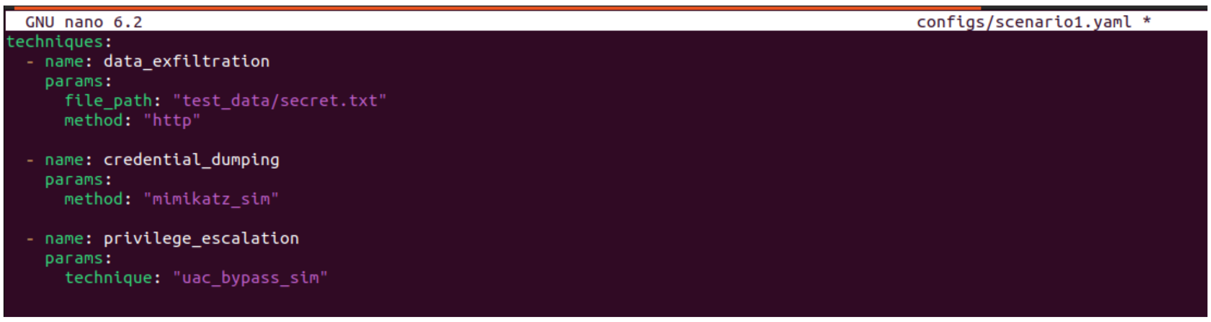
5. Methodology

5.1. Environment Setup

- Python virtual environment initialized
- Directory structure created with modules, configs, and logs
- Dependencies installed using pip install pyyaml

5.2. Configuration Template

A YAML file defines the technique sequence and parameters:

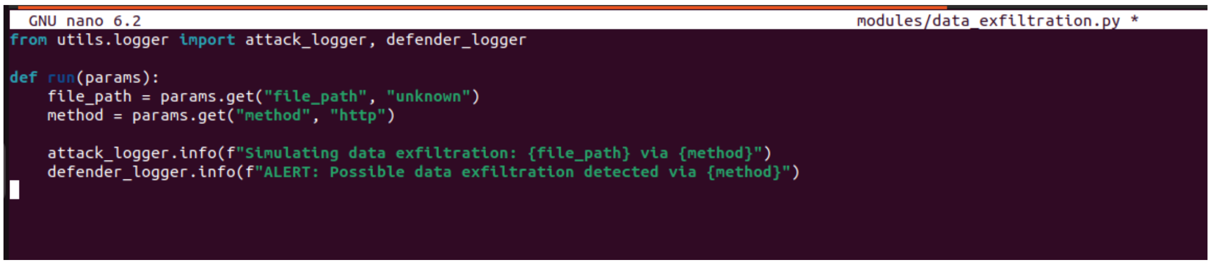


```
  GNU nano 6.2                                                               configs/scenario1.yaml *
techniques:
  - name: data_exfiltration
    params:
      file_path: "test_data/secret.txt"
      method: "http"

  - name: credential_dumping
    params:
      method: "mimikatz_sim"

  - name: privilege_escalation
    params:
      technique: "uac_bypass_sim"
```

5.3. Technique Modules

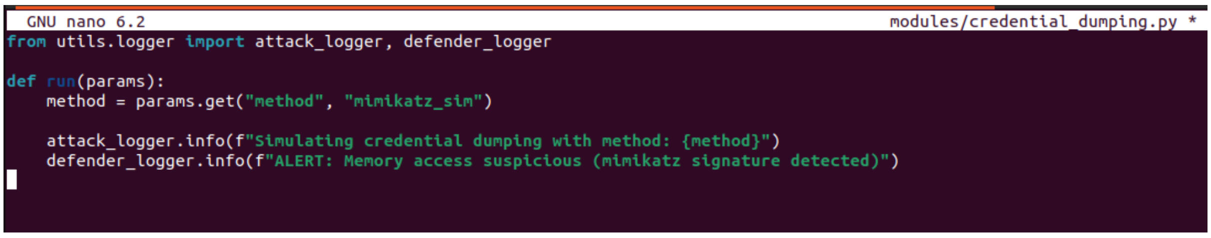
Each simulation technique is implemented in a separate module:

- data_exfiltration.py: Simulates file exfiltration via HTTP



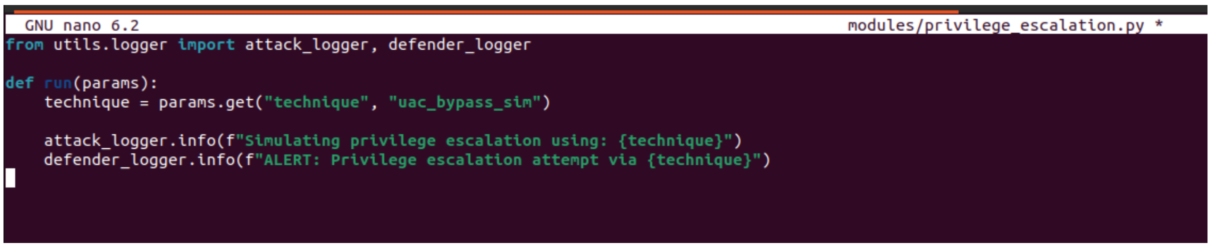
```
  GNU nano 6.2                                                          modules/data_exfiltration.py *
from utils.logger import attack_logger, defender_logger

def run(params):
    file_path = params.get("file_path", "unknown")
    method = params.get("method", "http")

    attack_logger.info(f"Simulating data exfiltration: {file_path} via {method}")
    defender_logger.info(f"ALERT: Possible data exfiltration detected via {method}")
```

- credential_dumping.py: Simulates credential theft using memory access



```
  GNU nano 6.2                                                               modules/credential_dumping.py *
from utils.logger import attack_logger, defender_logger

def run(params):
    method = params.get("method", "mimikatz_sim")

    attack_logger.info(f"Simulating credential dumping with method: {method}")
    defender_logger.info(f"ALERT: Memory access suspicious (mimikatz signature detected)")
```

- privilege_escalation.py: Simulates escalation using a UAC bypass method



```
  GNU nano 6.2                                                        modules/privilege_escalation.py *
from utils.logger import attack_logger, defender_logger

def run(params):
    technique = params.get("technique", "uac_bypass_sim")

    attack_logger.info(f"Simulating privilege escalation using: {technique}")
    defender_logger.info(f"ALERT: Privilege escalation attempt via {technique}")
```

5.4. Execution

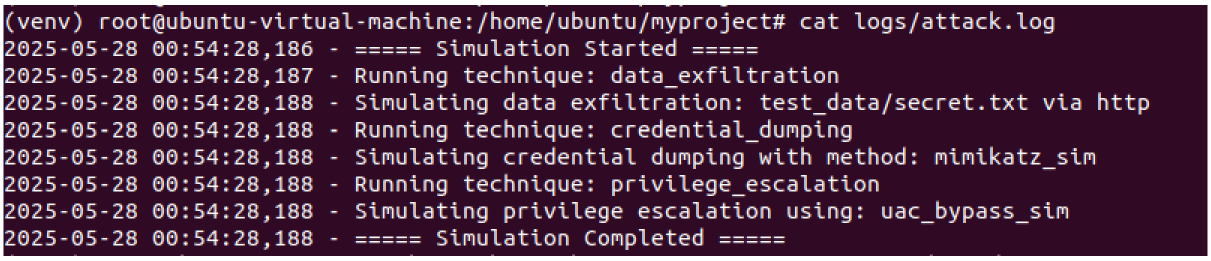
The tool is executed using:

python main.py --config configs/scenario1.yaml

Each technique is executed in order as defined in the YAML file, and logs are recorded.

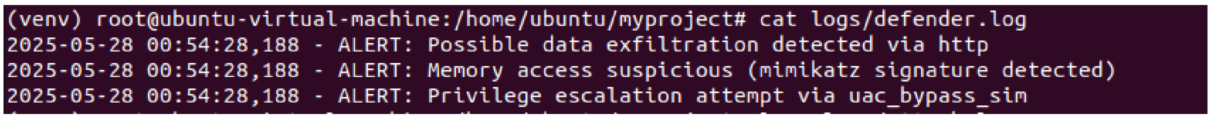
6. Results and Findings

Attack Simulation Log (logs/attack.log)



```
(venv) root@ubuntu-virtual-machine:/home/ubuntu/myproject# cat logs/attack.log
2025-05-28 00:54:28,186 - ===== Simulation Started =====
2025-05-28 00:54:28,187 - Running technique: data_exfiltration
2025-05-28 00:54:28,188 - Simulating data exfiltration: test_data/secret.txt via http
2025-05-28 00:54:28,188 - Running technique: credential_dumping
2025-05-28 00:54:28,188 - Simulating credential dumping with method: mimikatz_sim
2025-05-28 00:54:28,188 - Running technique: privilege_escalation
2025-05-28 00:54:28,188 - Simulating privilege escalation using: uac_bypass_sim
2025-05-28 00:54:28,188 - ===== Simulation Completed =====
```

Defensive Detection Log (logs/defender.log)



```
(venv) root@ubuntu-virtual-machine:/home/ubuntu/myproject# cat logs/defender.log
2025-05-28 00:54:28,188 - ALERT: Possible data exfiltration detected via http
2025-05-28 00:54:28,188 - ALERT: Memory access suspicious (mimikatz signature detected)
2025-05-28 00:54:28,188 - ALERT: Privilege escalation attempt via uac_bypass_sim
```

Logs validate that the emulation ran successfully and each technique was simulated and recorded.

7. Recommendations

- Extend support to additional techniques such as persistence and lateral movement.
- Integrate output with SIEM platforms for real-time detection testing.

- Utilize this platform as a base for red vs. blue team exercises.

8. Conclusion

The threat emulation platform meets all specified objectives. It supports modular simulation of common attack techniques and provides clear red/blue team logging. This implementation can be extended further and used as a repeatable, configurable tool for improving detection engineering and security control validation.