Section: **PUBP**
Project Title: **Enterprise-Grade Security on an SME Budget: Lightweight IDS with Suricata**
Student Name: **Grant Fitzgerald**

**Problem Statement:**

Small and medium-sized enterprises (SMEs) face mounting cybersecurity threats – including ransomware, credential theft, and web application exploitation – but often lack the financial and technical resources to implement traditional enterprise-grade intrusion-detection systems (IDS). Per the 2025 Verizon Data Breach Investigations Report (DBIR), 88% of ransomware incidents targeting small businesses result in successful breaches, underscoring a critical gap in prevention capabilities. Industry literature, including the Ponemon SMB study, reinforces that SMEs struggle with vendor security oversight and timely breach response. These findings highlight an urgent need for affordable, accessible IDS solutions tailored to this demographic.

**Solution Statement:**

This research aims to design and prototype a lightweight, open-source intrusion detection system (IDS) using Suricata to equip SMEs with an affordable, user-friendly cybersecurity option. Key deliverables include a Suricata configuration optimized for low-resource environments, a natural language rule generator, a web-based GUI dashboard, and test scenarios that simulate real-world threat detection. Suricata was selected for its multithreaded architecture, native JSON logging, and prior exposure in my Spring Security Incident Response course – offering both technical advantages and educational continuity. This project addresses an imminent cybersecurity need while building meaningfully on a topic I found engaging in prior coursework but had limited opportunity to explore in depth.

**Completed Tasks (Last 2 Weeks):**

- Reviewed the 2023-2025 Verizon DBIRs and 2019 Ponemon SMB report to contextualize IDS relevance and SME breach trends
- Finalized the problem and solution statements based on instructor feedback, adding industry literature support
- Evaluated open-source IDS options and documented rationale for selecting Suricata over Snort and Zeek
- Installed and ran Suricata in test mode on a Kali Linux VM to verify initial rule processing and logging
- Cleaned up and terminated multiple redundant Suricata instances to regain control over system logging and resource usage
- Created a Suricata kill switch/validator script to ensure future rogue instances can easily be detected and stopped, improving operational control
- Manually cleared and reset the eve.json log file to recover from corrupted JSON entries and restore clean alert logging
- Verified and tested proper Suricata shutdown procedures, including systemctl, pkill, and process validation using ps, lsof, and tail
- Tested custom Suricata rule output and confirmed alert generation and file behavior using ICMP traffic and basic packet inspection
- Documented logging behavior and system interactions to inform future GUI dashboard integration and logging design

**Tasks for the Next Project Report:**

- Deploy Suricata in Live Mode on Bridged Interface
    - Enable promiscuous mode on your VM NIC
    - Confirm that Suricata captures and logs real-time traffic (e.g. web browsing, pings, port scans)
- Develop the First Iteration of Natural Language Rule Translator
    - Built to support basic phrases like:
      "alert on SSH from external IP" → generates Suricata rule syntax
    - Implement as Python script with hard-coded templates
- Design Initial GUI Dashboard Layout
    - Choose stack (likely Flask)
    - Create basic HTML wireframe with:
        - Natural language input field
        - Rule preview panel
        - Log viewer
- Create and Document Test Scenarios
    - Generate controlled test traffic: ICMP, port scans, SSH attempt
    - Capture logs and confirm that Suricata alert behavior matches expectations
- Define Draft Evaluation Metrics
    - Usability: rule creation speed, error tolerance
    - Detection: true positives and log clarity

**Questions I have or Issues I'm running into:**

- Suricata process management required some troubleshooting to get back on track. Multiple instances were not shut down correctly and ended up running simultaneously. This caused unexpected logging behavior and corruption in the eve.json file, which I was eventually able to resolve with a custom kill script.
- While I'm going to be relying on Python code for rule translation and GUI tasks, I'm not fluent in it. I do understand the logic at a high level (have C# scripting experience that carries over) and am actively learning but will need to tap into other resources to translate design ideas into working scripts.
- I'm still deciding on how detailed the GUI will be. There is going to be a balance between being able to show functionality of the tool and staying within reasonable scope for the time we have to work on this project.
- I am considering building a dedicated inline hardware setup (e.g. mini-PC), but I want to make sure that it aligns with time constraints and project requirements before committing the time and funds, although it would more closely reflect best practices for how this should be set up between the modem and router.

**Methodology Paragraph Summary:**

This project is going to follow an iterative development methodology that is based on real-world cybersecurity use cases. First, Suricata will be deployed in live mode in a bridged Linux environment to simulate SME network monitoring conditions. From there, a series of test scenarios – ICMP pings, SSH attempts, and port scans – will be used to verify rule accuracy and system responsiveness. A natural language rule translator will be developed using Python templates to simplify rule creation for non-technical users, and a lightweight web-based GUI will be built (likely in Flask) to allow rule input and alert viewing. Evaluation will be based on usability (ease of rule creation/user interaction) and detection accuracy (% of true/false positive rates under simulated conditions). All components will be unit tested as they're developed to ensure system stability.

**Timeline:**

*Enter tasks for every week of this semester. One task per row and only include tasks related to your project (for example, do not include "Peer Feedback" or "Progress Report"). You'll most likely have multiple tasks per week. When you submit your first progress report, we're expecting to see a timeline of all tasks related to your project for the full length of the semester. It's expected that the tasks will develop and get more detailed over time but you must start with something. Any task you list should be actionable (for example, do not have a task "continue research" or "working on xyz"). For the Status column, let us know if you've completed the task, if it's still in-progress, maybe cancelled or whatever the status is.*

| **Week #** | **Description of Task** | **Status** |
|---|---|---|
| W1 (May 19-25) | Review DBIR and Ponemon literature for context | Complete |
| W1 | Select and justify Suricata over other open-source IDS tools | Complete |
| W1 | Finalize problem and solution statements | Complete |
| W1 | Install and run Suricata in test mode on Kali Linux | Complete |
| W1 | Investigate eve.json structure and Suricata logging behavior | Complete |
| W1 | Create kill switch and validator script for Suricata process management | Complete |
| W1 | Manually clear and verify Suricata logs and control instance spawning | Complete |
| W2 (May 26-June 1) | Begin drafting natural language rule mapping logic | In Progress |
| W2 | Deploy Suricata in live mode on bridged interface | In Progress |
| W2 | Develop first version of natural language rule parser (template-based) | Not Started |
| W2 | Choose GUI framework (Flask vs Node.js) and build basic layout skeleton | Not Started |
| W2 | Create test scenarios (e.g., port scan, SSH, ICMP) and verify alert logging | Not Started |
| W2 | Draft evaluation metrics for usability and detection accuracy | Not Started |
| W3 (Jun 2-8) | Refine rule translation logic and expand supported templates | Upcoming |
| W3 | Integrate rule engine with GUI rule input field | Upcoming |
| W3 | Implement log viewer component in GUI | Upcoming |
| W3 | Begin full-system test loop (traffic → alert → GUI) | Upcoming |
| W4 (June 9-15) | Begin writing final report introduction, methods, and design documentation | Upcoming |
| W4 | Collect performance metrics from traffic simulations | Upcoming |

| | | |
|---|---|---|
| W4 | Conduct usability testing or walkthrough with a peer | Upcoming |
| W5+ | TBD based on project pacing | Expandable |

**Evaluation:**

[Include any evaluation plans and/or results by Progress Report 4. This may expand as you finalize the report.]

**Report Outline:**

[Include an outline of your final report by Progress Report 4. This may expand as you finalize the report.]

**References:**

Verizon. (2025). *2025 Data Breach Investigations Report (DBIR)*. Retrieved from https://verizon.com/dbir

Verizon. (2024). *2024 Data Breach Investigations Report (DBIR)*. Retrieved from https://verizon.com/dbir

Verizon. (2023). *2023 Data Breach Investigations Report (DBIR)*. Retrieved from https://verizon.com/dbir

Ponemon Institute. (2019). *2019 Global State of Cybersecurity in Small and Medium-Sized Businesses (SMBs)*. Sponsored by Keeper Security.

OISF (Open Information Security Foundation). *Suricata Documentation*. Retrieved from https://suricata.io/docs

Emerging Threats. *Suricata Ruleset*. Retrieved from https://rules.emergingthreats.net
Suricata GitHub repository. https://github.com/OISF/suricata

## Appendix

*If there are notes, sources, figures you want to reference please include that here.*

### A. Custom Suricata Alert and Example Rule

Suricata Rule Used (custom.rules)

```
alert icmp any any -> any any (msg:"ICMP ping detected"; sid:1000001;
rev:1;)
```

*This rule generates an alert for any ICMP traffic, often used for testing with ping.*

Resulting eve.json Alert:

```
{
  "timestamp": "2025-05-23T23:32:28.510559-0400",
  "event_type": "alert",
  "src_ip": "108.157.142.105",
  "dest_ip": "192.168.6.134",
  "proto": "ICMP",
  "alert": {
    "signature_id": 1000001,
    "signature": "ICMP ping detected",
    "severity": 3
  }
}
```

*This log entry confirms that Suricata correctly loaded and triggered the custom rule during live traffic inspection on the eth0 interface.*

[The inclusion of draft work product should begin by Progress Report 3.]