Liam Calder V00963268
CSC 466
March 7 2025

Lightweight DDoS Alarm System:
Comparative Analysis of Detection Methods for Home Networks
Bi-Weekly Update 2 (Midterm Update)

Planned Deliverables for Mar 7 (Midterm Update):

- **Working Prototypes:**
  - Fully functional detection modules for threshold-based and anomaly-based (K-Means) detection integrated into a single Python tool.
- **Preliminary Experimental Data:**
  - Share initial test results and observations from synthetic DoS traffic experiments.

Extra Additions Made:

- **User-Configurable Settings:**
  - Added interactive prompts to allow users to input or auto-recommend threshold values.
  - Enabled customization of the monitoring window size (WINDOW_SIZE).
- **Enhanced Anomaly Detection:**
  - Updated the K-Means method to use a distance-to-centroid approach.
  - Introduced configurable parameters (n_clusters, distance_threshold, max_history) for tuning the anomaly detection.
  - Modified the system to update the anomaly detector only once per time window, ensuring a robust and gradual baseline build-up.
- **Improved Performance Logging:**
  - Detailed logs of attack type, start time, end time, duration, sources, and peaks for each attack are now stored in performance_metrics.txt.

Collected Performance Metrics

Below are selected logs that illustrate how the system recorded both UDP Flood and SYN Flood attacks. Each attack was configured to run for 30 seconds at 1000 packets/second.

```
DoS Alarm System Performance Metrics

=====================================

Log Entry Time: 2025-03-06 15:29:25

Attack Type: UDP Flood

Start Time: 2025-03-06 15:28:21

End Time: 2025-03-06 15:29:25

Duration: 63.95 seconds

Sources: xxx.x.xxx.xxx, xxx.xxx.xxx.xxx, xxx.xxx.xxx.xxx

Details: Peak SYN: 0, Peak UDP: 5664

-----------------------------------------------------

Log Entry Time: 2025-03-06 15:30:34

Attack Type: UDP Flood

Start Time: 2025-03-06 15:29:51

End Time: 2025-03-06 15:30:34

Duration: 43.05 seconds

Sources: xxx.x.x.xxx, xxx.xxx.xxx.xxx

Details: Peak SYN: 0, Peak UDP: 5703

-----------------------------------------------------

Log Entry Time: 2025-03-07 12:15:10

Attack Type: Anomaly

Start Time: 2025-03-07 12:14:09

End Time: 2025-03-07 12:15:10

Duration: 61.43 seconds

Detection Method: anomaly

Sources: xxx
```

```
Details: Peak SYN: 0, Peak UDP: 5532
--------------------------------------------------
Log Entry Time: 2025-03-07 12:19:37
Attack Type: Anomaly
Start Time: 2025-03-07 12:17:12
End Time: 2025-03-07 12:19:37
Duration: 144.99 seconds
Detection Method: anomaly
Sources: x,x,x,x,x,...
Details: Peak SYN: 3408, Peak UDP: 1
--------------------------------------------------
```

Note: IPs have been hidden for anonymity.

Despite each simulated attack running for 30 seconds, durations sometimes exceed 40 seconds. This “residual lag” often results from the detection window capturing post-attack traffic. Future refinements will tighten the window and better differentiate ongoing attacks from normal traffic fluctuations.

Console Alerts (Threshold & Anomaly)

Below are console logs that confirm the detections made by both the threshold-based and anomaly-based alarms.

DoS Attack Details (Console) (Synthetic attacks started by me)

```
Starting UDP flood on x
Attack started at 18:19:41.
Duration: 30 seconds
Packet rate: 1000 packets/second
UDP flood completed.
Attack ended at 18:20:11.
Starting SYN flood attack on x
Attack started at 18:21:43.
```

```
Duration: 30 seconds
Packet rate: 1000 packets/second
SYN flood completed.
Attack ended at 18:22:13.
Starting UDP flood on x
Attack started at 12:13:59.
Duration: 30 seconds
Packet rate: 1000 packets/second
UDP flood completed.
Attack ended at 12:14:29.
Starting SYN flood attack on x
Attack started at 12:17:02.
Duration: 30 seconds
Packet rate: 1000 packets/second
SYN flood completed.
Attack ended at 12:17:32.
```

What the alarm picked up:

```
=== ALERT ===
DoS Alert: UDP Flood Detected
Attack started at 18:19:42.
Source(s): xxx.x.x.xxx, xxx.xxx.xxx.xxx
==============
=== ALERT ===
DoS Alert: UDP Flood Ended
Attack ended at 18:20:16.
Duration: 34.30 seconds.
Source(s): xxx.x.x.xxx, xxx.xxx.xxx.xxx
==============
=== ALERT ===
DoS Alert: SYN Flood Detected
Attack started at 18:21:44.
Source(s): x,x,x,x,x…
==============
=== ALERT ===
DoS Alert: SYN Flood Ended
Attack ended at 18:22:34.
Duration: 50.27 seconds.
Source(s): x,x,x,x,x…
==============
=== ALERT ===
DoS Alert: Anomalous Traffic Detected
```

```
Anomalous traffic in last 10 seconds.
==============
=== ALERT ===
DoS Alert: Anomaly Ended
Attack ended at 12:15:10.
Duration: 61.43 seconds.
Source(s): 192.168.86.22, 192.168.56.1
==============
=== ALERT ===
DoS Alert: Anomalous Traffic Detected
Anomalous traffic in last 10 seconds.
==============
=== ALERT ===
DoS Alert: Anomaly Ended
Attack ended at 12:19:37.
Duration: 144.99 seconds.
Source(s): x,x,x,x,x…
==============
```

Analysis of Threshold-Based Detection

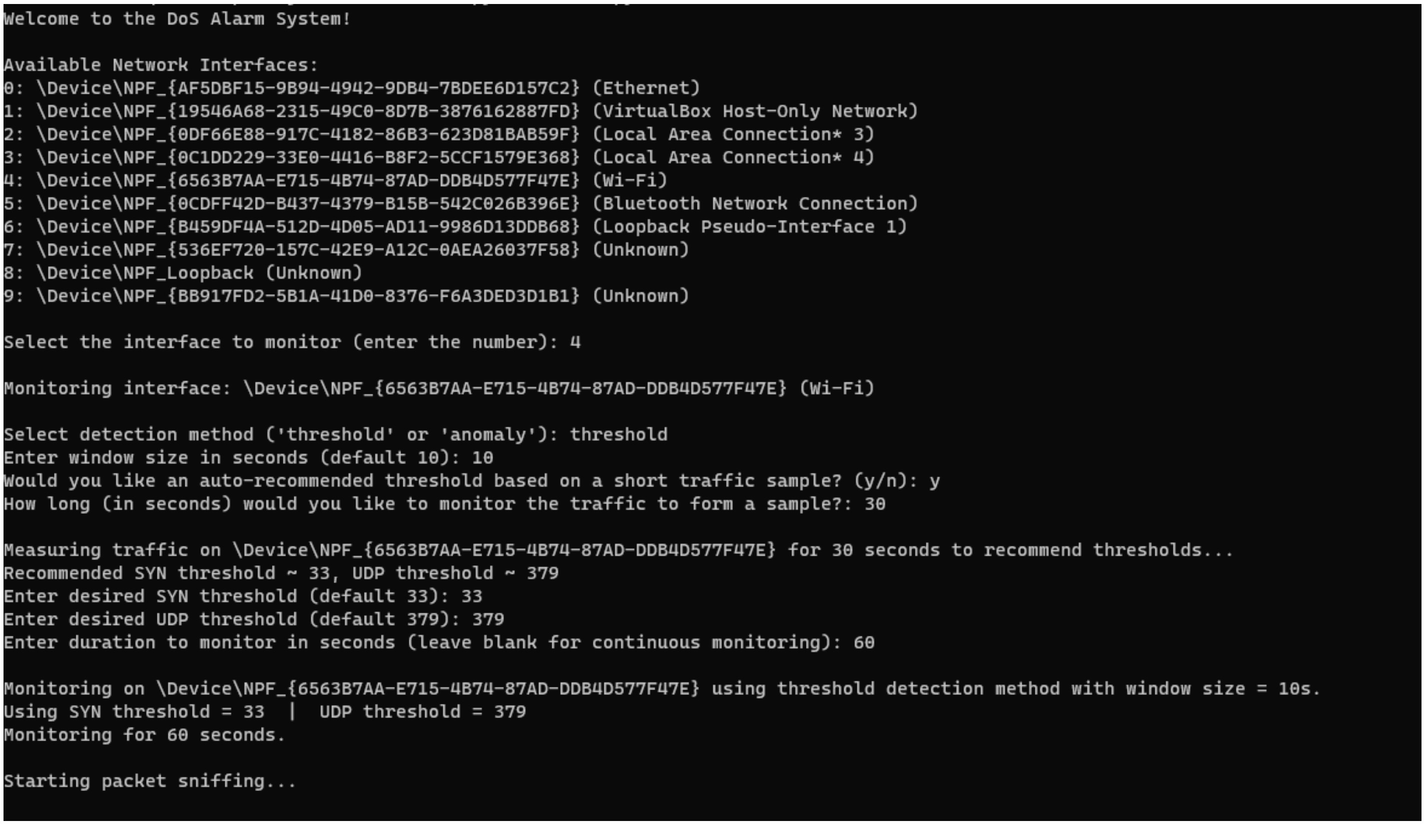
- **Rapid Detection:**
  The system raised an alert within one second of the UDP flood initiating (18:19:41 → 18:19:42) and did similarly well for the SYN flood. This confirms the threshold triggers are effective in quickly identifying abnormal surges.

- **Residual Lag:**
  Although attacks are designed to last 30 seconds, end-of-attack alerts recorded 34.30 seconds for UDP and 50.27 seconds for SYN. This suggests the system struggles to promptly recognize when an attack subsides, likely due to:

  1. Lingering traffic in the detection window after the actual attack ends.
  2. Conservative thresholds to prevent false negatives.

Future Considerations:
To address this residual lag, I plan to refine the criteria for ending alerts. Options include narrowing the detection window or adding logic to differentiate normal post-attack traffic from active attacks. Long-term, I aim to improve overall accuracy and expand the system's ability to switch seamlessly between threshold-based and anomaly-based detection in real-time.

The current alarm in action showcasing its function and customization:



```
Welcome to the DoS Alarm System!

Available Network Interfaces:
0: \Device\NPF_{AF5DBF15-9B94-4942-9DB4-7BDEE6D157C2} (Ethernet)
1: \Device\NPF_{19546A68-2315-49C0-8D7B-3876162887FD} (VirtualBox Host-Only Network)
2: \Device\NPF_{0DF66E88-917C-4182-86B3-623D81BAB59F} (Local Area Connection* 3)
3: \Device\NPF_{0C1DD229-33E0-4416-B8F2-5CCF1579E368} (Local Area Connection* 4)
4: \Device\NPF_{6563B7AA-E715-4B74-87AD-DDB4D577F47E} (Wi-Fi)
5: \Device\NPF_{0CDFF42D-B437-4379-B15B-542C026B396E} (Bluetooth Network Connection)
6: \Device\NPF_{B459DF4A-512D-4D05-AD11-9986D13DDB68} (Loopback Pseudo-Interface 1)
7: \Device\NPF_{536EF720-157C-42E9-A12C-0AEA26037F58} (Unknown)
8: \Device\NPF_Loopback (Unknown)
9: \Device\NPF_{BB917FD2-5B1A-41D0-8376-F6A3DED3D1B1} (Unknown)

Select the interface to monitor (enter the number): 4

Monitoring interface: \Device\NPF_{6563B7AA-E715-4B74-87AD-DDB4D577F47E} (Wi-Fi)

Select detection method ('threshold' or 'anomaly'): threshold
Enter window size in seconds (default 10): 10
Would you like an auto-recommended threshold based on a short traffic sample? (y/n): y
How long (in seconds) would you like to monitor the traffic to form a sample?: 30

Measuring traffic on \Device\NPF_{6563B7AA-E715-4B74-87AD-DDB4D577F47E} for 30 seconds to recommend thresholds...
Recommended SYN threshold ~ 33, UDP threshold ~ 379
Enter desired SYN threshold (default 33): 33
Enter desired UDP threshold (default 379): 379
Enter duration to monitor in seconds (leave blank for continuous monitoring): 60

Monitoring on \Device\NPF_{6563B7AA-E715-4B74-87AD-DDB4D577F47E} using threshold detection method with window size = 10s.
Using SYN threshold = 33  |  UDP threshold = 379
Monitoring for 60 seconds.

Starting packet sniffing...
```

Analysis of Anomaly-Based Detection

- **Detection Onset:** The anomaly alarm generally flags attacks within ~10 seconds, slightly slower than the near-instant threshold alerts. However, it's still fast enough to be practical for DoS defences.
- **Extended Durations:** The system often logs over 30 seconds for a 30-second attack because it waits for traffic to stabilize before ending the alert. This mirrors the "residual lag" seen with threshold-based detection.
- **Adaptive Modeling:** The anomaly method detects spikes in both SYN and UDP traffic, relying on learned normal patterns rather than fixed thresholds. However, if attack data bleeds into the baseline, it can distort future detections. I took precautions to prevent this bleed but it might need to be improved

Comparison:

Threshold-Based Alarm

- **Strengths:**
  - Detects attacks almost instantly (within 1–2 seconds) when packet rates surpass predefined thresholds.
  - Straightforward configuration (e.g., setting SYN/UDP packet thresholds).
- **Limitations:**
  - Residual network traffic often prolongs the recorded attack duration beyond the actual 30-second window.
  - Needs careful threshold tuning to avoid false positives/negatives in bursty or evolving traffic conditions.

Anomaly-Based Alarm

- **Strengths:**
  - Learns normal traffic patterns and flags significant deviations, enabling it to catch a variety of DoS methods.
  - Doesn't rely on fixed thresholds, making it more adaptive over time.
- **Limitations:**
  - Slightly slower detection (~10 seconds) due to the need for a rolling window or baseline update.
  - If attack data contaminates the baseline, it can distort future detections.

Conclusion:
Threshold-based detection excels at immediate recognition of large spikes but struggles to terminate alerts promptly. Anomaly-based detection adapts to changing traffic but may have

slower onset detection. Both methods show promise, and refining “attack end” criteria and baseline maintenance could significantly improve overall accuracy.

How I Met These Goals:

- **Working Prototypes:**
  - Developed a modular Python script that encapsulates both a ThresholdDetector and an AnomalyDetector, allowing easy switching between methods.
  - Successfully integrated both detection methods into a single tool, running concurrently with Scapy-based packet sniffing and multi-threaded execution.
- **User-Configurable Detection Settings:**
  - Implemented prompts to input or auto-calculate threshold values based on a short traffic sample.
  - Added functionality for users to specify the monitoring window (WINDOW_SIZE), enhancing detection granularity.
- **Anomaly Detection via K-Means:**
  - Updated the anomaly detection logic to use distance from the centroid instead of a simple ratio.
  - Provided configuration options for key parameters: n_clusters, distance_threshold, and max_history.
  - Designed the system to update the anomaly detection model once per time window, allowing gradual baseline formation from historical data.
  - Tests on a local network flagged artificially induced SYN/UDP flood spikes.
- **Performance Metrics Logging:**
  - Implemented detailed logging that records attack metrics after each attack window.
  - Metrics now include attack type, start time, end time, duration, involved sources, and peak packet counts, as shown above.
  - Initial tests have shown promising results, although further fine-tuning is required to improve accuracy in bursty network conditions. Window length still seems to be the issue that is affecting accuracy
- **Preliminary Experimental Data & Observations:**
  - Short traffic sampling (~30 seconds) for auto-recommended thresholds effectively reduced false positives during normal traffic.
  - Observations indicate both detection methods detect UDP Flood and SYN Flood attacks. More tools will be used for testing as I optimize and integrate.

Next Steps:

- **Seamless Detection Switching:**
  Implement a mechanism that allows real-time switching between threshold-based and anomaly-based detection without stopping the program.

- **Enhanced Alerting:**
  Upgrade the alerting system (e.g., integrating email or SMS notifications but most likely console pop-ups.) to ensure critical alerts are promptly delivered.

- **Improved Detection Performance:**
  Utilize DoS simulation tools to systematically stress test the system and further fine-tune detection parameters for optimal performance and accuracy.