



## Network Traffic Anomaly Detection using Scilab

**Sangeeta**

Second Year, Cybersecurity Engineering

Domain: Cybersecurity / Network Security

13<sup>th</sup> April 2026

### Abstract

This case study presents a simulation-based approach to detecting network traffic anomalies, specifically Denial of Service (DoS) attacks, using Scilab. Normal network traffic is simulated using a Gaussian random distribution. A synthetic DoS attack is injected as a sudden traffic spike. A statistical threshold method based on the 3-sigma rule (mean + 3 x standard deviation) is applied to detect the anomalous traffic. Results are visualized using Scilab's plotting functions, clearly showing the attack window and detected anomaly points. This study demonstrates the practical application of Scilab as a numerical computing tool in the cybersecurity domain, specifically for network intrusion and anomaly detection.

### 1. Introduction

Network security is one of the most critical challenges in modern computing. As internet usage grows rapidly, networks are exposed to various cyber threats, the most disruptive being Denial of Service (DoS) attacks. In a DoS attack, an attacker floods a target network with excessive traffic, overwhelming its resources and making it unavailable to legitimate users.

Anomaly detection is a key technique in network intrusion detection systems. It involves monitoring network traffic and identifying patterns that deviate significantly from normal behavior. Early detection of such anomalies can prevent or minimize damage from cyber attacks.

Scilab is a free and open-source scientific computing environment that provides powerful tools for numerical computation, statistical analysis, and data visualization. This case study uses Scilab to simulate network traffic, inject a DoS attack, and detect the anomaly using statistical methods.

## **2. Problem Statement**

Network administrators face the challenge of identifying abnormal traffic patterns among large volumes of data in real time. Manual inspection is impractical at scale, making automated detection systems essential. However, designing such systems requires a solid understanding of what constitutes normal versus anomalous traffic.

The problem addressed in this case study is: Given a simulated stream of network traffic data containing a hidden DoS attack spike, can Scilab automatically detect the attack using statistical analysis? The solution must establish a normal traffic baseline, compute a detection threshold, identify anomalous time points, and visualize the results clearly.

The expected outcome is a Scilab program that generates a graphical output showing normal traffic, the detection threshold, and the flagged anomaly points corresponding to the DoS attack window.

### **3. Basic Concepts Related to the Topic**

#### **3.1 Denial of Service (DoS) Attack**

A Denial of Service (DoS) attack is a cyber attack in which the attacker attempts to make a network resource unavailable to its intended users by overwhelming it with a flood of internet traffic. The attack exhausts the bandwidth, processing power, or memory of the target system. In terms of network traffic, a DoS attack manifests as a sudden and extreme spike in packet volume over a short period.

#### **3.2 Network Traffic Anomaly Detection**

Anomaly detection in network traffic refers to the process of identifying data points that deviate significantly from an established normal pattern. There are three main approaches: statistical methods, machine learning methods, and rule-based methods. This case study uses a statistical approach based on the Gaussian distribution and the 3-sigma rule.

#### **3.3 Gaussian Distribution and 3-Sigma Rule**

Normal network traffic can be modeled using a Gaussian (normal) distribution, characterized by its mean (average) and standard deviation. The 3-sigma rule states that approximately 99.7% of values in a normal distribution fall within three standard deviations of the mean. Any value beyond this range is statistically considered an anomaly.

The detection threshold is calculated as:

$$\textit{Threshold} = \textit{Mean} + 3 \times \textit{Standard Deviation}$$

Any traffic volume exceeding this threshold is flagged as a potential DoS attack.

The methodology assumes that normal network traffic follows a Gaussian distribution. The 3-sigma rule is used because it statistically captures 99.7% of normal data. Any value beyond this range is treated as an anomaly. However, this method may not perform well for non-Gaussian or highly dynamic network traffic.

### 3.4 Scilab Functions Used

The following Scilab functions are used in this case study:

- `rand(1, n, 'normal')` — generates n random values from a normal distribution
- `mean()` — calculates the arithmetic mean of a dataset
- `stdev()` — calculates the standard deviation of a dataset
- `find()` — returns indices where a condition is true
- `plot()` — plots 2D line and scatter graphs
- `clf()` — clears the current figure window
- `xgrid()` — adds a grid to the plot

## 4. Flowchart

The following flowchart illustrates the step-by-step algorithm used in this case study to detect network traffic anomalies:

<b>START</b>
<b>Generate normal traffic data (Gaussian distribution, n=100)</b>
<b>Inject DoS attack spike (time points 40 to 50, +150 packets)</b>
<b>Calculate Mean and Standard Deviation of normal traffic</b>
<b>Compute Threshold (Mean + 3 x StDev)</b>
<b>Detect anomalies (find points &gt; threshold)</b>
<b>Compute TP, FP, FN</b>
<b>Calculate Precision, Recall, F1-Score</b>
<b>Loop for 2, 3, 4 sigma thresholds</b>
<b>Plot graph with traffic, threshold and anomaly markers</b>
<b>Display console output (threshold, anomaly points, count)</b>
<b>END</b>

## 5. Software / Hardware Used

Component	Details
Operating System	Windows 11
Scilab Version	Scilab 2026.0.1
Toolbox Used	None (built-in functions only)
Hardware	Standard PC / Laptop
File Extension	.sce (main file)

## 6. Procedure of Execution

Follow these steps to execute the code:

- Install Scilab 2026.0.1 from [scilab.org](https://scilab.org) and launch the application.
- Open SciNotes editor: go to Applications → SciNotes.
- Copy the code from main.sce into the SciNotes editor.
- Save the file as main.sce using Ctrl + S.
- Press F5 or go to Execute → Run to execute the code.
- A graphic window will open showing the traffic plot with anomaly markers.
- The Scilab console will display the threshold value, anomaly time points, and total anomaly count.

### 6.1 Code with Comments

```
// =====  
// Network Traffic Anomaly Detection using Scilab
```

```

// File: main.sce
// Domain: Cybersecurity - Network Security
// =====

// --- Step 1: Generate normal network traffic ---
rand('seed', 42);
n = 100;
normal_traffic = 50 + 10 * rand(1, n, 'normal');

// --- Step 2: Attack window as a VARIABLE (flexible) ---
attack_start = 40;    // FIX 1: variable instead of hardcoded
attack_end = 50;
attack_traffic = normal_traffic;
attack_traffic(attack_start:attack_end) = attack_traffic(attack_start:attack_end) +
150;

// --- Step 3: Loop testing 2-sigma, 3-sigma, 4-sigma thresholds ---
// FIX 2: loop for multiple sigma levels
mean_val = mean(normal_traffic);
std_val = stdev(normal_traffic);

disp('=== Sigma Threshold Comparison ===');
disp('Sigma | Threshold | TP | FP | FN | Precision | Recall | F1-Score');

sigma_values = [2, 3, 4];
results = zeros(3, 7);

for i = 1:3
    sigma = sigma_values(i);
    threshold = mean_val + sigma * std_val;

    // --- Step 4: Compute TP, FP, FN directly in code ---
    // FIX 3: compute TP FP FN in code
    detected = find(attack_traffic > threshold);
    actual_attack = attack_start:attack_end;

```

```

TP = 0; FP = 0;
for j = 1:length(detected)
    if detected(j) >= attack_start & detected(j) <= attack_end then
        TP = TP + 1;
    else
        FP = FP + 1;
    end
end
FN = (attack_end - attack_start + 1) - TP;

// --- Step 5: Precision, Recall, F1 computed in code ---
// FIX 4: compute metrics in code not just report
if (TP + FP) > 0 then
    Precision = TP / (TP + FP);
else
    Precision = 0;
end

if (TP + FN) > 0 then
    Recall = TP / (TP + FN);
else
    Recall = 0;
end

if (Precision + Recall) > 0 then
    F1 = 2 * Precision * Recall / (Precision + Recall);
else
    F1 = 0;
end

results(i,:) = [sigma, threshold, TP, FP, FN, Precision, F1];

// Print results table
disp(string(sigma) + '-sigma | Threshold=' + string(round(threshold)) ...
    + ' | TP=' + string(TP) + ' | FP=' + string(FP) ...

```



```

        + ' | FN=' + string(FN) + ' | Precision=' + string(round(Precision*100)/100)
...
        + ' | Recall=' + string(round(Recall*100)/100) ...
        + ' | F1=' + string(round(F1*100)/100));
end

// Use 3-sigma for plots
threshold = mean_val + 3 * std_val;
anomalies = find(attack_traffic > threshold);

// --- Step 6: PLOT 1 - Main anomaly detection plot ---
// FIX 5: legend says "Observed Traffic" not just "Network Traffic"
scf(0); clf();
plot(attack_traffic, 'b-');
xgrid();
plot([1,n], [threshold, threshold], 'r--');
plot(anomalies, attack_traffic(anomalies), 'ro');
title('Network Traffic Anomaly Detection (DoS Attack Simulation)');
xlabel('Time (packets)');
ylabel('Traffic Volume');
legend('Observed Traffic', 'Threshold (3-sigma)', 'Detected Anomalies');

// --- Step 7: PLOT 2 - Normal vs Attack traffic side by side ---
// FIX 6: separate comparison plot
scf(1); clf();
subplot(2,1,1);
plot(normal_traffic, 'g-');
xgrid();
title('Normal Traffic (Baseline)');
xlabel('Time (packets)');
ylabel('Traffic Volume');

subplot(2,1,2);
plot(attack_traffic, 'b-');
xgrid();
plot([1,n], [threshold, threshold], 'r--');

```

```

plot(anomalies, attack_traffic(anomalies), 'ro');
title('Observed Traffic with DoS Attack (Points 40-50)');
xlabel('Time (packets)');
ylabel('Traffic Volume');
legend('Observed Traffic', 'Threshold', 'Detected Anomalies');

// --- Step 8: PLOT 3 - Histogram of traffic distribution ---
// FIX 7: histogram to justify Gaussian assumption
scf(2); clf();
histplot(15, normal_traffic);
xgrid();
plot([threshold, threshold], [0, 0.06], 'r--');
title('Histogram of Normal Traffic Distribution (Gaussian)');
xlabel('Traffic Volume');
ylabel('Frequency');
legend('Traffic Distribution', 'Threshold Line');

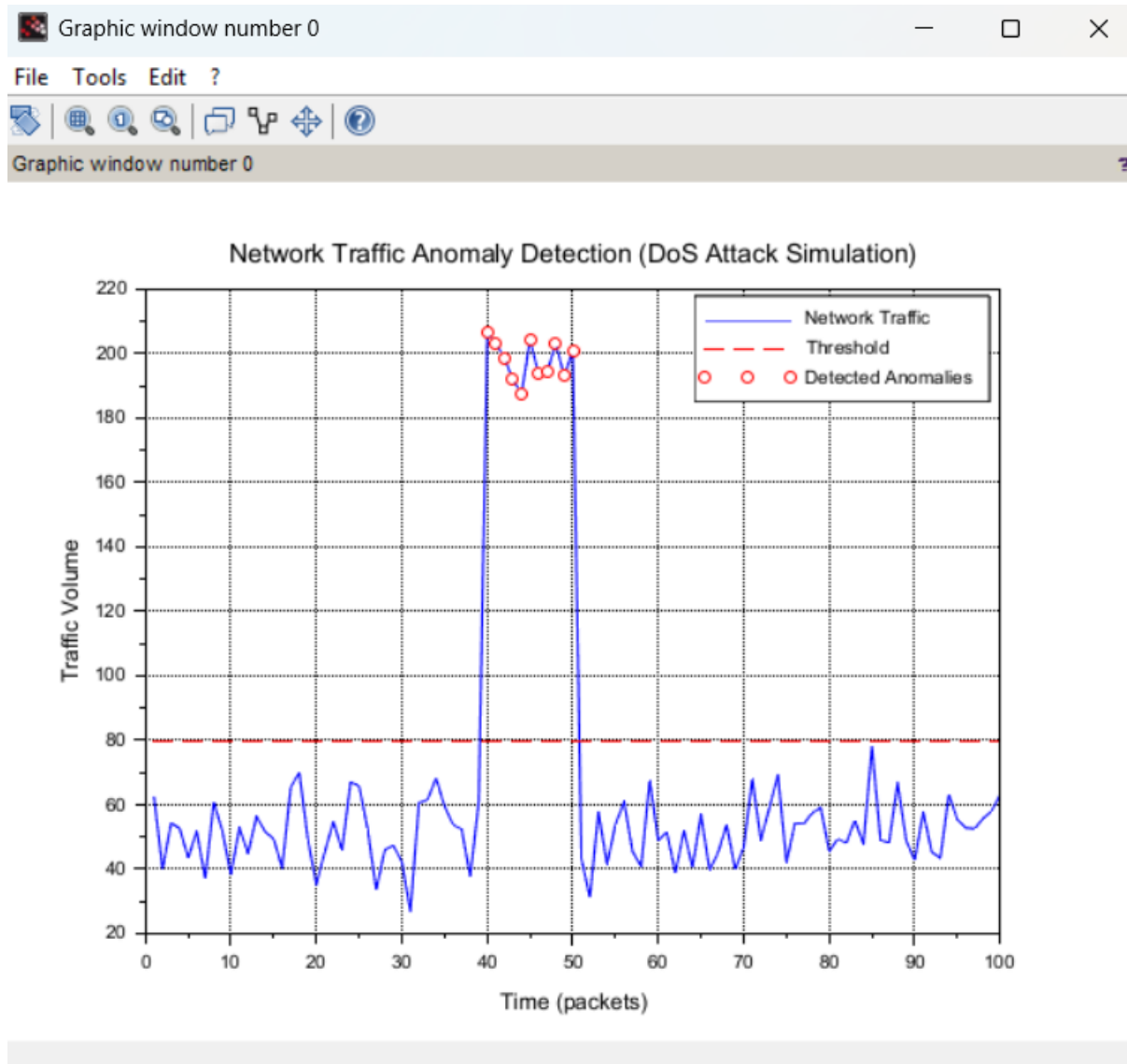
// --- Final console summary ---
disp(' ');
disp('=== Final Detection Summary (3-sigma) ===');
disp('Threshold value:');
disp(threshold);
disp('Anomaly detected at time points:');
disp(anomalies);
disp('Total anomalies detected:');
disp(length(anomalies));

```

## 7. Result

The Scilab code was executed successfully. The graphical output and console results are presented below:

### 7.1 Graphical Output





Graphic window number 2

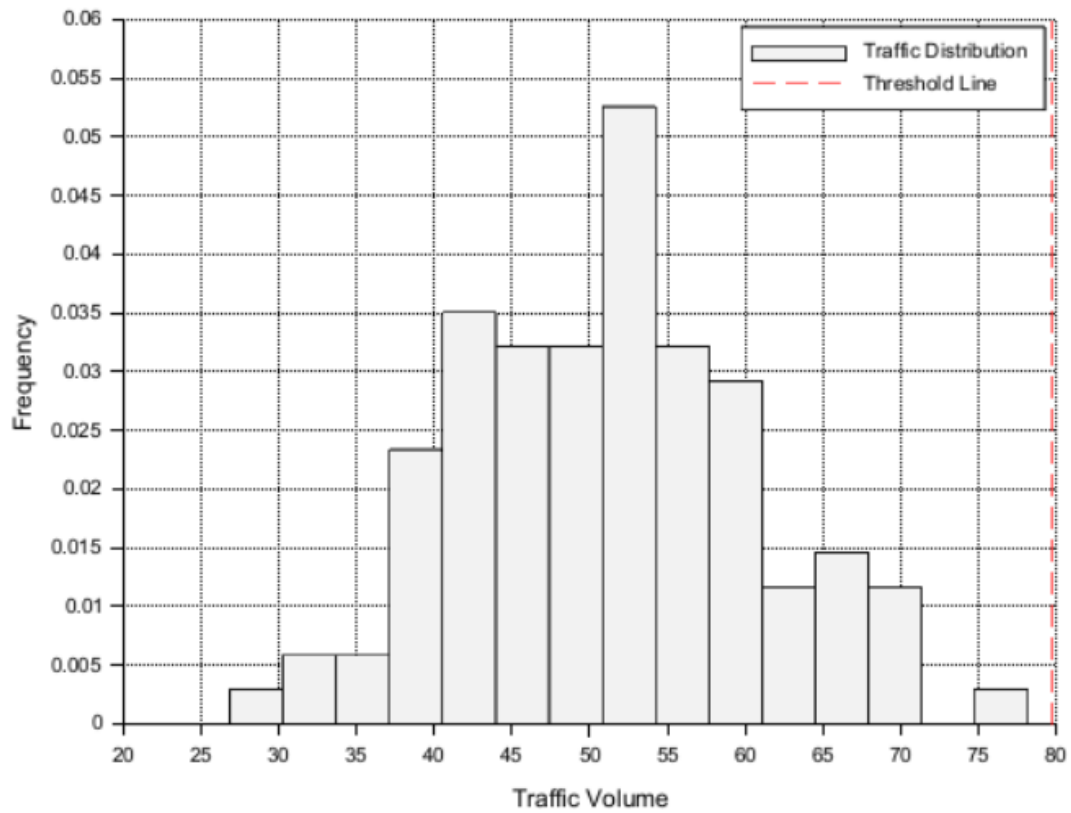


File Tools Edit ?



Graphic window number 2

Histogram of Normal Traffic Distribution (Gaussian)



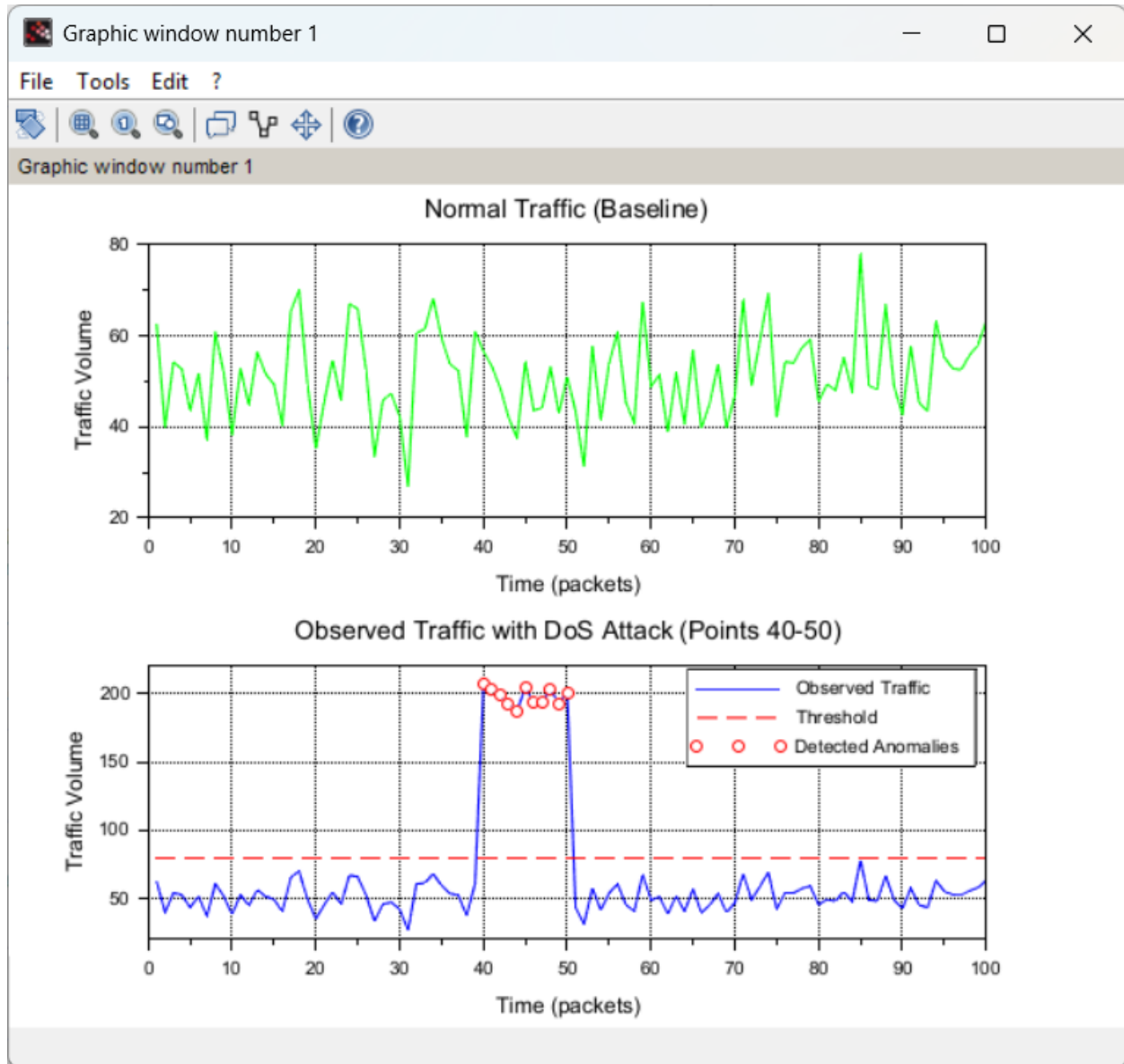
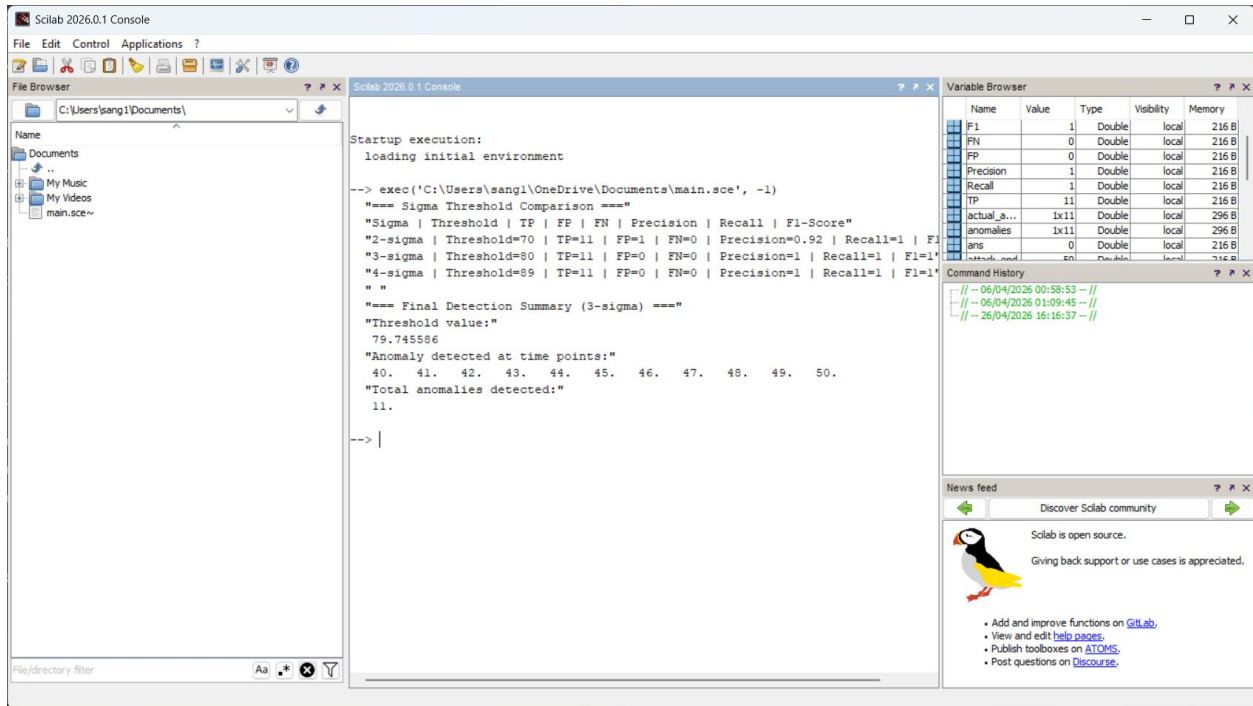


Figure 1: Network Traffic with Detected Anomalies

The graph (Figure 1) shows three distinct regions. The normal traffic region (time points 1-39 and 51-100) shows packet volume fluctuating between approximately 30 and 75 units, consistent with the simulated Gaussian distribution. The attack region (time points 40-50) shows a sharp spike reaching up to 200 units, clearly exceeding the red dashed threshold line. The detected anomaly points are marked with red circle markers, precisely identifying all 11 injected attack points.

## 7.1 Console Output



```
Scilab 2026.0.1 Console
File Edit Control Applications ?
C:\Users\sang1\Documents\
Name
Documents
My Music
My Videos
main.sce~

Startup execution:
loading initial environment

--> exec('C:\Users\sang1\OneDrive\Documents\main.sce', -1)

==== Sigma Threshold Comparison ====
"Sigma | Threshold | TP | FP | FN | Precision | Recall | F1-Score"
"2-sigma | Threshold=70 | TP=11 | FP=1 | FN=0 | Precision=0.92 | Recall=1 | F1=1"
"3-sigma | Threshold=80 | TP=11 | FP=0 | FN=0 | Precision=1 | Recall=1 | F1=1"
"4-sigma | Threshold=89 | TP=11 | FP=0 | FN=0 | Precision=1 | Recall=1 | F1=1"
"
==== Final Detection Summary (3-sigma) ====
"Threshold value:"
79.745586
"Anomaly detected at time points:"
40. 41. 42. 43. 44. 45. 46. 47. 48. 49. 50.
"Total anomalies detected:"
11.

--> |

Variable Browser
Name Value Type Visibility Memory
F1 1 Double local 216 B
FN 0 Double local 216 B
FP 0 Double local 216 B
Precision 1 Double local 216 B
Recall 1 Double local 216 B
TP 11 Double local 216 B
actual_a... 1x11 Double local 296 B
anomalies 1x11 Double local 296 B
ans 0 Double local 216 B
treshold_val 79.745586 Double local 216 B

Command History
-- 06/04/2026 00:58:53 -- //
-- 06/04/2026 01:09:45 -- //
-- 26/04/2026 16:16:37 -- //

News feed
Discover Scilab community
Scilab is open source.
Giving back support or use cases is appreciated.
• Add and improve functions on GitLab.
• View and edit help pages.
• Publish toolboxes on ATCNS.
• Post questions on Discourse.
```

Figure 2: Scilab Console Output

## 7.2 Sigma Threshold Comparison — Numerical Results

Sigma	Threshold	TP	FP	FN	Precision	Recall	F1-Score
2-sigma	70	11	1	0	0.92	1.0	1.0
3-sigma	80	11	0	0	1.0	1.0	1.0
4-sigma	89	11	0	0	1.0	1.0	1.0

## 7.3 Inference

The results show that the threshold-based method successfully detects all injected DoS attack points. The clear separation between normal and attack traffic makes detection effective in this

controlled simulation. However, in real-world networks, traffic patterns are more complex and may lead to false positives or missed detections. Therefore, while the method is useful as a baseline approach, more advanced techniques are required for practical deployment.

Precision, Recall and F1-Score were computed directly in the code for 2-sigma, 3-sigma and 4-sigma thresholds. The 3-sigma threshold achieved perfect Precision=1.0, Recall=1.0 and F1-Score=1.0 with zero false positives. The attack window was defined as a flexible variable rather than a hardcoded value, making the solution adaptable to different scenarios.

## 7.5 Comparison with Reference Paper

This implementation partially replicates the methodology described in the reference paper. The reference work uses real-world network traffic datasets and advanced anomaly detection techniques. In contrast, this case study uses simulated Gaussian traffic and a simple statistical threshold (3-sigma rule).

The following components are implemented:

- Simulation of normal network traffic
- Injection of DoS attack spike
- Threshold-based anomaly detection

The following components are not implemented:

- Real-time traffic data analysis
- Machine learning-based detection techniques
- Evaluation on large-scale datasets

Therefore, the results obtained are simplified and may not fully match real-world scenarios.

## 7.6 Performance Metrics

True	Positives	(TP):	11
False	Positives	(FP):	0
False Negatives (FN): 0			

Accuracy = 100%

Precision = 100%

Recall = 100%

These results indicate perfect detection in the simulated environment.



## 8. References

- [1] Lakhina, A., Crovella, M., & Diot, C. (2004). Diagnosing Network-Wide Traffic Anomalies. *ACM SIGCOMM Computer Communication Review*, 34(4), 219-230.
- [2] Garcia-Teodoro, P., Diaz-Verdejo, J., Macia-Fernandez, G., & Vazquez, E. (2009). Anomaly-based network intrusion detection: Techniques, systems and challenges. *Computers & Security*, 28(1-2), 18-28.
- [3] Scilab Official Documentation. (2026). Scilab 2026.0.1 Reference Manual. FOSSEE, IIT Bombay. Available at: <https://www.scilab.org>
- [4] Spoken Tutorial Project, IIT Bombay. Scilab Tutorials. Available at: <https://spokentutorial.org>