Enhancing Intrusion Detection System using XDP with eBPF

K.Geethika, G.Krishna Prathibha, G.Sneha

Under the esteemed guidance of

Mr. N. Anand

Assistant Professor

Bachelor of Technology

Department of Information Technology

BVRIT HYDERABAD college of engineering for Women

September 21, 2023

Overview

Introduction

Network filtering is essential for securing computer networks, and there are various types of attacks that network filtering aims to mitigate. Here are some common network attacks:

- **DOS (Denial of Service) Attack:**
  Attackers flood a network or system with excessive traffic, causing it to become overwhelmed and unavailable to legitimate users.
- **DDOS (Distributed Denial of Service) Attack:** Similar to DoS but conducted from multiple sources, making it even more challenging to mitigate.

Intrusion Detection System :

IDS stands for Intrusion Detection System. It is a critical component of a cybersecurity strategy designed to detect and respond to unauthorized access or suspicious activities on a network. IDS is required for several reasons:

- **Security Threat Detection:** IDS helps in detecting and identifying security threats and unauthorized activities within a network or computer system.
- **Network Visibility and Risk Mitigation:** IDS offers insights into network traffic, aids in understanding normal network behavior, and helps reduce the attack surface by identifying vulnerabilities and deviations from the norm.

What if we use general packet filters ? :

- Detection Precision
- Resource Efficiency
- Network Performance

eBPF and XDP

eBPF (extended Berkely Packet Filtering) :

- eBPF is a revolutionary technology which origins in the Linux kernel.
- eBPF extends kernel capabilities without altering kernel source code or loading kernel modules.
- It operates as a bytecode-based virtual machine for running programs.
- A verifier assesses the safety of eBPF programs when loaded into the kernel and rejects them if found unsafe.

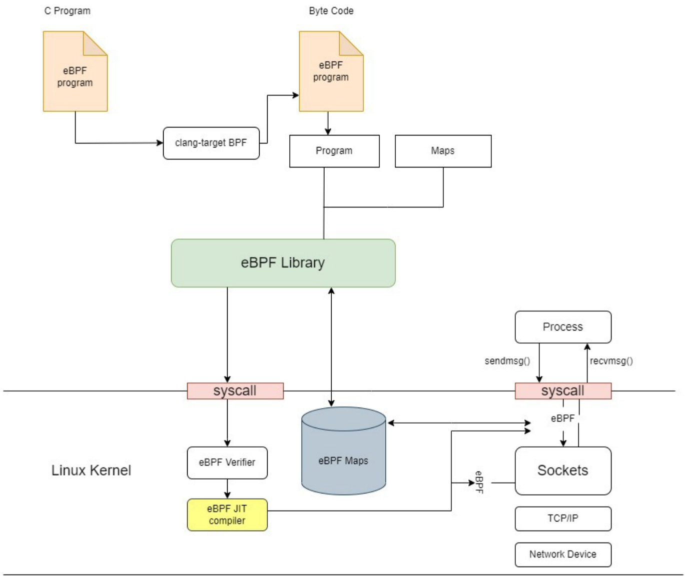


Figure: eBPF Architecture

XDP (eXpress Data Path) :

- XDP (eXpress Data Path) is a high-performance data processing framework in the Linux kernel.
- It operates at an extremely low level, providing fast packet processing capabilities.
- XDP is commonly used for network filtering, load balancing, and DDoS mitigation.
- It allows for efficient packet handling with minimal overhead, making it ideal for high-speed networking applications.

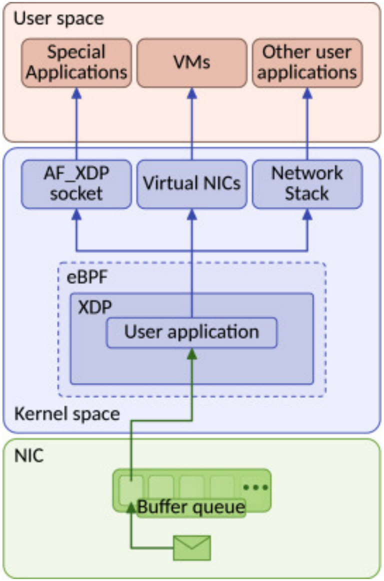


Figure: XDP Architecture

Literature Survey

S.No	Title	Author	Description	Year
1	High-performance Intrusion Detection System using eBPF with Machine Learning algorithms [1]	Nemalikanti Anand, Saifulla M A, Pavan Kumar Aakula	Intrusion Detection system using eBPF	2023
2	Intrusion Detection Systems, Issues, Challenges, and Needs [2]	Mohammad Aljanabi, Mohd Arfian Ismail, Ahmed Hussein Ali	Machine Learning Algorithms like Decision Tree and Support Vector Machine	2021

S.No	Title	Author	Description	Year
3	Fast Packet Processing with eBPF and XDP: Concepts, Code, Challenges and Applications [3]	MARCOS A. M. VIEIRA , MATHEUS S. CASTANHO, RACYUS D. G. PACÍFICO	Introduction to eBPF and XDP	2020

Problem Statement

- Attack detecting methods(like IDS, IPS, etc) can detect various application-level attacks (eg.HTTP attacks, DHCP attacks, etc) at network layer.
- In network layer, during packet filtering, there may be loss of data/useful information. (due to DOS attacks, Resource Limitations, Inadequate Throughput, etc).
- Machine learning algorithms help to classify the normal data and attack data. So that the performance may increase compared to existing methods.

"So, At what level can we filter the attacks ??"

Proposed Method

- The proposed solution is the eBPF (extended Berkely Packet Filtering) technology along with XDP (eXpress Data Path)
- eBPF technology with XDP is used to detect the attacks at kernel level without modifying the source code in the kernel
- Kernel-level packet filtering provides more efficiency in detecting and handling attacks by controlling packet flow at a lower system level ensuring robust security.
- eBPF with XDP analyzes target values from trained models and connects with network devices to make decisions on packet dropping or forwarding.

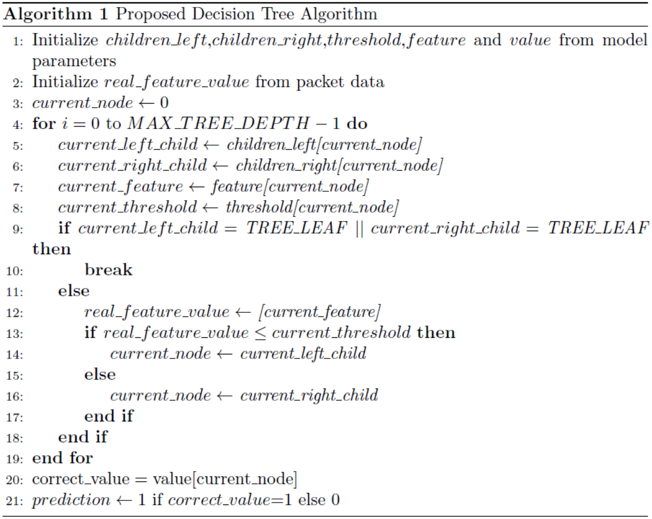
The steps involved in this process are:

- Identify CIC-IDS-2017 dataset.
- The raw data is then pre-processed, that includes data transmission, cleaning (infinity, NULL values, etc.), and reduction (size).
- Apply ANOVA technique to find top rated features from the pre-processed data
- The extracted features are analyzed for intrusion detection, using ML algorithms (DT, RF, SVM, TwinSVM).
- write an eBPF program(C program) that is converted to bytecode which utilizes trained model parameters to detect attacks.
- write a XDP code with eBPF program that utilizes trained model parameters to detect attacks (filter packets) faster and accurate within the kernel.

Machine Learning Algorithms

Decision Tree Algorithm

- Decision trees are a popular machine learning algorithm used for both classification and regression tasks.
- They work by recursively partitioning data into subsets based on the most informative features, creating a tree-like structure.
- Decision trees are interpretable and can be visualized, making them valuable for understanding decision-making processes.



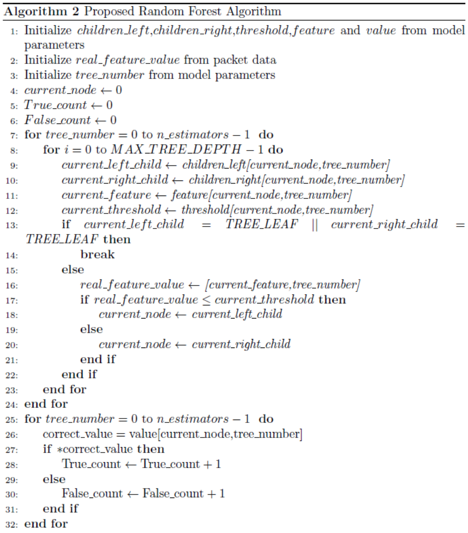
Algorithm 1 Proposed Decision Tree Algorithm

```
1:  Initialize $children\_left$, $children\_right$, $threshold$, $feature$ and $value$ from model parameters
2:  Initialize $real\_feature\_value$ from packet data
3:  $current\_node \leftarrow 0$
4:  for $i = 0$ to $MAX\_TREE\_DEPTH - 1$ do
5:      $current\_left\_child \leftarrow children\_left[current\_node]$
6:      $current\_right\_child \leftarrow children\_right[current\_node]$
7:      $current\_feature \leftarrow feature[current\_node]$
8:      $current\_threshold \leftarrow threshold[current\_node]$
9:      if $current\_left\_child = TREE\_LEAF \mid\mid current\_right\_child = TREE\_LEAF$ then
10:         break
11:     else
12:         $real\_feature\_value \leftarrow [current\_feature]$
13:         if $real\_feature\_value \leq current\_threshold$ then
14:             $current\_node \leftarrow current\_left\_child$
15:         else
16:             $current\_node \leftarrow current\_right\_child$
17:         end if
18:     end if
19: end for
20: correct_value = value[current_node]
21: $prediction \leftarrow 1$ if $correct\_value$=1 else 0
```

Figure: Decision Tree Algorithm

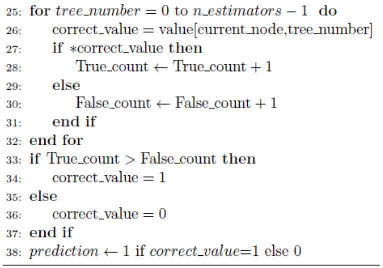
Random Forest Algorithm

- Random Forest is an ensemble machine learning algorithm that combines multiple decision trees to make more accurate predictions.
- It introduces randomness during the tree-building process by selecting random subsets of features and data samples, which helps reduce overfitting.
- Random Forest is effective for both classification and regression tasks.



Algorithm 2 Proposed Random Forest Algorithm

```
1:  Initialize $children\_left$,$children\_right$,$threshold$,$feature$ and $value$ from model
    parameters
2:  Initialize $real\_feature\_value$ from packet data
3:  Initialize $tree\_number$ from model parameters
4:  $current\_node \leftarrow 0$
5:  $True\_count \leftarrow 0$
6:  $False\_count \leftarrow 0$
7:  for $tree\_number = 0$ to $n\_estimators - 1$ do
8:      for $i = 0$ to $MAX\_TREE\_DEPTH - 1$ do
9:          $current\_left\_child \leftarrow children\_left[current\_node,tree\_number]$
10:         $current\_right\_child \leftarrow children\_right[current\_node,tree\_number]$
11:         $current\_feature \leftarrow feature[current\_node,tree\_number]$
12:         $current\_threshold \leftarrow threshold[current\_node,tree\_number]$
13:         if $current\_left\_child = TREE\_LEAF \mid\mid current\_right\_child =$
    $TREE\_LEAF$ then
14:             break
15:         else
16:             $real\_feature\_value \leftarrow [current\_feature,tree\_number]$
17:             if $real\_feature\_value \leq current\_threshold$ then
18:                 $current\_node \leftarrow current\_left\_child$
19:             else
20:                 $current\_node \leftarrow current\_right\_child$
21:             end if
22:         end if
23:     end for
24: end for
25: for $tree\_number = 0$ to $n\_estimators - 1$ do
26:     correct_value = value[current_node,tree_number]
27:     if *correct_value then
28:         True_count ← True_count + 1
29:     else
30:         False_count ← False_count + 1
31:     end if
32: end for
```



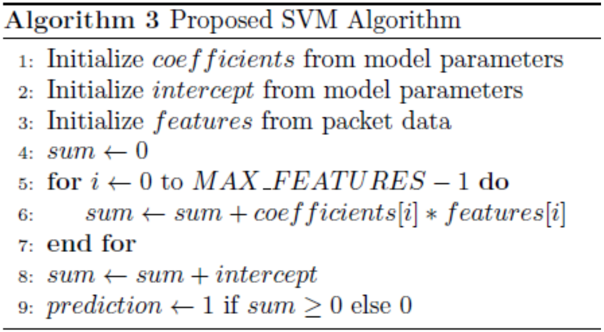
```
25: for $tree\_number = 0$ to $n\_estimators - 1$ do
26:     correct_value = value[current_node,tree_number]
27:     if *correct_value then
28:         True_count ← True_count + 1
29:     else
30:         False_count ← False_count + 1
31:     end if
32: end for
```

```
33: if True_count > False_count then
34:     correct_value = 1
35: else
36:     correct_value = 0
37: end if
38: $prediction \leftarrow 1$ if $correct\_value$=1 else 0
```

Figure: Random Forest Algorithm

Support Vector Machine Algorithm

- SVM is a supervised machine learning algorithm used for classification and regression tasks.
- It works by finding a hyperplane that best separates data points into different classes, maximizing the margin between the classes.
- SVM can handle linear and nonlinear data separation through the use of different kernel functions, such as linear, polynomial, and radial basis function (RBF) kernels.



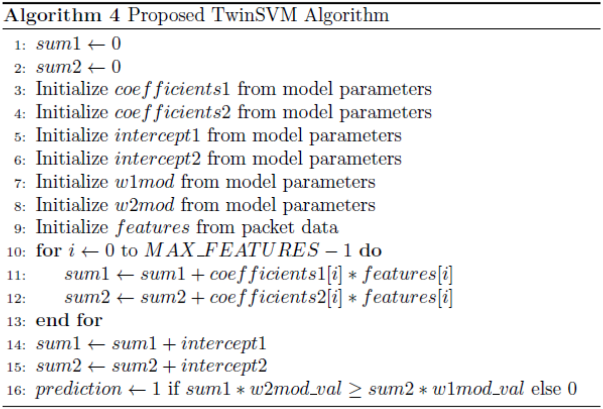
Algorithm 3 Proposed SVM Algorithm

```
Initialize $coefficients$ from model parameters
Initialize $intercept$ from model parameters
Initialize $features$ from packet data
$sum \leftarrow 0$
for $i \leftarrow 0$ to $MAX\_FEATURES - 1$ do
    $sum \leftarrow sum + coefficients[i] * features[i]$
end for
$sum \leftarrow sum + intercept$
$prediction \leftarrow 1$ if $sum \geq 0$ else 0
```

Figure: Support Vector Machine Algorithm

Twin Support Vector Machine Algorithm

- TWSVM is a machine learning technique that extends traditional Support Vector Machines (SVMs) for binary classification.
- It simultaneously learns two SVMs, one for each class, with the objective of maximizing the margin between both class boundaries.
- TWSVM is particularly useful when dealing with imbalanced datasets or situations where the margins of both classes need to be optimized independently.



Algorithm 4 Proposed TwinSVM Algorithm

```
1:  $sum1 \leftarrow 0$
2:  $sum2 \leftarrow 0$
3:  Initialize $coefficients1$ from model parameters
4:  Initialize $coefficients2$ from model parameters
5:  Initialize $intercept1$ from model parameters
6:  Initialize $intercept2$ from model parameters
7:  Initialize $w1mod$ from model parameters
8:  Initialize $w2mod$ from model parameters
9:  Initialize $features$ from packet data
10: for $i \leftarrow 0$ to $MAX\_FEATURES - 1$ do
11:     $sum1 \leftarrow sum1 + coefficients1[i] * features[i]$
12:     $sum2 \leftarrow sum2 + coefficients2[i] * features[i]$
13: end for
14: $sum1 \leftarrow sum1 + intercept1$
15: $sum2 \leftarrow sum2 + intercept2$
16: $prediction \leftarrow 1$ if $sum1 * w2mod\_val \geq sum2 * w1mod\_val$ else 0
```

Figure: Twin Support Vector Machine Algorithm

Implementation

- Analyzed the dataset CIC IDS 2017
- Top 15 Features are Extracted from dataset using ANOVA F-test method.
- These features are destination port, total forward packets, total backward packets, minimum packet length, etc.
- Working on eBPF Bytecode

References

Nemalikanti Anand Pavan Kumar Aakula, Saifulla M A.
High-performance intrusion detection system using ebpf with machine learning algorithms.
pages 1–25, 2023.

Ahmed Hussein Ali Mohammad Aljanabi, Mohd Arfian Ismail.
Intrusion detection systems, issues, challenges, and needs.
14:560 – 571, 2021.

RACYUS D. G. PACÍFICO ELERSON R. S. SANTOS EDUARDO P. M. CÂMARA JÚNIOR MARCOS A. M. VIEIRA, MATHEUS S. CASTANHO and LUIZ F. M. VIEIRA.
Fast packet processing with ebpf and xdp: Concepts, code,challenges and applications.
53:1–36, 2020.

Any Questions?

Thank you