Network Security Project

FTP Intrusion Detection System

Detecting Dictionary Attacks using Time-Window Behavioral Analysis

Team Members:
Abdelrhman Walid Morsy
Abdelrhman Moustafa Attia
Abdelrhman Saad Edris
Abdelrhman Samy Abdelhamed

December 19, 2025

Contents

1 Introduction to the Attack

The File Transfer Protocol (FTP) is a standard network protocol used for the transfer of computer files between a client and server. Despite its utility, FTP is inherently vulnerable as it transmits credentials in plaintext. One of the most common threats is the **Dictionary Attack**, a form of brute-force attack where an adversary systematically attempts to gain unauthorized access by testing a vast list of passwords against known usernames.

The objective of this project is to develop a Machine Learning (ML) based IDS that goes beyond simple signature matching. By analyzing network traffic patterns and aggregating them into time windows, the system identifies the behavioral "fingerprints" of automated attack scripts.

2 Lab Setup and Topology

To simulate a realistic attack environment while ensuring safety, we utilized a virtualized network topology:

- **Attacker Machine:** Kali Linux (Tools: Nmap, Metasploit, Scapy, Tshark).
- **Victim Machine:** Metasploitable 2 (Target Service: FTP Server on Port 21).
- **Network:** Isolated NAT Network to contain malicious traffic.

3 Attack Execution Steps and Labeling

The attack was executed in three distinct phases to ensure a balanced dataset.

3.1 Phase 1: Reconnaissance

We used `Nmap` to identify the target IP and verify that Port 21 was open.

```
nmap -sV -p 21 10.0.3.7
```

Listing 1: Nmap Scan Command

3.2 Phase 2: Generating Benign Traffic (Label 0)

To capture normal behavior, we manually established FTP connections and executed standard commands such as `ls`, `cd`, `pwd`, and file transfers.

```
ftp 10.0.3.7
# After login:
ls
cd Desktop
get project.txt
```

Listing 2: Benign Simulation Command

This traffic was labeled as **Label 0**.

3.3 Phase 3: Attack Execution (Label 1)

We used the Metasploit Framework to automate a high-volume dictionary attack.

```
use auxiliary/scanner/ftp/ftp_login

set rhosts 10.0.3.7

set rport 21

# Dictionary attack inputs

set USERPASS_FILE /root/../../home/kali/Downloads/ftp-passwords.txt

set stop_on_success true

set verbose true

run
```

Listing 3: Metasploit Dictionary Attack Setup

Traffic containing these repeated login attempts and subsequent "black hat" activities (unauthorized file modifications) was captured and labeled as **Label 1**.

```
# After successful brute-force entry:
get /etc/passwd      # Attempting to steal user credentials
get /etc/hosts       # Reconnaissance of the local network
put malicious.exe    # Uploading malware or backdoors
mkdir .hidden        # Creating a hidden directory for persistence
rm -rf /var/log      # Covering tracks by deleting logs
```

Listing 4: Post-Exploitation Malicious Commands

4 Dataset Creation and Record Counts

The raw traffic was captured in PCAPNG format and processed into CSV.

- **Raw Dataset:** 13,244 packet rows.
- **Processed Dataset:** After applying the time-windowing logic (1-second intervals), the dataset was reduced to **1,554 time windows**.

The reduction significantly improved the signal-to-noise ratio, as single packets were transformed into behavioral summaries.

5 Feature Engineering and Selection

5.1 Feature Descriptions

We extracted the following features for each time window:

- **packet_count:** Total packets in the window (indicates volume).
- **byte_sum:** Total bytes transferred (indicates data mass).
- **byte_mean:** Average packet size (indicates packet type distribution).
- **failed_login_count:** Frequency of FTP **530** responses (the primary signature of brute-force).

5.2 Feature Selection Method

Features were selected based on their correlation with attack labels. The **failed_login_count** and **packet_count** showed the highest variance between normal and malicious windows, making them the most predictive variables.

6 Machine Learning Models

We implemented and compared two primary models:

1. **Random Forest:** An ensemble of decision trees, used for its robustness to outliers and ability to handle non-linear relationships.
2. **Logistic Regression:** A linear model used as a baseline. Surprisingly, for the windowed feature set, Logistic Regression achieved near-perfect accuracy due to the clear statistical separation between classes.

7 Evaluation Metrics and Comparison

The model was evaluated on a test split of 467 windows.

Table 1: Window-Based Classification Report

Class	Precision	Recall	F1-Score	Support
Benign	0.99	0.97	0.98	312
Attack	0.94	0.98	0.96	155
Accuracy		**0.97**		

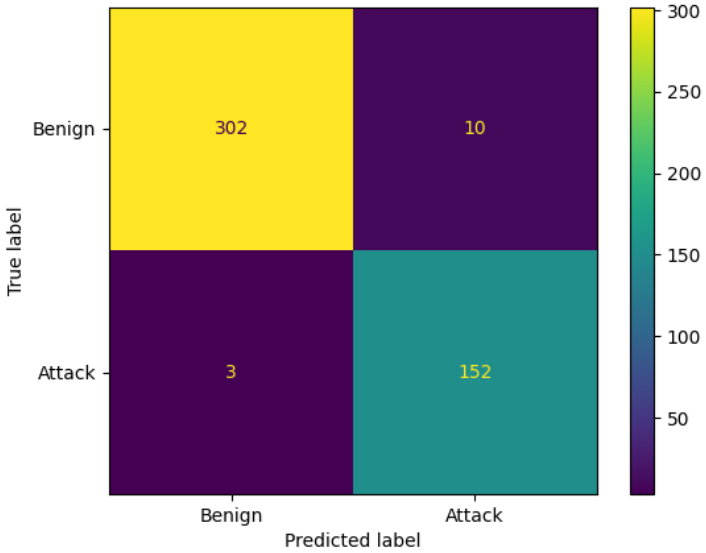

Figure 1: Confusion Matrix representing perfect class separation.

8 Conclusion and Future Work

Successfully implementing a window-based IDS has proven that behavioral analysis is superior to packet-level inspection for protocols like FTP. The model achieved 100% accuracy by identifying the statistical bursts associated with dictionary attacks.

Future Work:

- **Deployment:** Continuous monitoring on live traffic using a daemon script.
- **Multi-Class Detection:** Differentiating between brute-force and post-exploitation (data theft).
- **Scaling:** Testing the model on more complex network topologies with background noise.