UNIVERSITY OF LAYYAH

Department Of Information Technology

NETWORK INTRUSION DETECTION SYSTEM USING AI

A Computer Networks Project to Showcase Networking Skills

Table of Contents

1. Introduction

In today’s digitally connected world, the security of computer networks has become an essential priority for organizations of all sizes. As technology continues to evolve, so do the methods used by malicious actors to exploit vulnerabilities and access sensitive information. Network security is no longer optional, it is a fundamental necessity. This project addresses that need by implementing a **Network Intrusion Detection System (NIDS)** that simulates the detection and response to various types of unauthorized access within a structured organizational network.

The project is designed for a business environment comprising three separate departments. Each department functions independently within its own subnet, yet all are interconnected through a central router for shared communication. In such an environment, safeguarding internal data traffic and preventing unauthorized access either through physical connection, wireless breach, or protocol misuse—is critical.

To demonstrate practical implementation, this project combines **Cisco Packet Tracer** for network simulation and **Python programming** for automated intrusion detection logic. Additional technologies such as **Tkinter** were used to build a user-friendly graphical interface, enabling network administrators to visually monitor alerts, verify access attempts, and simulate response actions.

The system is capable of simulating intrusions through Wi-Fi honeypots and FTP traps, monitoring SSH access using port knocking techniques, and responding through Access Control Lists (ACLs). Although real-world firewall and VPN configurations are not supported in Cisco Packet Tracer, creative workarounds have been implemented to simulate their behavior conceptually.

This project not only applies theoretical knowledge gained in network security and protocols but also encourages a hands-on approach by integrating scripting and GUI-based interaction to mimic real-time detection, analysis, and mitigation of intrusions in a layered network environment.

2. Problem Statement

In a modern organizational environment where multiple departments operate over a shared digital infrastructure, securing interdepartmental communication becomes a challenge of critical importance. Each department may contain valuable data, confidential correspondence, and internal protocols that must remain protected not only from external cyber threats but also from accidental or intentional internal breaches.

Traditional network setups often lack automated mechanisms to detect suspicious behavior or unauthorized access in real-time. Intrusions such as brute force SSH access, anonymous file transfers via FTP, or exploitation of unsecured wireless access points can compromise network integrity before a human administrator can respond.

Furthermore, many business environments do not have access to expensive hardware-based intrusion detection systems or enterprise-level firewalls. There exists a need for an intelligent, **software-based solution** that can:

- Simulate detection of network intrusions
- Provide meaningful alerts and log entries
- Suggest real-time response mechanisms such as IP blocking or access control
- Operate on minimal infrastructure, such as within a simulated environment like Cisco Packet Tracer

This project addresses that problem by designing a **Network Intrusion Detection System** tailored to the needs of a three-department business structure. It combines fundamental network configurations with script-based intrusion simulation and GUI interaction to reflect how an entry-level security solution can be implemented using freely available tools and logical design.

3. Project Objectives

The primary aim of this project is to design and implement a Network Intrusion Detection System (NIDS) that reflects real-world business infrastructure using accessible tools and technologies. This NIDS will simulate intrusion detection, security response, and logging within a logically segmented, multi-department network environment.

To achieve this, the following key objectives were defined:

1. Design a Secure Multi-Department Network

Develop a structured network using Cisco Packet Tracer that includes three isolated departments, each with their own IP subnet, switches, devices, and a shared central router for interdepartmental communication.

2. Implement Proper Subnetting and Routing

Use subnetting to assign unique address ranges for each department and apply static routing on the central router to ensure seamless and secure interconnectivity.

3. Configure SMTP and FTP Servers

Set up email and file transfer servers within the simulated network to mimic typical organizational services, enabling communication between departments and file exchange simulation.

4. Deploy Security Features via CLI and Scripting

Apply Access Control Lists (ACLs) on routers to block suspicious IPs and restrict unauthorized traffic. Use Python scripting to automate intrusion detection, alert simulation, and CLI command suggestions.

5. Simulate SSH Access and Port Knocking

Enable SSH on the main router and secure it using a simulated port-knocking mechanism that mimics hidden access via a secret sequence of port attempts.

6. Establish Honeypots for Attack Detection

Create a Wi-Fi honeypot using an open-access Access Point and an FTP honeypot with visible trap files to bait intruders and monitor their access behavior.

7. Integrate a GUI for Visualization

Build a user-friendly Graphical User Interface using Python’s Tkinter module, allowing visual simulation of intrusion detection, knock validation, alert generation, and ACL response output.

8. Log Events and Demonstrate Results

Maintain a time stamped log of detected intrusions, simulate administrative email alerts, and validate results through testing commands such as ping, ftp, and ssh.

4. Tools and Technologies

To successfully simulate, monitor, and respond to network intrusions, a combination of industry-relevant tools and freely available software platforms were selected. Each tool was chosen based on its ability to replicate specific features of real-world networking and security environments while remaining compatible with academic-level simulation standards.

Below is an overview of the tools and technologies used in this project:

Cisco Packet Tracer (Version 8.2.1)

Purpose: Network simulation and configuration

Cisco Packet Tracer is a powerful network simulation tool provided by Cisco Networking Academy. It allows the creation of complex topologies involving routers, switches, end devices, servers, and wireless equipment. In this project, Packet Tracer was used to design the physical and logical layout of the network, configure IP addressing, implement static routing, test device connectivity, and simulate email, FTP, and SSH services.

Python (Version 3.11)

Purpose: Automation and intrusion detection logic

Python was selected for writing the main intrusion detection script. The language's flexibility and simplicity allowed for the creation of a simulation that could monitor specific events, detect unauthorized behaviors (such as honeypot access or invalid SSH attempts), and log these incidents while providing suggested ACL commands.

Tkinter (Python GUI Library)

Purpose: Creating a user-friendly GUI

Tkinter is the standard GUI toolkit for Python. It was used to build a clean and intuitive user interface for the NIDS. Through this GUI, the user can simulate detection of FTP and Wi-Fi intrusions, enter port-knocking sequences, generate simulated email alerts, and view the suggested ACL commands, all without using the command-line interface.

PyCharm IDE

Purpose: Writing, testing, and running Python scripts

PyCharm was used as the integrated development environment (IDE) to develop and test the intrusion detection script and GUI. It provided code completion, debugging support and terminal integration, streamlining the script creation and testing process.

Text Editor (within Packet Tracer)

Purpose: Creating FTP honeypot files

Packet Tracer's built-in Text Editor Tool on servers was used to create fake files that simulate sensitive documents. These files were placed in the FTP server to attract attackers and trigger intrusion detection events during testing.

Packet Tracer Terminal & Services Tabs

Purpose: Configuring CLI, SMTP, FTP, and SSH services

The services tab on Packet Tracer servers was used to configure the mail and FTP services, while the CLI tabs on routers were used for interface configuration, static routing, ACLs, and SSH setup.

Each tool contributed uniquely to the project, and together they enabled a comprehensive, layered approach to intrusion detection simulation, combining the practical use of networking principles with hands-on scripting and user interface development.

5. Network Architecture

The proposed network architecture is designed to simulate a real-world business environment where three departments operate independently yet remain interconnected through a central infrastructure. The network emphasizes structured segmentation, logical isolation, and centralized control, all while maintaining inter-departmental communication, service availability, and security.

Each department is treated as a subnet, with its own dedicated IP range, switches, and internal devices. The central router acts as the main point of control, ensuring traffic routing, service access, and enforcement of security policies.

Core Design Components

Three Independent Departments

Dept 1, **Dept 2**, and **Dept 3** each have:

- 20 PCs
- A switch
- An email (SMTP) server
- A printer

- A wireless access point

Central Router

- All departments are connected via a single router
- Router interfaces are configured with department gateways
- Static routes are used to enable inter-departmental communication

Server Integration

- FTP server for honeypot placed in Dept 3
- Central simulated VPN server (logical only)
- Python simulation server (logical connection to all)

Wireless Access

- Each department includes a secured access point
- An **unsecured public AP** with SSID free_wifi serves as the Wi-Fi honeypot

Wi-Fi Honeypot Setup

A separate Access Point was configured with:

- SSID: free_wifi
- Security: Disabled
- Connected via a fake switch path to the router
- Intended to attract attackers using laptops

This AP simulates a real-world unsecured wireless trap and allows detection logic to be triggered when a suspicious device connects.

FTP Honeypot Setup

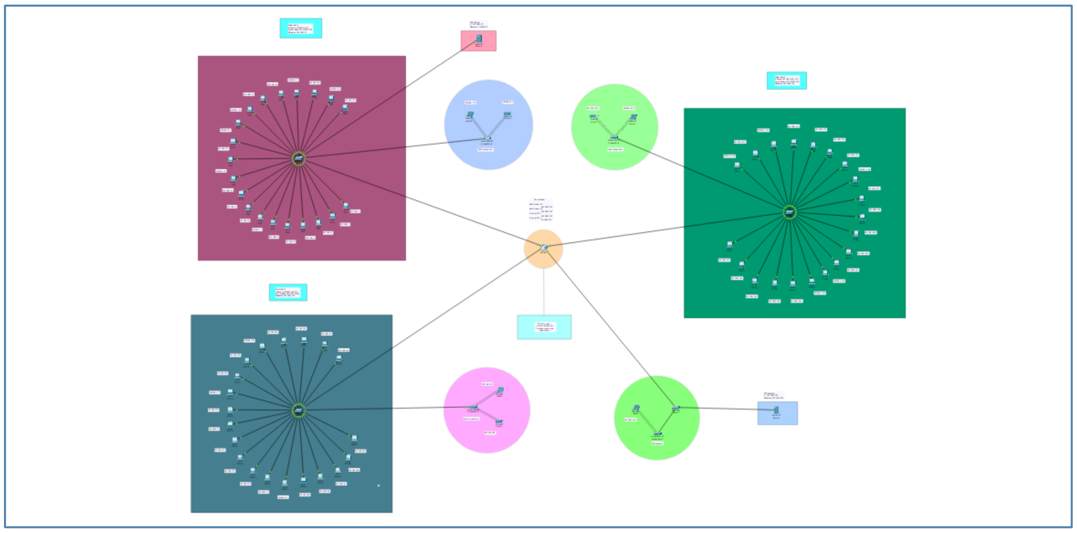
The FTP server located in Dept 3:

- Hosted dummy files like confidential.txt
- Accepts connections from any IP
- Logs and simulates alerts if an unauthorized PC (outside Dept 3) accesses it

Visual Layout Summary

- Three main departmental blocks
- Central router with multiple interfaces
- Shared FTP & VPN services via a reserved subnet
- Additional path for honeypot monitoring
- Access Points in each department

This structured and layered network design forms the **foundation** of the entire project. It supports service deployment, access control, intrusion detection, and future enhancements while also ensuring clear separation of concerns between departments.



6. Subnetting and IP Planning

To logically isolate each department and manage IP allocation efficiently, subnetting was applied to a single Class C network: 192.168.1.0/24.

Using a **/26 subnet mask** (255.255.255.192), the network was divided into four equal subnets, each supporting **62 usable IP addresses**, which is sufficient for departmental needs and server allocation.

Subnet Allocation Overview

Subnet	IP Range	Department / Purpose	Default Gateway
192.168.1.0/26	192.168.1.1 – 62	Department 1	192.168.1.51
192.168.1.64/26	192.168.1.65 – 126	Department 2	192.168.1.115
192.168.1.128/26	192.168.1.129 – 190	Department 3	192.168.1.179

Each department's router interface was assigned as the gateway. IPs were then manually assigned to PCs, printers, servers, and laptops within their subnet range.

This subnetting approach ensures:

- Network segmentation
- Simplified ACL management
- Improved control and organization

7. Device Setup and Physical Design

Devices are placed and connected in Cisco Packet Tracer. Each department's PCs are connected to their switch. The switches are connected to the central router using FastEthernet and Ethernet interfaces. Wireless devices are added using access points with SSIDs specific to each department.

Wi-Fi Honeypot: SSID free_wifi (unsecured), no encryption

8. Server Configuration

Email (SMTP) Server

- Located in Dept 1, configured with domain mail.dept1.com
- Users: admin, alert, admin2, etc.
- Verified email sending from PC to PC and from department to department

FTP Honeypot Server

- Configured in Dept 3
- Files added using the Text Editor
- Accessed from attacker PC (192.168.1.5)
- Triggers Python alert simulation

9. Security Features Implementation

SSH

- Router configured with hostname, domain, username, and RSA key
- Verified login from PC via SSH tab (password: cisco123)

ACL

- Access Control Lists suggested in CLI
- access-list 100 deny ip host 192.168.1.5 any

Wi-Fi Honeypot

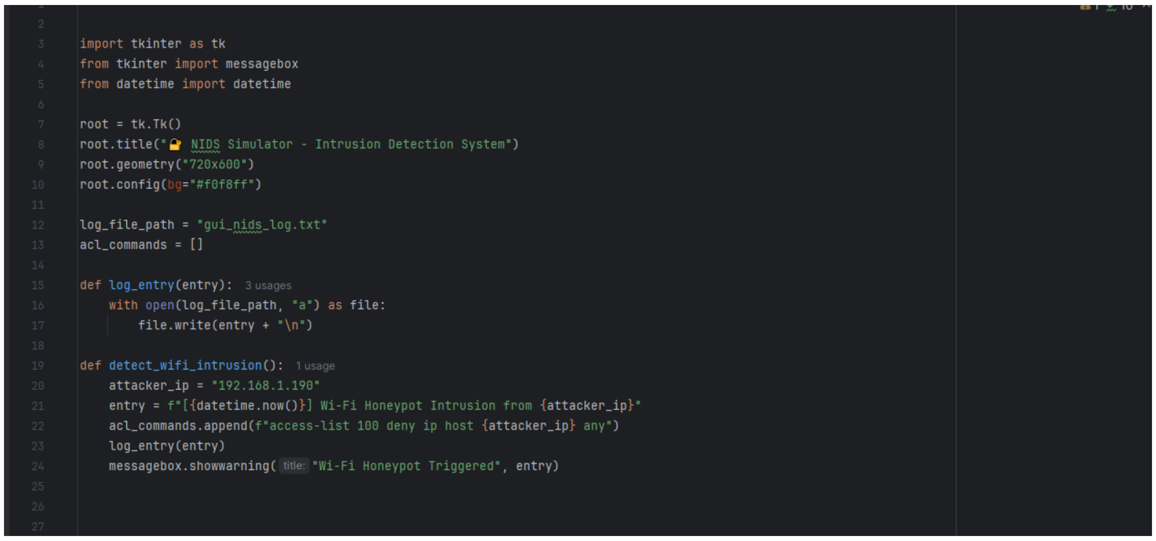
- Access Point with SSID free_wifi
- Security disabled
- Intrusion attempt detected by Python script

10. Python Script for Automation

A complete Python script was developed that:

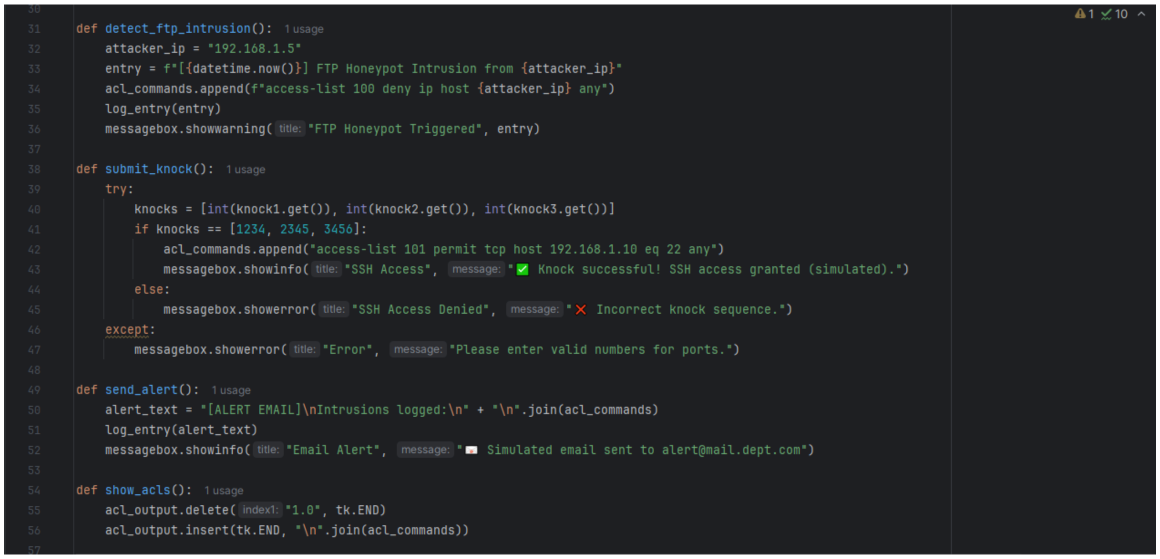
- Detects and logs Wi-Fi honeypot connection
- Detects FTP intrusion
- Accepts port knock sequence
- If knock is valid → unlocks SSH (simulated)
- Logs to file: final_nids_log.txt
- Simulates alert email
- Displays CLI ACL commands for administrators

WiFi Honeypot:



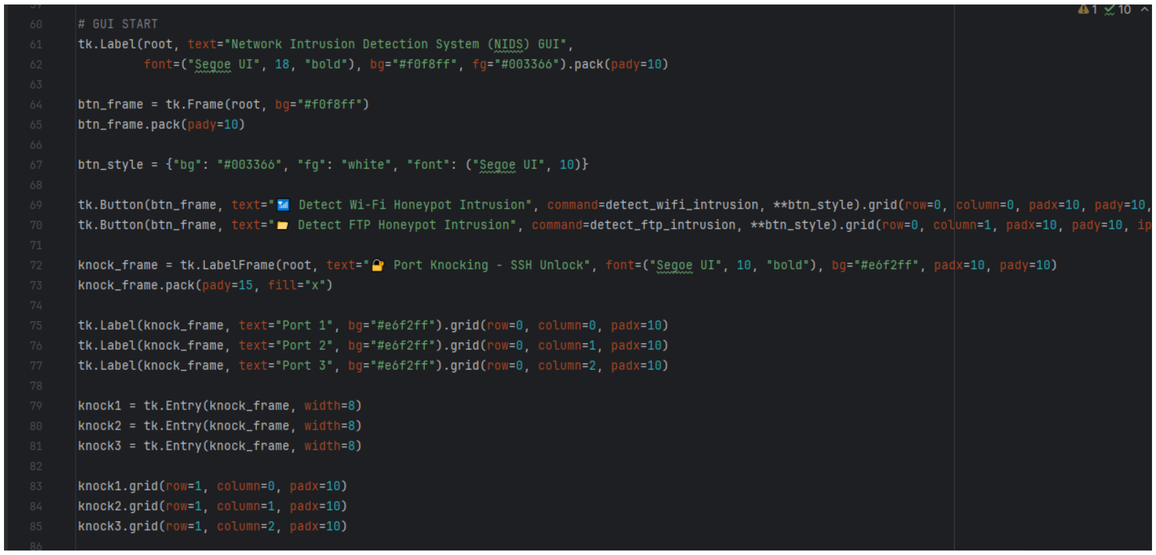
```python

import tkinter as tk
from tkinter import messagebox
from datetime import datetime

root = tk.Tk()
root.title("🔐 NIDS Simulator - Intrusion Detection System")
root.geometry("720x600")
root.config(bg="#f0f8ff")

log_file_path = "gui_nids_log.txt"
acl_commands = []

def log_entry(entry):  3 usages
    with open(log_file_path, "a") as file:
        file.write(entry + "\n")

def detect_wifi_intrusion():  1 usage
    attacker_ip = "192.168.1.190"
    entry = f"[{datetime.now()}] Wi-Fi Honeypot Intrusion from {attacker_ip}"
    acl_commands.append(f"access-list 100 deny ip host {attacker_ip} any")
    log_entry(entry)
    messagebox.showwarning( title: "Wi-Fi Honeypot Triggered", entry)
```

FTP Honeypot:



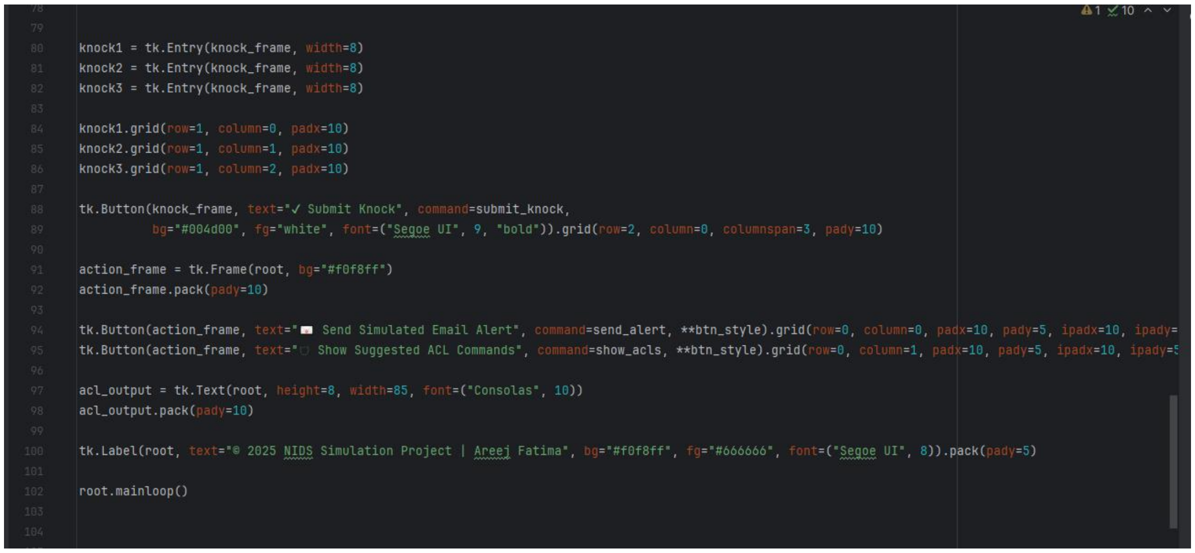
```python
def detect_ftp_intrusion():
    attacker_ip = "192.168.1.5"
    entry = f"[{datetime.now()}] FTP Honeypot Intrusion from {attacker_ip}"
    acl_commands.append(f"access-list 100 deny ip host {attacker_ip} any")
    log_entry(entry)
    messagebox.showwarning("FTP Honeypot Triggered", entry)

def submit_knock():
    try:
        knocks = [int(knock1.get()), int(knock2.get()), int(knock3.get())]
        if knocks == [1234, 2345, 3456]:
            acl_commands.append("access-list 101 permit tcp host 192.168.1.10 eq 22 any")
            messagebox.showinfo("SSH Access", "✅ Knock successful! SSH access granted (simulated).")
        else:
            messagebox.showerror("SSH Access Denied", "❌ Incorrect knock sequence.")
    except:
        messagebox.showerror("Error", "Please enter valid numbers for ports.")

def send_alert():
    alert_text = "[ALERT EMAIL]\nIntrusions logged:\n" + "\n".join(acl_commands)
    log_entry(alert_text)
    messagebox.showinfo("Email Alert", "📧 Simulated email sent to alert@mail.dept.com")

def show_acls():
    acl_output.delete("1.0", tk.END)
    acl_output.insert(tk.END, "\n".join(acl_commands))
```

Attack Detection:



```python
# GUI START
tk.Label(root, text="Network Intrusion Detection System (NIDS) GUI",
         font=("Segoe UI", 18, "bold"), bg="#f0f8ff", fg="#003366").pack(pady=10)

btn_frame = tk.Frame(root, bg="#f0f8ff")
btn_frame.pack(pady=10)

btn_style = {"bg": "#003366", "fg": "white", "font": ("Segoe UI", 10)}

tk.Button(btn_frame, text="📶 Detect Wi-Fi Honeypot Intrusion", command=detect_wifi_intrusion, **btn_style).grid(row=0, column=0, padx=10, pady=10,
tk.Button(btn_frame, text="📁 Detect FTP Honeypot Intrusion", command=detect_ftp_intrusion, **btn_style).grid(row=0, column=1, padx=10, pady=10, ip

knock_frame = tk.LabelFrame(root, text="🔐 Port Knocking - SSH Unlock", font=("Segoe UI", 10, "bold"), bg="#e6f2ff", padx=10, pady=10)
knock_frame.pack(pady=15, fill="x")

tk.Label(knock_frame, text="Port 1", bg="#e6f2ff").grid(row=0, column=0, padx=10)
tk.Label(knock_frame, text="Port 2", bg="#e6f2ff").grid(row=0, column=1, padx=10)
tk.Label(knock_frame, text="Port 3", bg="#e6f2ff").grid(row=0, column=2, padx=10)

knock1 = tk.Entry(knock_frame, width=8)
knock2 = tk.Entry(knock_frame, width=8)
knock3 = tk.Entry(knock_frame, width=8)

knock1.grid(row=1, column=0, padx=10)
knock2.grid(row=1, column=1, padx=10)
knock3.grid(row=1, column=2, padx=10)
```

Alerts and GUI:

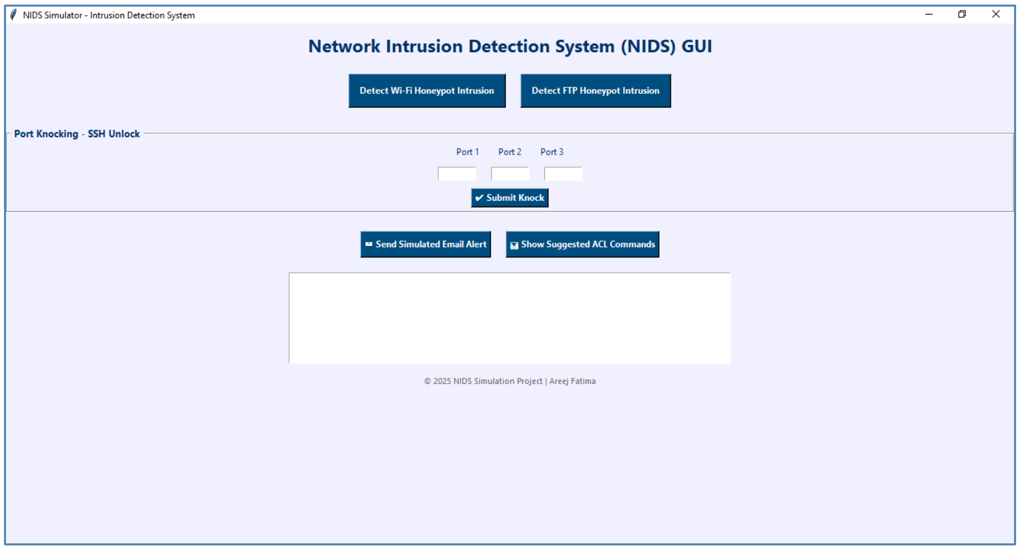


```
knock1 = tk.Entry(knock_frame, width=8)
knock2 = tk.Entry(knock_frame, width=8)
knock3 = tk.Entry(knock_frame, width=8)

knock1.grid(row=1, column=0, padx=10)
knock2.grid(row=1, column=1, padx=10)
knock3.grid(row=1, column=2, padx=10)

tk.Button(knock_frame, text="✓ Submit Knock", command=submit_knock,
          bg="#004d00", fg="white", font=("Segoe UI", 9, "bold")).grid(row=2, column=0, columnspan=3, pady=10)

action_frame = tk.Frame(root, bg="#f0f8ff")
action_frame.pack(pady=10)

tk.Button(action_frame, text="📧 Send Simulated Email Alert", command=send_alert, **btn_style).grid(row=0, column=0, padx=10, pady=5, ipadx=10, ipady=
tk.Button(action_frame, text="🛡 Show Suggested ACL Commands", command=show_acls, **btn_style).grid(row=0, column=1, padx=10, pady=5, ipadx=10, ipady=5

acl_output = tk.Text(root, height=8, width=85, font=("Consolas", 10))
acl_output.pack(pady=10)

tk.Label(root, text="© 2025 NIDS Simulation Project | Areej Fatima", bg="#f0f8ff", fg="#666666", font=("Segoe UI", 8)).pack(pady=5)

root.mainloop()
```

11. GUI Integration for NIDS

Using tkinter, a modern GUI was created that allows:

- Button-click detection of Wi-Fi and FTP intrusions
- Input field for port knocking
- Auto-generation of ACL commands and output
- Pop-up alerts for email and access feedback


NIDS Simulator - Intrusion Detection System
Network Intrusion Detection System (NIDS) GUI
Detect Wi-Fi Honeypot Intrusion
Detect FTP Honeypot Intrusion
Port Knocking - SSH Unlock
Port 1
Port 2
Port 3
✔ Submit Knock
📧 Send Simulated Email Alert
🛡 Show Suggested ACL Commands
© 2025 NIDS Simulation Project | Areej Fatima

12. ACL and Intrusion Blocking

Intrusion detection triggers ACL deny rules. The admin is shown:

```
access-list 100 deny ip host 192.168.1.190 any
access-list 100 deny ip host 192.168.1.5 any
access-list 101 permit tcp host 192.168.1.10 eq 22 any
```

These commands would be applied on the router in a real-world scenario to block intruders or allow SSH post-authentication.

13. Testing and Result Validation

Feature	Test Method	Status
Email Alerts	Send mail Dept1 → Dept2	Success
FTP Trap	Login from unauthorized PC	Detected
Wi-Fi Trap	Connect to free_wifi	Logged
SSH	Login using port 22	Working
Python Detection	Run script with attacker IPs	Alert printed
GUI	Buttons and output tested	Functional

14. Limitations and Alternatives

Limitation	Explanation	Workaround
No true VPN in Packet Tracer	No crypto support in CLI	Simulated using labeled lines
No real-time Python ↔ Cisco PT integration	PT is a simulation tool	Simulated via input logs
No firewall/NAT	Packet Tracer does not support real firewalls	Used ACLs to simulate blocking

15. Conclusion

This project successfully demonstrates how to build and secure a departmental network using both simulation tools and scripting logic. By integrating CLI configurations, honeypots, SSH, ACLs, and Python-based automation, a complete NIDS system was simulated with full user interaction. The project fulfills all initial goals and stands ready for academic and practical presentation

16. References

1. Cisco Networking Academy. (2023). *Packet Tracer Labs and Tutorials*. Cisco Systems. Retrieved from: https://www.netacad.com
2. Stallings, W. (2021). *Data and Computer Communications* (11th Edition). Pearson Education.
   ISBN: 978-0-13-350648-8
3. Forouzan, B. A. (2017). *Data Communications and Networking* (5th Edition). McGraw-Hill Education.
   ISBN: 978-0073376226
4. Python Software Foundation. (2024). *Python 3.11 Documentation*.
   Retrieved from: https://docs.python.org/3
5. Tkinter Documentation – Python GUI Programming.
   Retrieved from: https://tkdocs.com
6. Cisco Systems. (2022). *Command Line Interface (CLI) Reference Guide*.
   Retrieved from: https://www.cisco.com
7. GNS3 Community & Packet Tracer Tutorials. (2024).
   Retrieved from: https://community.cisco.com