# Basic Intrusion Detection System

## Definitions

### 1. **Module**

A **module** in Python is a file that contains Python code (functions, classes, variables) which can be reused in other programs.

- It helps organize code into separate files instead of writing everything in one file.
- Example: `ids.py` is a custom module that contains the IDS logic, which is imported into `main.py`.

---

### 2. **Data Structures**

**Data structures** are ways to store and organize data so that they can be used efficiently.

- **List** → Used to store multiple items in a sequence. Here, we used a list to store simulated network traffic.
- **Dictionary** → Stores data in key–value pairs. Here, we used a dictionary to map **attack types** to their **malicious patterns**.

---

### 3. **Object-Oriented Programming (OOP)**

**OOP** is a programming paradigm based on the concept of "objects", which bundle data (attributes) and behavior (methods).

- **Class** → A blueprint for creating objects.
- **Inheritance** → A class can inherit properties and methods from another class.
- **Polymorphism** → Methods can be redefined (overridden) in child classes to provide different behavior.

In this IDS:

- `Packet` class models network packets.
- `Rule` class inherits from `Packet` and overrides the `is_malicious` method (polymorphism).

---

### 4. **Exception Handling**

**Exception handling** is the process of managing errors in a program so that it doesn't crash unexpectedly.

- Python uses `try`, `except`, and `finally` blocks.
- Example: if an IP address is invalid or a protocol is unsupported, a `ValueError` is raised and caught, so the program continues running instead of stopping.

---

⚡ These four concepts (Modules, Data Structures, OOP, and Exception Handling) together make the **Basic Intrusion Detection System** structured, reusable, and robust.

## File 1: `ids.py`

```
import re

# ----------------------------
# Step 2: Malicious Patterns
# ----------------------------
# Dictionary of known malicious payload patterns
MALICIOUS_PATTERNS = {
    "SQL Injection": r"(?:\bSELECT\b|\bINSERT\b|\bUPDATE\b|\bDELETE\b).*?(?:\bFROM\b|\bWHERE\b)",
    "XSS Attack": r"<script>.*?</script>",
    "Path Traversal": r"(\.\./|\.\.\\)+",
    "Command Injection": r"(;|&&|\||\`)\s*(rm|cat|ls|echo|wget|curl|whoami)"
}

# ----------------------------
# Step 3: Object-Oriented Design
# ----------------------------
class Packet:
    """Represents a network packet with attributes like source IP, destination IP, protocol, and payload."""
```

```
    def __init__(self, source_ip, destination_ip, protocol, payload):
        self.source_ip = source_ip
        self.destination_ip = destination_ip
        self.protocol = protocol
        self.payload = payload

    def is_malicious(self):
        """Checks if the payload contains malicious patterns."""
        for attack_type, pattern in MALICIOUS_PATTERNS.items():
            if re.search(pattern, self.payload, re.IGNORECASE):
                return attack_type  # Return the type of attack detected
        return None


class Rule(Packet):
    """Extends the Packet class to define detection rules with custom alert
s."""

    def is_malicious(self):
        """Custom implementation to check for malicious activity in network tr
affic."""
        attack = super().is_malicious()
        if attack:
            return f"🚨 ALERT: {attack} detected from {self.source_ip} to {self.d
estination_ip}"
        return None


# ----------------------------
# Step 4: Exception Handling
# ----------------------------
def validate_ip(ip_address):
    """Validates if the given IP address is correctly formatted."""
    ip_pattern = r"^\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}$"
    if not re.match(ip_pattern, ip_address):
        raise ValueError(f"Invalid IP address format: {ip_address}")
    return True
```

```
def validate_protocol(protocol):
    """Ensures only TCP or HTTP protocols are accepted."""
    if protocol.upper() not in ["TCP", "HTTP"]:
        raise ValueError(f"Unsupported protocol: {protocol}")
    return True
```

## File 2: main.py

```
from ids import Packet, Rule, validate_ip, validate_protocol

# ----------------------------
# Step 2: Simulated Network Traffic
# ----------------------------
network_traffic = [
    {
        "source_ip": "192.168.1.101",
        "destination_ip": "192.168.1.1",
        "protocol": "HTTP",
        "payload": "SELECT * FROM users WHERE username='admin'"
    },
    {
        "source_ip": "192.168.1.102",
        "destination_ip": "192.168.1.2",
        "protocol": "TCP",
        "payload": "<script>alert('XSS')</script>"
    },
    {
        "source_ip": "192.168.1.103",
        "destination_ip": "192.168.1.3",
        "protocol": "HTTP",
        "payload": "../../etc/passwd"
    },
    {
        "source_ip": "192.168.1.104",
        "destination_ip": "192.168.1.4",
```

```
        "protocol": "HTTP",
        "payload": "echo Hello && rm -rf /"
    },
    {
        "source_ip": "192.168.1.105",
        "destination_ip": "192.168.1.5",
        "protocol": "TCP",
        "payload": "New user"
    }
]

# ----------------------------
# Step 5: Process Traffic and Detect Anomalies
# ----------------------------
for packet_data in network_traffic:
    try:
        # Validate IP and Protocol
        validate_ip(packet_data["source_ip"])
        validate_ip(packet_data["destination_ip"])
        validate_protocol(packet_data["protocol"])

        # Create a Packet Object (Rule subclass used for detection)
        packet = Rule(
            packet_data["source_ip"],
            packet_data["destination_ip"],
            packet_data["protocol"],
            packet_data["payload"]
        )

        # Detect malicious activity
        alert = packet.is_malicious()

        # Display results
        if alert:
            print(alert)
        else:
            print(f"✅ Normal traffic from {packet.source_ip} to {packet.destinat
ion_ip}")
```

```
    except ValueError as e:
        print(f"❌ Error: {e}")

    finally:
        # Optional: log completion of packet check
        pass
```

---

## Sample Output

If you run `python main.py`, you might see:

```
🚨 ALERT: SQL Injection detected from 192.168.1.101 to 192.168.1.1
🚨 ALERT: XSS Attack detected from 192.168.1.102 to 192.168.1.2
🚨 ALERT: Path Traversal detected from 192.168.1.103 to 192.168.1.3
🚨 ALERT: Command Injection detected from 192.168.1.104 to 192.168.1.4
✅ Normal traffic from 192.168.1.105 to 192.168.1.5
```

---

This covers:

✔️ **Modules** (`ids.py`, `main.py`)

✔️ **Data Structures** (list for traffic, dictionary for patterns)

✔️ **OOP** (Packet class, Rule subclass, polymorphism)

✔️ **Exceptions** (invalid IP, protocol check, empty payload)