AI vs. AI: Defending Against Malicious AI Agents

This presentation explores the rising threat of malicious AI agents and AI-driven defense models. We'll cover the problem, solutions, real-world examples, and future enhancements. Prepare to dive into the evolving battle of AI vs AI in cybersecurity.

The Problem: Malicious AI Agents

AI-Powered Attacks

Phishing, malware, and DDoS attacks enhanced with AI techniques.

Adversarial Attacks

Tricking AI systems like image recognition and NLP with crafted inputs.

Data Poisoning

Corrupting training data to weaken AI defenses and breach security.

Deepfake Threats

Used in social engineering attacks, causing increasing concern in 2024.

The Solution: AI-Driven Defense Models

Anomaly Detection
Identifies unusual patterns to flag potential threats automatically.

Adversarial Training
Strengthens AI models by training them against crafted attack examples.

Reinforcement Learning
Enables dynamic and adaptive responses to evolving threats in real-time.

Explainable AI (XAI)
Provides transparency, helping analysts understand AI defense decisions.

Code/Tool Breakdown: Building Defense Systems

Python Libraries

- TensorFlow
- PyTorch
- Scikit-learn

Defense Tools

- Adversarial training code snippets
- AI-powered IDS with Snort and Zeek
- MITRE ATT&CK framework integration

Real-World Use Cases: Defensive AI in Action

Fraud Detection

AI significantly reduces global credit card fraud losses.

Network Security

Identifies and blocks malicious traffic in enterprise networks.

Endpoint Protection

AI-based antivirus software improves threat detection accuracy.

Darktrace Antigena

Prevented WannaCry-style ransomware attacks in 2024 effectively.

Future Enhancements: The Next Frontier

1 **Automated Adversarial Training**
Enables continuous model hardening without manual intervention.

2 **Self-Evolving Defenses**
Uses meta-learning for adaptive, autonomous threat responses.

3 **Quantum Security Integration**
Potentially revolutionizes encryption and AI threat detection.

4 **Decentralized AI**
Blockchain-powered defense systems offer robust tamper resistance.

Challenges and Considerations

Ethical Concerns

Ensuring AI-driven defense respects privacy and rights.

Transparency

Explainability is critical for trust and user confidence.

Bias Mitigation

Addressing biases to prevent unfair or ineffective defenses.

Continuous Adaptation

Monitoring and updating AI defenses to handle new threats.

Conclusion: Embracing AI for a Secure Future

New Cyber Reality

AI vs AI is reshaping cybersecurity battles worldwide.

Proactive Defense

Organizations must adopt AI-based proactive security strategies.

Ongoing Research

Collaboration and innovation are keys to staying ahead.

Q&A and Resources

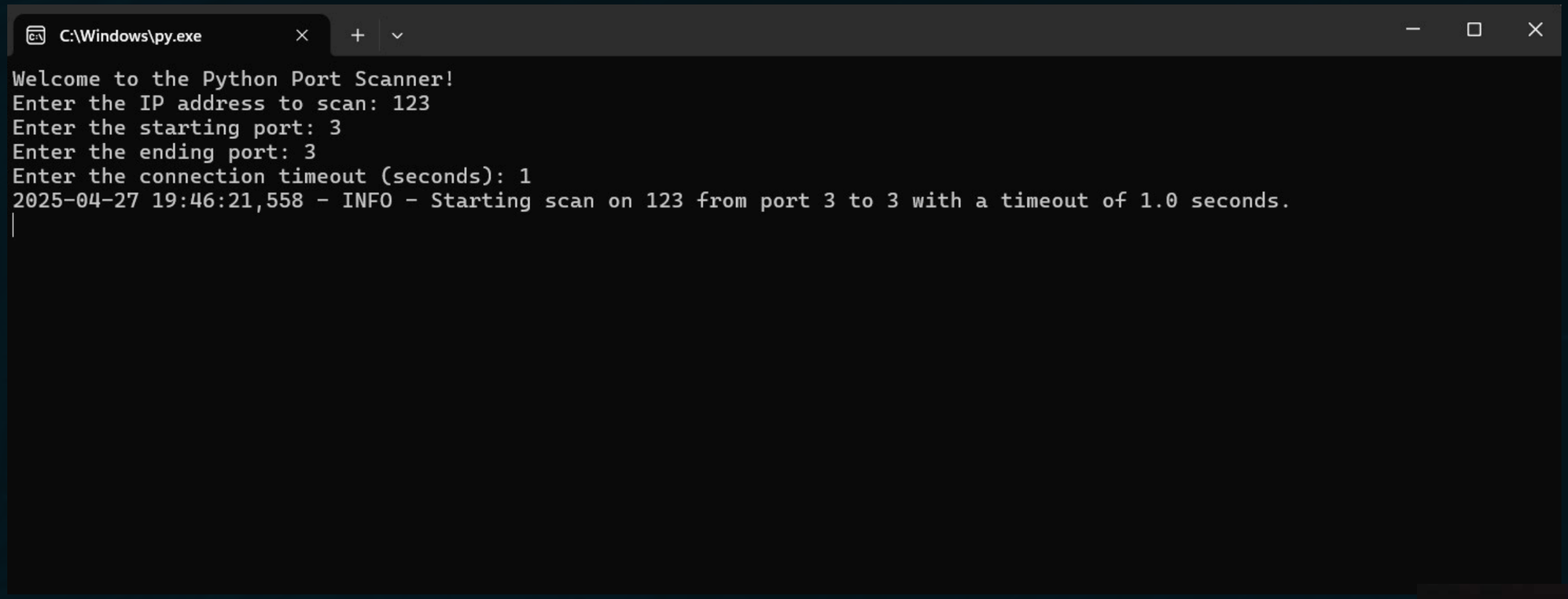
Open for questions, further learning, and discussions.

```python
import socket
import threading
import time
import logging
from concurrent.futures import ThreadPoolExecutor

# Configure logging
logging.basicConfig(level=logging.INFO, format='%(asctime)s - %(levelname)s - %(message)s')

class PortScanner:
    def __init__(self, ip, start_port, end_port, timeout=1.0):
        """
        Initialize the PortScanner with the target IP, port range, and timeout.

        :param ip: Target IP address to scan.
        :param start_port: Starting port number for the scan.
        :param end_port: Ending port number for the scan.
        :param timeout: Timeout for socket connections.
        """
        self.ip = ip
        self.start_port = start_port
        self.end_port = end_port
        self.timeout = timeout
        self.open_ports = []

    def scan_port(self, port):
        """
        Scan a single port on the target IP address.

        :param port: Port number to scan.
        :return: Tuple of (port, service) if open, else (None, None).
        """
        with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as sock:
            sock.settimeout(self.timeout)
            try:
                result = sock.connect_ex((self.ip, port))
                if result == 0:
                    service = self.get_service_name(port)
                    logging.info(f"Port {port} is open (Service: {service})")
                    return port, service
            except socket.error as e:
                logging.error(f"Socket error on port {port}: {e}")
            except Exception as e:
                logging.error(f"Error scanning port {port}: {e}")
        return None, None
```

```python
    @staticmethod
    def get_service_name(port):
        """
        Get the service name for a given port number.

        :param port: Port number.
        :return: Service name or 'Unknown Service'.
        """
        try:
            return socket.getservbyport(port)
        except OSError:
            return "Unknown Service"

    def scan_ports(self):
        """
        Scan the range of ports on the target IP address using multithreading.

        :return: List of open ports and their services.
        """
        with ThreadPoolExecutor(max_workers=100) as executor:
            futures = {executor.submit(self.scan_port, port): port for port in range(self.start_port, self.end_port + 1)}
            for future in futures:
                port, service = future.result()
                if port is not None:
                    self.open_ports.append((port, service))
        return self.open_ports

    def display_results(self):
        """
        Display the results of the port scan.
        """
        if self.open_ports:
            print("\nOpen ports found:")
            for port, service in self.open_ports:
                print(f"Port {port}: {service}")
        else:
            print("No open ports found.")

def main():
    """
    Main function to execute the port scanner.
    """
    print("Welcome to the Python Port Scanner!")
    ip = input("Enter the IP address to scan: ")
    start_port = int(input("Enter the starting port: "))
    end_port = int(input("Enter the ending port: "))
    timeout = float(input("Enter the connection timeout (seconds): "))

    logging.info(f"Starting scan on {ip} from port {start_port} to {end_port} with a timeout of {timeout} seconds.")

    scanner = PortScanner(ip, start_port, end_port, timeout)
```

```python
    def scan_ports(self):
        """
        Scan the range of ports on the target IP address using multithreading.

        :return: List of open ports and their services.
        """
        with ThreadPoolExecutor(max_workers=100) as executor:
            futures = {executor.submit(self.scan_port, port): port for port in range(self.start_port, self.end_port + 1)}
            for future in futures:
                port, service = future.result()
                if port is not None:
                    self.open_ports.append((port, service))
        return self.open_ports

    def display_results(self):
        """
        Display the results of the port scan.
        """
        if self.open_ports:
            print("\nOpen ports found:")
            for port, service in self.open_ports:
                print(f"Port {port}: {service}")
        else:
            print("No open ports found.")

def main():
    """
    Main function to execute the port scanner.
    """
    print("Welcome to the Python Port Scanner!")
    ip = input("Enter the IP address to scan: ")
    start_port = int(input("Enter the starting port: "))
    end_port = int(input("Enter the ending port: "))
    timeout = float(input("Enter the connection timeout (seconds): "))

    logging.info(f"Starting scan on {ip} from port {start_port} to {end_port} with a timeout of {timeout} seconds.")

    scanner = PortScanner(ip, start_port, end_port, timeout)

    start_time = time.time()
    open_ports = scanner.scan_ports()
    end_time = time.time()

    scanner.display_results()
    logging.info(f"Scan completed in {end_time - start_time:.2f} seconds.")

if __name__ == "__main__":
    main()
```

```
Welcome to the Python Port Scanner!
Enter the IP address to scan: 4553
Enter the starting port: 231
Enter the ending port: 122
Enter the connection timeout (seconds): 1
2025-05-12 20:14:11,744 - INFO - Starting scan on 4553 from port 231 to 122 with a timeout of 1.0 seconds.
No open ports found.
2025-05-12 20:14:11,755 - INFO - Scan completed in 0.00 seconds.
```


C:\Windows\py.exe
Welcome to the Python Port Scanner!
Enter the IP address to scan: 123
Enter the starting port: 3
Enter the ending port: 3
Enter the connection timeout (seconds): 1
2025-04-27 19:46:21,558 - INFO - Starting scan on 123 from port 3 to 3 with a timeout of 1.0 seconds.

https://youtu.be/sm8fvVYaT6Q?si=e8_JQhztRY19GCbi

https://agileblue.com/ai-threats-vs-ai-defenses-the-battle-for-cyber-supremacy/

https://right-hand.ai/blog/ai-agents-vs-agentic-ai-in-cybersecurity/

https://www.lesswrong.com/posts/LPHMMMZFAWog6ty5x/ctrl-z-controlling-ai-agents-via-resampling

Github Link:-

https://github.com/Vineet1395/Vineet13