

How to code AI agent for detecting security vulnerability

ChatGPT 의 말:

Creating an AI agent for detecting security vulnerabilities involves data collection, feature engineering, model selection, training, and deployment.

- 1. Define Scope & Threat Model**
- 2. Data Collection & Preprocessing**
- 3. Feature Engineering**
- 4. AI/ML Model Selection**
- 5. Code**

Step 1: Install Dependencies

Step 2: Data Preprocessing

Step 3: Train the AI Model

Step 4: Real-Time Vulnerability Detection

1. Define Scope & Threat Model

Before coding, define the type of vulnerabilities you want to detect:

- Network vulnerabilities (e.g., open ports, misconfigured firewalls)
 - Software vulnerabilities (e.g., SQL injection, buffer overflow)
 - IoT vulnerabilities (e.g., weak authentication, insecure APIs)
 - Cloud vulnerabilities (e.g., misconfigured storage, API abuse)
-

2. Data Collection & Preprocessing

To train an AI model, you need labeled security datasets. Common sources include:

- CVE (Common Vulnerabilities and Exposures) database
- CWE (Common Weakness Enumeration) dataset
- Kaggle cybersecurity datasets
- MITRE ATT&CK framework
- Public penetration testing reports

Data types:

- Log files: System logs, network traffic logs
- Code repositories: Analyzing source code for insecure patterns
- Network traffic: Packet captures (e.g., using Wireshark)

Preprocessing techniques:

- Tokenization for NLP-based vulnerability detection
 - Feature extraction (e.g., opcode sequences, API call patterns)
 - Normalization and one-hot encoding for categorical features
-

3. Feature Engineering

- Static Analysis Features: Code structure, API calls, syntax analysis
- Dynamic Analysis Features: Runtime behavior, memory usage, system calls
- Network Traffic Features: Packet size, destination IPs, protocol types
- Log-Based Features: Timestamps, access logs, error logs

For IoT security, include:

- Device fingerprinting
- Behavior-based anomaly detection
- Cryptographic strength analysis

4. Model Selection

Choose AI/ML models based on your problem type:

Model	Use Case
Random Forest	Simple anomaly detection
Support Vector Machines (SVM)	Malware detection, intrusion detection
Deep Learning (CNN/RNN/LSTMs)	Analyzing network traffic, IoT vulnerabilities
Transformer-based NLP (BERT, GPT)	Code vulnerability detection
Graph Neural Networks (GNNs)	Identifying security issues in large software dependencies
Reinforcement Learning (RL)	Adaptive penetration testing

5. AI agent code model

An example of a simple AI model for detecting software vulnerabilities using NLP-based classification:

Step 1: Install Dependencies

```
bash
pip install scikit-learn pandas numpy tensorflow torch transformers
```

Step 2: Data Preprocessing

```
python
복사편집
```

```

import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

# Load vulnerability dataset
df = pd.read_csv("vulnerability_dataset.csv") # Assume it contains "code" and
"label" columns
X = df["code"]
y = df["label"]

# Convert text data to numerical format using TF-IDF
vectorizer = TfidfVectorizer(stop_words='english', max_features=5000)
X_vectorized = vectorizer.fit_transform(X)

# Split dataset
X_train, X_test, y_train, y_test = train_test_split(X_vectorized, y, test_size=0.2,
random_state=42)

```

Step 3: Train the AI Model

```

python
복사편집
# Train a simple Random Forest classifier
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

# Evaluate performance
y_pred = model.predict(X_test)
print(f'Accuracy: {accuracy_score(y_test, y_pred):.2f}')

```

Step 4: Real-Time Vulnerability Detection

```

python
복사편집
def detect_vulnerability(code_snippet):
    input_vectorized = vectorizer.transform([code_snippet])
    prediction = model.predict(input_vectorized)
    return "Vulnerable" if prediction == 1 else "Secure"

# Example usage
sample_code = "char buffer[10]; strcpy(buffer, user_input);" # Example of buffer
overflow
print(detect_vulnerability(sample_code))

```

6. Advanced Enhancements

To improve accuracy and real-world applicability:

1. **Deep Learning Approach:** Use BERT for analyzing vulnerabilities in code.
 2. **Reinforcement Learning:** Train an agent to find vulnerabilities dynamically.
 3. **Integrate with SIEM systems:** Collect logs and detect real-time threats.
 4. **Deploy in a cloud environment:** Use AWS Lambda or Azure Security Center.
-

7. Deployment & Integration

- **Containerize:** Use Docker to deploy in cloud environments.
 - **API Exposure:** Create a REST API using Flask or FastAPI.
 - **CI/CD Integration:** Automate updates with GitHub Actions.
-
-

AI driven code for detecting security vulnerability of IP127.0.0.1 using AI agent

ChatGPT 의 말:

Here's a basic AI-driven Python script using an AI agent to detect security vulnerabilities on 127.0.0.1. This script leverages OpenAI's transformers for analyzing responses and scapy for network scanning. You can extend it with deep learning-based models for more advanced analysis.

Ai Vuln Detection

대신 채팅에서 답합니다

Explanation:

1. **Port Scanning:** Uses scapy to scan common ports on 127.0.0.1.
2. **AI Analysis:** Sends open ports to an AI model for vulnerability analysis.
3. **Output Report:** Provides security insights and recommendations.

Next Steps:

- Integrate with **Shodan API** for real-world vulnerability assessments.
- Enhance with **ML-based anomaly detection**.
- Automate **patch recommendations** based on CVEs.

Would you like any additional functionality, such as deeper packet inspection or integration with threat intelligence feeds?

```
import scapy.all as scapy
```

```
import openai
```

```
import re
```

```
def scan_ports(target_ip):
```

```
    """Scan common ports on the target IP."""
```

```
    open_ports = []
```

```
    common_ports = [22, 80, 443, 8080, 3306] # Expand as needed
```

```
    for port in common_ports:
```

```
        syn_packet = scapy.IP(dst=target_ip)/scapy.TCP(dport=port, flags="S")
```

```
        response = scapy.sr1(syn_packet, timeout=1, verbose=0)
```

```
        if response and response.haslayer(scapy.TCP) and  
response.getlayer(scapy.TCP).flags == 0x12:
```

```
            open_ports.append(port)
```

```
    return open_ports
```

```
def analyze_vulnerabilities(open_ports):
```

```
    """Use an AI model to analyze security vulnerabilities based on open  
ports."""
```

```
    prompt = f"Analyze security vulnerabilities for open ports: {open_ports}.  
Provide recommendations."
```

```
    response = openai.ChatCompletion.create(  
        model="gpt-4",  
        messages=[{"role": "system", "content": "You are a cybersecurity  
expert."},  
                  {"role": "user", "content": prompt}]  
    )  
    return response["choices"][0]["message"]["content"]
```

```
def detect_vulnerabilities(target_ip="127.0.0.1"):  
    """Main function to detect security vulnerabilities."""  
    open_ports = scan_ports(target_ip)  
    if open_ports:  
        analysis = analyze_vulnerabilities(open_ports)  
        print("AI Security Analysis:\n", analysis)  
    else:  
        print("No open ports detected. System appears secure.")  
  
if __name__ == "__main__":  
    detect_vulnerabilities()
```
