

Proyecto Final

Bootcamp Ciberseguridad

Keepcoding

Fecha: 17 de Julio de 2024

Autor: **Azael Ramírez Pérez**

Mail: **keepcoder_test@gmail.com** (ficticio)

Empresa: **KeepCoder.inc** (ficticio)

Contenido

Objetivo.....	3
Alcance.....	5
Análisis de la muestra de Malware.....	6
Base de Conocimiento.....	10
App GenAI.....	11
Anexos.....	16

Objetivo

El panorama de las ciberamenazas está en constante cambio, y el malware moderno se ha vuelto cada vez más sofisticado, lo que presenta enormes desafíos para su análisis y neutralización.

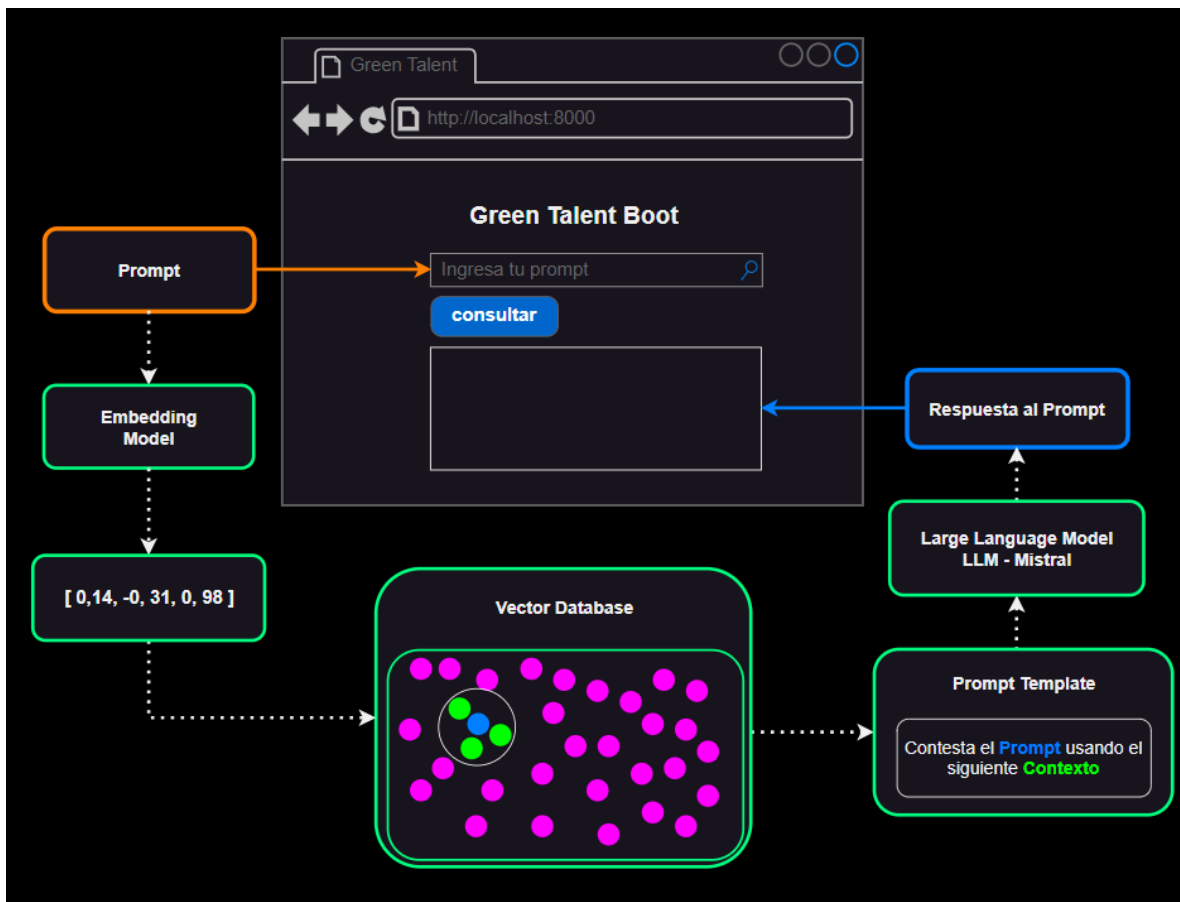
El malware actual no solo es más complejo en su código, sino que también emplea técnicas de ofuscación y evasión más elaboradas para eludir los métodos de detección tradicionales. Esto dificulta la comprensión de su funcionamiento y la identificación de sus puntos vulnerables.

El presente trabajo propone aprovechar las capacidades de la **Inteligencia Artificial Generativa (GenAI)** para identificar y analizar en profundidad 3 aspectos principales de una muestra de malware:

- **Datos Generales (SHA256):** Identificar el hash único del malware para su posterior rastreo y clasificación.
- **Tácticas y Técnicas de Mitre Attack:** Relacionar el malware con las tácticas y técnicas conocidas del marco MITRE ATT&CK, permitiendo comprender su modus operandi y objetivos.
- **Comportamiento y Capacidades Maliciosas:** Analizar el comportamiento del malware en ejecución, identificando sus capacidades maliciosas, como la propagación, la extracción de datos, la exfiltración de información o la toma de control de sistemas.

Componentes de la aplicación

- Documentos que conformaran la Base de conocimiento
- Embedding Model (hkunlp/instructor-large)
 - Prompts
 - Documentos
- Base de Datos Vectorial
- Backed (Python)
- SDK AWS
- Bedrock AWS
- LLM - Mistral AI
- FrontEnd (uso de frameworks)



Alcance

Es cierto que el Malware tiene esa capacidad de para propagarse a gran velocidad a través de redes y dispositivos conectados lo cual representa un grave peligro. En cuestión de horas, una nueva amenaza puede infectar miles de sistemas, causando daños considerables y dificultando la contención.

Debido a ello es importante poder utilizar nuevas tecnologías, estrategias innovadoras y la colaboración global es fundamental para proteger a las personas, las empresas y las infraestructuras críticas de las ciberamenazas en constante cambio.

A continuación, listamos los beneficios que presenta el desarrollo de la aplicación que se ha propuesto usando Inteligencia Artificial Generativa (GenAI):

- **Automatización de tareas:** GenAI puede automatizar tareas repetitivas de análisis de malware, liberando tiempo a los analistas para que se centren en tareas más complejas.
- **Automatización de informes:** GenAI puede automatizar la creación de informes detallados acerca del análisis dinámico realizado previamente del malware, esto brinda la posibilidad de determinar el comportamiento y las capacidades del malware, facilitando su neutralización y prevención de futuros ataques.

Análisis de la muestra de Malware

El análisis de muestras de malware es un proceso crucial en la lucha contra las ciberamenazas. Permite a los investigadores y profesionales de la seguridad cibernética comprender mejor el funcionamiento del malware, identificar sus objetivos y desarrollar estrategias efectivas para combatirlo.

Para llevar a cabo el análisis de las diferentes muestras de malware realizamos las siguientes actividades:

1. Uso de un ambiente Sandbox configurado con CAPE
2. Uso de la herramienta de Mandiant (CAPA)
3. Programación de un script de Python para parsear los resultados que arroja CAPA para poder construir la base de conocimiento

Uso de un ambiente Sandbox configurado con CAPE

El objetivo de esta actividad fue analizar una muestra de malware en un entorno controlado (sandbox) para comprender su comportamiento y obtener información valiosa para su neutralización.

Herramienta utilizada: CAPE (Common Analysis Platform for Execution Environments), una herramienta de código abierto para automatizar el análisis de archivos sospechosos.

Metodología:

- 1. Preparación del entorno:** Se configuró un entorno sandbox aislado para ejecutar la muestra de malware de forma segura.
- 2. Ejecución de la muestra:** Se utilizó CAPE para ejecutar la muestra de malware en el entorno sandbox y monitorizar su comportamiento.
- 3. Recopilación de datos:** CAPE recopiló información detallada sobre el comportamiento del malware, incluyendo las acciones realizadas, los recursos utilizados y las interacciones con el sistema.
- 4. Generación de informe:** CAPE generó un informe en formato JSON que contiene la información recopilada durante el análisis.

Uso de la herramienta de Mandiant (CAPA)

Se automatizó el uso de Capa mediante un script de Python para extraer del análisis realizado con Capa los principales comportamientos de la muestra de malware. Esto permite agilizar el proceso de análisis y obtener información relevante de manera más rápida.

Mandiant ha presentado la versión 7.0 de Capa, una herramienta de código abierto para analizar programas maliciosos. Esta nueva versión incorpora la capacidad de identificar las capacidades de los programas a partir de informes de análisis dinámico generados a través de CAPE Sandbox.

Principales características de Capa 7.0:

- **Identificación de capacidades:** Capa 7.0 permite identificar las capacidades de un programa, como la capacidad de robar datos, instalar malware o comunicarse con servidores remotos.
- **Informes detallados:** Capa 7.0 genera informes detallados que contienen información sobre las capacidades del programa y su comportamiento.
- **Marco para la comunidad:** Capa 7.0 proporciona un marco para que la comunidad pueda reconocer y compartir los comportamientos que ha encontrado en diversas muestras de malware.

Beneficios de utilizar CAPA 7.0:

- **Mejora del análisis de malware:** Capa 7.0 permite a los analistas comprender mejor el comportamiento del malware y sus capacidades.
- **Detección más precisa:** La identificación de las capacidades del malware puede ayudar a detectar y clasificar el malware con mayor precisión.
- **Investigación más rápida:** Capa 7.0 puede automatizar el proceso de análisis de malware, lo que permite a los investigadores dedicar más tiempo a la investigación y la respuesta.

Programación de un script de Python para procesar los resultados de CAPA para poder construir la base de conocimiento.

Se diseñó un script de Python para procesar los resultados de CAPA y poder construir una base de conocimiento, esta última es una herramienta valiosa para mejorar la eficiencia del análisis de malware y la investigación de ciberamenazas. La información almacenada en la base de conocimiento puede ser utilizada para diversos fines, como la identificación de patrones, el desarrollo de detecciones más precisas y la investigación forense de incidentes de seguridad.

Beneficios de utilizar CAPA 7.0:

- **Automatización del proceso:** El script automatiza la tarea repetitiva de procesar los resultados de CAPA, liberando tiempo a los analistas para que se centren en tareas más complejas.
- **Mejora de la eficiencia:** El procesamiento automatizado permite analizar y almacenar grandes cantidades de datos de manera eficiente.
- **Construcción de una base de conocimiento valiosa:** La información almacenada en la base de conocimiento puede ser utilizada para:
 - **Investigación de malware:** Identificar patrones, tendencias y nuevas amenazas.
 - **Mejora de las detecciones de malware:** Desarrollar firmas y detecciones más precisas.
 - **Investigación forense:** Analizar incidentes de seguridad y determinar el origen y el impacto del malware.

Base de Conocimiento

En el panorama actual de las ciberamenazas, la necesidad de contar con herramientas de seguridad robustas y eficientes es más apremiante que nunca. En este contexto, la integración de Modelos de Lenguaje Grande (LLM) como Mistral IA junto a herramientas de análisis de malware como CAPE abre un abanico de posibilidades para fortalecer la protección de nuestras aplicaciones, infraestructuras, empresas y personas.

Al combinar el análisis detallado que proporciona CAPE sobre una muestra de malware con las capacidades de procesamiento de lenguaje y generación de conocimiento de Mistral IA, podemos construir una base de conocimiento externa que enriquece el entrenamiento del LLM. Esta base de conocimiento, alimentada con información actualizada sobre las últimas amenazas y vulnerabilidades, permite a Mistral IA tomar decisiones más precisas y oportunas para proteger nuestros sistemas.

Esta sinergia entre LLM y herramientas de análisis de malware representa un salto cualitativo en la lucha contra las ciberamenazas. La capacidad de Mistral IA para procesar grandes volúmenes de datos, identificar patrones y generar insights relevantes nos permite anticiparnos a las amenazas, detectarlas de forma más rápida y precisa, y tomar medidas preventivas más efectivas.

App GenAI

La aplicación que se ha desarrollado implementa IA y estamos convencidos que esta tecnología es un aliado invaluable en la lucha contra el malware, proporcionando una comprensión profunda de su funcionamiento, objetivos y capacidades. Esta información es esencial para desarrollar estrategias de seguridad efectivas, proteger nuestros sistemas y defendernos contra las ciberamenazas en constante evolución.

La inteligencia artificial (IA) se presenta como una herramienta poderosa para desentrañar los misterios del malware y fortalecer nuestras defensas digitales.

Alcanzando una comprensión profunda del malware:

Mediante el análisis impulsado por IA, podemos identificar diferentes muestras de malware y descifrar sus objetivos. Este análisis profundo nos permite identificar datos muy importantes del malware, a continuación, se abordan en 4 rubros.

Datos generales de la muestra:

- **Función hash:** Un identificador único que permite rastrear y clasificar el malware.
- **Sistema operativo:** El sistema operativo al que apunta el malware.
- **Formato:** El formato del archivo que contiene el malware (ejecutable, script, etc.).
- **Arquitectura:** La arquitectura del procesador para la que está diseñado el malware.

Datos MITRE ATT&CK:

- **Tácticas, técnicas y procedimientos (TTP):** Un marco de referencia global que clasifica las técnicas utilizadas por los atacantes.
- **Conexión con amenazas conocidas:** Comparación del malware con TTP conocidos para identificar posibles actores y métodos de ataque.

Objetivos y comportamientos de MBC:

- **Intención del actor de la amenaza:** Determinar los objetivos que persigue el malware, como el robo de datos, la interrupción del servicio o el espionaje.
- **Acciones maliciosas:** Identificar las acciones específicas que realiza el malware para lograr sus objetivos.

Descubriendo las capacidades del malware:

La IA nos permite desvelar las capacidades del malware, comprendiendo las habilidades y funciones que posee para causar daño. Entre estas capacidades se encuentran:

- **Mecanismos de infección:** Cómo el malware logra infiltrarse en un sistema.
- **Evasión de detección:** Técnicas utilizadas para evitar ser detectado por antivirus y otras herramientas de seguridad.
- **Persistencia:** Métodos para mantenerse en un sistema infectado incluso después de un reinicio o una eliminación parcial.

- Comunicación y control remoto: Cómo el malware se comunica con sus operadores y recibe instrucciones.
- Daños potenciales: El tipo de daño que puede causar el malware, como la pérdida de datos, la corrupción de archivos o el control del sistema.

Funcionalidad de la aplicación

Ingresando el prompt a consultar en la aplicación

Green Talent

CAPE + CAPA + Python + RAG + ChromaDB + LangChain + AWS Bedrock + Mistral + Streamlit

Ingresa tu prompt:

Actúa como analista experto en ciberseguridad, ayúdame creando un informe en el que indiques lo siguientes rubros:

- explícame 3 tácticas y técnicas de MITRE ATT&CK
- explícame 3 comportamientos críticos (MBC Behavior)
- explícame 3 habilidades maliciosas

El hash: sha256 076abc443c05871e2e638ec146f791b06084aafb9dc200410c6aaaacef934239 es el que debes analizar, finalmente dame 3 recomendaciones para poder detectar este malware en mi infraestructura.

Enviar

Consultando...

Respuesta al prompt

CAPE + CAPA + Python + RAG + ChromaDB + LangChain + AWS Bedrock + Mistral + Streamlit

Ingresa tu prompt:

Actúa como analista experto en ciberseguridad, ayúdame creando un informe en el que indiques lo siguientes rubros:

- explícame 3 tácticas y técnicas de MITRE ATT&CK
- explícame 3 comportamientos críticos (MBC Behavior)
- explícame 3 habilidades maliciosas

El hash: sha256 076abc443c05871e2e638ec146f791b06084aafb9dc200410c6aaaaacef934239 es el que debes analizar, finalmente dame 3 recomendaciones para poder detectar este malware en mi infraestructura.

Enviar

Informe de Análisis de Malware

Hash Analizado: SHA256 - 076abc443c05871e2e638ec146f791b06084aafb9dc200410c6aaaaacef934239

1. Tácticas y Técnicas de MITRE ATT&CK:

a. Defense Evasion - File and Directory Permissions Modification (T1222): El malware modifica los permisos de archivos y directorios para evadir la detección y análisis de seguridad.

b. Defense Evasion - Obfuscated Files or Information (T1027): El malware utiliza ofuscación para ocultar archivos o información, dificultando así su detección y análisis.

c. Discovery - System Information Discovery (T1082): El malware recopila información del sistema, como la versión del sistema operativo, arquitectura, y otros detalles, para adaptar sus acciones y mejorar su eficacia.

2. Comportamientos Críticos (MBC Behavior):

a. ANTI-BEHAVIORAL ANALYSIS - Virtual Machine Detection (B0009): El malware realiza comprobaciones para detectar si se está ejecutando en un entorno de máquina virtual, con el fin de evadir el análisis.

b. COMMAND AND CONTROL - C2 Communication::Receive Data (B0030.002): El malware recibe datos desde un servidor de comando y control, lo que indica que está bajo el control de un atacante remoto.

c. COMMUNICATION - HTTP Communication (C0002): El malware utiliza el protocolo HTTP para comunicarse con servidores remotos, lo que le permite transferir datos y recibir instrucciones.

3. Habilidades Maliciosas:

a. check for unmoving mouse cursor: El malware verifica si el cursor del ratón no se ha movido durante un período de tiempo, lo que sugiere que se está ejecutando en un entorno de análisis automatizado (sandbox).

b. receive data: El malware tiene la capacidad de recibir datos desde un servidor remoto, lo que indica que puede recibir instrucciones adicionales o cargas útiles.

c. communicate using DNS and HTTP: El malware utiliza los protocolos DNS y HTTP para comunicarse con servidores remotos, lo que le permite transferir datos y recibir instrucciones.

Recomendaciones para detectar el malware:


1. Implementar herramientas de seguridad que monitoreen y analicen el tráfico de red en busca de actividades sospechosas, como comunicaciones con servidores de comando y control conocidos o patrones de tráfico inusuales.
2. Utilizar soluciones de seguridad que incluyan análisis de comportamiento y detección de amenazas avanzadas, capaces de identificar técnicas de ofuscación y evasión utilizadas por el malware.
3. Mantener actualizados los sistemas y aplicaciones, así como las firmas de las soluciones de seguridad, para garantizar una protección adecuada contra las últimas amenazas y vulnerabilidades.

Respuesta entregada.

Anexos

Análisis dinámico en el entorno Sandbox usando CAPE

- Reporte que nos arroja CAPE de la muestra **SmokeLoader**

Detection(s): **SmokeLoader**

Sandbox Info

Category	Started On	Completed On	Duration	Cuckoo Version
FILE	2023-12-02 00:12:37	2023-12-02 00:18:14	337 seconds	2.4-CAPE

Machine	Label	Manager	Started On	Shutdown On
win7	win7	KVM	2023-12-02 00:12:37	2023-12-02 00:18:14

Malware config(s)

Type	SmokeLoader Config						
C2s	<table><tbody><tr><td>http://go-piratia.ru/tmp/index.php</td></tr><tr><td>http://humydrrole.com/tmp/index.php</td></tr><tr><td>http://pirateking.online/tmp/index.php</td></tr><tr><td>http://piratia.pw/tmp/index.php</td></tr><tr><td>http://trunk-co.ru/tmp/index.php</td></tr><tr><td>http://weareelight.com/tmp/index.php</td></tr></tbody></table>	http://go-piratia.ru/tmp/index.php	http://humydrrole.com/tmp/index.php	http://pirateking.online/tmp/index.php	http://piratia.pw/tmp/index.php	http://trunk-co.ru/tmp/index.php	http://weareelight.com/tmp/index.php
http://go-piratia.ru/tmp/index.php							
http://humydrrole.com/tmp/index.php							
http://pirateking.online/tmp/index.php							
http://piratia.pw/tmp/index.php							
http://trunk-co.ru/tmp/index.php							
http://weareelight.com/tmp/index.php							

File Details

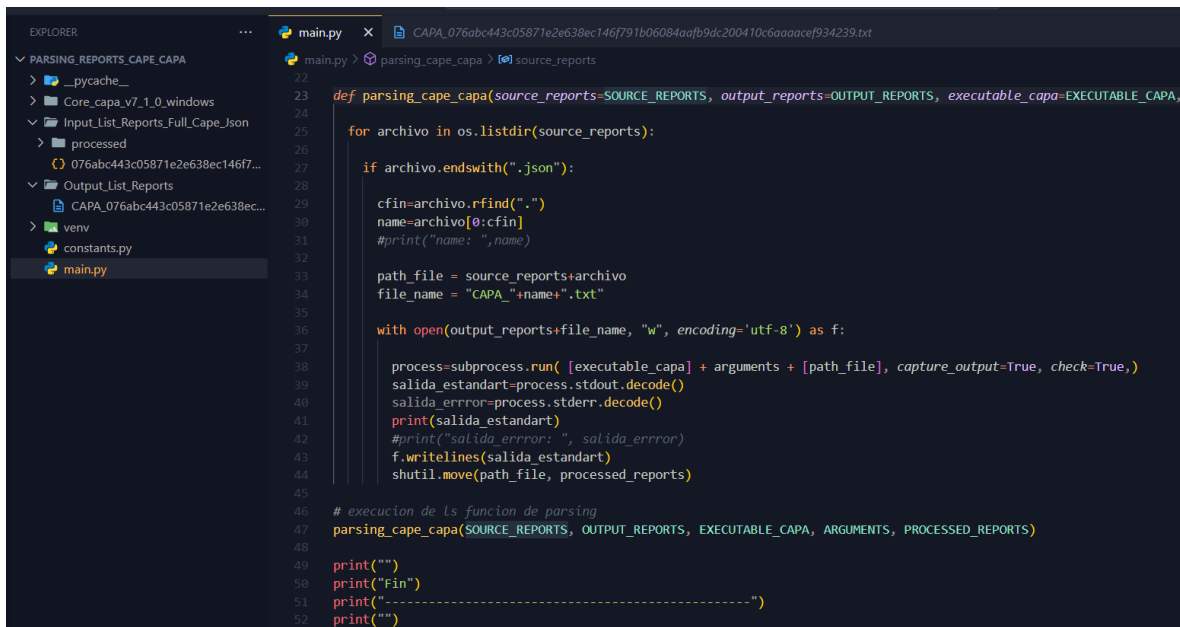
Filename	3928.exe
File Type	PE32 executable (GUI) Intel 80386, for MS Windows
File Size	311808 bytes
MD5	e0e783bba2f8e3f0d2da2bded27eceed
SHA1	a723dea176c400de9bdd169b703eb283032ed2cb
SHA256	076abc443c05871e2e638ec146f791b06084aafb9dc200410c6aaaacef934239 [VT] [MWDB] [Bazaar]
SHA3-384	47e1eea30594c469186f60ea30931c83c81d927936354ffe3e6e6e810e8ea85eb97f82b9b59fb53228fdaf50ea80cc87
CRC32	1D33105F
TLSH	T112641A1392A17C55E9164B328E1FC6E8761EFB518F7D77AA1218AB2F04731B2C263F11
Ssdeep	6144:f3JXcSiTU5/67tt575Wh5oLzrFV2qIpp69a:f3/5/67ttD8Czr+5II

Reporte en formato Json de la muestra SmokeLoader

usando CAPE

```
report.json
{
  "decompression": [],
  "target": {
    "category": "file",
    "file": {
      "name": "3928.exe",
      "path": "/opt/CAPEv2/storage/binaries/076abc443c05871e2e638ec146f791b06084aafb9dc200410c6aaaaacef934239",
      "guest_paths": "",
      "size": 311808,
      "crc32": "1D33105F",
      "md5": "e0e783bba2f8e3f0d2da2bde27eceed",
      "sha1": "a723dea176c400de9hdd169b703eh283032ed2ch",
      "sha256": "076abc443c05871e2e638ec146f791b06084aafb9dc200410c6aaaaacef934239",
      "sha512": "d9908973f5cba81e6ffda944a4f0a968b2306f7945fea0bcdbf8f1d1c086b4c586277a33213971eee6b98665de08fb9e942baae23f6960f12651ec5ffeb595be",
      "rh_hash": null,
      "ssdeep": "6144:f3JXc5iTu5/67tt575Wh5oLzrFV2oIp69a:f3/5/67ttD8Czr+SII",
      "type": "PE32 executable (GUI) Intel 80386, for MS Windows",
      "yara": [],
      "cape_yara": [],
      "clamav": [],
      "tlsh": "T112641A1392A17C55E9164B328E1FC6E8761EFB518F7D77AA1218AB2F04731B2C263F11",
      "sha3_384": "47e1eea30594c469186f60ea30931c83c81d927936354ffe3e6e6e810e8ea85eb97f82b9b59fb53228fdaf50ea80cc87",
      "pe": {
        "guest_signers": {
          "aux_shal": null,
          "aux_timestamp": null,
          "aux_valid": false,
          "aux_error": true,
          "aux_error_desc": "No signature found.",
          "aux_signers": []
        },
        "digital_signers": [],
        "imagebase": "0x00400000",
        "entrypoint": "0x0000c3a",
        "ep_bytes": "e8f4390000e989feffff8bfff558bec8b",
        "peid_signatures": null,
        "reported_checksum": "0x004d042",
        "actual_checksum": "0x004d042",
        "osversion": "5.1",
        "pdbpath": null,
        "imports": {
          "KERNEL32": {
            "dll": "KERNEL32.dll",
            "imports": [
              {
                "address": "0x401000",
                "name": "GetComputerNameA"
              },
              {
                "address": "0x401004",
                "name": "CreateFileA"
              }
            ]
          }
        }
      }
    }
  }
}
```

Procesamiento del reporte de CAPE (formato Json) de la muestra SmokeLoader usando la herramienta de Mandiant (CAPA) a través de Python



```
22
23 def parsing_cape_capa(source_reports=SOURCE_REPORTS, output_reports=OUTPUT_REPORTS, executable_capa=EXECUTABLE_CAPA,
24
25
26
27     if archivo.endswith(".json"):
28
29         cfin=archivo.rfind(".")
30         name=archivo[0:cfin]
31         #print("name: ",name)
32
33         path_file = source_reports+archivo
34         file_name = "CAPA_"+name+".txt"
35
36         with open(output_reports+file_name, "w", encoding='utf-8') as f:
37
38             process=subprocess.run( [executable_capa] + arguments + [path_file], capture_output=True, check=True,)
39             salida_estandart=process.stdout.decode()
40             salida_error=process.stderr.decode()
41             print(salida_estandart)
42             #print("salida error: ", salida_error)
43             f.writelines(salida_estandart)
44             shutil.move(path_file, processed_reports)
45
46 # ejecucion de ls funcion de parsing
47 parsing_cape_capa(SOURCE_REPORTS, OUTPUT_REPORTS, EXECUTABLE_CAPA, ARGUMENTS, PROCESSED_REPORTS)
48
49 print("")
50 print("Fin")
51 print("-----")
52 print("")
```


DISCOVERY	Account Discovery T1087 File and Directory Discovery T1083 Process Discovery T1057 Query Registry T1012 Software Discovery T1518 System Information Discovery T1082 System Location Discovery::System Language Discovery T1614.001 System Network Configuration Discovery T1016 System Owner/User Discovery T1033
EXECUTION	Command and Scripting Interpreter T1059 Shared Modules T1129
MBC Objective	MBC Behavior
ANTI-BEHAVIORAL ANALYSIS	Virtual Machine Detection [B0009] Virtual Machine Detection::Human User Check [B0009.012]
COMMAND AND CONTROL	C2 Communication::Receive Data [B0030.002] C2 Communication::Send Data [B0030.001]
COMMUNICATION	DNS Communication::Resolve [C0011.001] HTTP Communication [C0002] HTTP Communication::Create Request [C0002.012] HTTP Communication::Get Response [C0002.017] HTTP Communication::Send Request [C0002.003] HTTP Communication::WinHTTP [C0002.008]

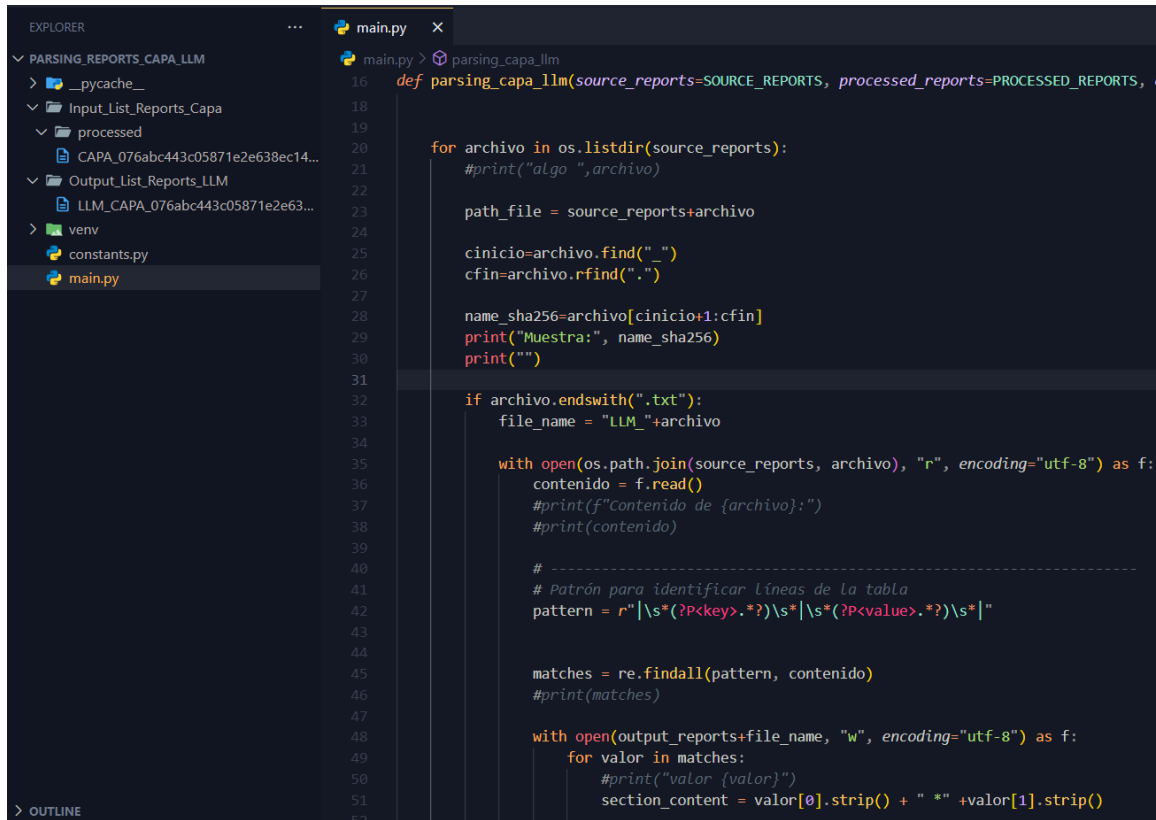
COMMUNICATION	DNS Communication::Resolve [C0011.001] HTTP Communication [C0002] HTTP Communication::Create Request [C0002.012] HTTP Communication::Get Response [C0002.017] HTTP Communication::Send Request [C0002.003] HTTP Communication::WinHTTP [C0002.008] Socket Communication::Connect Socket [C0001.004] Socket Communication::Create TCP Socket [C0001.011] Socket Communication::Receive Data [C0001.006] Socket Communication::Send Data [C0001.007] Socket Communication::Set Socket Config [C0001.001] Socket Communication::TCP Client [C0001.008]
CRYPTOGRAPHY	Cryptographic Hash [C0029] Cryptographic Hash::MD5 [C0029.001] Encryption Key [C0028] Generate Pseudo-random Sequence::Use API [C0021.003]
DISCOVERY	File and Directory Discovery [E1083] System Information Discovery [E1082]
EXECUTION	Command and Scripting Interpreter [E1059]
FILE SYSTEM	Copy File [C0045] Delete File [C0047] Get File Attributes [C0049] Read File [C0051] Set File Attributes [C0050]
MEMORY	Allocate Memory [C0007]
OPERATING SYSTEM	Registry::Query Registry Key [C0036.005] Registry::Query Registry Value [C0036.006] Registry::Set Registry Key [C0036.001]

OPERATING SYSTEM	Registry::Query Registry Key [C0036.005] Registry::Query Registry Value [C0036.006] Registry::Set Registry Key [C0036.001]
PROCESS	Create Thread [C0038]

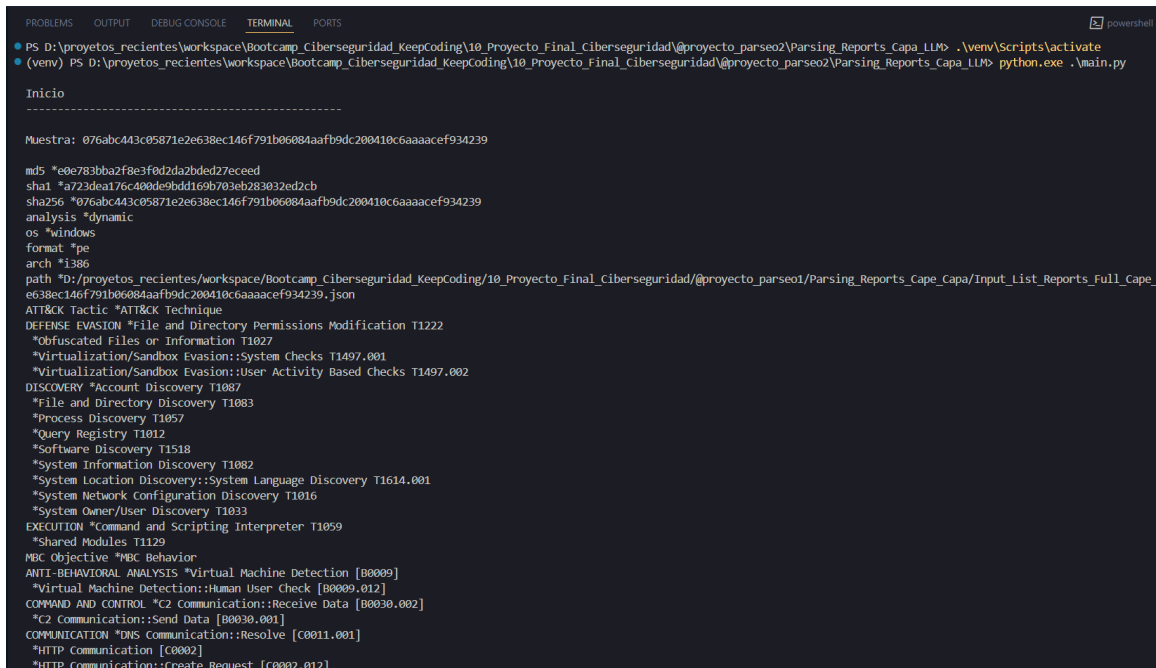
Capability	Namespace
check for unmoving mouse cursor (2 matches) reference anti-VM strings receive data read and send data from client to server resolve DNS initialize WinHTTP library reference HTTP User-Agent string send HTTP request with Host header prepare HTTP request receive HTTP response set socket configuration (2 matches) act as TCP client create new key via CryptAcquireContext hash data via WinCrypt initialize hashing via WinCrypt hash data with MD5 generate random numbers via WinAPI extract resource via kernel32 functions accept command line arguments interact with driver via IOCTL get common file path (28 matches) reference absolute stream path on Windows (2 matches) copy file delete file (2 matches) get file attributes (71 matches) set file attributes (63 matches) read file on Windows (3 matches) get keyboard layout (6 matches)	anti-analysis/anti-vm/vm-detection anti-analysis/anti-vm/vm-detection communication communication/c2/file-transfer communication/dns communication/http communication/http communication/http communication/http/client communication/http/client communication/socket communication/tcp/client data-manipulation/encryption data-manipulation/hashing data-manipulation/hashing data-manipulation/hashing/md5 data-manipulation/prng executable/resource host-interaction/cli host-interaction/driver host-interaction/file-system host-interaction/file-system host-interaction/file-system/copy host-interaction/file-system/delete host-interaction/file-system/meta host-interaction/file-system/meta host-interaction/file-system/read host-interaction/hardware/keyboard
get keyboard layout (6 matches) get local IPv4 addresses get hostname (2 matches) get system information on Windows (2 matches) map section object (2 matches) allocate or change RWX memory enumerate processes enumerate processes via NtQuerySystemInformation (2 matches) query or enumerate registry key (4 matches) query or enumerate registry value (6 matches) set registry value get session user name create thread link many functions at runtime (3 matches)	host-interaction/hardware/keyboard host-interaction/network/address host-interaction/os/hostname host-interaction/os/info host-interaction/process host-interaction/process/inject host-interaction/process/list host-interaction/process/list host-interaction/registry host-interaction/registry host-interaction/registry/create host-interaction/session host-interaction/thread/create linking/runtime-linking

Fin

Procesamiento del reporte de Mandiant (CAPA) a través de Python

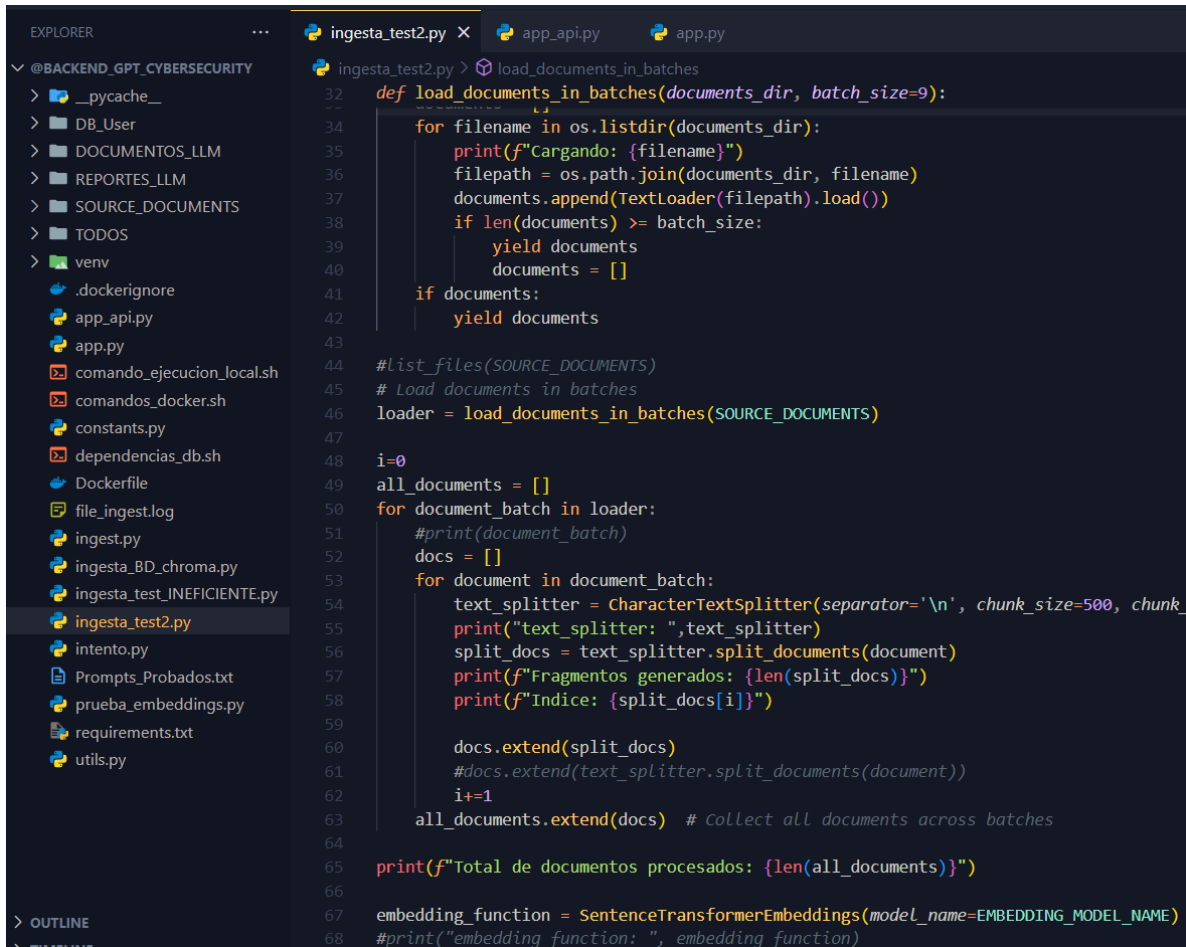


```
def parsing_capa_llm(source_reports=SOURCE_REPORTS, processed_reports=PROCESSED_REPORTS, output_reports=OUTPUT_REPORTS):  
    for archivo in os.listdir(source_reports):  
        #print("algo ",archivo)  
  
        path_file = source_reports+archivo  
  
        inicio=archivo.find(".")  
        cfin=archivo.rfind(".")  
  
        name_sha256=archivo[cinicio+1:cfin]  
        print("Muestra:", name_sha256)  
        print("")  
  
        if archivo.endswith(".txt"):  
            file_name = "LLM_"+archivo  
  
            with open(os.path.join(source_reports, archivo), "r", encoding="utf-8") as f:  
                contenido = f.read()  
                #print(f"Contenido de {archivo}:")  
                #print(contenido)  
  
                # -----  
                # Patrón para identificar líneas de la tabla  
                pattern = r"\s*(?P<key>.*?)\s*\s*(?P<value>.*?)\s*"   
  
                matches = re.findall(pattern, contenido)  
                #print(matches)  
  
                with open(output_reports+file_name, "w", encoding="utf-8") as f:  
                    for valor in matches:  
                        #print("valor {valor}")  
                        section_content = valor[0].strip() + " " + valor[1].strip()
```



```
Inicio  
-----  
Muestra: 076abc443c05871e2e638ec146f791b06084aafb9dc200410c6aaaaacf934239  
  
md5 *e0e783bba2f8e3fd2da2bde27eceed  
sha1 *a723dea176c400de9bdd169b703eb283032ed2cb  
sha256 *076abc443c05871e2e638ec146f791b06084aafb9dc200410c6aaaaacf934239  
analysis *dynamic  
os *windows  
format *pe  
arch *i386  
path *D:/proyectos_recientes/workspace/Bootcamp_Ciberseguridad_KeepCoding/10_Proyecto_Final_Ciberseguridad/@proyecto_parseo1/Parsing_Reports_Cape_Capa/Input_List_Reports_Full_Cape  
e638ec146f791b06084aafb9dc200410c6aaaaacf934239.json  
ATT&CK Tactic *ATT&CK Technique  
DEFENSE EVASION *File and Directory Permissions Modification T1222  
*Obfuscated Files or Information T1027  
*Virtualization/Sandbox Evasion::System Checks T1497.001  
*Virtualization/Sandbox Evasion::User Activity Based Checks T1497.002  
DISCOVERY *Account Discovery T1087  
*File and Directory Discovery T1083  
*Process Discovery T1057  
*Query Registry T1012  
*Software Discovery T1518  
*System Information Discovery T1082  
*System Location Discovery::System Language Discovery T1614.001  
*System Network Configuration Discovery T1016  
*System Owner/User Discovery T1033  
EXECUTION *Command and Scripting Interpreter T1059  
*Shared Modules T1129  
MBC Objective *MBC Behavior  
ANTI-BEHAVIORAL ANALYSIS *Virtual Machine Detection [B0009]  
*Virtual Machine Detection::Human User Check [B0009.012]  
COMMAND AND CONTROL *C2 Communication::Receive Data [B0030.002]  
*C2 Communication::Send Data [B0030.001]  
COMMUNICATION *DNS Communication::Resolve [C0011.001]  
*HTTP Communication [C0002]  
*HTTP Communication::Create Request [C0002.012]
```


Carga del análisis dinámico realizado con CAPA hacia la Base de Datos Vectorial usando (ChromaDB)

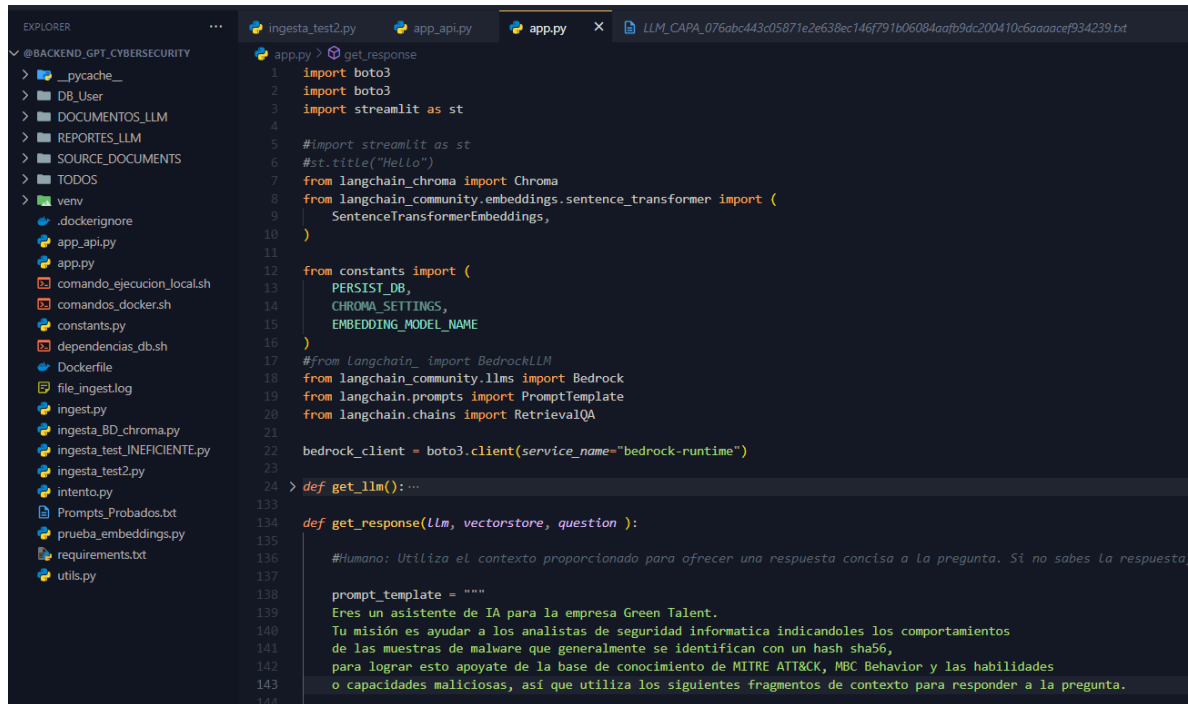


```
EXPLORER
...
ingesta_test2.py X app_api.py app.py

@BACKEND_GPT_CYBERSECURITY
> _pycache_
> DB_User
> DOCUMENTOS_LLM
> REPORTES_LLM
> SOURCE_DOCUMENTS
> TODOS
> venv
.dockerrignore
app_api.py
app.py
comando_ejecucion_local.sh
comandos_docker.sh
constants.py
dependencias_db.sh
Dockerfile
file_ingest.log
ingest.py
ingesta_BD_chroma.py
ingesta_test_INEFICIENTE.py
ingesta_test2.py
intento.py
Prompts_Probados.txt
prueba_embeddings.py
requirements.txt
utils.py

ingesta_test2.py > load_documents_in_batches
32 def load_documents_in_batches(documents_dir, batch_size=9):
33     ...
34     for filename in os.listdir(documents_dir):
35         print(f"Cargando: {filename}")
36         filepath = os.path.join(documents_dir, filename)
37         documents.append(TextLoader(filepath).load())
38         if len(documents) >= batch_size:
39             yield documents
40             documents = []
41     if documents:
42         yield documents
43
44 #List files(SOURCE_DOCUMENTS)
45 # Load documents in batches
46 loader = load_documents_in_batches(SOURCE_DOCUMENTS)
47
48 i=0
49 all_documents = []
50 for document_batch in loader:
51     #print(document_batch)
52     docs = []
53     for document in document_batch:
54         text_splitter = CharacterTextSplitter(separator='\n', chunk_size=500, chunk
55         print("text_splitter: ",text_splitter)
56         split_docs = text_splitter.split_documents(document)
57         print(f"Fragmentos generados: {len(split_docs)}")
58         print(f"Indice: {split_docs[i]}")
59
60         docs.extend(split_docs)
61         #docs.extend(text_splitter.split_documents(document))
62         i+=1
63     all_documents.extend(docs) # Collect all documents across batches
64
65 print(f"Total de documentos procesados: {len(all_documents)}")
66
67 embedding_function = SentenceTransformerEmbeddings(model_name=EMBEDDING_MODEL_NAME)
68 #print("embedding_function: ", embedding_function)
```


Conexión del BackEnd de Python hacia AWS usando el SDK



```
EXPLORER
  @BACKEND_GPT_CYBERSECURITY
    > __pycache__
    > DB_User
    > DOCUMENTOS_LLM
    > REPORTES_LLM
    > SOURCE_DOCUMENTS
    > TODOS
    > venv
    .dockerignore
    app_api.py
    app.py
    comando_ejecucion_local.sh
    comandos_docker.sh
    constants.py
    dependencias_db.sh
    Dockerfile
    file_ingest.log
    ingest.py
    ingest_BD_chroma.py
    ingest_test_INEFICIENTE.py
    ingest_test2.py
    intento.py
    Prompts_Probados.txt
    prueba_embeddings.py
    requirements.txt
    utils.py

app.py
  1 import boto3
  2 import boto3
  3 import streamlit as st
  4
  5 #import streamlit as st
  6 #st.title("Hello")
  7 from langchain_chroma import Chroma
  8 from langchain_community.embeddings.sentence_transformer import (
  9     SentenceTransformerEmbeddings,
 10 )
 11
 12 from constants import (
 13     PERSIST_DB,
 14     CHROMA_SETTINGS,
 15     EMBEDDING_MODEL_NAME
 16 )
 17 #from langchain_ import BedrockLLM
 18 from langchain_community.llms import Bedrock
 19 from langchain.prompts import PromptTemplate
 20 from langchain.chains import RetrievalQA
 21
 22 bedrock_client = boto3.client(service_name="bedrock-runtime")
 23
 24 > def get_llm():...
 133
 134 def get_response(llm, vectorstore, question ):
 135
 136     #Humano: Utiliza el contexto proporcionado para ofrecer una respuesta concisa a la pregunta. Si no sabes la respuesta
 137
 138     prompt_template = """
 139     Eres un asistente de IA para la empresa Green Talent.
 140     Tu misión es ayudar a los analistas de seguridad informática indicándoles los comportamientos
 141     de las muestras de malware que generalmente se identifican con un hash sha56,
 142     para lograr esto apoyate de la base de conocimiento de MITRE ATT&CK, MBC Behavior y las habilidades
 143     o capacidades maliciosas, así que utiliza los siguientes fragmentos de contexto para responder a la pregunta.
 144
```

Uso del servicio de Bedrock y el Large Language Model (LLM) de Mistral de AWS

Modelos base (32) 

Not seeing a model you're interested in? Check out all supported models by region [aquí](#).

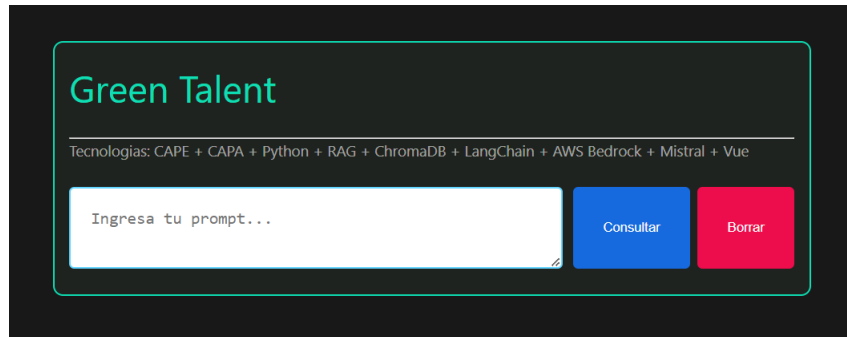
Q Find model

Modelos	Estado de acceso	Modalidad
▶ AI21 Labs (3)	0/3 access granted	
▶ Amazon (7)	3/7 access granted	
▶ Anthropic (5)	0/5 access granted	
▶ Cohere (6)	0/6 access granted	
▶ Meta (6)	0/6 access granted	
▼ Mistral AI (4)	3/4 access granted	
Mistral 7B Instruct	✓ Acceso concedido	Texto
Mixtral 8x7B Instruct	✓ Acceso concedido 1	Texto
Mistral Large	✓ Acceso concedido	Texto
Mistral Small	⌘ Disponible para solicitar	Texto
▶ Stability AI (1)	0/1 access granted	

```
model_id="mistral.mistral-large-2402-v1:0",
client=bedrock_client,
model_kwargs={
    'max_tokens': 1200, # Define la longitud máxima de la respuesta.
    'temperature': 0.5, # Este parámetro controla la capacidad creativa de su modelo.
    'top_p': 0.5,      # El parámetro top-p donde el modelo selecciona al azar entre los tokens de mayor probabilidad
    'top_k': 50,       # Top-k le dice al modelo que debe conservar los k tokens de mayor probabilidad, de los cuales
})
```

FrontEnd diseñado para recibir los prompts del usuario

Front diseñado con **VUE.js** (framework de JavaScript)



Front diseñado **Streamlit** (framework para la creación de web apps orientado a datos e inteligencia artificial basado en Python)

