

Projeto Final

Sistema Automatizado para Análise de E-mails

Francisco Rafael Carocinho Ribeiro – Nº 2024123

Tiago Filipe Sousa Carvalho – Nº 2024180

Vanina Kollen – Nº 2024056

2º Ano de CTeSP de Cibersegurança

Sistemas de Controlo

André Ribeiro

Porto, 2026

Índice

Projeto Final.....	1
Índice.....	1
1. Introdução.....	1
1.1. Enquadramento.....	1
1.2. Objetivos.....	1
1.3. Estrutura do Relatório.....	1
2. Desenvolvimento.....	2
2.1. Ferramentas.....	2
2.2. Desenvolvimento do Projeto.....	2
2.3. Funcionamento.....	7
2.4. Diagramas.....	13
2.5. Resultados.....	14
3. Conclusão.....	15
3.1. Limitações do Projeto.....	15

1. Introdução

Este projeto consiste no desenvolvimento de um sistema de scripts que automatiza o processo de análise de emails através de uma sandbox, na qual os ficheiros anexos são analisados e classificados como infetados ou não infetados, ou seja, conforme contenham conteúdo malicioso ou potencialmente danoso.

1.1. Enquadramento

Este trabalho encontra-se inscrito dentro do contexto do 2º ano do curso de CTeSP de Cibersegurança, dentro da disciplina de Sistemas de Controlo. Tendo isto em conta, a realização deste projeto serve como exercício prático para elaborar um sistema de segurança para a análise de e-mails, resultando fundamental para a nossa área.

1.2. Objetivos

O objetivo deste trabalho envolve levar a cabo um exercício para a elaboração de scripts para melhor perceber o seu funcionamento, além de focar-se na área de cibersegurança. Este estudo serve para adquirir maiores conhecimentos acerca da automatização de processos e sistemas de segurança.

1.3. Estrutura do Relatório

Neste relatório será apresentada uma introdução (já abordada), na qual se explica em que consiste o projeto e quais os seus objetivos; a descrição do desenvolvimento, incluindo as ferramentas utilizadas, o processo, bem como o código e os diagramas que explicam o funcionamento e o fluxo do sistema; e, por fim, a conclusão, onde são apresentados os resultados obtidos.

2. Desenvolvimento

Neste segmento abordaremos como foi desenvolvido este sistema, a sua estrutura, como funciona, o que realiza, para que serve, as ferramentas utilizadas, além de perceber a lógica que segue o sistema.

2.1. Ferramentas

Neste projeto, foram utilizadas diversas ferramentas e recursos para o desenvolvimento do sistema, tais como a CAPE Sandbox, o comando *cron job* para automatizar a execução dos scripts, bem como ficheiros infetados e uma caixa de e-mail para a realização dos testes.

A execução do projeto foi realizada num ambiente isolado, nomeadamente numa máquina virtual Ubuntu e uma máquina Windows. Foi desenvolvido um script responsável pela comunicação entre a caixa de e-mails e a CAPE Sandbox. Esta sandbox permite a análise dos e-mails e dos possíveis malwares neles contidos. A CAPE Sandbox é uma versão melhorada da Cuckoo Sandbox, sendo de código aberto e baseada em Python.

O *cron job*, conforme mencionado anteriormente, é utilizado para configurar a periodicidade de execução do script de organização das análises, facilitando a automatização do processo e permitindo ajustar a frequência com que a organização é executada. A caixa de e-mail funciona como o ambiente onde são recebidos os ficheiros infetados para posterior análise.

2.2. Desenvolvimento do Projeto

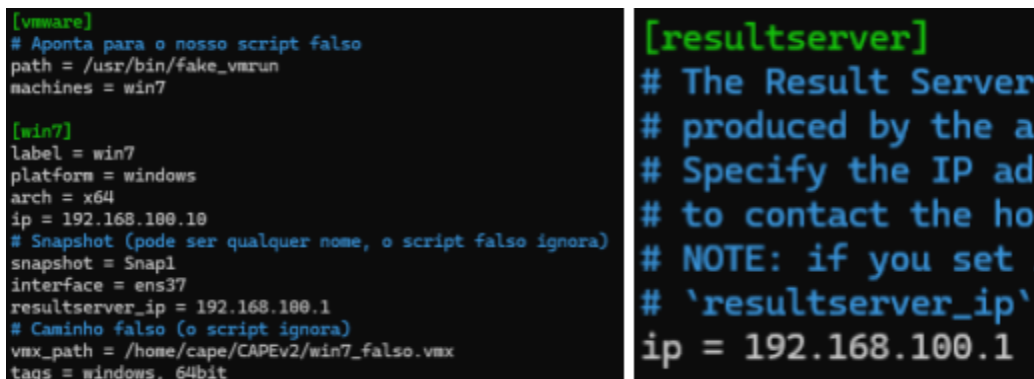
Para a criação do sistema, foi necessário realizar uma série de configurações. Primeiramente, foi estabelecida a infraestrutura de virtualização, consistindo num servidor Ubuntu (CAPE Host) com um ambiente virtual Python (*venv*) para o isolamento de dependências. Posteriormente, configurou-se o VMware como hypervisor para gerir a máquina vítima. Para garantir que o malware não tivesse acesso à internet real, foi criada uma rede isolada (*Host-Only*), permitindo apenas uma comunicação controlada entre o Ubuntu (Controlador) e o Windows (Vítima).

De seguida, procedeu-se à instalação do CAPE v2 e à configuração de um utilizador dedicado (*cape*), bem como à instalação de bibliotecas de sistema críticas, como *tcpdump* e *libcap2-bin*, e

à gestão de pacotes Python através do *pip* e do *poetry*. Posteriormente, configuraram-se as bases de dados PostgreSQL, utilizada para o armazenamento de metadados das tarefas, e MongoDB, responsável pelo armazenamento dos relatórios em formato JSON e HTML.

No que diz respeito à configuração da máquina vítima (*Guest VM*), foi instalado o Windows 7 Ultimate x64, escolhido pela facilidade de desativar mecanismos de proteção modernos, como o Windows Defender e o *Tamper Protection*. Foi igualmente instalado o Python 3.8 (x86) para permitir a execução do agente do CAPE. Ao nível da rede, definiu-se um endereço IP estático (192.168.100.10), garantindo uma comunicação estável com o CAPE Controller. Foram ainda desativados a Firewall do Windows, o Windows Update, o UAC (*User Account Control*) e o Windows Defender.

Após estas configurações, relativamente aos scripts, para o seu correto funcionamento, foi preciso realizar as necessárias alterações no ficheiro de configuração do VMware, além de modificar o IP dentro do ficheiro de configuração do Cuckoo, estas mudanças são mostradas nas seguintes capturas:



```
[vmware]
# Aponta para o nosso script falso
path = /usr/bin/fake_vmrun
machines = win7

[win7]
label = win7
platform = windows
arch = x64
ip = 192.168.100.10
# Snapshot (pode ser qualquer nome, o script falso ignora)
snapshot = Snap1
interface = ens37
resultserver_ip = 192.168.100.1
# Caminho falso (o script ignora)
vmx_path = /home/cape/CAPEv2/win7_falso.vmx
tags = windows, 64bit

[resultserver]
# The Result Server
# produced by the an
# Specify the IP add
# to contact the hos
# NOTE: if you set
# 'resultserver_ip'
ip = 192.168.100.1
```

Para os scripts, criou-se um responsável de enviar o conteúdo (ficheiros anexos) para a sandbox CAPE e de processar os e-mails não lidos e que possua conteúdo em formato .zip, precisando configurar o IMAP server, o email usado, a App password que gerada, o intervalo para verificar emails novos (20 segundos), pasta que guarda os e-mails coletados e a possível password do ZIP (caso o anexo esteja comprimido); um segundo script que dá aviso à CAPE sandbox das snapshots que deve restaurar, caso deseje verificar se a máquina virtual encontra-se ativa, é preciso notificar sempre com 0 para que tente iniciá-la, no entanto, será

necessário ligar a máquina virtual manualmente enquanto o script atrasa o cape e desativa por 45s, uma vez que o tempo acaba, a sandbox deteta que a máquina encontra-se ligada e inicia a análise; e um script final, o qual está encarregado de classificar os ficheiros. Posteriormente, executa-se o comando crontab para que esse script seja executado a cada minuto, além de gerar um ficheiro .log para confirmar o seu correto funcionamento. O código destes scripts encontra-se a continuação:

- Script 1

```
# --- CONFIGURAÇÃO ---
IMAP_SERVER = 'imap.gmail.com'
EMAIL_USER = 'tiago.fsc.00@gmail.com' # o email
EMAIL_PASS = 'bbrv bstq tfjj dgelw' # A App Password
CHECK_INTERVAL = 20
TEMP_DIR = '/tmp/cape_emails'
ZIP_PASSWORD = b'infected' # Password padrão para malware
# -----

def connect_imap():
    try:
        mail = imaplib.IMAP4_SSL(IMAP_SERVER)
        mail.login(EMAIL_USER, EMAIL_PASS)
        return mail
    except Exception as e:
        print(f'[-] Erro crítico ao ligar ao email: {e}')
        return None

def clean_filename(filename):
    return ''.join([c for c in filename if c.isalpha() or c.isdigit() or c in '._-']).rstrip()

def submit_to_cape(filepath):
    print(f'[+] A preparar envio de: {filepath}')

    # Garantir caminhos absolutos
    abs_filepath = os.path.abspath(filepath)
    cwd = os.getcwd() # Pasta onde o script está a correr (deve ser ~/CAPEv2)
    submit_script = os.path.join(cwd, "utils", "submit.py")

    print(f'  -> Executando: {sys.executable} {submit_script} {abs_filepath}')

    try:
        # Executa e captura o que o comando responde
        result = subprocess.run(
            [sys.executable, submit_script, abs_filepath],
            capture_output=True,
            text=True,
            cwd=cwd
        )

        if result.returncode == 0:
            print(f'[+] SUCESSO CAPE diz: {result.stdout}')
            # Tenta encontrar o ID na resposta
            if "ID" in result.stdout or "ids" in result.stdout:
                return True
            else:
                print(f'[-] AVISO: O comando correu mas não retornou ID. Estranho.')
        else:
            print(f'[-] AVISO: O comando correu mas não retornou ID. Estranho.')

def process_emails():
    if not os.path.exists(TEMP_DIR):
        os.makedirs(TEMP_DIR)

    mail = connect_imap()
    if not mail: return

    try:
        mail.select("inbox")
        status, messages = mail.search(None, 'UNSEEN')
        email_ids = messages[0].split()

        if email_ids:
            print(f'[+] Encountered {len(email_ids)} unseen emails:')
            for e_id in email_ids:
                msg_data = mail.fetch(e_id, '(RFC822)')
                for response_part in msg_data:
                    if isinstance(response_part, tuple):
                        msg = email.message_from_bytes(response_part[1])
                        subject = decode_header(msg['subject'])[0][0]
                        if isinstance(subject, bytes): subject = subject.decode()
                        print(f'  -> Email: {subject}')

                        for part in msg.walk():
                            if part.get_content_mimetype() == 'multipart' or part.get('Content-Disposition') is None:
                                continue

                            filename = part.get_filename()
                            if filename:
                                filename = clean_filename(filename)
                                content = part.get_payload(decode=True)
                                handle_attachment(filename, content)
```

```
# Se for ZIP, tentar extrair
if filename.lower().endswith('.zip'):
    print(f" -> Detetado ficheiro ZIP: {filename}. A tentar extrair com password...")
    try:
        with zipfile.ZipFile(io.BytesIO(content)) as z:
            # Tentar com password 'infected'
            try:
                z.setpassword(ZIP_PASSWORD)
                # Extrair o primeiro ficheiro do zip
                for file_in_zip in z.namelist():
                    print(f" -> Extraindo do ZIP: {file_in_zip}")
                    z.extract(file_in_zip, TEMP_DIR)
                    extracted_path = os.path.join(TEMP_DIR, file_in_zip)
                    submit_to_cape(extracted_path)
                    try:
                        os.remove(extracted_path)
                    except:
                        pass
                return # Processa apenas o primeiro e sai
            except RuntimeError:
                print(f" [-] Password incorreta ou ZIP sem password. A tentar sem password...")
                z.extractall(TEMP_DIR)
                # (Lógica simplificada para ZIPs sem pass)
            except Exception as e:
                print(f" [-] Falha ao abrir ZIP: {e}. A enviar original.")
                # Se falhar a extração, envia o zip normal
                with open(filepath, 'wb') as f:
                    f.write(content)
                submit_to_cape(filepath)
    else:
        # Ficheiro normal (não ZIP)
        with open(filepath, 'wb') as f:
            f.write(content)
        submit_to_cape(filepath)

# Limpeza final
if os.path.exists(filepath):
    try:
        os.remove(filepath)
    except:
        pass
```

- Script 2

```
# --- Script Assistente CAPE ---

# 1. Se o CAPE perguntar snapshots
if [[ "$*" == *"listSnapshots"* ]]; then
    echo "Total snapshots: 1"
    echo "Snap1"
    exit 0
fi

# 2. Se perguntar o que esta a correr (Dizemos sempre 0 para ele tentar iniciar)
if [[ "$*" == *"list"* ]]; then
    echo "Total running VMs: 0"
    exit 0
fi

# 3. START ou REVERT
# Aqui desviamos o texto para o ecrã (/dev/tty) para o CAPE não ler
if [[ "$*" == *"start"* || "$*" == *"revertToSnapshot"* ]]; then
    echo "-----" > /dev/tty
    echo "▲ AÇÃO MANUAL NECESSÁRIA ▲" > /dev/tty
    echo "O CAPE quer analisar uma amostra." > /dev/tty
    echo "Por favor, vai ao VMware no Host:" > /dev/tty
    echo "1. Reverte o Windows 7 para o snapshot 'Snap1'." > /dev/tty
    echo "2. Inicia a VM." > /dev/tty
    echo "3. Certifica-te que o agent.py está a correr." > /dev/tty
    echo "4. O IP tem de ser 192.168.100.10." > /dev/tty
    echo "-----" > /dev/tty

    echo "A contar 45 segundos..." > /dev/tty
    sleep 45

    echo "Tempo esgotado. A dizer ao CAPE que foi um sucesso." > /dev/tty

    # Sair em silêncio absoluto para o CAPE ficar feliz
    exit 0
fi

# 4. Stop
if [[ "$*" == *"stop"* ]]; then
    # Opcional: Avisar-te que acabou
    echo "O CAPE pediu para parar a VM. Podes desligar." > /dev/tty
    exit 0
fi
```

- Script 3

```
# --- CONFIGURAÇÃO ---
# O expanduser(~) vai buscar a pasta /home/cape automaticamente
CAPE_STORAGE = os.path.expanduser("~/CAPEv2/storage/analyses")
DEST_MALWARE = os.path.expanduser("~/Relatorios_Finais/Inseguro")
DEST_SEGURO = os.path.expanduser("~/Relatorios_Finais/Seguro")
SCORE_LIMIT = 5.0

def organizar():
    if not os.path.exists(CAPE_STORAGE):
        print(f"Erro: Pasta não encontrada em {CAPE_STORAGE}")
        return

    for task_id in os.listdir(CAPE_STORAGE):
        task_path = os.path.join(CAPE_STORAGE, task_id)
        report_path = os.path.join(task_path, "reports", "report.json")
        binary_path = os.path.join(task_path, "binary")

        if os.path.exists(report_path) and os.path.exists(binary_path):
            try:
                with open(report_path, "r") as f:
                    data = json.load(f)

                score = data.get("malware", 0)
                # Tenta obter o nome, senão inventa um
                try:
                    filename = data.get("target", {}).get("file", {}).get("name", f"amostra_{task_id}.bin")
                except:
                    filename = f"amostra_{task_id}.bin"

                if score >= SCORE_LIMIT:
                    destino_dir = DEST_MALWARE
                    tipo = "INSEGURO"
                else:
                    destino_dir = DEST_SEGURO
                    tipo = "SEGURO"

                destino_final = os.path.join(destino_dir, filename)

                # Copia apenas se ainda não existir no destino
                if not os.path.exists(destino_final):
                    shutil.copy(binary_path, destino_final)
                    print(f"[NOVO] ID {task_id}: {filename} -> {tipo} (Score: {score})")

            except Exception as e:
                print(f"Erro ID {task_id}: {e}")
```

- Comando crontab

```
* * * * * /usr/bin/python3 /home/cape/organizador.py >> /home/cape/organizador.log 2>&1
```

A seguir, instalou-se e executou-se o script *agent.py*, com privilégios de administrador, no sistema Windows. Este script permanece à escuta na porta 8000. Foi então criado um ponto de restauro (*snapshot*), denominado “Snap1”, com o *agent.py* em execução. O CAPE utiliza este *snapshot* para reverter a máquina ao seu estado limpo após cada análise.

Para a automação da submissão de e-mails, foi desenvolvido o script *utils/email_to_cape.py*, recorrendo à biblioteca *imaplib*. Este script monitoriza uma caixa de correio Gmail dedicada, filtra e-mails não lidos, extrai automaticamente os anexos e submete-os à API do CAPE através

do *submit.py*. Em termos de segurança, foi utilizada uma *App Password* do Google para autenticação, evitando a exposição da palavra-passe principal da conta.

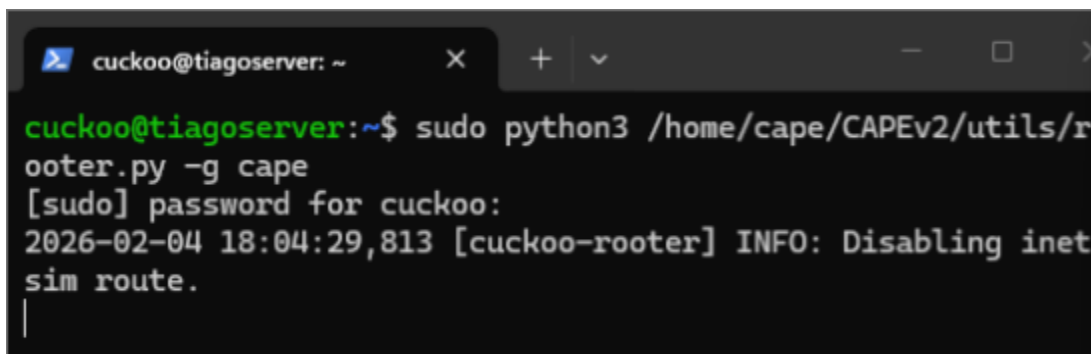
Por fim, relativamente ao sistema de relatórios (*Reporting*), procedeu-se à correção de erros, ao ajuste da versão da biblioteca *pymongo* (atualizada para a versão 4) para garantir compatibilidade com versões modernas do CAPE, e à correção do ficheiro *conf/reporting.conf*, de modo a apontar para a base de dados correta (*db = cape*) e ativar a geração de relatórios em HTML e JSON. Adicionalmente, foi configurado um servidor Django para disponibilizar uma interface web que permite a visualização gráfica dos relatórios através do navegador.

2.3. Funcionamento

Nesta secção será explicada a execução do sistema passo a passo, com o acompanhamento de capturas, oferecendo assim, um apoio visual para perceber e demonstrar o funcionamento do programa desenvolvido.

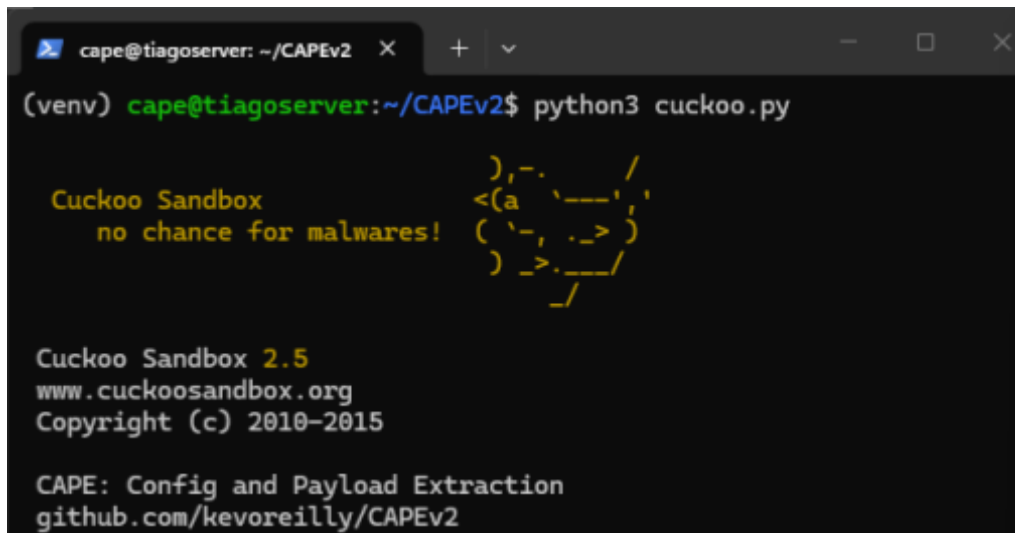
1.- Primeiramente, é preciso abrir 6 terminais dentro da máquina Ubuntu. A seguir, em todos os terminais tirando o primeiro e último, mediante o utilizador “cape” usando o comando “sudo su - cape”, ir à pasta do CAPE v2 através do comando “cd CAPEv2” e executar o comando “source venv/bin/activate”.

2.- No terminal 0, o qual é responsável por permissões de rede e captura de pacotes, executar “sudo python3 /home/cape/CAPEv2/utils/rooter.py -g cape”.



```
cuckoo@tiagoserver: ~  
cuckoo@tiagoserver:~$ sudo python3 /home/cape/CAPEv2/utils/rooter.py -g cape  
[sudo] password for cuckoo:  
2026-02-04 18:04:29,813 [cuckoo-rooter] INFO: Disabling inet  
sim route.  
|
```

3.- No terminal 1, o qual Inicia o motor principal de análise, executar “python3 cuckoo.py”.



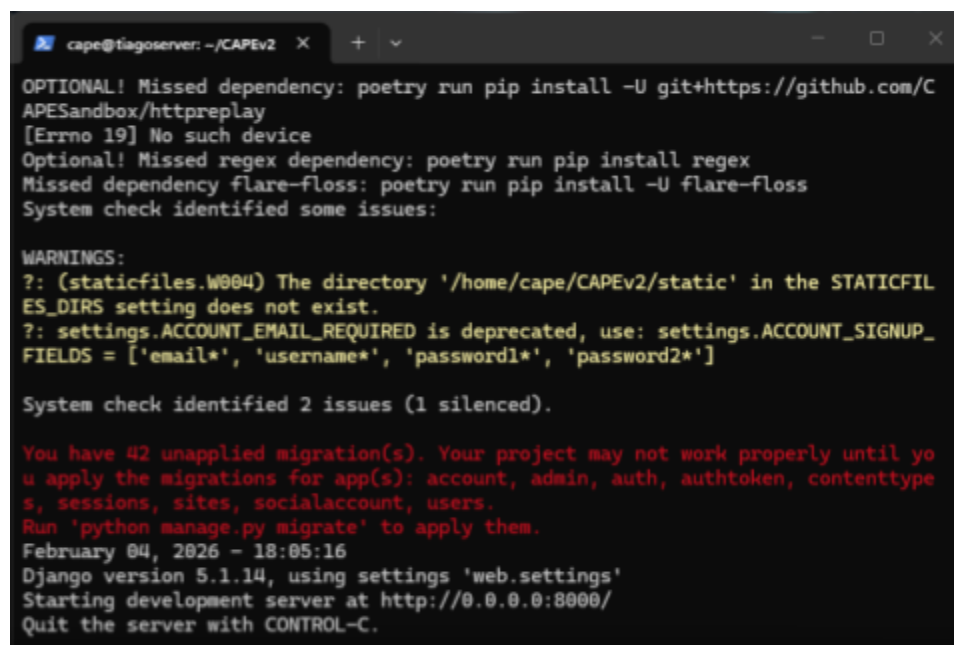
```
cape@tiagoserver: ~/CAPEv2
(venv) cape@tiagoserver:~/CAPEv2$ python3 cuckoo.py

Cuckoo Sandbox
no chance for malwares!

Cuckoo Sandbox 2.5
www.cuckoosandbox.org
Copyright (c) 2010-2015

CAPE: Config and Payload Extraction
github.com/kevoreilly/CAPEv2
```

4.- No terminal 2, o qual arranca o site do CAPEv2, executar “python3 web/manage.py runserver 0.0.0.0:8000”.



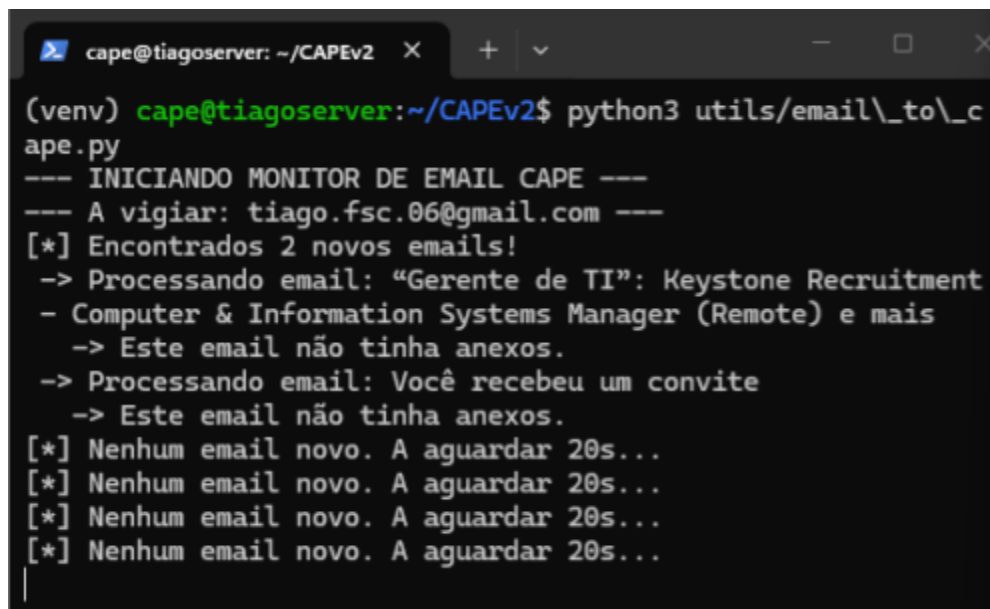
```
cape@tiagoserver: ~/CAPEv2
OPTIONAL! Missed dependency: poetry run pip install -U git+https://github.com/C
APESandbox/httpreplay
[Errno 19] No such device
Optional! Missed regex dependency: poetry run pip install regex
Missed dependency flare-floss: poetry run pip install -U flare-floss
System check identified some issues:

WARNINGS:
?: (staticfiles.W004) The directory '/home/cape/CAPEv2/static' in the STATICFIL
ES_DIRS setting does not exist.
?: settings.ACCOUNT_EMAIL_REQUIRED is deprecated, use: settings.ACCOUNT_SIGNUP_
FIELDS = ['email*', 'username*', 'password1*', 'password2*']

System check identified 2 issues (1 silenced).

You have 42 unapplied migration(s). Your project may not work properly until yo
u apply the migrations for app(s): account, admin, auth, authtoken, contenttype
s, sessions, sites, socialaccount, users.
Run 'python manage.py migrate' to apply them.
February 04, 2026 - 18:05:16
Django version 5.1.14, using settings 'web.settings'
Starting development server at http://0.0.0.0:8000/
Quit the server with CONTROL-C.
```

5.- No terminal 3, responsável por monitorizar e recolher emails com anexos maliciosos, executar “python3 utils/email_to_cape.py”.



```
(venv) cape@tiagoserver: ~/CAPEv2$ python3 utils/email\_to\_cape.py
--- INICIANDO MONITOR DE EMAIL CAPE ---
--- A vigiar: tiago.fsc.06@gmail.com ---
[*] Encontrados 2 novos emails!
-> Processando email: “Gerente de TI”: Keystone Recruitment
- Computer & Information Systems Manager (Remote) e mais
-> Este email não tinha anexos.
-> Processando email: Você recebeu um convite
-> Este email não tinha anexos.
[*] Nenhum email novo. A aguardar 20s...
[*] Nenhum email novo. A aguardar 20s...
[*] Nenhum email novo. A aguardar 20s...
[*] Nenhum email novo. A aguardar 20s...
```

6.- No terminal 4, o qual gera os relatórios, executar “python3 utils/process.py <Nº da análise> --report” caso o cape não faça o relatório sozinho.

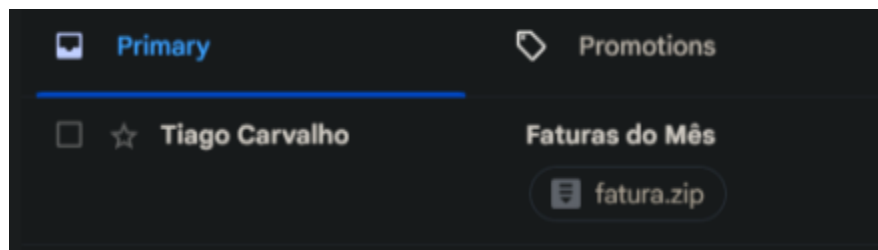
```
python3 utils/process.py 12 --report
```

7.- O terminal 5 permanecerá vazio, já que servirá para verificar logs, testar conectividade, corrigir erros manualmente, verificar IPs ou processos, etc.



```
cuckoo@tiagoserver: ~$
```

8.- Após, descarregar um *sample* (ex: MalwareBazaar), e mudar o formato para .txt, .png or .jpg, etc, para disfarçar. Posteriormente, enviar o e-mail para o endereço de teste.



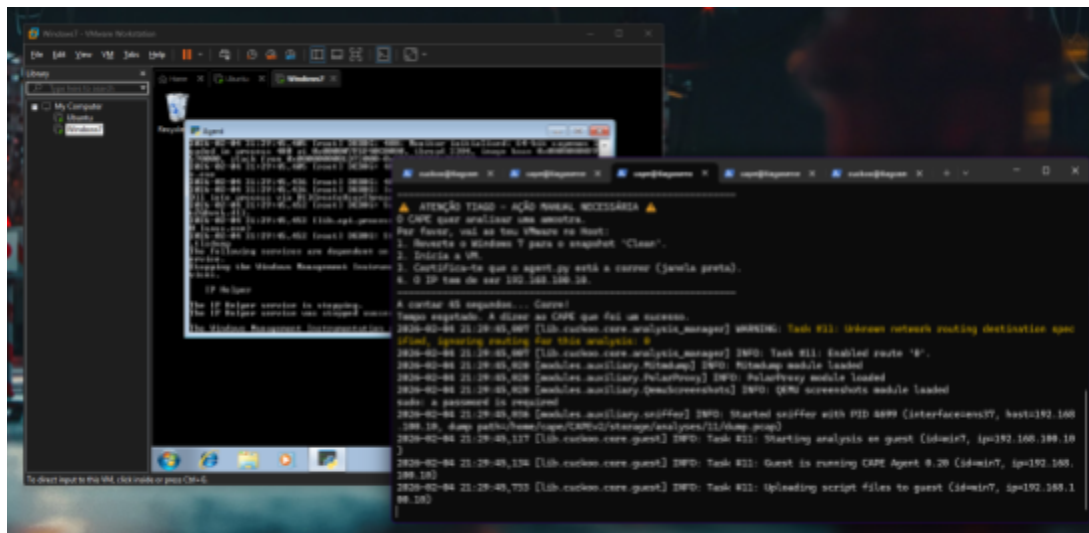
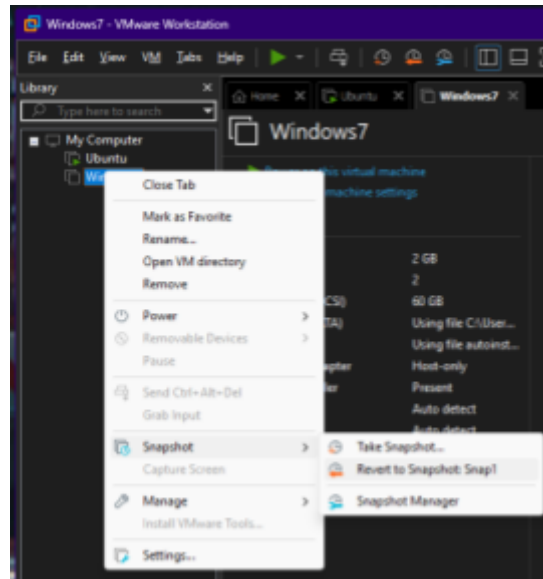
9.- A seguir, realizar o monitoramento: no terminal 1 mostrará a mensagem "Task #X: Starting analysis" e pedirá para reverter e ligar a máquina virtual e será preciso realizá-lo na VMware; no terminal 3 mostrará as mensagens "Anexo extraído" e "Submissão enviada" e no terminal 4, poderão ser gerados os relatórios.

```
cuckoo@tiagos: X cape@tiagoserver X cape@tiagoserver X cape@tiagoserver X cuckoo@tiagos X + v
OPTIONAL: Missed dependency: poetry run pip install -U git+https://github.com/Diasect/Malware/hatch_desofuscator
2026-02-04 21:26:50,220 [modules_processing.network] ERROR: OPTIONAL: Missed dependency: poetry run pip install -U git+https://github.com/CAPEsandbox/httpreplay
2026-02-04 21:26:50,496 [lib.cuckoo.core.machinery_manager] INFO: Using MachineryManager[vmware] with max_machines_count=10
2026-02-04 21:26:50,499 [lib.cuckoo.core.scheduler] INFO: Creating scheduler with max_analysis_count=unlimited
2026-02-04 21:26:50,533 [lib.cuckoo.core.machinery_manager] INFO: Loaded 1 machine
2026-02-04 21:26:50,633 [lib.cuckoo.core.machinery_manager] INFO: max_vmstartup_count for BoundedSemaphore = 5
2026-02-04 21:26:50,637 [lib.cuckoo.core.scheduler] INFO: Waiting for analysis tasks
2026-02-04 21:26:51,961 [lib.cuckoo.core.machinery_manager] INFO: Task #11: found usable machine win7 (arch=x86, platform=windows)
2026-02-04 21:26:51,961 [lib.cuckoo.core.scheduler] INFO: Task #11: Processing task
2026-02-04 21:26:51,982 [lib.cuckoo.core.analysis_manager] INFO: Task #11: File already exists at '/home/cape/CAPEv2/storage/binaries/1ca88c889640252a61840f48be052ba52d7fedaa67ed83e8f0fe3d4d0808cb3d4'
2026-02-04 21:26:51,982 [lib.cuckoo.core.analysis_manager] INFO: Task #11: Starting analysis of FILE '/tmp/cuckoo-sfloc/tmppyw05u_/fatura.pdf'

⚠️ ATENÇÃO TIAGO - AÇÃO MANUAL NECESSÁRIA ⚠️
O CAPE quer analisar uma amostra.
Por favor, vá ao teu iPhone no Host:
1. Reverte o Windows 7 para o snapshot 'Clean'.
2. Inicia a VM.
3. Certifica-te que o agent.py está a correr (janela preta).
4. O IP tem de ser 192.168.100.10.

A contar 45 segundos... Corre!
```

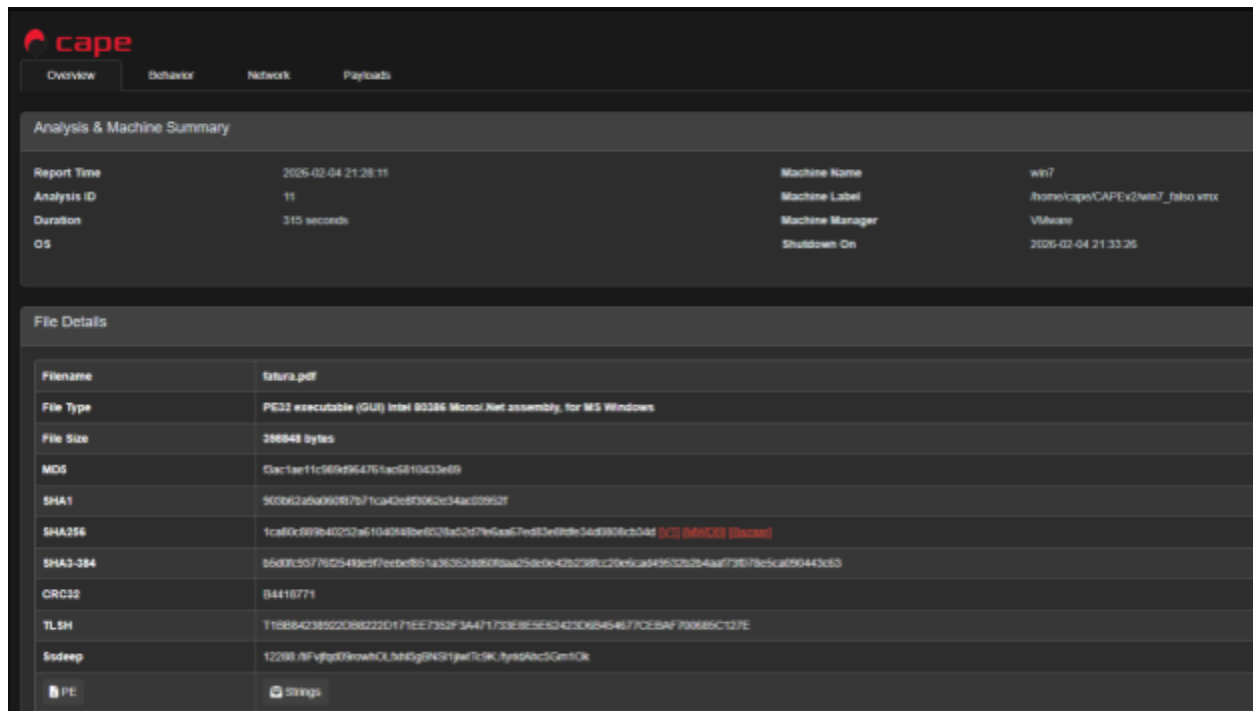
```
cuckoo@tiagos: X cape@tiagoserver X cape@tiagoserver X cape@tiagoserver X cuckoo@tiagos X + v
(venv) cape@tiagoserver:~/CAPEv2$ python3 utils/email_to_cape.py
--- MONITOR DE EMAIL 2.0 (Suporte a ZIP Encriptado) ---
[*] Encontrados 1 novos emails!
-> Email: Faturas do Mês
-> Detetado ficheiro ZIP: fatura.zip. A tentar extrair com password...
-> Extraíndo do ZIP: fatura.pdf
[-] Password incorreta ou ZIP sem password. A tentar sem password...
[-] Falha ao abrir ZIP: That compression method is not supported. A enviar original.
[*] A preparar envio de: /tmp/cape_emails/fatura.zip
-> Executando: /home/cape/CAPEv2/venv/bin/python3 /home/cape/CAPEv2/utils/submit.py /tmp/cape_emails/fatura.zip
[*] SUCESSO CAPE diz:
Success: File "/tmp/cape_emails/fatura.zip" added as task with ID 11
```



```

2026-02-04 21:33:26,747 [lib.cuckoo.core.guest] INFO: Task #11: Analysis completed successfully (id=win7, ip=192.168.1
00.10)
2026-02-04 21:33:26,778 [modules.machinery.vmware] WARNING: Trying to stop an already stopped machine: /home/cafe/CAPE
v2/win7_false.vmx
2026-02-04 21:33:26,788 [lib.cuckoo.core.analysis_manager] INFO: Task #11: Completed analysis successfully.
2026-02-04 21:33:26,794 [lib.cuckoo.core.analysis_manager] INFO: Task #11: analysis procedure completed
    
```

10.- Finalmente, ir ao site “http://127.0.0.1:8000/” clicar em “Recent Analyses” e aceder ao relatório.



The screenshot shows the CAPE web interface. The top navigation bar includes 'Overview', 'Behavior', 'Network', and 'Payloads'. The main content area is divided into two sections: 'Analysis & Machine Summary' and 'File Details'.

Analysis & Machine Summary

Report Time	2026-02-04 21:28:11	Machine Name	win7
Analysis ID	11	Machine Label	/home/cape/CAPE-x2/win7_falso_vmx
Duration	310 seconds	Machine Manager	VMware
OS		Shutdown On	2026-02-04 21:33:36

File Details

Filename	fatura.pdf
File Type	PE32 executable (GUI) Intel 80386 Mono/Net assembly, for MS Windows
File Size	38888 bytes
MD5	E2c1ae11c569d95475fac5810433e69
SHA1	90862ab06087b71ca40e8f36c2c34ac0952f
SHA256	1ca80c899e40252a6104048be6328e2d7b65a67e833e8de34d808e3d4d
SHA3-384	96d8f35f762549c9f7e6e851a36362386f8a25de4e42c238f29efca45632b24aaf79b78dca690443c63
CRC32	B4418771
TLSH	T19884238522080222D171EE7952F3A471733E9E5E32423068454677CEBAF700685C12FE
Sddeep	12268.7Fvfp9E9xowhOLfddGg9K9H1jwTc9K_AyrdAec3Gm10k
PE	Strings

Podemos verificar que a CAPE classificou as várias ações do malware de verde a vermelho sendo pouco a muito suspeito.



The screenshot shows a list of signatures detected by CAPE. Each signature is preceded by a colored bar indicating its risk level: green for 'low', yellow for 'medium', and red for 'high'.

Signature	Risk Level
Checks available memory	Low
Queries computer hostname	Low
Queries the keyboard layout	Low
Queries the computer locale (possible geolocation)	Low
Introduces/unintended scriptlets (possible anti-debug)	Low
Checks adapter addresses which can be used to detect virtual network interfaces	Low
Checks system language via registry key (possible geolocation)	Low
Overlaid pages size detected - possible anti-debugging	Medium
Dynamic (reported) function loading detected	Medium
Performs HTTP requests potentially not found in PCAP	Medium
Reads from the memory of another process	Medium
The binary body contains encrypted or compressed data	Medium
Looks up the external IP address	Medium
Creates RMM memory	Medium
Detects Sandboxes through the presence of a library	High
Attempted to write directly to a physical drive	High

2.4. Diagramas

Diagrama de Fluxograma

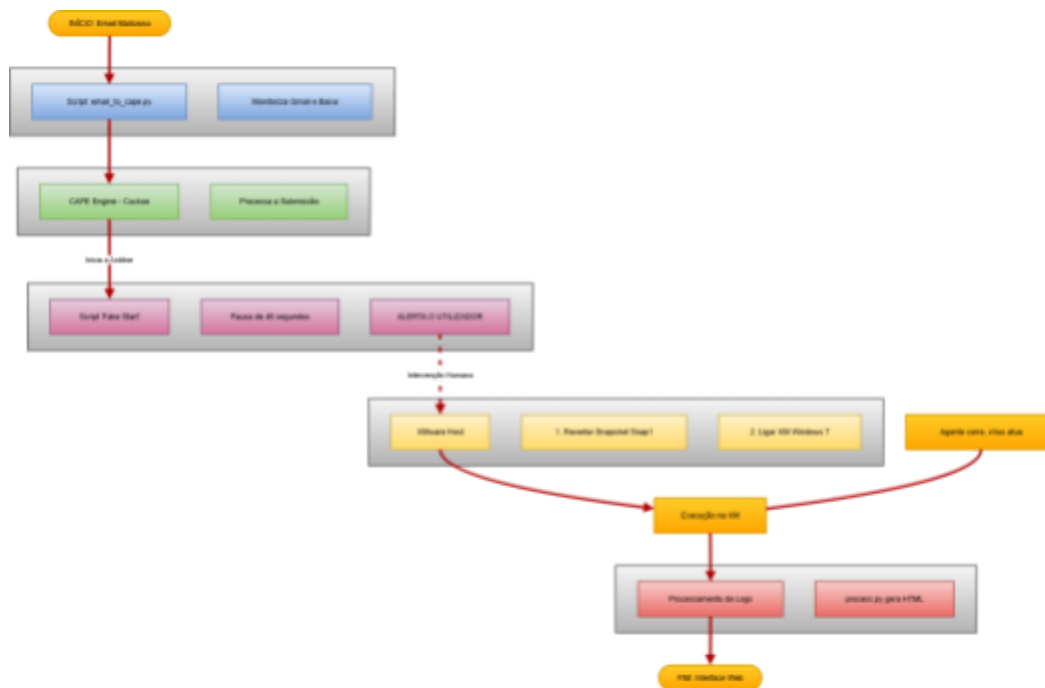
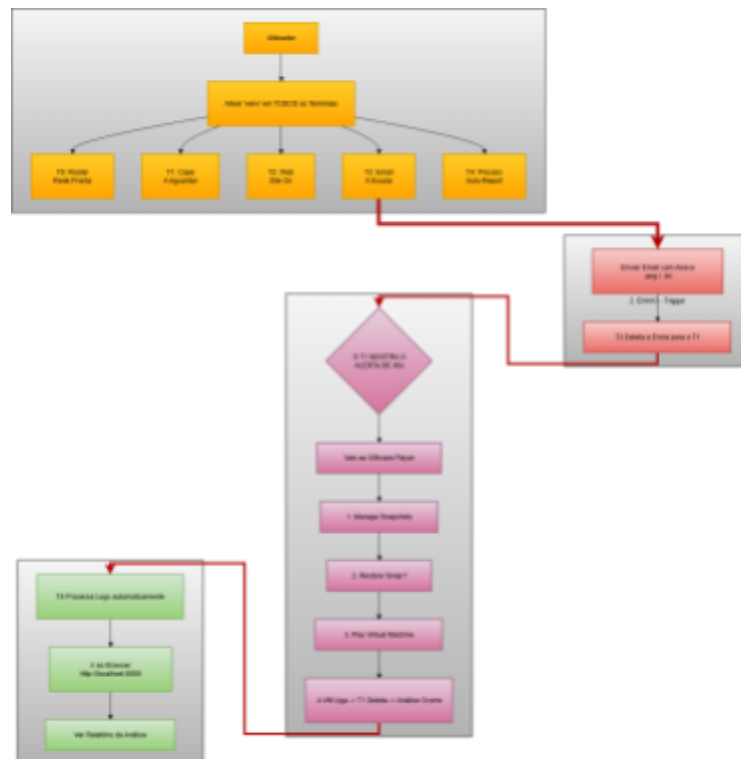


Diagrama de Caso de Uso



2.5. Resultados

Como resultado, foi elaborado um sistema de análise de e-mails que permite detetar malware ou vírus e classificar os ficheiros recebidos na caixa de e-mail em ficheiros infetados e ficheiros "limpos", obtendo assim um projeto de segurança focado na criação, configuração e gestão de scripts dentro de ambientes isolados.

Além disto, foram realizados testes, com o objetivo de comprovar o correto funcionamento do programa e scripts, mediante o envio de ficheiros infetados e não infetados. A seguir mostra-se uma imagem do conteúdo do ficheiro de logs onde pode-se observar a classificação realizada pelo sistema. Podemos ver que o cape classificou ficheiros inofensivos com pontuação baixa e o malware com pontuação quase máxima.

```
cape@tiagoserver:~$ cat ~/organizador.log
[NOVO] ID 7: binary -> SEGURO (Score: 1.799999999999998)
[NOVO] ID 11: fatura.pdf -> INSEGURO (Score: 9.0)
[NOVO] ID 8: IMG_4231.JPG -> SEGURO (Score: 3.199999999999997)
cape@tiagoserver:~$ ls -R ~/Relatorios_Finais/
/home/cape/Relatorios_Finais/:
Inseguro Seguro

/home/cape/Relatorios_Finais/Inseguro:
fatura.pdf

/home/cape/Relatorios_Finais/Seguro:
binary IMG_4231.JPG
```


3. Conclusão

Em suma, este projeto permitiu, conforme mencionado na introdução, adquirir melhores conhecimentos e prática na criação de scripts e sistemas automatizados, bem como compreender a lógica por detrás desses sistemas, as configurações necessárias para a execução deste tipo de projetos e a utilização de novas ferramentas, como a sandbox.

3.1. Limitações do Projeto

Devido à impossibilidade de deteção do *Nested VT*, não foi possível criar a máquina vítima diretamente dentro do Ubuntu. Como alternativa, a máquina Windows foi criada separadamente no VMware. No entanto, desta forma, o CAPE não conseguia aceder diretamente ao ficheiro *.vmx* da máquina vítima. Para contornar esta limitação, foi desenvolvido um script que simula o arranque automático da máquina virtual. Este script introduz um ciclo de espera de 45 segundos, durante o qual é apresentada uma notificação no terminal a indicar que a máquina vítima deve ser ligada manualmente, permitindo posteriormente que o CAPE prossiga a execução como se tivesse sido responsável pelo arranque da VM.