3º WEBINÁRIO

PARA EQUIPES DE PREVENÇÃO, TRATAMENTO E RESPOSTA A INCIDENTES CIBERNÉTICOS (ETIR) DOS ÓRGÃOS PERTENCENTES À REDE FEDERAL DE GESTÃO DE INCIDENTES CIBERNÉTICOS (REGIC)

Data: 1º Out 24

Local: On-line - Plataforma Teams

GABINETE DE SEGURANÇA INSTITUCIONAL

GOVERNO FEDERAL
BRASIL
UNIÃO E RECONSTRUÇÃO

TEMA: Python para executar rotinas de forma rápida e eficiente

09:30 (BRT, UTC -3) – Abertura da sala virtual e admissão dos participantes;

10:00 (BRT, UTC -3) – Abertura do 3º Webinário CTIR Gov 2024;

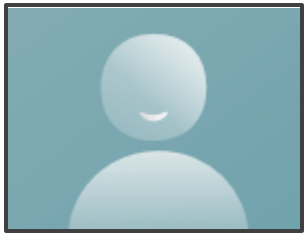
10:10 (BRT, UTC -3) – Introdução;
– Normativas;
– FEEDS Informativos para ETIR;

10:50 (BRT, UTC -3) – Intervalo;

11:00 (BRT, UTC -3) – Demonstração de manipulação de arquivos utilizando Python;

11:40 (BRT, UTC -3) – Perguntas e respostas;

11:50 (BRT, UTC -3) – Encerramento.



3° Webinário para ETIRs integrantes da ReGIC (2024)

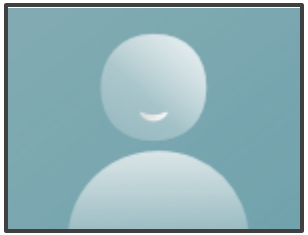
Juliano Prestes Brum e
Leonardo Fagundes

Apresentação

Formação
- Gestão de Tecnologia da Informação - UNISUL
- Atualmente estou cursando Pós Graduação de Privacidade e Seg da Info/UNB

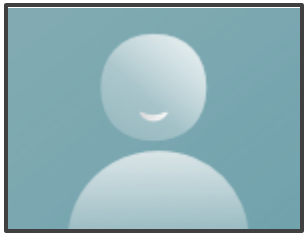
Na área do Tratamento e Resposta a Incidentes
- Curso Fundamentals of Incident Handling
- Curso Advanced Topics in Incident Handling

Software Livre
- Usuário e entusiasta



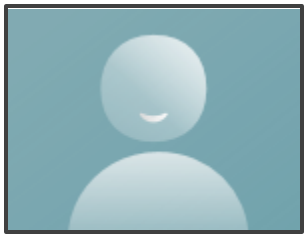
Tema

Python para executar rotinas de forma rápida e eficiente



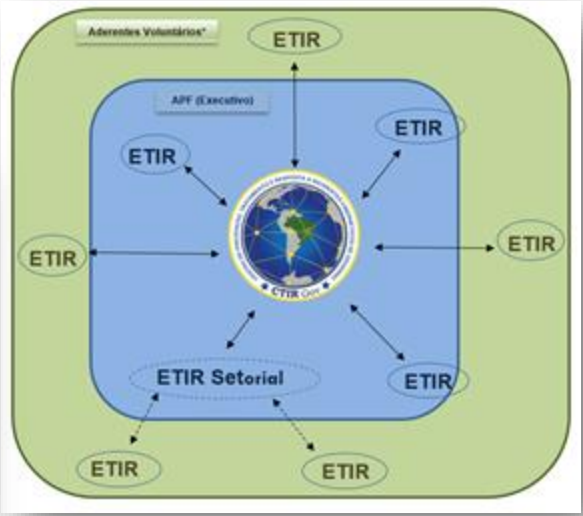
Conteúdo Programático

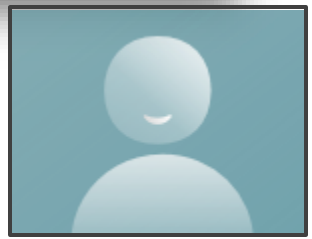
- Instrução Normativa
- Feeds Informativos do CTIR GOV
    - Parceiros
    - Feeds Informativos
    - Implementação / Fluxo
    - Desafios
- Tratar feeds recebidos e analisar as informações utilizando Python
    - O que é Python? e como funciona.
    - Planejamento
    - Estrutura adequada
    - Bibliotecas
    - Variáveis de acesso/controle/saída
    - Manipulação de Arquivos
    - Envio de mensagens automáticas
    - Análise de Dados
    - Produtos
    - Seg e Integração com Ferramentas
    - Boas Práticas na Implementação
    - Automação de Processos



Base Normativa

- Decreto Nº 10.748, de 16 de julho de 2021 que institui a Rede Federal de Incidentes Cibernéticos **(ReGIC)**





Base Normativa

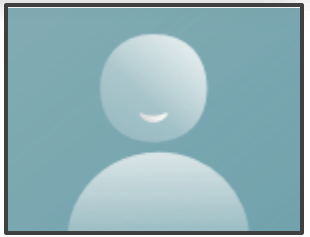
Norma Complementar nº 08/GSI, 19 Ago 10

- **Estabelecer diretrizes para o gerenciamento de incidentes de segurança em redes computacionais** na APF.

Destaque

- **Envolvimento da alta administração**

- **Troca de informações** entre ETIR e CGTIR por intermédio do CTIR GOV.



Base Normativa

Programa de Privacidade e Segurança da Informação **(PPSI)**

- Guia de Gerenciamento de Vulnerabilidades.

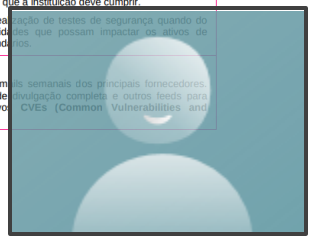
- **Nº 1 Ciclo de detecção**
  - Item 1.2.3 - Garantir o feed de vulnerabilidade mais recente

"**Assine o recebimento** de e-mails semanais dos principais fornecedores.

Assine o banco de dados de divulgação completa e **outros feeds** para acompanhar todos os novos CVEs (Common Vulnerabilities and Exposures)

1.2 Ferramentas

1.2	TAREFA	ENTRADA	SAÍDA
Otimizar ferramentas		1.1 Definir/Refinar escopo 1.4 Confirmar descobertas	1.3 Executar testes 1.4 Confirmar descobertas
#	AÇÃO	POR QUÊ	
1.2.1	Determinar o tipo de teste/varredura	O escopo define os ativos de informação que serão analisados e determina que tipo de teste de segurança será conduzido. Pode-se optar, por exemplo, pelos seguintes testes e varreduras: • Varreduras de rede: credencial vs. varreduras não credenciadas; Varreduras de aplicativos: Análise de Código Estático (**Static Code Analysis - SAST**[7]) vs. Varreduras Dinâmicas (**Dynamic Scans - DAST**[8]); • Testes de segurança de e-mail ou Engenharia Social (Social Engineering- SE). As varreduras de rede são preparadas para detectar correções (patches) não implementadas, erros de configuração e credenciais padrão em servidores web e dispositivos de rede. A varredura credenciada geralmente fornece resultados mais precisos que a não-credenciada. Recomenda-se usar varreduras não-credenciadas para varredura de ativos de informação expostos à Internet. Importante: Quando estiver implantando varreduras pela primeira vez (considerar uma primeira vez para algum grupo de ativos de informação), verificar a "saúde" dos ativos de informação antes e depois. Enquanto o SAST analisa a qualidade do código, o DAST simula ataques reais. Uma estratégia de varredura híbrida que contemple o SAST e o DAST também pode ser utilizada. Testes de segurança de e-mail, ou testes de phishing, são uma maneira de envolver o pensamento crítico dos usuários e evitar perdas. Testes de Engenharia Social (SE), embora não sejam muito comuns, são considerados uma maneira eficaz de conscientizar dos usuários, assim como treinamentos periódicos. **ATENÇÃO: Esteja atento, já que o DAST pode causar danos à aplicação ao servidor Web, e evite a execução do DAST no ambiente de produção.**	
1.2.2	Definir a frequência de seus testes de segurança	O escopo deve fornecer a entrada com base nos requisitos legais, regulamentares e contratuais que a instituição deve cumprir. É importante considerar a rea[illegible]ação de testes de segurança quando do conhecimento de vulnerabilid[illegible]des que possam impactar os ativos de informação primários e secund[illegible]rios.	
1.2.3	Garantir o feed de vulnerabilidade mais recente	Assine o recebimento de e-m[illegible]ils semanais dos principais fornecedores. Assine o banco de dados de divulgação completa e outros feeds para acompanhar todos os novos CVEs (**Common Vulnerabilities and**	



Base Normativa

ISO 27035-1 - Gestão de Incidentes de SI
- Parte 1: Princípios e Processos

- Prepara

- **Detectar**

- Relatar

- Avaliar e responder

- Aplicar lições aprendidas

Base Normativa

Parte 1: Princípios e Processos

- Detectar

- **Recolher informações** sobre consciência situacional a partir do ambiente local de **fontes externas e feeds de notícias.**

- Monitorar sistemas

- Detectar e alertar atividades, suspeitas ou anômalas

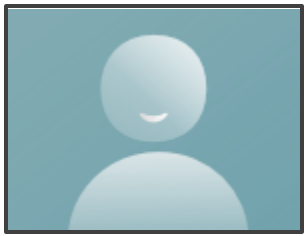
- **Coletar** relatórios de eventos de segurança da Informação de usuários, fornecedores, Outras organizações ou sensores automatizados.

- Relatar eventos

Base Normativa

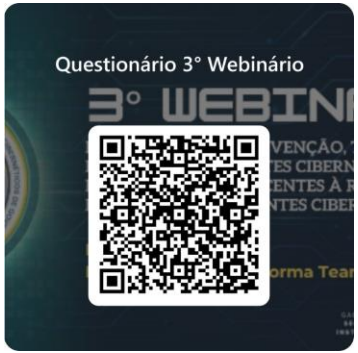
- A **detecção** rápida e precisa é crucial para limitar o impacto de um incidente, permitindo que a equipe responsável inicie os procedimentos de contenção o mais rápido possível. Além disso, uma **comunicação eficiente**, tanto interna quanto externa, garante que os atores certos sejam alertados para mitigar o risco de forma coordenada.

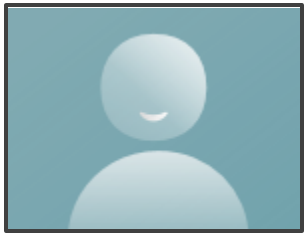
- A detecção através de **diversas fontes** de monitoramento, aliada a uma comunicação ágil, é o primeiro passo crítico no ciclo de resposta a incidentes,



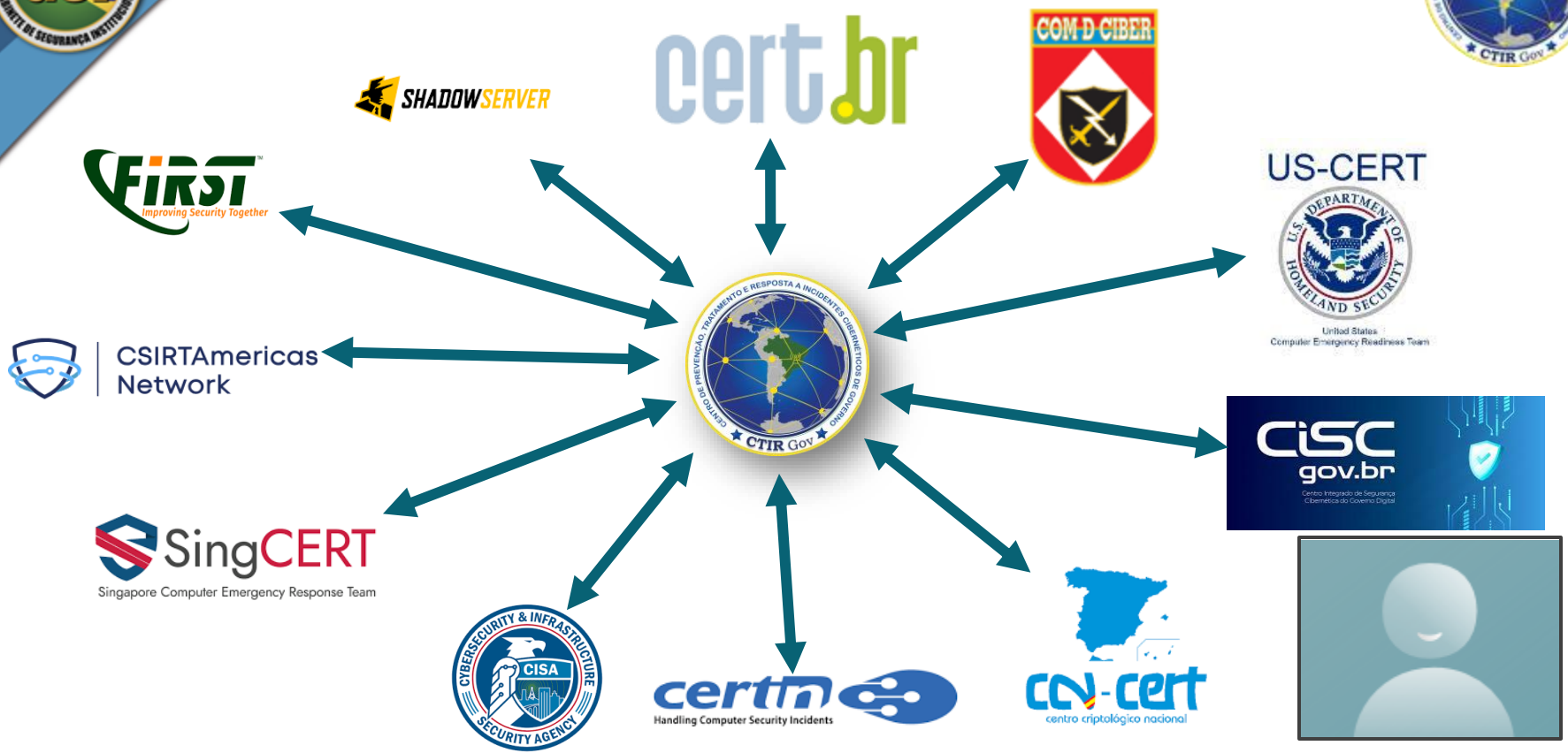
Conteúdo Programático

- Instrução Normativa
- Feeds Informativos do CTIR GOV
  - Parceiros
  - Feeds Informativos
  - Implementação / Fluxo
  - Desafios
- Tratar feeds recebidos e analisar as informações utilizando Python
  - O que é Python? e como funciona.
  - Planejamento
  - Estrutura adequada
  - Bibliotecas
  - Variáveis de acesso/controle/saída
  - Manipulação de Arquivos
  - Envio de mensagens automáticas
  - Análise de Dados
  - Produtos
  - Seg e Integração com Ferramentas
  - Boas Práticas na Implementação
  - Automação de Processos





Parceiros


SHADOWSERVER
cert.br
COM D CIBER
FIRST
Improving Security Together
US-CERT
U.S. DEPARTMENT OF HOMELAND SECURITY
United States Computer Emergency Readiness Team
CSIRTAmericas Network
CTIR Gov
CISC gov.br
Centro Integrado de Segurança Cibernética do Governo Digital
SingCERT
Singapore Computer Emergency Response Team
CYBERSECURITY & INFRASTRUCTURE SECURITY AGENCY
CISA
certn
Handling Computer Security Incidents
CCN-CERT
centro criptológico nacional

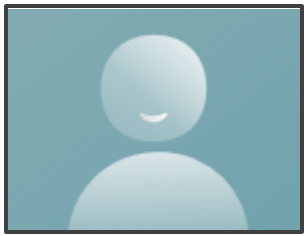
2. Feeds Informativos

1. **A ETIR** precisa coletar feeds informativos de ativos para assegurar uma gestão eficiente de vulnerabilidades, além de manter o inventário atualizado, identificar ameaças emergentes e priorizar a proteção dos ativos críticos

2. **O CTIR GOV**, em parceria com uma ampla rede de colaboradores, recebe um grande volume de informações, que precisam ser verificadas pelos responsáveis pelos ativos para garantir sua precisão e relevância.

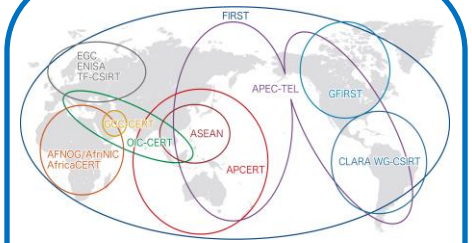
3. Os **Feeds Informativos do CTIR GOV** podem oferecer dados valiosos que permitem à equipe dos órgãos agir de forma proativa, antecipando ataques e fortalecendo a postura de segurança da organização, além de reduzir significativamente a superfície de ataque.

4. Os **Eventos** contribuir para iniciar **Plano de Tratamento e Resposta de Incidentes** e **Planos de Tratamento de Vulnerabilidades**.



Implementação / Fluxo

Parcerias



CSV

PDF

Processamento no CTIR

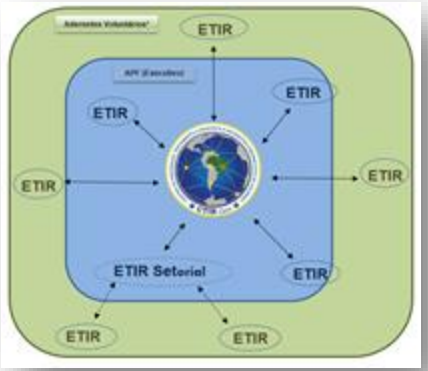
CTIR Gov

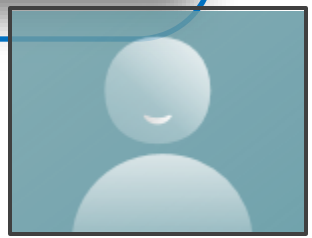
python™

Domínios de interesse

*.gov.br, *.mil.br, *.jus.br, *.leg.br, *.mp.br, *.def.br, *.tc.br, *.eb.br, *.mar.br, *.edu.br, Universidades federais, outros.

Analise nas ETIRs



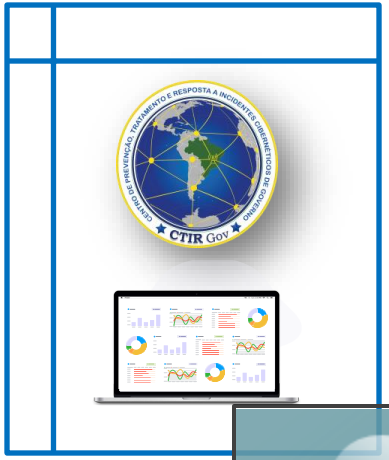


Implementação / Fluxo

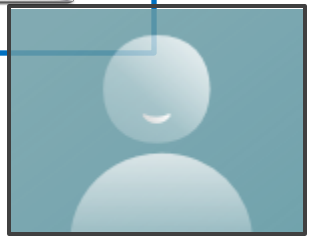
Após analise

Compartilhar com sua constituinte

Registra e coordena



Encarregado "Confirmado Incidente"



Desafios

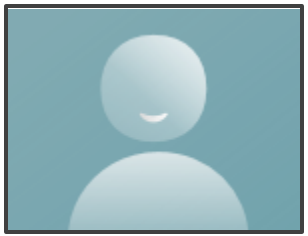
Mapeamento de **fontes confiáveis** e seleção **adequada** dos Feeds.

Tempestividade e consciência situacional.

Mapeamento de domínios e subdomínios.

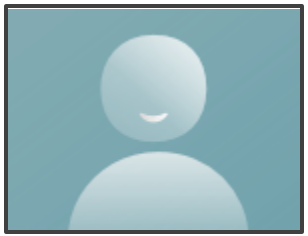
Mapeamento de **IPs**, **ASN**.

Informações a respeito de sistemas que podem estar sendo abusados.



Conteúdo Programático

- **Instrução Normativa**
- **Feeds Informativos do CTIR GOV**
  - Parceiros
  - Feeds Informativos
  - Implementação / Fluxo
  - Desafios
- **Tratar feeds recebidos e analisar as informações utilizando Python**
  - O que é Python? E como funciona.
  - Planejamento
  - Estrutura adequada
  - Bibliotecas
  - Variáveis de acesso/controle/saída
  - Manipulação de Arquivos
  - Envio de mensagens automáticas
  - Análise de Dados
  - Produtos
  - Seg e Integração com Ferramentas
  - Boas Práticas na Implementação
  - Automação de Processos

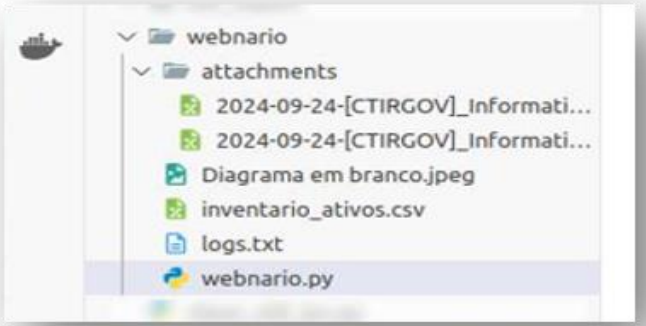


O que é Python?

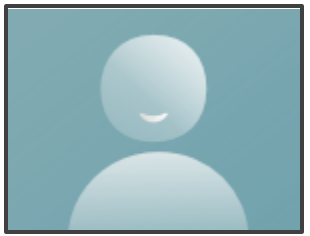
- **Linguagem de Programação Interpretada:**
  - Python é uma linguagem de alto nível, interpretada, o que significa que seu código é executado linha por linha, sem a necessidade de compilação.

- Sintaxe Simples e Legível:
  - O design da linguagem **foca na simplicidade** e na **clareza**, tornando o código fácil de ler e escrever.

- Possui uma enorme variedade de bibliotecas para automação, ciência de dados, inteligência artificial, desenvolvimento web.



- Interpretação do Código:
  - O Python Interpreter lê e executa o código diretamente do arquivo **".py"**, convertendo-o em código executável na máquina.



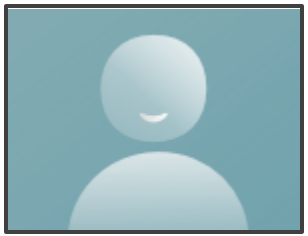
- Python é multiplataform:
  - Significa que o mesmo código pode ser executado em diferentes sistemas operacionais, como Windows, MacOS e Linux.

Programar em Python?

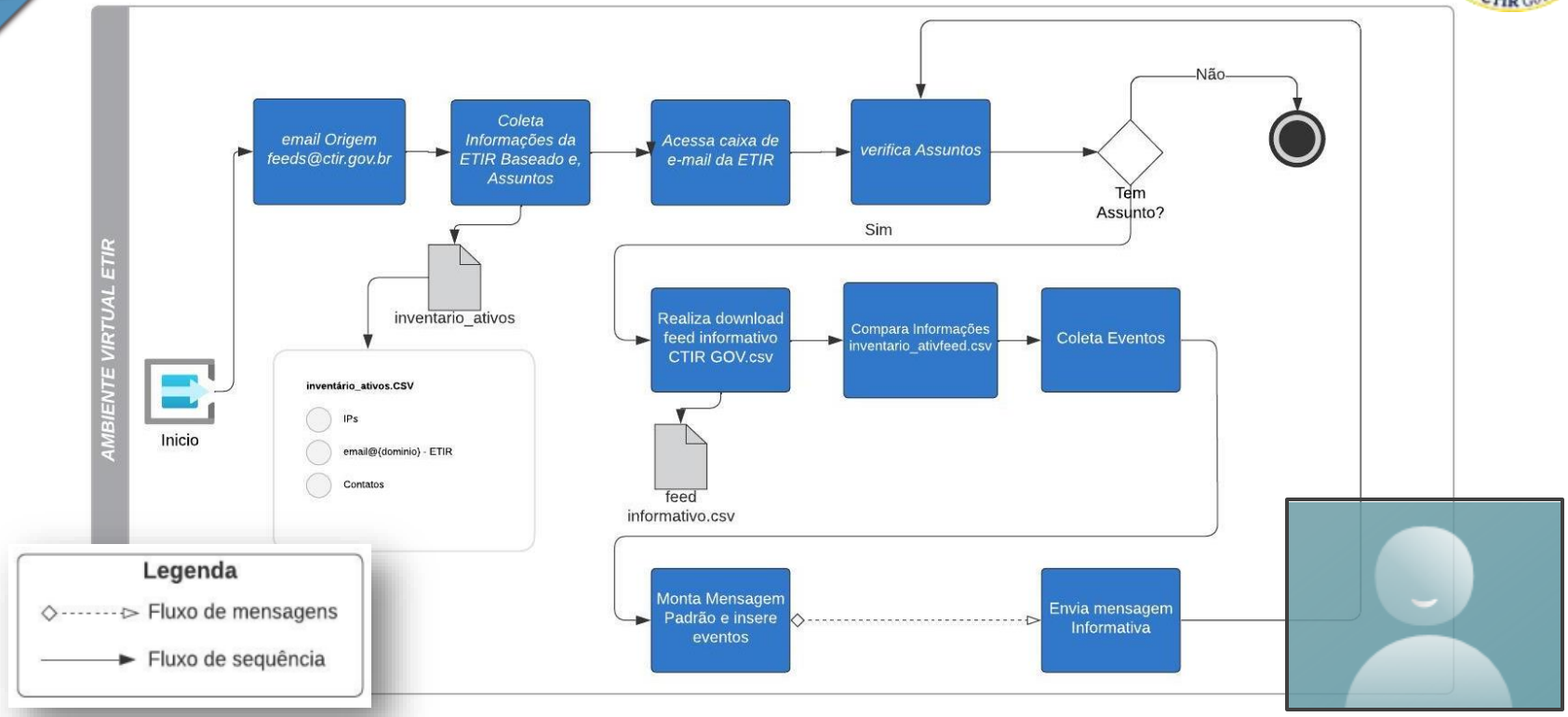
Para programar em Python, é possível utilizar:

- Editor de texto simples como o Bloco de Notas

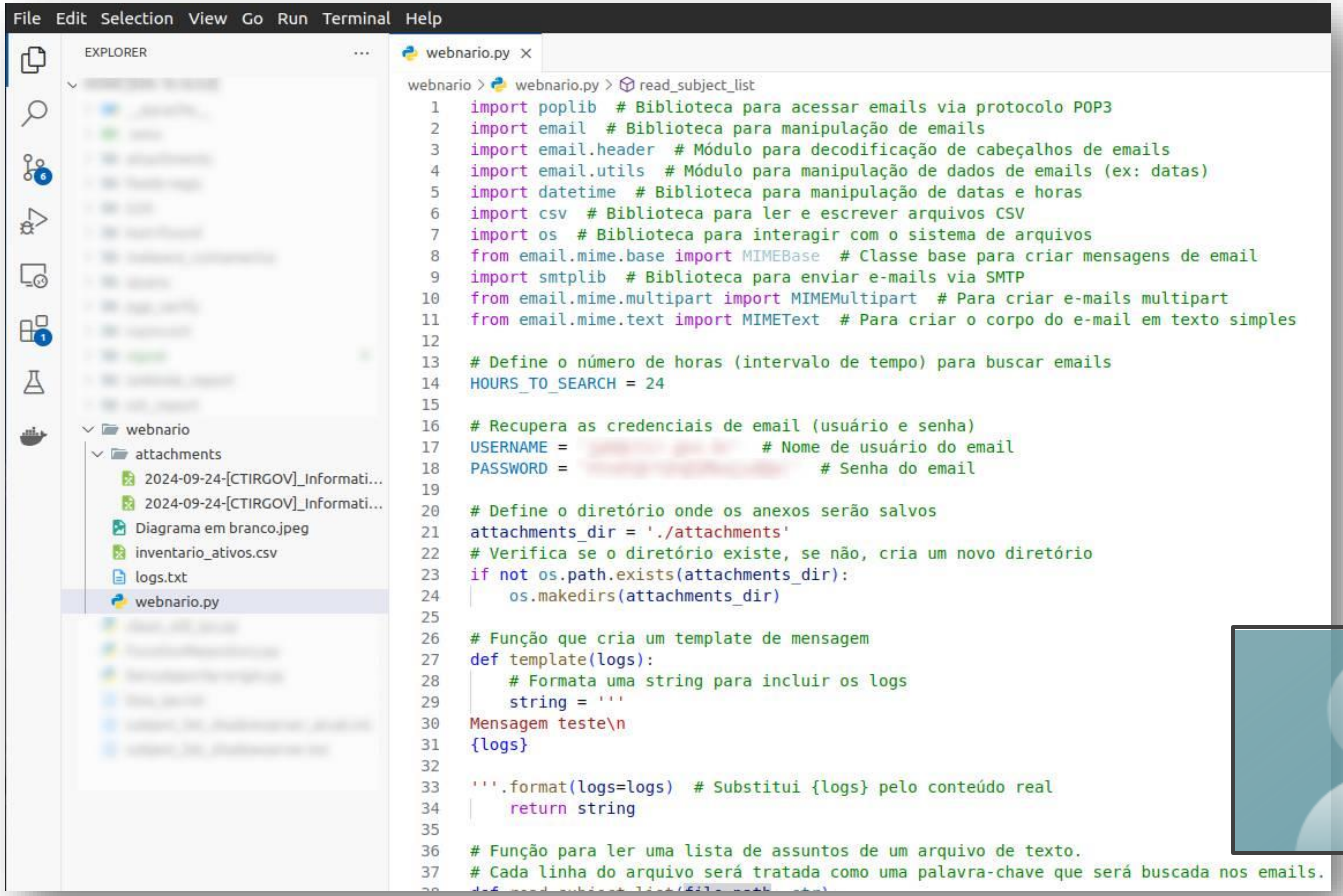
- Ambiente de Desenvolvimento Integrado (IDE).
  - VSCode e PyCharm



Planejamento



Estrutura adequada

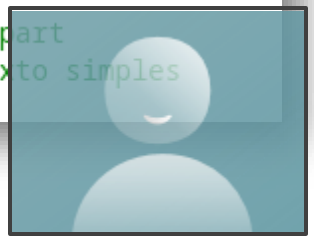


Bibliotecas

webnario.py | 2024-09-24-[CTIRGOV]_Information_feeds-Accessible ICS.csv

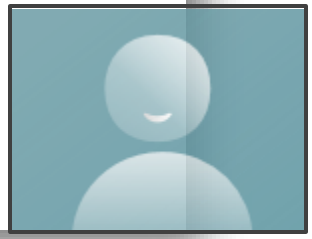
webnario > webnario.py > template

```python
import poplib  # Biblioteca para acessar emails via protocolo POP3
import email  # Biblioteca para manipulação de emails
import email.header  # Módulo para decodificação de cabeçalhos de emails
import email.utils  # Módulo para manipulação de dados de emails (ex: datas)
import datetime  # Biblioteca para manipulação de datas e horas
import csv  # Biblioteca para ler e escrever arquivos CSV
import os  # Biblioteca para interagir com o sistema de arquivos
from email import encoders  # Para codificar anexos de email
from email.mime.base import MIMEBase  # Classe base para criar mensagens de email
import smtplib  # Biblioteca para enviar e-mails via SMTP
from email.mime.multipart import MIMEMultipart  # Para criar e-mails multipart
from email.mime.text import MIMEText  # Para criar o corpo do e-mail em texto simples

```



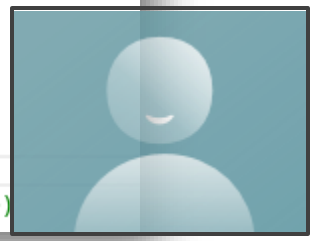
Variáveis de acesso/controle/saída

```
14  import csv  # Biblioteca para ler e escrever arquivos CSV
15  import os  # Biblioteca para interagir com o sistema de arquivos
16  from email.mime.base import MIMEBase  # Classe base para criar mensagens de email
17  import smtplib  # Biblioteca para enviar e-mails via SMTP
18  from email.mime.multipart import MIMEMultipart  # Para criar e-mails multipart
19  from email.mime.text import MIMEText  # Para criar o corpo do e-mail em texto simples
20
21  # Define o número de horas (intervalo de tempo) para buscar emails
22  HOURS_TO_SEARCH = 24
23
24  # Recupera as credenciais de email (usuário e senha)
25  USERNAME = 'seu username'  # Nome de usuário do email
26  PASSWORD = 'sua senha'  # Senha do email
27
28  # Define o diretório onde os anexos serão salvos
29  attachments_dir = './attachments'
30  # Verifica se o diretório existe, se não, cria um novo diretório
31  if not os.path.exists(attachments_dir):
32      os.makedirs(attachments_dir)
33
34  # Função que cria um template de mensagem
35  def template(logs):
```



Manipulação de Arquivos

```python
# Função para ler uma lista de assuntos de um arquivo de texto.
# Cada linha do arquivo será tratada como uma palavra-chave que será buscada nos emails.
def read_subject_list(file_path: str):
    try:
        # Abre o arquivo especificado no caminho e lê suas linhas
        with open(file_path, 'r') as file:
            return file.read().splitlines()  # Retorna as linhas como uma lista de strings
    except FileNotFoundError:
        # Se o arquivo não for encontrado, exibe uma mensagem de erro e retorna uma lista vazia
        print(f"Arquivo não encontrado: {file_path}")
        return []

# Função para ler um arquivo CSV e retornar seu conteúdo como uma lista de dicionários
def read_data_base_csv(file_path):
    data = []  # Inicializa uma lista vazia para armazenar os dados
    try:
        # Abre o arquivo CSV em modo leitura com a codificação UTF-8
        with open(file_path, 'r', encoding='utf-8') as file:
            reader = csv.DictReader(file)  # Lê o arquivo como um dicionário
            for row in reader:
                data.append(row)  # Adiciona cada linha à lista de dados
        return data  # Retorna os dados lidos
    except:
        pass  # Se ocorrer um erro, nada será feito (pode ser melhor detalhar a exceção)
```



Envio de mensagens automáticas

```python
# Função para enviar um email
def send_email(email_content, subject, email_to, email_cc=[]):
    smtp_server = "seu endereço"  # Endereço do servidor SMTP
    smtp_port = 587  # Porta do servidor SMTP
    sender_email = USERNAME  # Email do remetente
    password = PASSWORD  # Senha do remetente

    recipient_email = email_to  # Destinatários principais
    cc_recipient_email = email_cc  # Destinatários em cópia

    msg = MIMEMultipart()  # Cria uma mensagem do tipo multipart
    msg['From'] = sender_email  # Define o remetente
    msg['To'] = ", ".join(recipient_email)  # Define os destinatários principais
    msg['Cc'] = ", ".join(cc_recipient_email)  # Define os destinatários em cópia
    msg['Subject'] = f"[TLP:AMBER] Compartilhamento de informações - {subject} - {datetime.date.today()}

    msg.attach(MIMEText(email_content, 'plain'))  # Anexa o conteúdo do email no formato texto simples

    try:
        server = smtplib.SMTP(smtp_server, smtp_port)  # Cria uma conexão SMTP
        server.starttls()  # Ativa a criptografia TLS
        server.login(sender_email, password)  # Realiza o login no servidor SMTP

        server.send_message(msg)  # Envia a mensagem
        print("\033[1m" + f"Email enviado com sucesso para {', '.join(recipient_email)}!\n" + "\033[0m\n
    except Exception as e:
        print(f"Erro ao enviar e-mail: {e}")  # Exibe mensagem de erro caso ocorra
    finally:
        server.quit()  # Fecha a conexão com o servidor SMTP
```

Análise de Dados

Função:

-Função save_csv_attachments

tem como objetivo salvar anexos do tipo CSV de um email e retornar o caminho do arquivo salvo.

- Criação de diretório:

Verifica se o diretório de anexos existe; se não, cria-o.

- Iteração sobre partes:

Para emails multipart, percorre cada parte em busca de anexos.

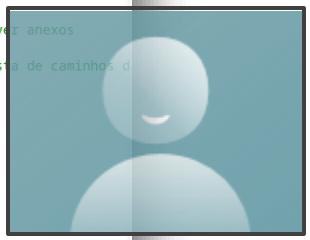
Filtro de CSV:

Verifica se o anexo é um arquivo .csv antes de salvá-lo.

Salvar arquivo:

Escreve o conteúdo do anexo no disco.

```python
# Função para salvar anexos CSV de um email e retornar uma lista com os caminhos dos arquivos salvos
def save_csv_attachments(msg, attachments_dir='./attachments'):
    # Cria o diretório de anexos, se não existir
    if not os.path.exists(attachments_dir):
        os.makedirs(attachments_dir)

    saved_files = []  # Lista para armazenar os caminhos dos arquivos CSV salvos

    # Verifica se o email é multipart (contém anexos ou partes diferentes)
    if msg.is_multipart():
        # Itera sobre as partes do email
        for part in msg.walk():
            # Se a parte é um anexo
            if part.get_content_disposition() == 'attachment':
                # Obtém o nome do arquivo anexo
                filename = part.get_filename()

                # Decodifica o nome do arquivo, se necessário
                if filename:
                    filename = email.header.decode_header(filename)[0][0]
                    if isinstance(filename, bytes):
                        filename = filename.decode('utf-8')  # Decodifica bytes para string

                # Verifica se o arquivo é um CSV
                if filename and filename.endswith('.csv'):
                    # Caminho completo onde o anexo será salvo
                    file_path = os.path.join(attachments_dir, filename)
                    # Salva o anexo no diretório de anexos
                    with open(file_path, 'wb') as f:
                        f.write(part.get_payload(decode=True))  # Salva o conteúdo do anexo
                    print(f"Anexo '{filename}' salvo em: {file_path}")  # Mensagem de confirmação
                    # Adiciona o caminho do arquivo à lista de arquivos salvos
                    saved_files.append(file_path)

    else:
        print("O email não contém partes multipart.")  # Mensagem se o email não tiver anexos

    return saved_files[0] if saved_files else None  # Retorna o primeiro item da lista de caminhos d

def extract_third_level_domain(email: str, level=3) -> str:
    """
    Extrai o nível especificado do domínio de um endereço de e-mail.

    Args:
        email (str): O endereço de e-mail.
        level (int, opcional): O nível do domínio a ser extraído. Padrão é 3.

    Returns:
        str: O domínio extraído.
    """
```



Análise de Dados

Função:

-Função extract_third_level_domain

Extrai o domínio de terceiro nível de um endereço de email.

Dependendo do nível solicitado, a função retorna a parte desejada do domínio (por exemplo, "gov.br" ou "empresa.com").

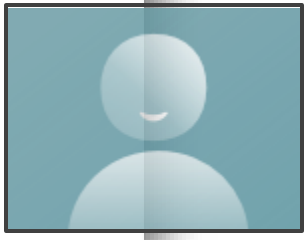
- Função match_domains_in_data

Compara duas listas de dicionários contendo hostnames.

Verifica se o domínio baixado coincide com o domínio presente na base de dados.

A função retorna uma **lista de dicionários**, onde cada dicionário contém um **email de contato** e todas as linhas de dados correspondentes.

```python
    # Divide o email em partes, obtendo o domínio após o símbolo '@'
    parts = email.split('@')[-1].split('.')
    return '.'.join(parts[-level:]) if len(parts) > level else email.split('@')[-1]

def match_domains_in_data(data_base: list[dict], data_downloaded: list[dict]) -> list[dict]:
    """
    Compara duas listas de dicionários e retorna uma lista com os hostnames que estão presentes em a
    juntamente com seus domínios de 3º nível.

    :param data_base: Lista de dicionários que contêm hostnames de referência.
    :param data_downloaded: Lista de dicionários que contêm hostnames baixados ou extraídos.
    :return: Lista de dicionários contendo hostnames correspondentes e seus domínios de 3º nível.
    """

    matches = []  # Inicializa uma lista para armazenar as correspondências

    # Itera sobre cada linha baixada
    for row_downloaded in data_downloaded:
        # Extrai o domínio de 3º nível do hostname baixado
        hostname_downloaded = extract_third_level_domain(row_downloaded.get('hostname', row_downloac

        # Itera sobre cada linha da base de dados
        for row_base in data_base:
            # Extrai o domínio de 3º nível do hostname da base
            hostname_base = extract_third_level_domain(row_base.get('hostname'))
            email_base = row_base.get("contato")  # Obtém o email de contato da base

            # Verifica se os hostnames baixado e da base são iguais
            if hostname_downloaded == hostname_base:
                # Verifica se o email já está em matches
                existing_match = next((match for match in matches if match['email'] == email_base),

                if existing_match:
                    # Se o email já existe, adiciona a nova linha ao 'row'
                    existing_match['rows'].append(row_downloaded)
                else:
                    # Se não existe, cria um novo item em matches
                    matches.append({
                        'email': email_base,
                        'rows': [row_downloaded]  # Coloca a row em uma lista
                    })

    return matches  # Retorna a lista de correspondências

# Lê dados de um arquivo CSV chamado 'inventario_ativos.csv'
data = read_data_base_csv('inventario_ativos.csv')
```



Análise de Dados

Função:

-Função extract_third_level_domain

Extrai o domínio de terceiro nível de um endereço de email.

Dependendo do nível solicitado, a função retorna a parte desejada do domínio (por exemplo, "gov.br" ou "empresa.com").

- Função match_domains_in_data

Compara duas listas de dicionários contendo hostnames.

Verifica se o domínio baixado coincide com o domínio presente na base de dados.

A função retorna uma **lista de dicionários**, onde cada dicionário contém um **email de contato** e todas as linhas de dados correspondentes.

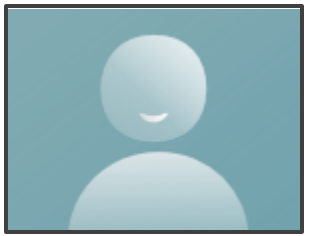
```python
# Inicia a conexão com o servidor POP3 usando SSL
mail = poplib.POP3_SSL(pop3_server, pop3_port)

# Realiza login no servidor POP3 com o nome de usuário e a senha fornecidos
mail.user(USERNAME)
mail.pass_(PASSWORD)

# Obtém o número de mensagens no servidor de email
num_messages = len(mail.list()[1])

# Define o intervalo de tempo a ser buscado (neste caso, 12 horas)
time_interval = datetime.timedelta(hours=HOURS_TO_SEARCH)

# Obtém a data e hora atuais no formato UTC
now = datetime.datetime.now(tz=datetime.timezone.utc)

# Define uma lista de palavras-chave para buscar nos assuntos dos emails
subject_keywords = ['Feeds-CTIR GOV - Accessible ICS', 'Feeds-CTIR GOV - Honeypot Brute Force']

# Loop para percorrer todas as mensagens de email
for i in range(num_messages):
    # Obtém o conteúdo de uma mensagem específica usando o índice da mensagem
    response, lines, octets = mail.retr(i+1)  # 'retr' recupera a mensagem, 'lines' contém as linhas

    try:
        # Tenta decodificar o conteúdo da mensagem como UTF-8
        msg_content = b'\r\n'.join(lines).decode('utf-8')
    except UnicodeDecodeError:
        # Caso UTF-8 falhe, tenta decodificar usando ISO-8859-1
        msg_content = b'\r\n'.join(lines).decode('iso-8859-1')

    # Cria um objeto de email a partir do conteúdo decodificado
    msg = email.message_from_string(msg_content)

    '''
    A partir dessa variável 'msg' conseguimos buscar diversos campos do email como:

    - `msg['Subject']`: O assunto do email.
    - `msg['From']`: O endereço de email do remetente.
    - `msg['To']`: O(s) destinatário(s) principais do email.
    - `msg['Cc']`: Os destinatários em cópia (cópia carbono).
    - `msg['Bcc']`: Os destinatários em cópia oculta (geralmente não disponível em emails recebidos)
    - `msg['Date']`: A data e hora em que o email foi enviado.
    - `msg['Reply-To']`: O endereço de email para onde as respostas devem ser enviadas (caso diferen
    - `msg['Message-ID']`: Um identificador único atribuído ao email.
    - `msg['In-Reply-To']`: Usado para identificar a mensagem original a que o email está respondend
    - `msg['References']`: Lista de mensagens anteriores no encadeamento de emails.
    - `msg['MIME-Version']`: A versão do protocolo MIME utilizada no email.
    - `msg['Content-Type']`: O tipo de conteúdo da mensagem (ex.: `text/plain`, `text/html`, `multip
    - `msg['Content-Transfer-Encoding']`: A codificação usada para transferir o conteúdo (ex.: `7bit
    - `msg['Return-Path']`: Endereço de retorno para mensagens de erro ou falhas na entrega.
    - `msg['Received']`: Um campo múltiplo que mostra o caminho do email através dos servidores.

    Esses campos podem variar dependendo do provedor de email e das configurações específicas,
    e é possível usar o método 'msg.items()' para obter todos os cabeçalhos disponíveis.
    '''
```

Análise de Dados

Variáveis:

- **data:** Carrega os dados de um arquivo CSV 'inventario_ativos.csv'.
- **mail:** Estabelece uma conexão segura com o servidor de e-mail POP3.
- **time_interval:** Define um intervalo de tempo de busca de 12 horas.
- **now:** Captura a data e hora atuais no formato UTC, usada para comparar com o horário de chegada dos e-mails.
- **subject_keywords:** Lista de palavras-chave específicas que serão buscadas no campo de assunto dos e-mails, para identificar mensagens relevantes para análise.

```
192  # Inicia a conexão com o servidor POP3 usando SSL
193  mail = poplib.POP3_SSL(pop3_server, pop3_port)
194
195  # Realiza login no servidor POP3 com o nome de usuário e a senha fornecidos
196  mail.user(USERNAME)
197  mail.pass_(PASSWORD)
198
199  # Obtém o número de mensagens no servidor de email
200  num_messages = len(mail.list()[1])
201
202  # Define o intervalo de tempo a ser buscado (neste caso, 12 horas)
203  time_interval = datetime.timedelta(hours=HOURS_TO_SEARCH)
204
205  # Obtém a data e hora atuais no formato UTC
206  now = datetime.datetime.now(tz=datetime.timezone.utc)
207
208  # Define uma lista de palavras-chave para buscar nos assuntos dos emails
209  subject_keywords = ['Feeds-CTIR GOV - Accessible ICS', 'Feeds-CTIR GOV - Honeypot Brute Force']
210
```



Análise de Dados

Resumida da Execução do Código

- Laço de repetição:

O código percorre todas as mensagens de email utilizando mail.retr(), que recupera o conteúdo de cada mensagem.

- Extração e decodificação do assunto:

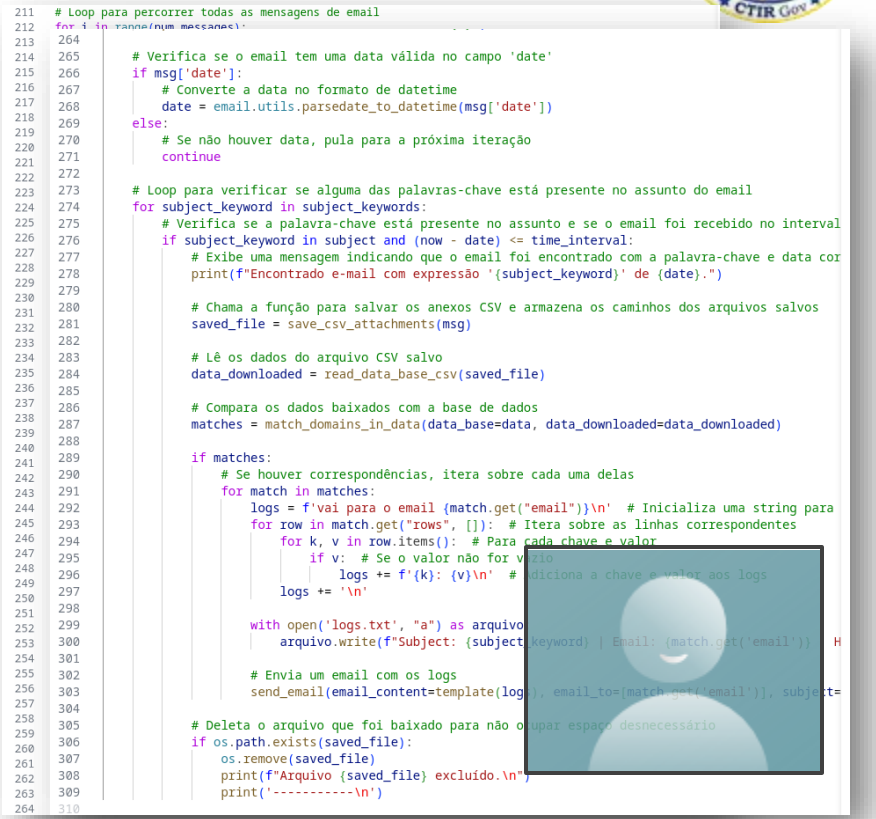
O código compara o assunto do e-mail com uma lista de palavras-chave e verifica se o e-mail foi recebido dentro de um intervalo de tempo específico.

- Processamento do anexo CSV

- Envio de e-mails de alerta

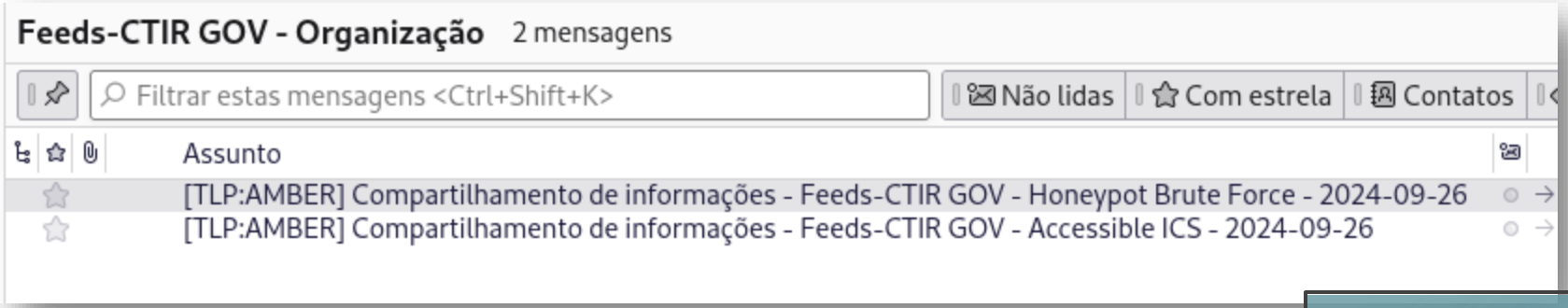
- Registro de logs:

O código também registra as informações importantes em um arquivo de logs (logs.txt) para fins de auditoria e rastreamento.



Produtos

Resultado da automação
- Mensagem e-mail com as informações filtradas
- Destinatário: definido no "inventario_ativos.csv"

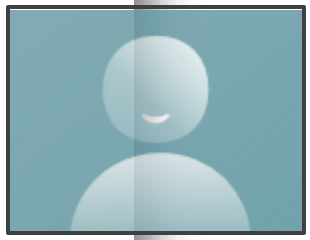


Produtos

Feeds-CTIR GOV - Organização | [TLP:AMBER] Feeds-CTIR GO | [TLP:AMBER] Compartilhame

De	Mim
Para	Mim
Assunto	**[TLP:AMBER] Compartilhamento de informações - Feeds-CTIR GOV - Honeypot Brute Force - 2024-09-26**
X-Account-Key	account1
X-UIDL	000091f662065d07
X-Mozilla-Status	0001
X-Mozilla-Status2	00000000
Return-Path	.gov.br>

```
Mensagem teste

vai para o email
timestamp: 2024-09-15 14:29:17
protocol: tcp
src_ip: 200.196.
src_asn: 10938
src_geo: BR
src_region: PERNAMBUCO
src_city: RECIFE
src_hostname: mail.
severity: critical
public_source: dataplane.org
infection: telnet-brute-force
```



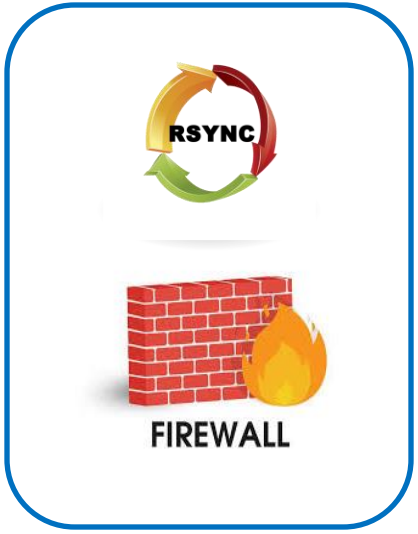
Seg e Integração com Ferramentas

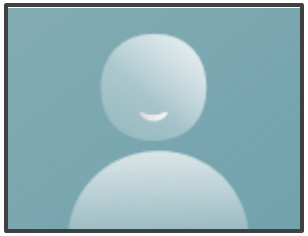
Identificação de Indicadores de Comprometimento:

- Possibilidade de identificação de IPs maliciosos, domínios suspeitos e outros IOCs (Indicadores de Comprometimento) através da comparação com feed.

- **Listas de Bloqueio e Respostas Automatizadas:**
  - Baseando-se nos feeds e nos dados locais, a equipe pode gerar listas de bloqueio (blacklists) ou enviar alertas automatizados.

Após analise

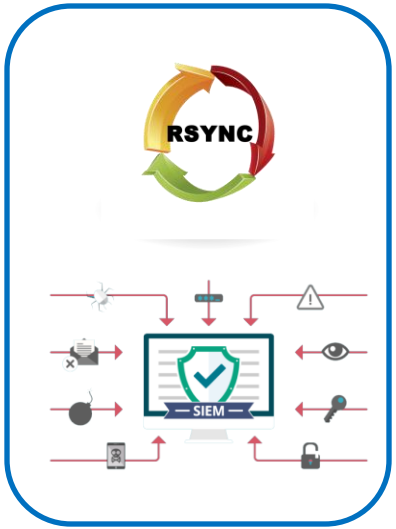


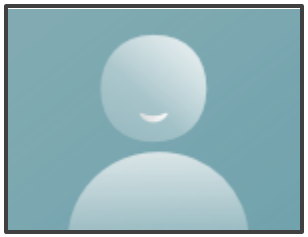


Seg e Integração com Ferramentas

- **Integração com Ferramentas de SIEM (Security Information and Event Management):**
  - Feeds de informações podem ser integrados com SIEM para fornecer dados de ameaças em tempo real, ajudando na automação da detecção e resposta a incidentes. Com a integração dos feeds, o SIEM pode gerar Alertas automáticos quando uma ameaça é detectada em tempo real.

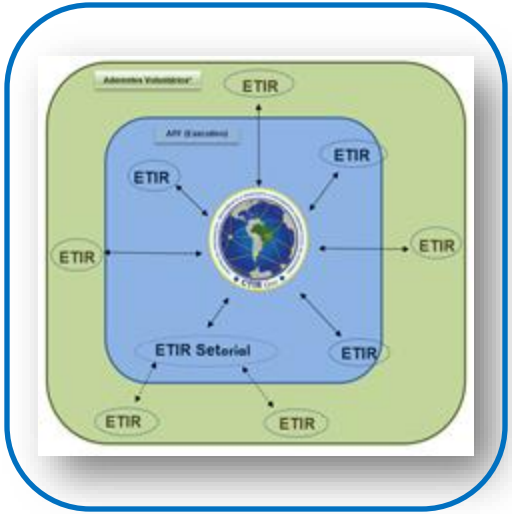
Após analise



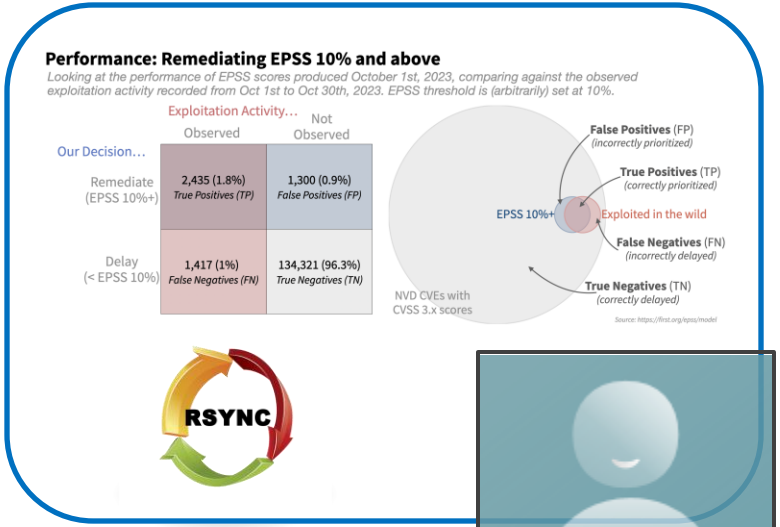


Seg e Integração com Ferramentas

- **Integração com Exploit Prediction Scoring System (EPSS):**
- O modelo serve como um medidor de **explorabilidade**, especificamente estimando a probabilidade de observar tentativas de exploração de uma vulnerabilidade no período subsequente de 30 dias.



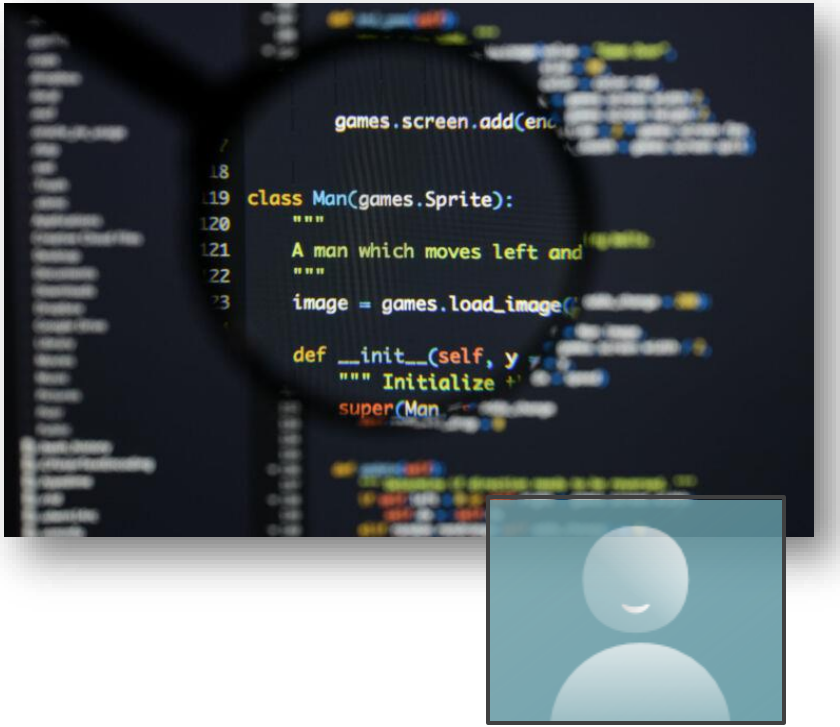
Após analise



Boas Práticas na Implementação

Criar scripts Python para automação exige **planejamento cuidadoso** e a adoção de boas práticas para garantir que o código seja

- **Eficiente**
- **Legível e**
- **Fácil de manutenção**

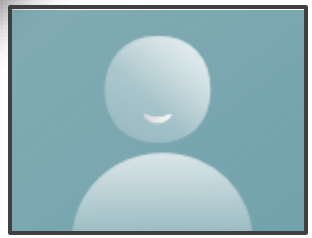


Boas Práticas na Implementação

Para documentar, utilize um sistema de **versionamento**:

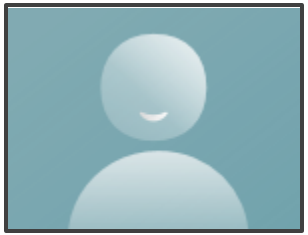
Git, Subversion (SVN), Mercurial

- **Controle de Versões**
- **Colaboração Eficiente**
- **Histórico Completo de Alterações**
- **Facilidade de Recuperação**
- **Integração com outras Ferramentas**
- **Resolução de Conflitos**

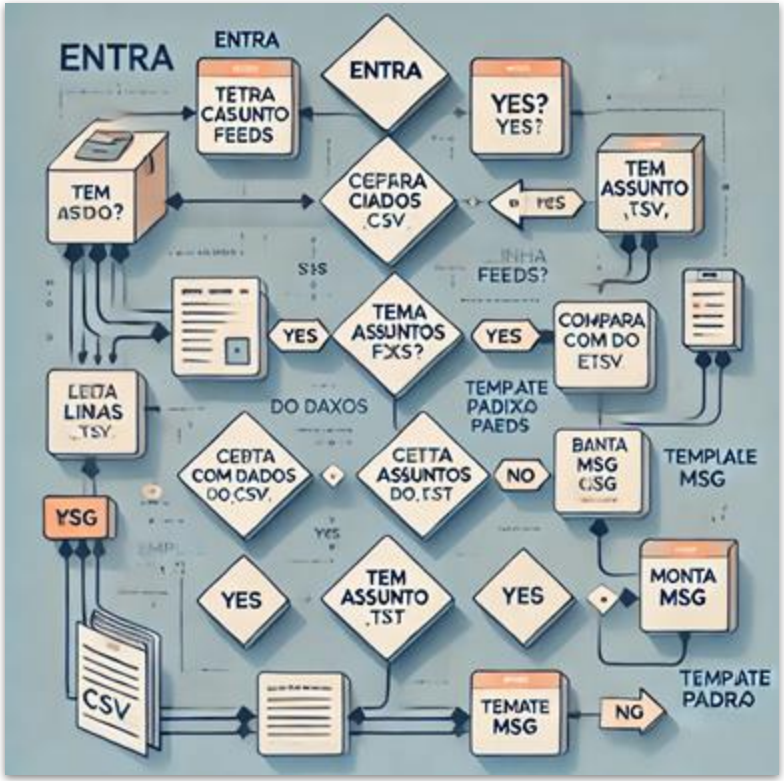


Automação de Processos

- **Automatização de Verificação e Atualização:** Automação para a verificação contínua de feeds recebidos e comparação automática com dados locais. Uso de loops, agendadores (como cron jobs), e bibliotecas como schedule para execução periódica.
- **Exemplo de automação com Crontab:**

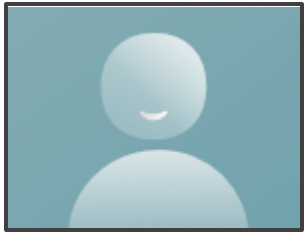


Conclusão



Em conformidade com o Decreto 10.748 e a Norma Complementar 08/IN01/DSIC/GSIPR, o planejamento e a modelagem de scripts em Python tornam-se fundamentais para a automação inteligente na análise de feeds.

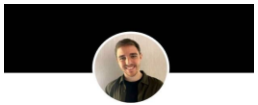
Essa abordagem permite respostas rápidas e precisas a possíveis incidentes, garantindo a integração entre segurança e eficiência na gestão de riscos.



Contatos

Juliano P Brum
Analista CTIR GOV
Juliano.brum@presidencia.gov.br

Formulário de Avaliação 3º Webinário para ETIRs integrantes da ReGIC (2024)



Leonardo Cantieri Taube Fagundes
Responsável pelo desenvolvimento do Script
Leonardo.fagundes@presidencia.gov.br