Ofuscação de código com pyArmor e pyMinifier

Live de Python #187

Roteiro

1. Ofuscação de código

Por que e quando você precisa disso?

2. Estágios da ofuscação

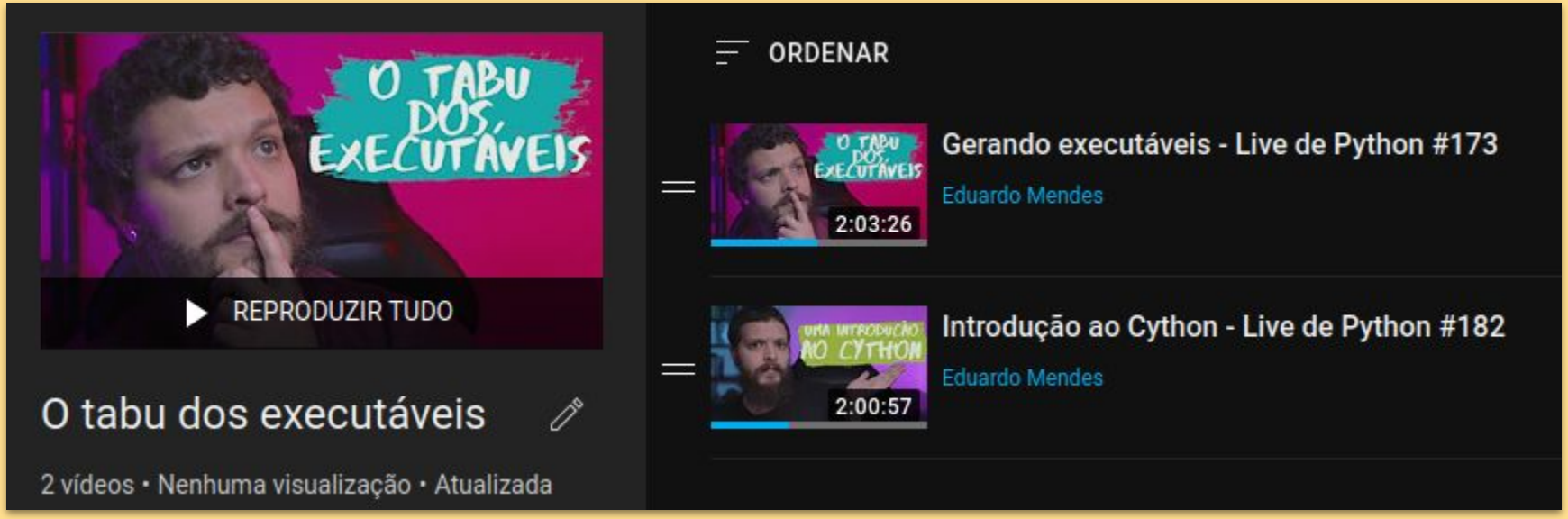
Passeando com pyMinifier

3. pyArmor

A ferramenta definitiva

4. Dicas valiosas

Ou pode ser que não



Disclaimer 1

Ofuscação dificulta, mas não resolve o problema da engenharia reversa!

Disclaimer 2

picpay.me/dunossauro

apoia.se/livedepython

PIX

Ajude o projeto <3

A Earth, Acássio Anjos, Ademar Peixoto, Alex Lima, Alexandre Harano, Alexandre Santos, Alexandre Takahashi, Alexandre Tsuno, Alexandre Villares, Alynne Ferreira, Alysson Oliveira, Amaziles Carvalho, Ana Carneiro, Andre Azevedo, André Rocha, Antonio Neto, Apolo Santos, Arnaldo Turque, Artur Zalewska, Bruno Barcellos, Bruno Freitas, Bruno Guizi, Bruno Oliveira, Bruno Ramos, Caio Nascimento, Carlos Chiarelli, Carlos Eduardo, Cleber Santos, César Almeida, Dartz Dartz, David Kwast, Diego Guimarães, Diego Ubirajara, Dilenon Delfino, Dino Aguilar, Donivaldo Sarzi, Douglas Zickuhr, Emerson Rafael, Eric Niens, Eugenio Mazzini, Euripedes Borges, Fabiano Gomes, Fabio Barros, Fabio Castro, Felipe Rodrigues, Fernando Silva, Flavkaze Flavkaze, Flávio Meira, Francisco Alencar, Franklin Silva, Fábio Barros, Gabriel Sarmento, Gabriel Simonetto, Geandreson Costa, Guilherme Castro, Guilherme Felitti, Guilherme Gall, Guilherme Ostrock, Gustavo Suto, Henrique Junqueira, Henrique Machado, Ismael Ventura, Israel Fabiano, Israel Gomes, Italo Silva, Jair Andrade, Jairo Rocha, Johnny Tardin, Jonatas Leon, Jonatas Oliveira, Jorge Plautz, Jose Mazolini, José Gomes, José Prado, João Lugão, Juan Gutierrez, Julio Silva, Jônatas Silva, Kaio Peixoto, Kaneson Alves, Leandro Miranda, Leonardo Cruz, Leonardo Mello, Lidiane Monteiro, Lucas Barros, Lucas Mello, Lucas Mendes, Lucas Oliveira, Lucas Polo, Lucas Teixeira, Lucas Valino, Luciano Ratamero, Luciano Silva, Luciano Teixeira, Maiquel Leonel, Marcela Campos, Marcelino Pinheiro, Marco Yamada, Marcos Ferreira, Maria Clara, Marina Passos, Matheus Vian, Murilo Cunha, Márcio Martignoni, Natan Cervinski, Nicolas Teodosio, Osvaldo Neto, Patric Lacouth, Patricia Minamizawa, Patrick Brito, Patrick Gomes, Paulo Tadei, Pedro Henrique, Pedro Kulaif, Pedro Pereira, Peterson Santos, Priscila Santos, Rafael Lino, Reinaldo Silva, Renan Gomes, Renan Moura, Revton Silva, Richard Nixon, Riverfount Riverfount, Robson Maciel, Rodrigo Ferreira, Rodrigo Freire, Rodrigo O'neal, Rodrigo Vaccari, Ronaldo Silva, Rui Jr, Samanta Cicilia, Sandro Mio, Sara Selis, Silvio Xm, Thiago Araujo, Thiago Borges, Thiago Bueno, Thiago Moraes, Tony Dias, Tyrone Damasceno, Victor Wildner, Vinícius Bastos, Vlademir Souza, Vladimir Lemos, Vítor Gomes, Wellington Abreu, Wesley Mendes, Willian Lopes, Willian Rosa, Wilson Duarte, Yuri Fialho, Yury Barros, Érico Andrei

Obrigado você

Como isso funciona?

Ofuscação

O que é ofuscação?

É uma técnica usada para deixar o código mais difícil de ser lido.

```
1   # exemplo_01.py
2   print('Live de Python')
```

O que é ofuscação?

```
1   # exemplo_01.py
2   print('Live de Python')
```

Ofuscação

```
1 # __pycache__/exemplo_01.cpython-39.pyc
2 a
3 ^@^@^@^@-
  ??a^X^@^@^@?^@^@^@^@^@^@^@^@^@^@^@^@^
  @^@^B^@^@^@@^@^@^@s^L^@^@^@e^@d^@?^A^A^@d
  ^AS^@)^Bz^NLive de
  pythonN)^A?^Eprint?^@r^B^@^@^@r^B^@^@^@?^
  Mexemplo_01.py<module>^A^@^@^@?^@^@^@^@
```

Mas eu preciso disso?

Em alguns casos e em algumas linguagens isso é um processo padrão.
Por dois motivos:

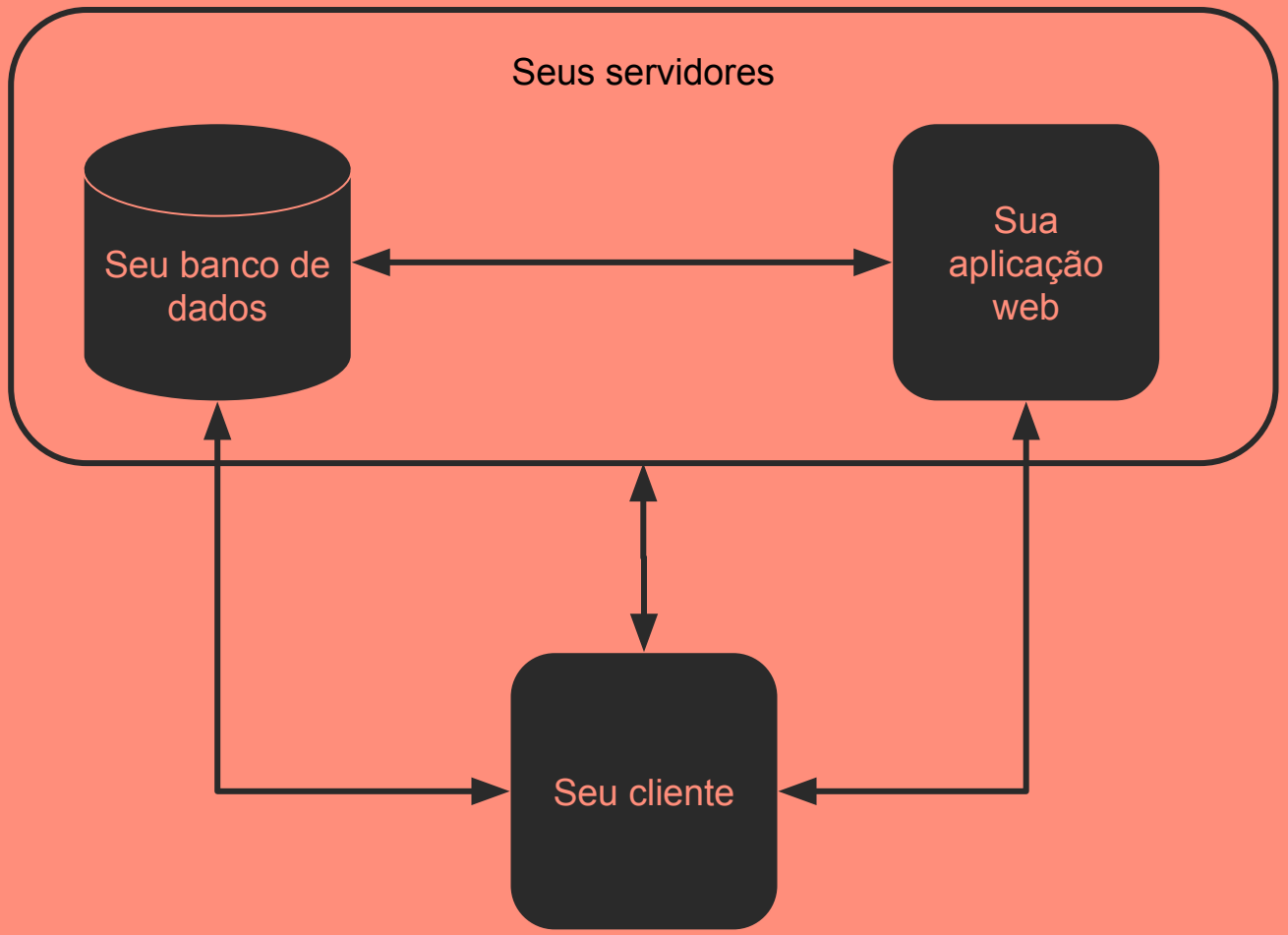
- Segurança
- Otimização

Mas eu preciso disso?

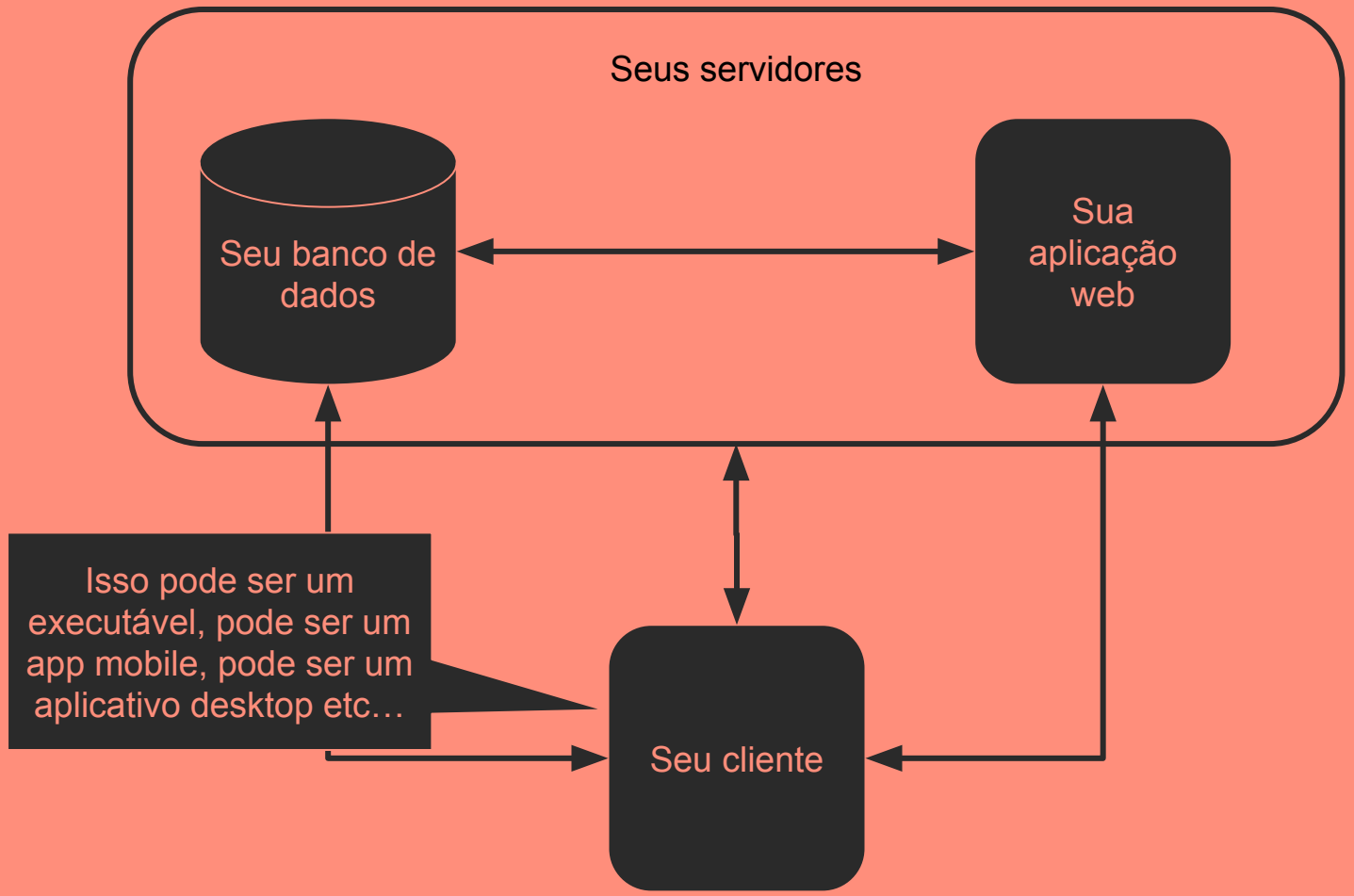
Em alguns casos e em algumas linguagens isso é um processo padrão.
Por dois motivos:

- Segurança
  - **Esconder dados sensíveis**
  - Proteção de propriedade (???)
  - Dificultar engenharia reversa
- Otimização
  - Diminuir tamanho do código-fonte (alô, JS)

Tá, vamos para o mundo real



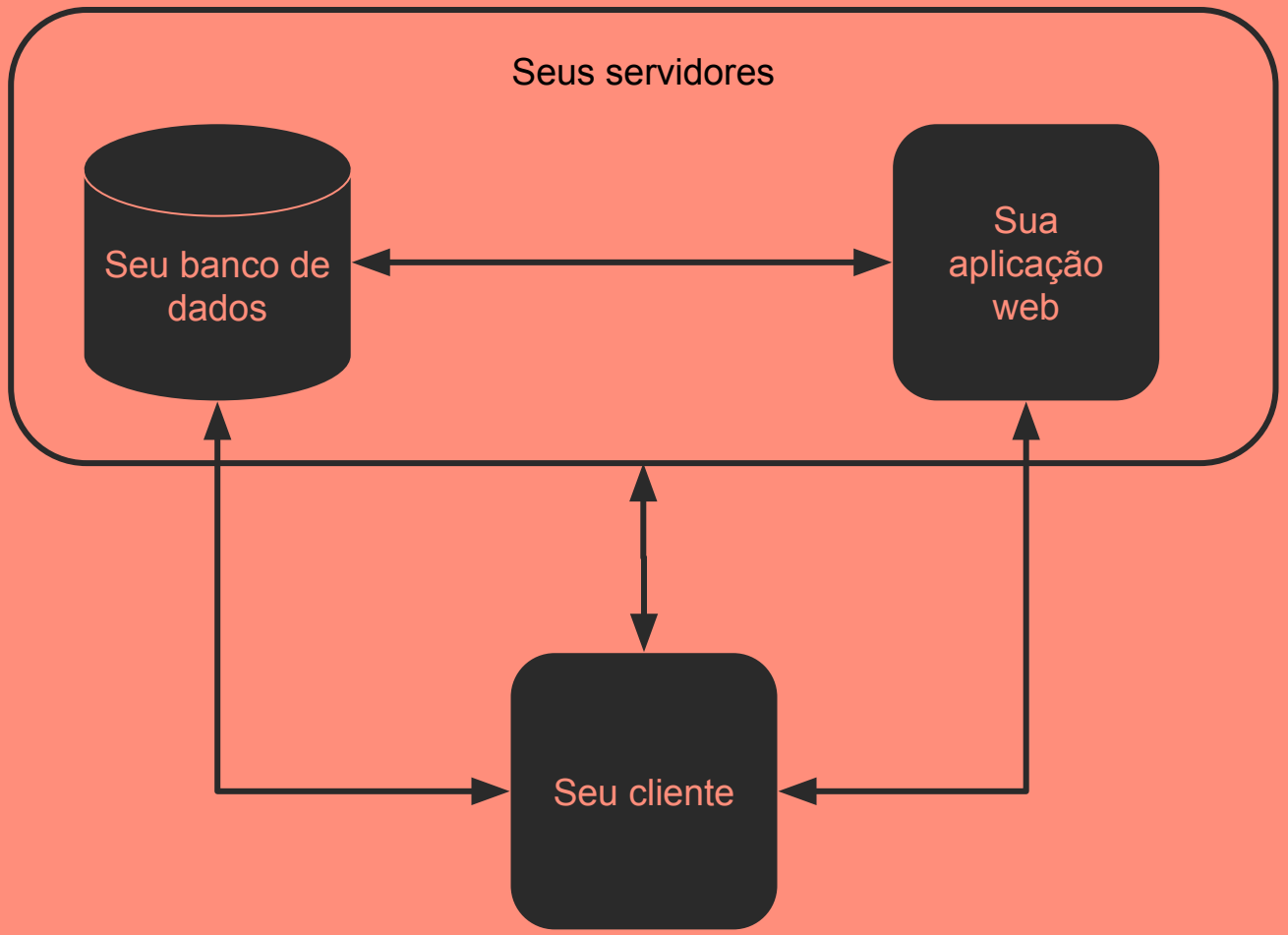
Tá, vamos para o mundo real



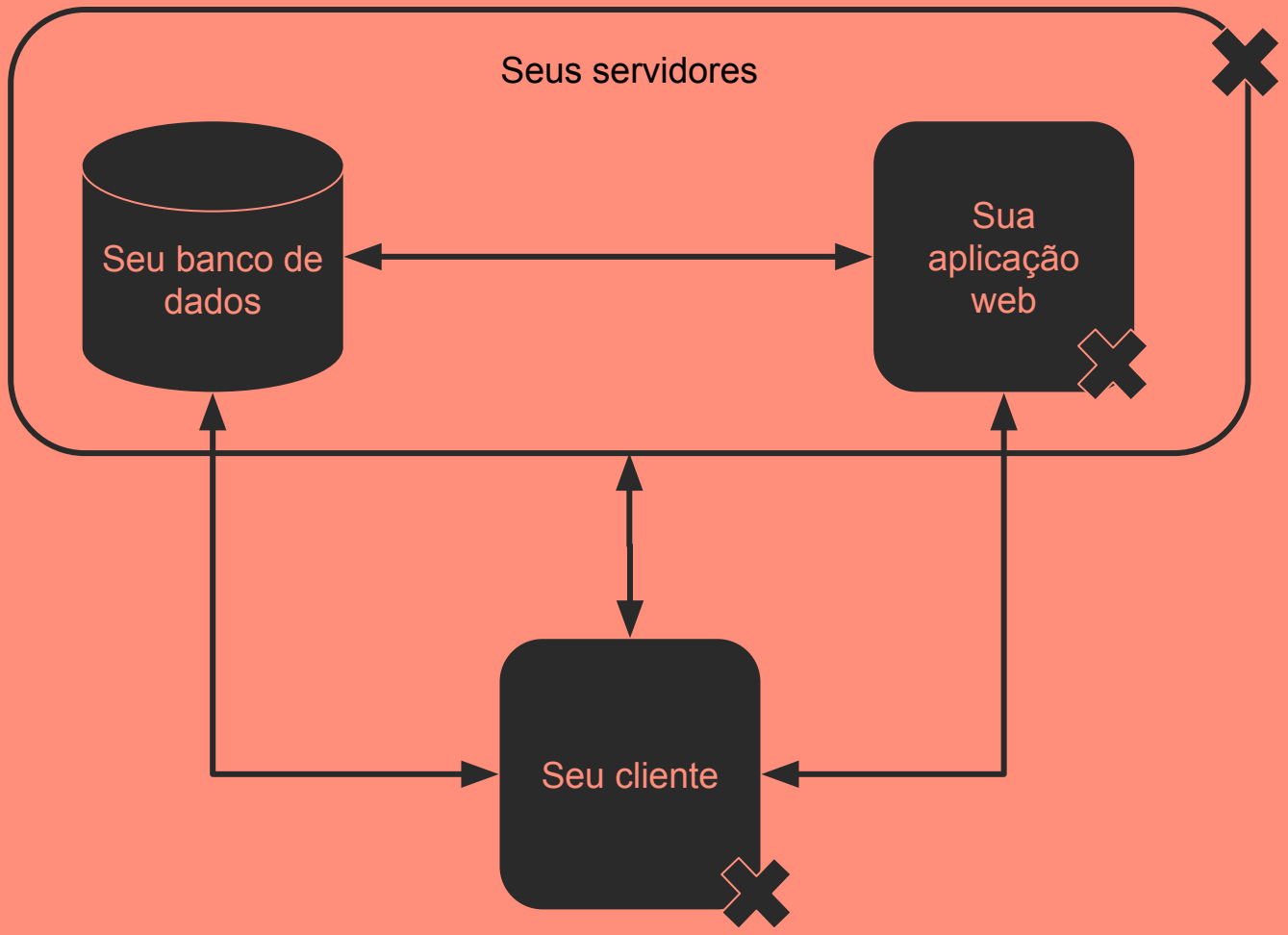
O nível de segurança geral da sua aplicação é equivalente ao seu elo mais fraco

Vamos pensar um pouco

Onde isso pode dar problema?



Onde isso pode dar problema?



Looking inside the (Drop) box

Dhiru Kholia
Openwall / University of British Columbia
dhiru@openwall.com

Przemysław Węgrzyn
CodePainters
wegrzyn@codepainters.com

Abstract

Dropbox is a cloud based file storage service used by more than 100 million users. In spite of its widespread popularity, we believe that Dropbox as a platform hasn't been analyzed extensively enough from a security standpoint. Also, the previous work on the security analysis of Dropbox has been heavily censored. Moreover, the existing Python bytecode reversing techniques are not enough for reversing *hardened* applications like Dropbox.

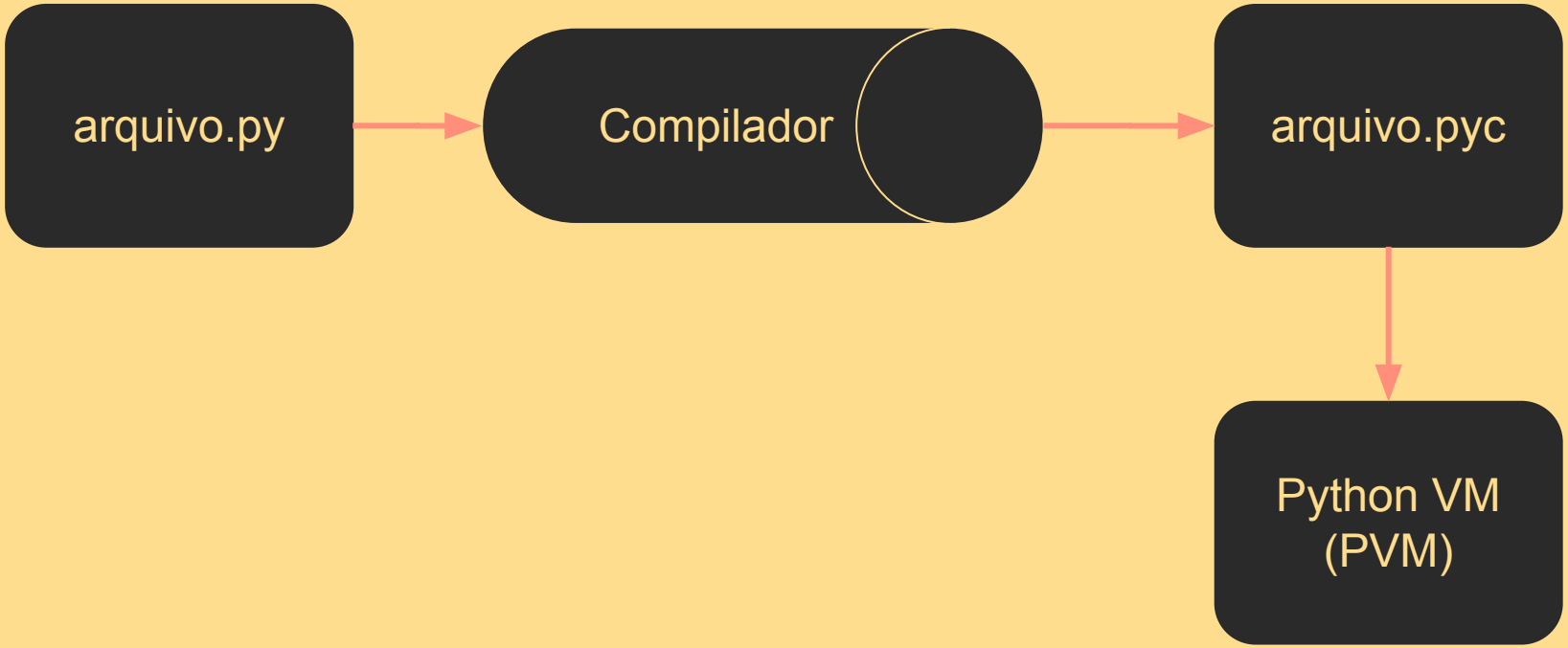
This paper presents new and generic techniques, to reverse engineer *frozen* Python applications, which are not more. In this paper, we show how to unpack, decrypt and decompile Dropbox from scratch and in full detail. This paper presents new and generic techniques to reverse engineer frozen Python applications. Once you have the decompiled source-code, it is possible to study how Dropbox works in detail. This Dropbox source-code reversing step is the foundation of this paper and is described in section 3.

Our work uses various code injection techniques and monkey-patching to intercept SSL data in Dropbox client. We have used these techniques successfully to snoop on SSL data in other commercial products as well.

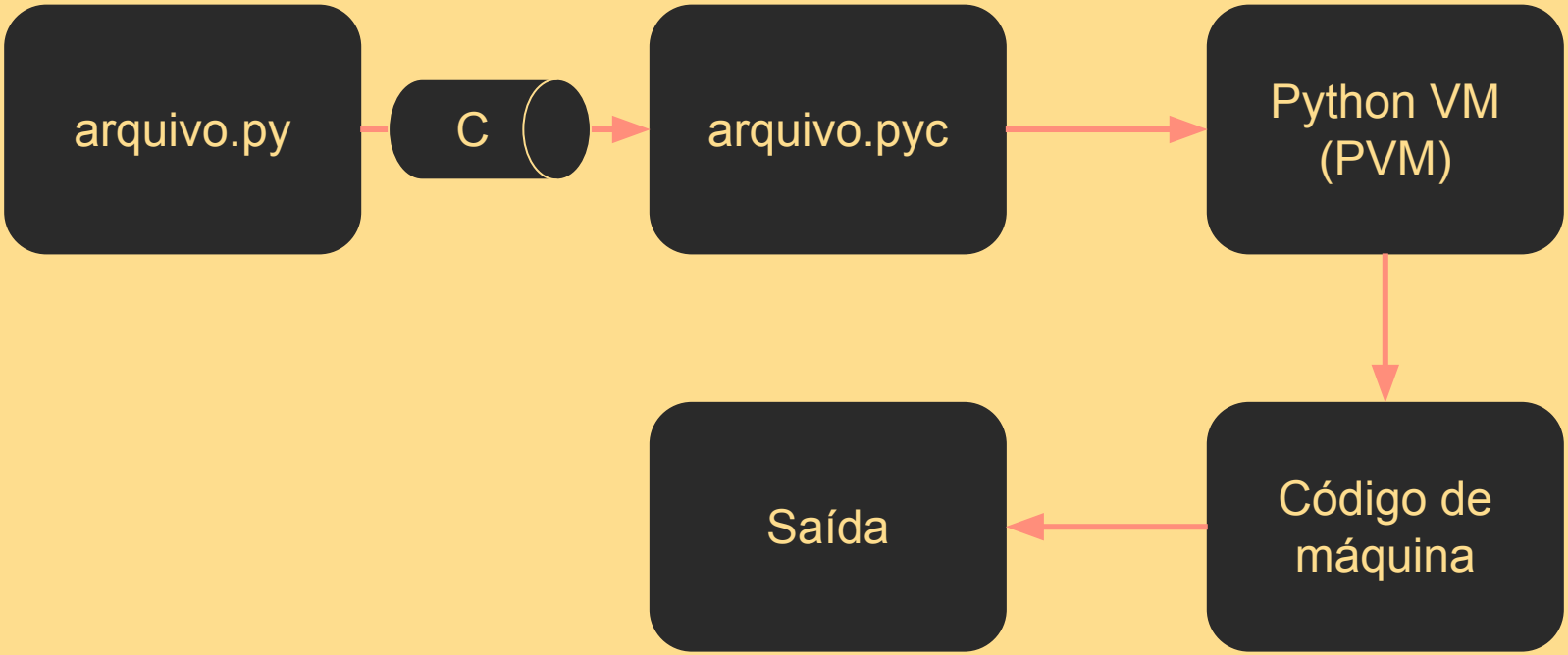
https://www.usenix.org/system/files/conference/woot13/woot13-kholia.pdf

Uma revisão básica

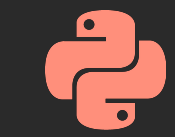
Python é uma linguagem compilada, não em linguagem de máquina, mas em byte-code. Esse byte-code é interpretado pela máquina virtual do python.



Uma revisão básica



Como posso compilar meu código?

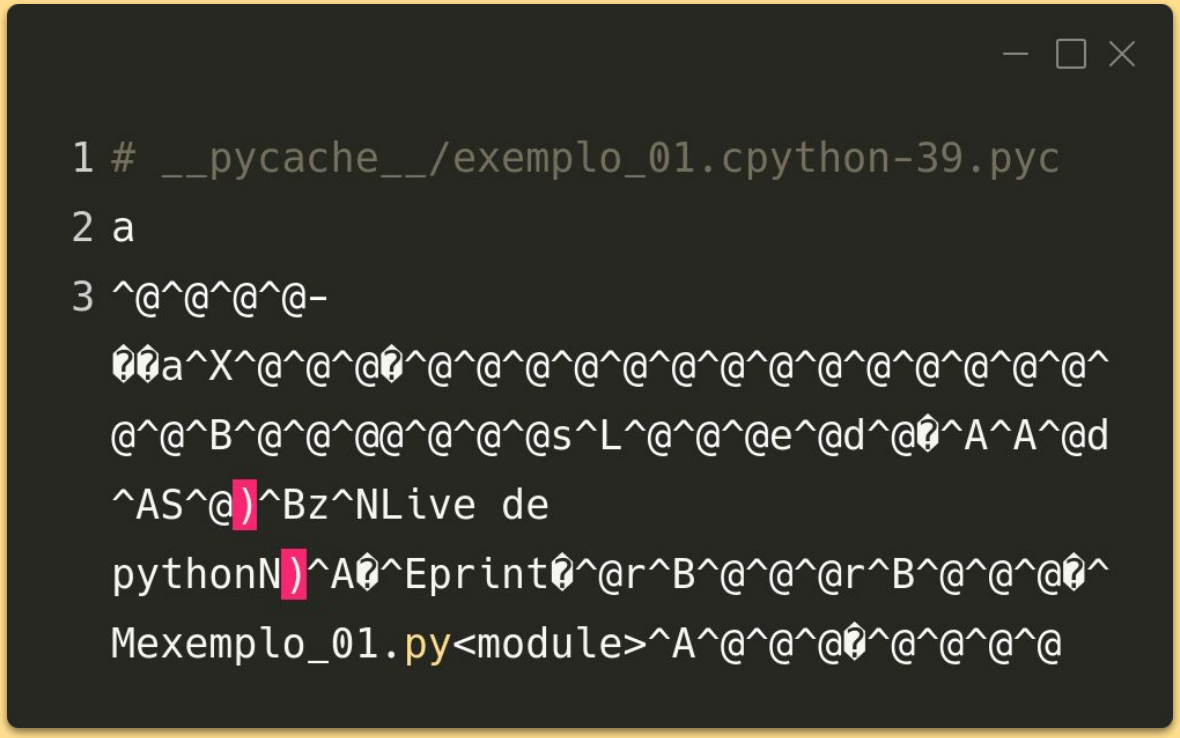


```
python -m compileall exemplo_01.py
```

```
1  # exemplo_01.py
2  print('Live de Python')
```

```
1 # __pycache__/exemplo_01.cpython-39.pyc
2 a
3 ^@^@^@^@-
  ��a^X^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^
  @^@^B^@^@^@@^@^@^@s^L^@^@^@e^@d^@@^A^A^@d
  ^AS^@)^Bz^NLive de
  pythonN)^A@^Eprint@^@r^B^@^@^@r^B^@^@^@@^
  Mexemplo_01.py<module>^A^@^@^@@^@^@^@^@
```

Mas o que é isso?



```
1 # __pycache__/exemplo_01.cpython-39.pyc
2 a
3 ^@^@^@^@-
  ??a^X^@^@^@?^@^@^@^@^@^@^@^@^@^@^@^@^@^@^
  @^@^B^@^@^@@^@^@^@s^L^@^@^@e^@d^@?^A^A^@d
  ^AS^@)^Bz^NLive de
  pythonN)^A?^Eprint?^@r^B^@^@^@r^B^@^@^@?^
  Mexemplo_01.py<module>^A^@^@^@?^@^@^@^@
```

Byte-code; IL; Intermediated Language, Linguagem intermediária e etc...

Não, um byte-code não é "seguro", as respostas do stackoverflow estão erradas!

DICA!

Não, um byte-code não é "seguro", as respostas do stackoverflow estão erradas!

Mas sim, para um humano está ofuscado xD

DICA!

Como não?

```
pip install decompile3
decompyle3 __pycache__/exemplo_01.cpython-38.pyc
```

```
# __pycache__/exemplo_01.cpython-39.pyc
a
^@^@^@^@-
  ??a^X^@^@^@?^@^@^@^@
```

pyMinifier

Ofuscando de verdade!

pyMinifier

Uma biblioteca de linha de comando para minificar, ofuscar e comprimir código python

```
$ pip install pyminifier
```

pyMinifier

O pyMinifier pode ofuscar camadas interessantes do seu código.
Exemplos interessantes:

- nomes de classes
  - --obfuscate-classes
- nomes de funções
  - --obfuscate-functions
- Variáveis
  - --obfuscate-variables
- imports
  - --obfuscate-import-methods
- builtins
  - --obfuscate-builtins

```
$ pyminifier --obfuscate-<algo> script.py -o ofuscado.py
```

```python
from httpx import get, post

username = 'dunossauro'
password = 'my_scret'

class MyClass:
    """Essa classe é só sucesso."""
    def __init__(self):
        self.atributo = 1

    def my_method(self):
        return self.atributo

def google_is_up():
    """Checa se o google está de pé."""
    response = get('http://google.com')
    return response

def login(user, password):
    """Faz um login maroto."""
    response = post(
        'http://meu_site.com',
        json={'username': user, 'password': password}
    )
    return response

print(google_is_up())
print(login(username, password))
```

Caso eu esqueça

Opções mega interessantes no minifier

- Ofuscação em código não latino
  - --nonlatin
- Tamanho da substituição
  - --replacement-length=x
- Compressão
  - --gzip

Isso impede de alguém
fazer engenharia reversa?

Não

Não!

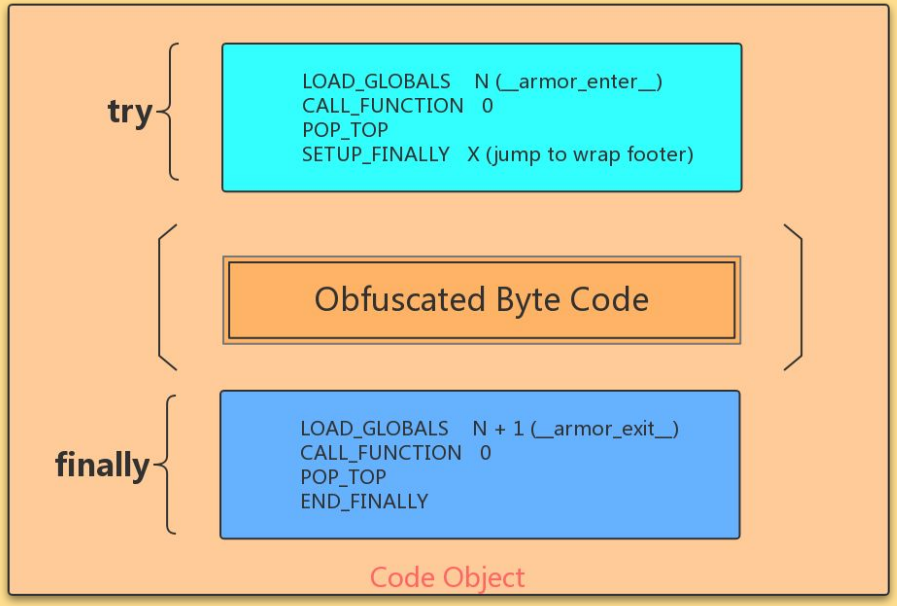
pyArmor

Ofuscação segura!

pyArmor

PyArmor é uma ferramenta feita e pensada para ofuscação segura.

Diferente do minifier, que ofusca o código python. O pyarmor modifica o byte-code usando a camada do Cpython



O que isso quer dizer?

Que a depuração do código desce um nível e não acontece mais na camada do Python e sim na camada do C.

A ofuscação, por padrão é feita em dois níveis.

- Função a função
- Depois ao nível de módulo

```
def google_is_up():
    """Checa se o google está de pé."""
    response = get('http://google.com')
    return response

def login(user, password):
    """Faz um login maroto."""
    response = post(
        'http://meu_site.com',
        json={'username': user, 'password': password}
    )
    return response
```

WHAT?

Isso significa que caso o código do módulo seja desofuscado via engenharia reversa, o código ainda estará ofuscado.

Bom, vamos lá

```
$ pip install pyarmor
$ pyarmor obfuscate script.py
```

O que aconteceu?

```python
# teste.py
from pytransform import pyarmor_runtime
pyarmor_runtime()
__pyarmor__(__name__, __file__, b'\x50\x59\x41\x52\x4d\x4f\x52\x00\x00...', 2)
```

```
$ lt dist/
dist
├── pytransform
│   ├── __init__.py
│   └── _pytransform.so
└── teste.py
```

O que aconteceu?

```python
# teste.py
from pytransform import pyarmor_runtime
pyarmor_runtime()
__pyarmor__(__name__, __file__, b'\x50\x59\x41\x52\x4d\x4f\x52\x00\x00...', 2)
```

```
$ lt dist/
dist
├── pytransform
│   ├── __init__.py
│   └── _pytransform.so
└── teste.py
```

o _pytranform pode mudar de SO para SO.

.so: linux
.dll: windows
.lib: macos

_pytransform

Que raios é esse diretório pytransform? Proteção cruzada.

- _pytransform é escrito em C
- Projete os scripts ofuscados
- JIT:
  - Evita que a chave DES usada para criptografar seja encontrar no debugger
  - Faz com que o código não possa ser alterado em memória, depois que estartado (memdump)

bootstrap

```
# teste.py
from pytransform import pyarmor_runtime
pyarmor_runtime()
__pyarmor__(__name__, __file__, b'\x50\x59\x41\x52\x4d\x4f\x52\x00\x00...', 2)
```

Licenças

Quando somente ofuscar não basta. As licenças podem ser usadas. Licenças de maneira geral são algumas limitações ou restrições que você pode adicionar ao seu código ofuscado. Como:

- Data de expiração
  - --expired YYYY-MM-DD
- Vincular ao número de série do HD
  - --bind-disk <Numero>
- Vincular ao ip
  - --bind-ipv4 IPV4
- Vincular ao mac address
  - --bind-mac MACADDR
- Execução por período
  - --enable-period-mode

Usando as licenças

```
$ pyarmor hdinfo # pega as informações
$ pyarmor licenses --expired 2019-01-01 deu_ruim_em_2019
$ pyarmor obfuscate --with-license <path>/license.lic teste.py
```

um minutinho, pelo menos

Atenção

Problemas ...

1. O melhor do pyArmor é pago
    a. O que implica e não poder nem testar todos os modos de ofuscação
    b. Ter que fazer vendoring na versão trial
2. Não lida bem com outros binários na ofuscação
3. Problemas com NVMe para licenças de disco
4. A geração dos executáveis só podem ser feitas com pyinstaller
    a. Ter que fazer vendoring em casos onde o pyinstaller falha
    b. Não existe uma integração clara com o pyinstaller

picpay.me/dunossauro

apoia.se/livedepython

PIX

Ajude o projeto <3