



A.A. 2022-2023



StealBot

Università degli studi di Napoli

Federico II



Valentino Bocchetti - N86003405



Valentina Annunziata - N86003280

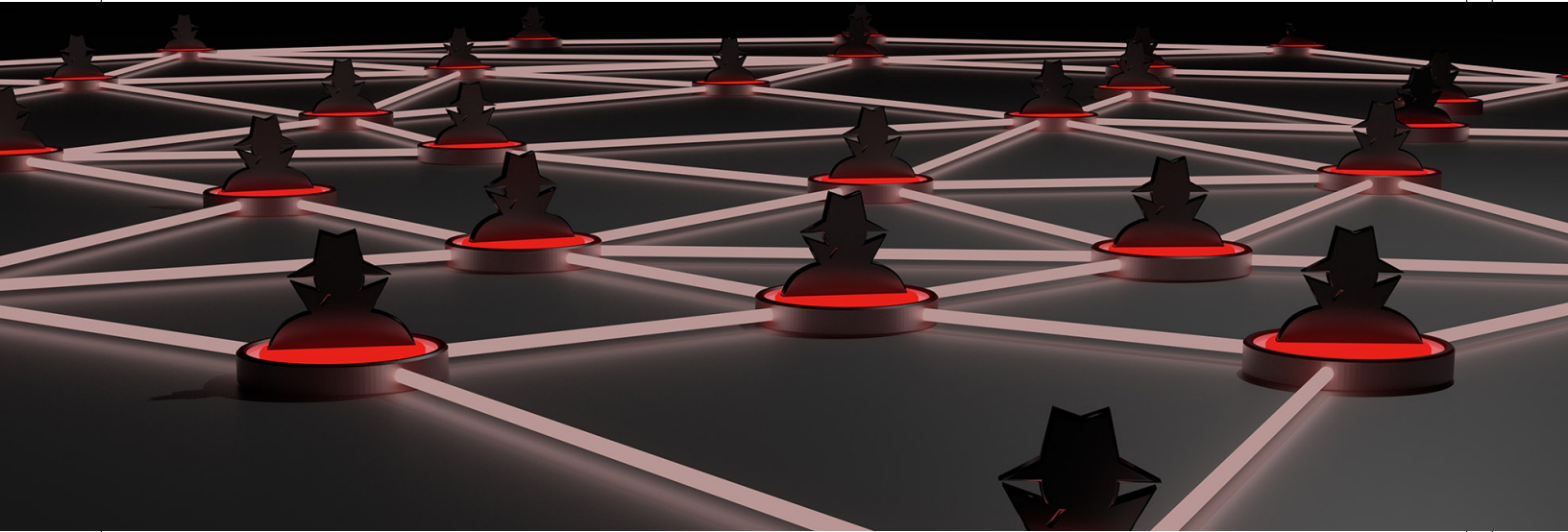


Francesco Ciccarelli - N86003285



Giulia Caputo - N86003429

1 Presentazione



1.1 Descrizione della traccia

Si richiede la realizzazione di una **BotNET**¹ per il recupero di quante più informazioni possibili sul dispositivo in cui una delle componenti della BotNET (a scelta dello studente) venga eseguito.

1.1.1 Tecnologie e linguaggi richiesti

Si richiede un applicativo scritto in **Python**² che utilizzi come strumento di comunicazione le **socket**³

1.2 Implementazione del sistema

Il progetto si concretizza in 2 componenti ben definite:

- Un **Bot Master** per la gestione dei dati ricevuti dal **bot slave** al quale inpartisce comandi sfruttando una connessione tramite socket asincrona;
- Il **Bot slave**, che ha il compito di ricavare quante più informazioni possibili sullo stato della macchina sul quale viene eseguito⁴.

¹ Per BotNET si intende una rete composta da dispositivi infettati da malware, detti bot o zombie, che agiscono tutti sotto lo stesso controllo di un unico dispositivo - detto botmaster - aumentando esponenzialmente le capacità dell'attaccante.

² Python è un linguaggio di programmazione di alto livello, orientato a oggetti, adatto, tra gli altri usi, a sviluppare applicazioni distribuite, scripting, computazione numerica e system testing.

³ Astrazione software progettata per utilizzare delle API standard e condivise per la trasmissione e la ricezione di dati attraverso una rete oppure come meccanismo di IPC..

⁴ Della quale non abbiamo nessun controllo diretto.

1.3 Guida al Bot Master

1.3.1 Primo avvio

Durante la fase di avvio il programma effettua le seguenti operazioni:

- ▶ Controlla che **host** e **porta**⁵ siano disponibili per la successiva creazione della socket;
- ▶ Esegue una connessione al dbms utilizzato per il salvataggio delle informazioni e inizializza la **tabella** utilizzata per lo scopo (se non precedentemente presente);
- ▶ Inizializza la socket in attesa di nuove connessioni dal client a cui impartirà comandi da eseguire;

Nel momento in cui viene effettuata una nuova connessione, il server invia la richiesta effettuata dall'utente al client e in base a questa automaticamente:

- ▶ Salva l'informazione sul database (che viene mostrata all'utente attraverso lo standard output);
- ▶ Nel caso in cui si trattasse di un file (identificato da un campo **Header** a inizio richiesta), lo salva automaticamente, per poter essere fruibile successivamente.

1.3.2 Memorizzazione dei dati

Il sistema permette inoltre utilizza un DBMS⁶ per il salvataggio dei dati ricavati dal *bot slave* durante la sua esecuzione.

1.3.3 Modalità di esecuzione

È possibile invocare il bot master con una serie di flag aggiuntive, che permettono di:

- ▶ Definire un host e porta su cui esporre il servizio (rispettivamente **--host** e **--port**)
 - ◊ Ricordiamo che di default il bot master utilizzerà rispettivamente **127.0.0.1** e la porta **9090**;
- ▶ Definire una cartella custom che verrà utilizzata per il salvataggio dei dati;⁷
- ▶ Gestire una connessione multi-client (invocando il bot master con **--supervisor=dispatcher**)
 - ◊ In questo modo il bot master fa da tramite per la connessione 1:1 tra **clientX** e **master**

1.4 Guida al Bot Slave

1.4.1 Primo avvio

Durante la fase di avvio il programma effettua le seguenti operazioni:

⁵Ricordiamo che in fase di lancio del programma è possibile definirne altri e sostituirli a quelli di default.

⁶Fa affidamento al DBMS (Database Management System) PostgreSQL.

⁷Dati che verranno recuperati dalla macchina in cui è eseguito il client (su specifica richiesta).

- ▶ Controlla che **host** e **porta**⁸ siano disponibili per la successiva creazione della socket;
- ▶ Esegue un test sull'effettivo stato di attività del server
 - ◊ In caso di esito negativo attende e ritenta;
 - ◊ In caso di esito positivo invece esegue le istruzioni impartite dal Master.

In base alle flag specificate è possibile:

- ▶ Definire un nuovo host e porta a cui connettersi (rispettivamente **--host** e **--port**);
- ▶ Ricercare automaticamente il bot master (flag **--finder**);
- ▶ Richiedere di essere accoppiato ad un **bot master** automaticamente (flag **-r**).

1.5 Analisi della struttura del progetto

La struttura del progetto è così strutturata:

- | | |
|---|---|
| ▶ Un file main.py , utilizzato per eseguire il tutto; | ▶ Un file main.py , utilizzato per eseguire il tutto; |
| ▶ Una cartella utilities , contenente: <ul style="list-style-type: none">◊ async_socket_server.py → Funzioni per la gestione della connessione socket;◊ bot_master_utility.py → Funzioni di supporto al server;◊ database_handler.py → Funzioni di supporto per la gestione del DBMS. | ▶ Una cartella utilities , contenente: <ul style="list-style-type: none">◊ async_socket_client.py → Funzioni per la gestione della connessione socket;◊ bot_master_utility.py → Funzioni di supporto al client; |

2 Codice sorgente sviluppato

Il codice sorgente prodotto durante lo sviluppo di *StealBot*[®] è disponibile sulla piattaforma [GitHub](#), che ne ha permesso anche il versionamento.

Di seguito riportiamo un link per il [download](#)⁹

È possibile leggere una attenta analisi del codice al seguente [link](#)

TODO: Aggiungere l'url del sito prima di consegnare

⁸ Così come per il Master anche in questo caso è possibile definirne altri e sostituirli a quelli di default.

⁹ Potrebbe non essere accessibile a tutti (il repository è per privacy privato).

3 Risultati ottenuti

Durante le prove di testing¹⁰, abbiamo recuperato le seguenti informazioni:¹¹

3.1 Informazioni sulla macchina (OS: Linux-5.15.0-52-generic-x86_64-with-glibc2.35)

CPU

Brand	CPU Count	CPU Count logical	Frequenza Minima	Frequenza Massima
Intel(R) Core(TM) i7-8569U	4	4	2.80GHz	4.70GHz

RAM

Memoria utilizzata	Memoria Totale
790.86MB	3.83GB

DISCO

Device	Mountpoint	Tipo di partizione
/dev/sda2	/boot/efi	vfat
/dev/sda3	/	ext4

STATO DEL DISCO

Letture	Scritture
691.32MB	31.47MB

¹⁰Effettuate il 16 novembre e il 13 dicembre.

¹¹Le due macchine in questione sono molto simili tra loro (ricordiamo che sono entrambe macchine virtuali eseguite su Hypervisor [VirtualBox](#)), tanto da supporre che siano una il clone dell'altra.

NETWORKING¹²

Interfaccia	IP	NetMask	Broadcast
loop	127.0.0.1	255.0.0.0	Nessuna
loop	::1	ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff	Nessuna
loop	00:00:00:00:00:00	Nessuna	Nessuna
enp0s3	10.0.2.15	255.255.255.0	10.0.2.255
enp0s3	fe80::9406:ff6d:57df:81b6%enp0s3	ffff:ffff:ffff:ffff::	Nessuna
enp0s3	08:00:27:63:f0:81	Nessuna	ff:ff:ff:ff:ff:ff
enp0s8	192.168.1.188	255.255.255.0	192.168.1.255
enp0s8	192.168.1.224	255.255.255.0	192.168.1.255
enp0s8	fdac:c077:5c58:0:7913:ba74:dcde:5157	ffff:ffff:ffff:ffff::	Nessuna
enp0s8	fdac:c077:5c58:0:3595:1b00:316b:ad04	ffff:ffff:ffff:ffff::	Nessuna
enp0s8	fe80::b224:2d33:82d5:b5de%enp0s8	ffff:ffff:ffff:ffff::	Nessuna
enp0s8	fdac:c077:5c58:0:7913:ba74:dcde:5157	ffff:ffff:ffff:ffff::	Nessuna
enp0s8	fdac:c077:5c58:0:3595:1b00:316b:ad04	ffff:ffff:ffff:ffff::	Nessuna
enp0s8	fe80::b224:2d33:82d5:b5de%enp0s8	ffff:ffff:ffff:ffff::	Nessuna
enp0s8	08:00:27:e5:6a:b8	Nessuna	ff:ff:ff:ff:ff:ff

UTENTI ATTIVI

Nome utente	Attivo da
alessio	2022-11-16 09:04:16

3.2 File recuperati durante l'esecuzione del bot slave

- ▶ **.bash_history** → Contiene tutta la cronologia dei comandi dati dall'utente;
- ▶ **.bash_logout** → Contiene le operazioni da eseguire durante il logout dell'utente;
- ▶ **.bashrc** → File di configurazione della shell bash;
- ▶ **bookmarks** → Contiene i segnalibri definiti dall'utente
- ▶ **meta-release-lts** e **ubuntu.22.04** → Contengono le informazioni aggiuntive della macchina su cui gira il bot slave;

¹²Onde evitare inutili ripetizioni abbiamo preferito compattare le informazioni riguardanti il networking di entrambe le macchine virtuali. Con l'IP **192.168.1.188** stiamo indicando la macchina della quale avevamo già informazioni; con **192.168.1.224** quella sulla quale non erano stati ancora effettuati test di alcun tipo.

- ▶ `.pam_environment` → Contiene variabili per la lingua;
- ▶ `.passwords` → File contenente eCambiata
- ▶ `.profile` → Impostazioni aggiuntive per la shell `bash`
- ▶ `.python_history` → Contiene la cronologia dei comandi effettuati dall'interprete interattivo `python`;
- ▶ `user-dirs.dirs` → Contiene le informazioni sulle variabili delle directory della home dell'utente.

3.3 Report dei dati recuperati

Di seguito vengono riportati alcuni estratti dei dati recuperati dall'applicativo [wireshark](#)¹³, ottenuti durante la prima prova:

	No 1 (Time 0.000000000)
Source	WistronN_73:f9:a6
Destination	Broadcast
Protocol	ARP
Lenght	42
Info	Who has 192.168.11.1? Tell 192.168.1.24
	No 13 (Time 3.176898029)
Source	192.168.1.72
Destination	192.168.1.188
Protocol	TCP
Lenght	120
Info	54654 → 9000 [PSH, ACK] Seq=1 Ack=8 Win=502 TSVal=1423094325 TSecr=3808546355
Data (54 bytes)	<OS-type>Linux-5.15.0-52-generic-x86_64-with-glibc2.35

¹³ Nel quale abbiamo impostato il filtro `ip.addr eq 192.168.1.72 and (tcp.port eq 9000 || tcp.port eq 9001)` (ottenendo in questo modo solo i pacchetti di nostro interesse).

	No 179 (Time 14.033129003)
Source	192.168.1.72
Destination	192.168.1.224
Protocol	TCP
Lenght	157
Info	54658 → 9001 [PSH, ACK] Seq=268 Ack=39 Win=502 TSVal=1423105176 TSecr=3808557231
Data (54 bytes)	<Partition-disk-info><Partition-Device>/dev/sda3<Partition-MountPoint>/<Partition-FSType>ext4

4 Ringraziamenti

Ringraziamo il professore [Alessio Botta](#) per lo splendido corso, che ci ha permesso di comprendere a pieno tecnologie di cui il mondo fa largo uso.