Relazione Home work Sicurezza

Realizzazione di una Botnet

Pizzari Federico 1936451

A.A 2022/2023

Introduzione e scelte progettuali

Lo studente è stato incaricato di realizzare una botnet comprendente di “Command & Control” (in breve C&C) e di Bot (o Slave) da installare sulle macchine infette.

Lo sviluppo è iniziato con la scelta del linguaggio di programmazione da utilizzare, è per questo progetto la scelta è ricaduta sul linguaggio Golang. La scelta è stata supportata da diverse motivazioni, tra cui:

- Linguaggio realizzato con il networking in mente, così facendo abbiamo a disposizione molte funzioni efficienti e native per gestire le comunicazioni tra i vari componenti della rete
- Mette a disposizione le goroutine, una tipologia di thread specifica per questo linguaggio, molto veloci, potenti e facilissimi da usare
- Le prestazioni del GO sono ottime, infatti nel caso di progetti del genere si avvicina se non supera le prestazioni di un linguaggio come il C++
- Il linguaggio, a differenza di altri linguaggi come il python, ci fornisce un compilatore con la quale realizzare i file eseguibili, potendo così avere la quasi certezza di poter eseguire il codice realizzato su tutte le macchine (o perlomeno quelle compatibili con quel compilato)

Per l’invio delle operazioni da effettuare e le comunicazioni tra il C&C e i bot, è stato deciso di utilizzare il protocollo HTTP per cercare di essere meno individuabili dai sistemi di rilevazione delle intrusioni (o IDS). Fa eccezione la connessione iniziale, che viene effettuata connettendosi ad un socket appositamente creato sulla porta 8080 dalla C&C.

Nota: Per testare gli eseguibili è necessario eseguire delle macchine virtuali che possano comunicare tra loro su rete NAT e bisogna assegnare alla VM che contiene la C&C l’indirizzo privato 192.168.178.64. I bot possono invece avere qualsiasi tipo di indirizzo.

Centro di Controllo

Il centro di controllo è la parte della rete che gestisce, controlla e mostra lo stato della botnet. Una botnet non deve però essere dipendente dal C&C, infatti nel caso il C&C dovesse andare offline, i bot devono comunque poter terminare il lavoro a loro assegnato e eventualmente restare in attesa di istruzioni.

L’avvio della C&C è caratterizzato immediatamente dalla creazione di un semaforo necessario per gestire alcuni casi di concorrenza che si verificano nel codice (come nel caso dell’aggiunta e la rimozione dei bot attivi) e successivamente dall’apertura dell'ascolto sulla porta 8080 per permettere ai bot di connettersi.
Subito dopo leggiamo da un file JSON (ConnectedBots.json), compilato dal C&C stesso durante l’ultima sua esecuzione, lo stato della botnet, caricandoci in questo modo tutti i bot registrati e in attesa di istruzioni.
Alcuni di questi bot potrebbero essere andati offline nel mentre, ma non ci interessa al momento. L’obiettivo è quello di rendere i bot presenti sugli host i più leggeri possibile; pertanto, è la C&C che deve scoprire in fase di lancio delle istruzioni se i bot sono ancora raggiungibili oppure se sono andati offline (e nel caso siano andati offline, aggiornare il JSON). Questa struttura non limita la botnet per due motivi:

- Non ci interessa sapere esattamente il numero di nodi online ad ogni istante, anche considerando la volatilità con la quale potrebbero andare online/offline in breve tempo. Inoltre, anche se fosse necessario, esiste un comando apposito che fa un polling dello stato della rete in tempo reale.
- Una struttura del genere è resistente al crash della C&C o di uno dei nodi e impedisce la creazione di nodi "zombie" che dichiarano di esistere ma in realtà non sono raggiungibili perché dietro rete NAT. Inoltre, limita le comunicazioni da/verso la C&C, rendendo il tutto leggermente meno individuabile.

Una volta fatto questo, avviamo un thread che itererà all'infinito in attesa della richiesta di connessione dei bot, che ci comunicherà il loro IP e la loro porta sulla quale sono in ascolto.
Appena abbiamo effettuato la parte utente della C&C, mostrando le istruzioni lanciabili.
Una volta selezionata un'operazione, facciamo scegliere all'utente l'insieme di bot che si vogliono utilizzare e infine si lancia l'operazione.
Le istruzioni sono le seguenti:

- **Invio di un batch di e-mail**
  L'invio del batch di e-mail viene effettuato controllando un file JSON che funge da database".
  Al suo interno sono presenti due oggetti:
  - L'oggetto users che contiene un array di e-mail
  - L'oggetto mailingListsBodies che contiene le mail effettive da inviare e a chi inviarle

  Una volta elaborato il file, la C&C itera sul numero di mail da inviare, e ad ogni iterazione sceglie il bot incaricato dell'invio e invia una richiesta HTTP PUT verso il bot, allegando le istruzioni dell'attacco nel body della richiesta sottoforma di JSON.
  Nel caso una mail non venga inviata perché il bot selezionato nel mentre è andato offline, eliminiamo dalla lista il bot e ritentiamo lo stesso invio con un bot diverso.
  L'attacco termina (con successo) quando vengono inviate tutte le mail oppure (con fallimento) quando tutti i bot sono divenuti irraggiungibili.

- **Inizio di un attacco DDoS verso uno specifico target**
  L'inizio dell'attacco DDoS viene effettuato prima di tutto chiedendo all'utente alcune informazioni comprendenti:
  - Il target da attaccare (Indirizzo HTTP della risorsa)
  - La porta dove è situato il target
  - Il numero di richieste da voler inviare (se il numero fornito è negativo, vengono effettuate richieste illimitate).

  Le informazioni vengono poi fornite ai bot utilizzando una richiesta HTTP PUT, allegandole nel body della richiesta sottoforma di JSON.
  Come sempre, se rileviamo un bot offline, lo eliminiamo dalla lista dei bot attivi.

- **Interruzione di un attacco DDoS**
  Questo comando è reso necessario dal fatto che l'attacco DDoS è l'unico che può avere una durata illimitata o non ben specificata in fase di creazione dell'attacco.
  L'interruzione dell'attacco viene effettuata realizzando una richiesta HTTP DELETE allo stesso entrypoint usato all'inizio dell'attacco

- **Ricezione di informazioni sul sistema host infettato**
  Questo comando utilizza una chiamata GET per ricevere in formato JSON informazioni sull'host. Le informazioni comprendono: Hostname, Piattaforma, CPU, Dimensione totale della RAM e Dimensione totale del disco

- **Visualizzare lo stato della bot-net**
  Questo comando funziona in modo simile al precedente, solamente che invece che ricevere le informazioni dello host, riceve le informazioni sullo stato attuale di ogni singolo bot, elencando quindi le eventuali operazioni attualmente in corso e la lista degli errori riscontrati durante l'esecuzione delle operazioni precedenti

- **Fare il polling dello stato della bot-net**
  Sostanzialmente come la chiamata precedente, ma semplicemente non mostra nessun output, solamente il numero di bot al momento online

Una volta finita un'operazione, l'utente ha una breve descrizione dell'esito dell'operazione e infine il programma itera all'inizio, dove potrà selezionare nuovamente un'operazione da effettuare

Bot

Il bot è l'elemento che svolge fisicamente le operazioni malevole impartite dalla C&C. Deve essere un componente flessibile, veloce e leggero in modo da potersi adattare e essere il meno individuabile possibile.
Nota: Tutti i messaggi mostrati a terminale dai bot sono solo a scopo didattico e non devono essere intesi come parte integrante del progetto. Inoltre, l'invio delle e-mail negli eseguibili consegnati è non funzionante, considerando che per poter inviare una mail abbiamo necessità di un account verso un server SMTP che non è stato incluso nell'eseguibile finale

All'avvio del bot come prima cosa vengono impostati immediatamente sia il semaforo che i vari handler necessari per definire gli endpoint del server HTTP. Viene poi chiamata una funzione che tenta di mettersi in ascolto sulla porta 80 o, nel caso di fallimento, una qualsiasi porta superiore fino a successo: sarà la porta sulla quale sarà presente il nostro server HTTP. Infine, tentiamo la connessione e la registrazione con il C&C.
Fino a quando la connessione non avrà esito positivo il bot resterà in attesa e non effettuerà alcun tipo di operazione, ne inizializzerà il server HTTP.
Solo una volta effettuata la registrazione sul C&C, chiamerà la funzione che inizializzerà il nostro server HTTP: adesso il bot è pronto per ricevere richieste.

Durante il tempo in idle, il bot non esegue assolutamente nessuna operazione, proprio come richiesto dalle specifiche del progetto.

Gli endpoint registrati sul bot per effettuare le varie operazioni sono i seguenti:

- **/e-mail PUT**
  Prende dal body della richiesta un JSON che rappresenta una mail, lo spacchetta e infine lancia una goroutine (thread) per effettuare l'invio vero e proprio utilizzando la libreria gomail e un server SMTP fornito.
  Restituisce 200 se l'invio è stato inoltrato correttamente o 400 se il JSON ricevuto è incorrettamente formattato
- **/site PUT**
  Prende dal body della richiesta un JSON contenente l'url della risorsa da attaccare, la porta da attaccare e il numero di richieste da effettuare. Si crea infine l'url completo da attaccare e lo passa ad una goroutine apposita che inizierà il ciclo con le richieste HTTP.
  Restituisce 200 se l'attacco è iniziato con successo, 400 se il JSON ricevuto è incorrettamente formattato oppure 503 se c'è già un attacco in corso sul bot

- **/site DELETE**
  Endpoint necessario per terminare un attacco DDoS con durata non specificata.
  Può anche terminare prematuramente un attacco DDoS con durata.
  Restituisce solo 200
- **/system GET**
  Restituisce le informazioni del sistema come illustrato nella parte di relazione relativa alla C&C
  Restituisce 200 se le informazioni sono state inviate correttamente, 500 altrimenti
- **/status GET**
  Restituisce le informazioni del sistema contenute nella variabile status
  Restituisce 200 se le informazioni sono state inviate correttamente, 500 altrimenti

Test della C&C

Il test della C&C è stato effettuato utilizzando delle VM istanziate con Oracle VirtualBox e legate tra loro utilizzando una rete NAT in cui la VM con la C&C aveva l'IP privato 192.168.178.64, mentre le altre VM ottenevano un indirizzo IP casuale dal server DHCP
Il sistema operativo scelto per il test è stata la distribuzione Bodhi Linux 6.0.0, basata su Ubuntu 20.04.
La scelta è ricaduta su questa distribuzione per due motivi:

- Bodhi Linux è una distribuzione molto leggera e con pochissime dipendenze preinstallate, è il campo di prova perfetto per testare il comportamento del nostro eseguibile.
- Inoltre, questo ci consente di poter creare delle VM con pochissime risorse (1GB di RAM e 2 core virtuali), in modo da testare come si comporta il nostro codice in situazioni di scarsità di potenza di calcolo.

Sono stati effettuati diversi test durante tutta il ciclo di sviluppo del progetto e non sono mai state rilevate criticità dovute all'ambiente di sviluppo.