Proiect ASO 2024
Management-ul unui *botnet*

1 Introducere

De multe ori, în securitate trebuie să înțelegem cum gândesc și cum acționează atacatorii, iar pentru asta trebuie să "ne punem pălăria neagră (eng. *black hat*)". În acest proiect vom învăța cum funcționează un bot, adică o categorie de malware ce păstrează legătura cu server-ul atacatorului și primește comenzi de la acesta.

Terminologie:

- **bot**[1] este un tip de malware (program care face rău în mod intenționat), având particularitatea că păstrează legătura cu *centrul de comandă și control* (un server controlat de atacator) și execută comenzile primite de la acesta;
- **botnet**[2] este o rețea formată din mai mulți boți (de obicei de ordinul miilor, sau chiar milioanelor), pe care un atacator o poate controla;
- **centru de comandă și control (C&C)**[3] este server-ul controlat de atacator, la care boții se conectează pentru a primi comenzi; în cazul botnet-uril mai sofisiticate, acest server este bine ascuns, fie prin proxy-uri, fie prin algoritmi de generare a domeniilor, dar în acest caz vom presupune că boții cunosc IP-ul C&C-ului;
- **Internet of Things (IoT)**[4] se referă la dispozitive inteligente, interconectate și de obicei conectate la internet;
- **Distributed-Denial-of-Service (DDoS)**[5] reprezintă un tip de atac prin care mai multe dispozitive comunică cu o victimă prin rețea (realizând conexiuni sau trimițând pachete), epuizând resursele computaționale ale acesteia;

Mirai[6] este o familie de malware de tip bot, orientată către dispozitivele din Internet of Things. Acest malware a apărut în 2016 și a profitat de vulnerabilitățile dispozitivelor inteligente conectate la internet, lansând atacuri devastatoare de tip DDoS.

[1] `https://en.wikipedia.org/wiki/Zombie_(computing)`
[2] `https://en.wikipedia.org/wiki/Botnet`
[3] `https://en.wikipedia.org/wiki/Botnet#Command_and_control`
[4] https://en.wikipedia.org/wiki/Internet_of_things
[5] `https://en.wikipedia.org/wiki/Denial-of-service_attack`
[6] `https://en.wikipedia.org/wiki/Mirai_(malware)`

Codul sursă al malware-ului este public pe GitHub: `https://github.com/jgamblin/Mirai-Source-Code`. Acest cod sursă este format din codul botului (scris în C) și codul centrului de comandă și control (scris în Go). Codul sursă al botului se poate compila pentru majoritatea platformelor existente pe dispozitive IoT (arm, mips, ...), dar și pe Intel x86. Instrucțiunile originale pentru deployment se găsesc în același repo[7] (*disclaimer*: limbajul folosit nu este unul academic).

Recomandare: pe durata lucrului la proiect există posibilitatea să rulați malware-ul din greșeală. Se recomndă realizarea periodică de snapshot-uri ale mașinii virtuale de lucru.

2 Cerințe

2.1 Etapa 1 – compilare și instalare (3.5p)

- Analizați codul sursă din repo-ul Mirai[8] identificând componentele acestuia; realizați o scurtă documentație în care să explicați ce rol are fiecare componentă (la nivel înalt, nu trebuie să detaliați fiecare funcționalitate).
- Realizați o mașină virtuală și instalați în ea Golang și alte pachete necesare pentru a putea rula server-ul C&C. Baza de date folosită de server poate rula pe același sistem.
- Compilați binarele malware-ului. E nevoie să modificați fișierul `mirai/bot/table.c` conform instrucțiunilor, pentru a se conecta la server-ul vostru.
- Compilați și instalați server-ul de comandă și control (`cnc`). Rulați-l doar pe acesta, fără a rula și malware-ul.

2.2 Etapa 2 – dockerizare (3.5p)

În această etapă veți face deploy malware-ului Mirai într-o infrastructură de imagini de Docker, formată din 3 imagini / containere (se recomandă *docker-compose* pentru orchestrarea acestora):

- *cnc* – în această imagine veți face deploy server-ului C&C;
- *patient-zero* – în această imagine va rula "pacientul zero", adică primul sistem infectat; imaginea nu va conține malware-ul propriu-zis, ci va aștepta ca *cnc* să pornească, apoi îl va descărca și rula de acolo;
- *victim* – această imagine va juca rolul de victimă; va conține un server de Linux, cu un server de Telnet a cărui credențiale sunt pe lista celor încercate de Mirai;

2.3 Etapa 3 – detecție (3p)

În această etapă veți captura traficul existent în rețeaua de tip docker și veți realiza un script (Python sau Bash) care joacă rolul unui IDS[9] și detectează atacurile de tip bruteforce.

[7] `https://github.com/jgamblin/Mirai-Source-Code/blob/master/ForumPost.md`
[8] `https://github.com/jgamblin/Mirai-Source-Code`
[9] `https://en.wikipedia.org/wiki/Intrusion_detection_system`

3 Livrabile

La fiecare etapă se va demonstra funcționalitatea cerințelor.

Tot pentru fiecare etapă se va scrie o documentație, în care se va descrie soluția aleasă. Documentația trebuie scrisă cât mai explicit, astfel încât cineva care o citește să poată reimplementa soluția.

Acolo unde se cer script-uri, acestea se vor urca pe Moodle, împreună cu documentația.

3.1 Termene limită

- Etapa 1: săptămâna 5
- Etapa 2: săptămâna 9
- Etapa 3: săptămâna 13