Fiche descriptive de projet HUB

cuddly-couscous

Contexte et but du projet

Pourquoi ce projet?

Étant intéréssé par la sécurité réseau, en particulier analyse du trafic sur le réseau, je souhaite tester mes conaissance en cyber-défence.

Et donc?

L'objectif du projet serait de créer un logiciel qui surveille le réseau cible (sniffer) et de détecter les attaques potentielles. Cela permettrait de détecter les attaques sous forme d'alertes envoyées par le logiciel.

Le logiciel peut analyser les paquets en temps réel ou analyser un fichier de logs déjà existant.

Afin de mieux comprendre les attaques, je reproduirai des logs d'attaques pour différents types d'attaques sur le réseau cible afin de tester le logiciel et ainsi garantir son efficacité.

L'idée est de développer cela de manière modulaire afin de pouvoir le mettre à jour et ajouter d'autres types d'attaques à l'avenir.

Porteur du projet

Guilhem Vinet (guilhem.vinet@epitech.eu) - Lead

Environnement technique / technologique

- **C++ (sotfware)**
- **Criterion (unitest)**
- **Bash (commande réseau, installation etc)**

Objectif fonctionnel

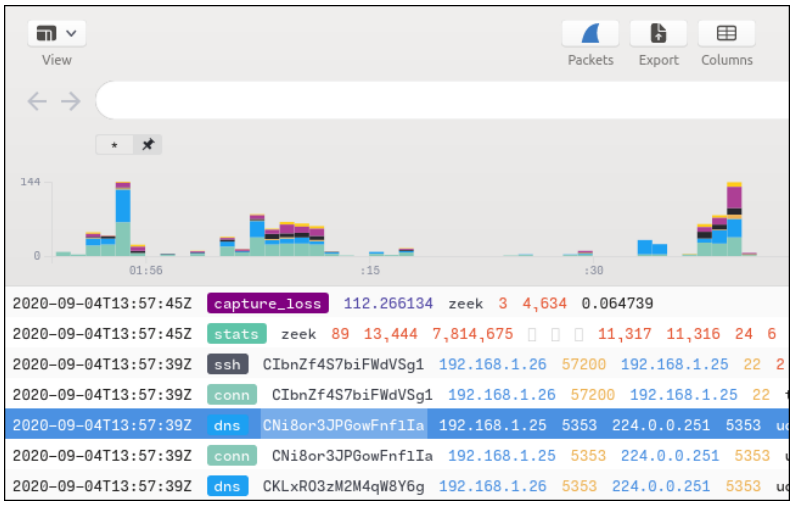
Ojectif dévelloper un sniffer pour récuperer le trafic du réseau et des detecter chacune des attaques. Afficher le trafic sur le réseau et d'afficher des notifications qui avertise la detection d'attaque

Les attaques reproduits sont les suivantes :

- deauthentication frames
- decoy scanning attempts
- SMURF attacks
- LAND attacks
- IP time to live attacks
- TCP Handshake Abnormalities
- TCP Connection Hijacking

Aperçu du projet

Voici un apperçu de l'interface graphique potentiel (visuel de Brim)



Estimation du projet

J'estimes à 40 **XP** le projet (entre 3 et 4h par jour du lundi au (samedi et dimanche peuvent comprendre plus d'heure quand semaine). Le nombre d'heure est estimé est 120h (3 à 4h par jours) et qu'il me prendra 30 jours.

Pourquoi ?

J'estime que le dévellopement de sniffer de packet va prendre 20**h** (comprend le temps de code et des recherches (5 jours):

→ avoir les infos essentiel ip source/destination, port source/destionation, le protocol utilisé, nombre packets/bytes échangé

→ durée de l'échange de packet/durée de connection d'une connection

→ choisir l'adapteur du réseaux du serveur

→ filtrer le trafic qu'on veux regarder

→ détecter les différentes connections/appareils sur le réseau

→ gestion des fichiers logs

Ensuite viens la parties de détections de différentes attaques temps estimé (16 jours):

- deauthentication frames (8h)

- decoy scanning attempts (8h)
- SMURF attacks (8h)
- ARP spoofing attacks (8h)
- LAND attacks (8h)
- IP time to live attacks (8h)
- TCP Handshake Abnormalities (8h)
- TCP Connection Hijacking (8h)

Automatisation de alerte et rendu dans le terminal en ncurse : (4 jours)

- interface graphique (8h)

- Implémentation des menus (5h)

- Implémentation des alertes (5h)

Gestion des risques : (5 jours)

- Les 5 jours restants seront utilisés pour tester et corriger les éventuels bugs du logiciel.

Organisation et temporalité

Outils : Wireshark, projet github sniffnet, HackTheBox. Notion

Temps estimé : 30 jours (du Lundi au dimanche) soit 120h (4h par jour) + temps disponible si les projets Epitech sont achevés en parallèle

Organisation : Tous est instentié dans un notion, gestion de tache journalière, documentation, liens des recherches, rapport de bugs/problème rencontré.

Etape du projet

- **ALPHA** : Intro -> 5 Jour

    - Dev du sniffer (2 jours)
    - mise en forme du fichier de logs (1 jours)
    - Parseur du fichier de logs (2 jours)

- **BETA 1.0** : Détecter les attaques -> 17 Jour

- deauthentication frames (2 jour)

- decoy scanning attempts (2 jour)
- SMURF attacks (2 jour)
- ARP spoofing attacks (2 jour)
- LAND attacks (2 jour)
- IP time to live attacks (2 jour)
- TCP Handshake Abnormalities (2 jour)
- TCP Connection Hijacking (2 jour)
- **Unitest** (1 jour)

- **BETA 2.0**: Interface graphique -> 3 Jour

- interface graphique (2 jours)

- Implémentation des alertes (1 jours)

- REALISE 1.0 → gestion de test, correction des bugs et rédiger une documentation→ 5 jours :

  Les 5 jours restants seront utilisés pour tester et corriger les éventuels bugs du logiciel.

- Batterie de test + correction de bugs (4 jours)
- Rédaction de la documentation (1 jours)