Projet CASSIOPEE® :
« Database Firewall »
Compte rendu d'avancement n°01

Liste de diffusion : Grégory Blanc, Grégoire Menguy, Baptiste Polvé

Ordre du jour

- **Planification du projet (achevé)**
- **État de l'art sur les parsers (achevé)**
- **État de l'art sur la détection d'intrusion (commencé)**
- **État de l'art sur les Database Firewall (commencé)**

Planification

Pour la planification, nous avons décidé ensemble, à partir des objectifs et sous-objectifs vus en réunion du 23/02/2017, de définir des sous-tâches que nous nous sommes répartis. Ensuite, nous avons pondéré le temps de travail de chacune des tâches sur une base globale de 370h. (à diviser en 2) Nous avons également réparti les 18h par semaine de travail en (5 * 3h30 par jour de travail) et supprimé tous les jours fériés/vacances/challenge comme jugé par Mme Kohlenberg pour tenter d'être au plus près de la réalité. Ainsi sur le logiciel lorsque l'on voit 1d cela représente 3h30.
C'est pourquoi, nous avons utilisé le coût du travail pondéré à 1 pour avoir la valeur visuellement en terme de temps de travail.
Nous avons utilisé « planner » car il nous est apparu simple d'utilisation pour inclure les membres du groupe, les horaires et qu'il rend un affichage interactif via une page html.
Ainsi, vous pouvez le retrouver en pièce jointe et sur Git dans gestion_de_projet/planning_cassiopee_allege.html . Allégé car ce planning n'inclut pas les réunions avec le client (évalué à 1h de travail supplémentaire, tout comme ce type de compte rendu).

État de l'art sur les parsers

En amont, Grégoire a étudié les parsers, voici donc un résumé de ses notes que l'on peut retrovuer plus complet dans le dossier Documentation.

- Pouquoi les parser ?

Les parsers permettent de vérifier qu'une certaine expression suit une grammaire définit au préalable. Cela nous sera très utile pour analyser les requêtes SQL et ne laisser passer que celles qui suivent la grammaire imposé par MySQL. Ainsi nous pourrons déjà droper certaines requêtes rien qu'en les lisant.

- Résultat des recherches

Il existe différents type de grammaires. Nous nous sommes donc intéressés à ces différents types et en avons conclut que nous devrions utiliser un parser pour context-free grammars.

Ils existent différents algorithmes pour parser ce type de langage dont le LL-parsing, le LR(1)-parsing et LALR(1)-parsing. Cependant aucun de ces algorithmes ne permettent de parser tous les context-free language. Il faudra donc s'assurer que la grammaire en entré n'est pas ambiguë et ne possède pas de conflit interne (shift/reduce conflicts et/ou shift/shift conflicts).

- Outils utilisés

Il est compliquer de programmer à la main un parser. C'est pourquoi, nous utiliserons Lex & Yacc pour générer le parser. Yacc gère les LALR(1) grammars et génère en sortie du code C. Il est également possible de l'utiliser pour créer un parser en C++.

Lex n'est pas open source mais certaines de ses versions le sont, notamment flex que nous utiliserons. Yacc est qu'en à lui un outil open source.

État de l’art sur la détection d’intrusion

Grégoire a fait des recherches sur les différents types d'IDS pour mieux comprendre les méthodes de détections d'attaques et pouvoir les incorporer dans notre outil. Les recherchers peuvent être retrouvées dans le dossier Documentation.

Ce qui a été fait :

- Recherche sur les différents type d'IDS (HIDS & NIDS)
- Recherche sur les différentes implémentations (stream-based, connexion-based)
- Recherche sur les différents types de signatures
- Recherche sur les faux positifs et comment les limiter
- Recherche d'IDS déjà existant (SURICATA, POSEIDON, SNORT)

Ce qu'il reste à faire :

- Approfondir les implémentations des signatures
- Comprendre les méthodes d'évasion

État de l’art sur les Databases Firewall

Baptiste a fait des recherches sur les différents Databases Firewall pour comprendre l’implémentation qui semblait la plus pertinente et ce qui pouvait être intéressant à ajouter pour justifier notre outil.

Ce qui a été fait :

- Recherche sur les Databases Firewall
- Recherche des Databases Firewall existants(non inclut dans la documentation)
- Choix de mettre en avant une solution propriétaire (Oracle) et une solution libre (GreenSQL/ HexaTier)
    - Ce sont les solutions les plus utilisées et celles qui semblent les plus performantes à ce jour.
- Présentation générale et technique du Database Firewall d’Oracle
- Présentation générale et technique générale de GreenSQL

Ce qu'il reste à faire :

- Approfondir le fonctionnement technique de GreenSQL
- Notifier certains Database Firewall autre

Ce CR sera considéré comme validé mardi 07/03/2017 en l’absence de remarque.