Projet Cassiopée – Groupe 35

Développement d'un DATABASE FIREWALL (DBF)

- ## Ce projet dans l’actualité

Les injections SQL font parties des attaques les plus utilisées et les plus dévastatrices à l'heure actuelle sur le web. On les retrouve notamment dans le top 10 des attaques les plus utilisées selon l'OWASP. Cela est dû a deux choses :

1. La plupart des développeurs ne testent pas suffisamment leur code et/ou ne suivent pas correctement les bonnes pratiques de sécurité liées au filtrage d'entrée, aux *prepared statements*...

2. Les outils les plus utilisés tels que les WAF(Web Application Firewall) ne sont pas suffisamment performants puisqu'ils doivent gérer une infinité de requêtes sous différentes formes, parfois mêmes offusquées.

Il existe cependant un outil plus performant que le WAF : le **Database Firewall**.
Celui-ci ne se place pas entre le serveur de l’application et le client, contrairement au WAF, mais entre le serveur de l’application et la/les base(s) de données.

Là ou le WAF échouait car les possibilités de messages envoyés via le formulaire du site sont infinies (notamment à cause des tautologies et de l'offuscation de code), le DBF lui, reçoit des requêtes SQL complètes, déjà traduites, qui sont plus simples à analyser. Finalement, en étant au plus proche de la base de données, le DBF va pouvoir d'avantage comprendre la requête et ainsi être plus efficace que le WAF dans son filtrage.

Aujourd’hui il existe quelques propositions de DBF, notamment les deux grands leaders que sont la solution “Audit Vault and Database Firewall” d’Oracle qui a un coût important ou encore la solution OpenSource “greensql” qui n'est plus suivie par la communauté open source mais a été repris par une entreprise Israélienne.

Pendant ce projet, nous nous appliquerons donc à développer un Database Firewall (DBF), à l’intégrer au sein d’une infrastructure faillible, et à tester ses capacités tant en terme de sécurisation que de temps d’exécution.

- ## Innovations / Valeurs ajoutées

Contrairement aux autres DBF existants qui ne traitent que les requêtes en provenance des applications(input), notre DBF filtrera les inputs et les outputs(ce qui provient de la base de données). Cela permettra de s'assurer qu'aucune information compromettante n’est retournée à l'attaquant. Avec cela, nous espérons améliorer l’efficacité de notre Database Firewall. Nous pourrons ainsi quantifier les biens faits de cette méthode en terme de sécurité et les comparer aux pertes de performances temporelles que cela occasionnera, ce qui nous permettra de justifier l’éventuel intérêt de notre approche.

Nous voulons également que notre outil fonctionne avec un système de liste blanche (« tout ce que j’autorise est sur cette liste, je refuse le reste ») et de liste noire(« tout ce que je refuse est sur cette liste, le reste je l’accepte »). En effet, le système de whitelist, qui nécessite une connaissance de son application, est beaucoup plus sécurisé que le système de blacklist, qui atteint la limite des failles connues et qui demande

moins de connaissances sur le fonctionnement réel de l'application. Nous allons mettre en place une combinaison de ces deux systèmes en intégrant une whitelist pour traiter le flux en provenance de l'application à sécuriser et de blacklist en sortie vers l'application pour ne pas laisser sortir des données privées.

Le choix de favoriser la whitelist est néanmoins beaucoup plus contraignant pour l'utilisateur de l'outil, il sera donc intéressant de réfléchir à simplifier cette mise en place.

De plus, aujourd'hui, les IDS fonctionnant sur le système de whitelist utilisent les machines learning. Nous ne désirons pas mettre en place un tel système, par manque de temps. Nous allons ainsi implémenter un mécanisme de whitelist qui n'utilise pas ce système.

Organisation du projet :

Nous avons divisé notre projet en 6 parties qui sont elles-mêmes divisées en sous-parties. Les 5 première parties suivent un ordre chronologique dans un gestionnaire de planning tandis que la dernière se retrouve au fil du projet. Ces différentes tâches ont été partagé entre nous mais pourront tout de même être sujette à changement selon l'avancement du projet.

		Groupe 35 - Projet DBF		
No	***Actions***		***Travail estimé(heures)***	***Responsable***
1	**Préparation du planning**		*8*	Grégoire[25], Baptiste[75]
2	**Etat de l'art**		*100*	Grégoire, Baptiste
	2.1	Fonctionnement des parsers	40	Grégoire
	2.2	Méthode de détections d'intrusions	20	Grégoire
	2.3	Fonctionnement des autres DBFs	40	Baptiste
3	**Pré-étude de développement**		*45*	Grégoire, Baptiste
	3.1	Définition format traitement/analyse	10	Grégoire, Baptiste
	3.2	Fonctionnement du protocole MySQL	20	Baptiste
	3.3	Fonctionnement et mise en oeuvre d'un proxy	15	Baptiste
4	**Développement de l'outil**		*165*	Grégoire, Baptiste
	4.1	Dev client/serveur intégrant échanges SQL	30	Baptiste
	4.2	Dev parseur	30	Grégoire
	4.3	Dev analyseur	70	Grégoire[60], Baptiste[40]
	4.4	Tests Unitaires	15	Grégoire, Baptiste
	4.5	Tests d'intégrations	20	Grégoire, Baptiste
5	**Evaluation de l'outil**		*13*	Grégoire, Baptiste
	5.1	Test performance en temps	5	Grégoire, Baptiste
	5.2	Test performance sécurité	8	Grégoire, Baptiste
6	**Gestion de projets (tout au long du projet)**		*39*	Grégoire, Baptiste