2013/2014

M2-Informatique SSI

GULDNER Geoffrey
HURIER Médéric
LACAVE Valentin
SOIHILI Mouchtali

[Rapport de faille]

SOMMAIRE

I. Faille n°F001

Description de la faille

Faille sur WkiWikiWeb qui permet le remplissage de la base de données automatiquement causant une impossibilité de création d'article.

Description d'attaque possible

Une attaque grâce à un plugin sur Firefox permet le remplissage de tous les champs d'un article (le formulaire de création d'article).

Contre mesure mise en place

La mise en place d'un capchat à la fin de la création d'un article pour vérifier que ce n'est pas un robot permet d'enrailler la création automatique d'article et donc le remplissage de la base de données.

II. Faille n°F002

Description de la faille

Notre Mediawiki est protégé contre les attaques automatiques de type remplissage du disque dur, mais nous avons dû mettre à jour notre application web en conséquence.

Description d'attaque possible

Un attaquant peut créer des articles de taille importante (> 3 Mo) grâce à des scripts automatiques. Au bout d'un certain temps, le disque dur de notre serveur peut être saturé.

Contre mesure mise en place

Nous avons rajouté une condition de validation: le texte de l'article ne doit pas dépasser 512 caractères pour être valide (comme sur le Mediawiki). Ce test a été déployé pour les fonctions d'import, de création, d'édition, de téléchargement et de rechargement.

Notez que plusieurs contre-mesures étaient déjà présentes sur notre serveur (captcha, limitation du nombre d'article par utilisateur, antispam ...). Elles sont consultables dans notre rapport de réalisation.

III. Faille n°F003

Description de la faille

Nous manquions d'un outil de supervision pour consulter l'état de notre site et de notre serveur. La consultation des logs était faite manuellement, ce qui était couteux en temps et en analyse.

Description d'attaque possible

Cette faille nous empêchait de détecter les attaques des autres groupes à cause du manque de visibilité.

Contre mesure mise en place

Nous avons mis en place le SIEM Splunk en local pour corriger ce problème.
Initialement, nous comptions utiliser Munin, un outil simple et facile à installer. Il s'est révélé peu sécurisé (pas de SSL, port exposé ...) et nous avons opté pour une autre solution Open Source: Shinken. Shinken est basé sur Nagios, dont il étend ses fonctionnalités grâce à des plugins en Python et une séparation plus fine entre ses composants (arbitrer, receiver, poller ...). De même que pour Munin, ce logiciel n'était pas sécurisé par défaut, et la configuration s'est révélée très compliquée.
Splunk s'est révélé très facile à installer. Il est extensible par des extensions installables par le navigateur pour monitorer tous les composants logiciels de notre système. C'est un outil très générique, orienté vers le Big Data, mais qui se révèle adapté pour un usage orienté sécurité.
Nous avons installé une application pour surveiller notre serveur Linux, notre application Django et l'état de notre site (Web Ping). Des exemples d'interfaces sont disponibles ci-dessous.