




Éditorial

Le pentester, un artisan ?

Louis Nyffenegger 

La réalisation de tests d'intrusion sur des applications Web est un domaine particulièrement complexe. Même si les attaques et méthodes d'attaques sont connues (injections SQL, exécution de code, XSS, CSRF, ...), il est souvent difficile d'arriver à automatiser la détection de ce type de vulnérabilités, c'est pourtant le rêve de tout informaticien : automatiser le travail récurrent, ne garder que les meilleurs moments.

Prenons l'exemple de la simple recherche de fichiers, en voulant par exemple rechercher si certains fichiers sont présents sur le serveur. Il est déjà difficile d'automatiser cette tâche, en effet, certains serveurs répondront par une erreur HTTP 404 alors que d'autres renverront un code 200 avec un message spécifique contenu dans la page, dès lors, on se rend rapidement compte des limites d'outils tel que Nikto [1] ou DirBuster [2].

Outre cette recherche d'informations, première phase de l'intrusion, la réelle recherche de vulnérabilité demeure un vrai défi pour les intrus. Alors que l'humain va facilement faire la différence entre une page vulnérable et une page non vulnérable par interprétation du résultat visuel de la page, un *robot* (script ou application) aura beaucoup plus de mal à réaliser cette analyse. Certains outils pour détecter une injection SQL recherche par exemple des messages d'erreurs dans les pages retournées par le serveur (*You have an error in your SQL syntax* par exemple).

Cependant de nombreux outils existent afin d'automatiser certaines parties du test d'intrusion, le nombre d'outils permettant d'automatiser l'exploitation d'injections SQL ne cesse de croître. La détection reste néanmoins un problème difficile (insoluble ?) même si beaucoup d'outils tentent d'y remédier : Wapiti [3], SQL Injection Brute-forcer [4], ... Du côté de l'automatisation de l'exploitation des vulnérabilités, beaucoup plus d'outils sont disponibles [5], et les possibilités offertes sont extrêmement impressionnantes [6].

De plus, certaines vulnérabilités critiques ne seront jamais découvertes par des outils automatiques : quantité négative ou nulle, erreur de gestion des droits sur les profils des utilisateurs, erreur de gestion de l'accès à certains fichiers sensibles, ... Une erreur de conception dans l'application dépend tellement de celle-ci que la

détection semble quasiment irréalisable par un outil automatique. En effet, comment un outil peut-il savoir qu'une valeur doit être positive quand on parle d'une quantité, ou qu'une information ne doit être accessible que par le profil administrateur ? Chaque application Web est unique et c'est ce qui rend l'automatisation de la détection de vulnérabilités réellement complexe.

La dernière limite de ces outils est la gestion du chiffrement, par exemple, très peu d'outils gèrent les *tokens* utilisés pour l'authentification sur certains sites sensibles et il est souvent nécessaire de développer un relais pour faire l'interface entre l'outil et l'application. Le seul avantage d'un outil de détection est une meilleure exhaustivité, là où un intrus pourra laisser passer quelques pages, un outil de détection suivant tous les liens du site et brute-forçant les noms de fichiers fera les tests sur un plus grand nombre de fichiers et de paramètres. Un algorithme, même limité, reste plus fiable et plus rapide que l'humain.

J'espère vous avoir convaincu qu'un test d'intrusion sur une application web est complexe et que son automatisation est quasiment impossible même si cela semble être indispensable, c'est l'alliance d'outils de qualité et de personnes compétentes qui permet la découverte d'un maximum de vulnérabilités. ●

À propos de l'auteur

Louis Nyffenegger est consultant en sécurité suisse travaillant chez HSC (*Hervé Schauer Consultants* – <http://www.hsc.fr>). Il réalise des audits, études et tests d'intrusion. Louis remercie toute l'équipe HSC pour son aide et les relectures. Louis peut être contacté à l'adresse suivante : Louis.Nyffenegger@hsc.fr

Références :

- [1] <http://www.cirt.net/code/nikto.shtml>
- [2] <http://www.sittinglittleduck.com/DirBuster/>
- [3] <http://wapiti.sourceforge.net/>
- [4] <http://www.open-labs.org/>
- [5] <http://www.security-hacks.com/2007/05/18/top-15-free-sql-injection-scanners>
- [6] <http://sqlninja.sourceforge.net/sqlninjademo.html>