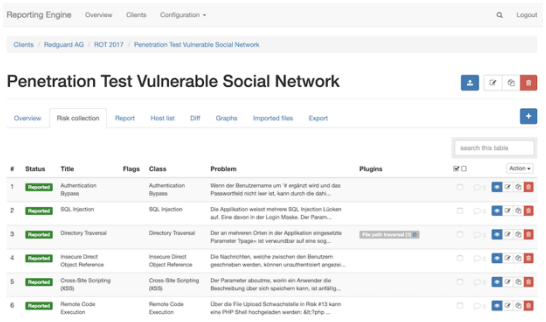
Un reporting efficace des tests d'intrusion

Feb 25, 2020 par Sven Vetsch, Dominique Meier

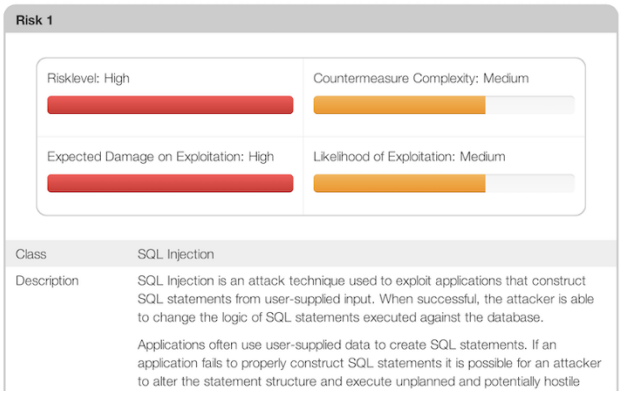


La phase la plus importante d'un projet de test d'intrusion est sans doute la rédaction du rapport final. Du point de vue du « pentester », la phase de reporting n'est pas très excitante en regard des tests d'intrusions proprement dit, mais le rapport est le « livrable » du projet et **son contenu est donc essentiel pour le client**. Il est dans l'intérêt bien compris du client et des « Pentester » de consacrer le plus de temps possible aux tests proprement dit. Chez Redguard, nous sommes convaincus que pour y parvenir, un reporting efficace et de qualité est la clé pour offrir des services de tests d'intrusion professionnels, ceci bien entendu, en sus de la qualité intrinsèque des tests de sécurité effectués par nos équipes.

Lors de la création de Redguard en 2012, ceci sur la base de nos expériences existantes, il a été décidé que Redguard ne produirait pas de rapports de tests d'intrusion sous la forme de documents Word, mais que l'on utiliserait les outils de reporting les plus appropriés. Non pas que Microsoft Word ne soit pas un bon outil pour écrire des textes, mais un rapport de test d'intrusion est bien plus qu'un simple texte et nécessite donc un outillage adéquat pour sa création.

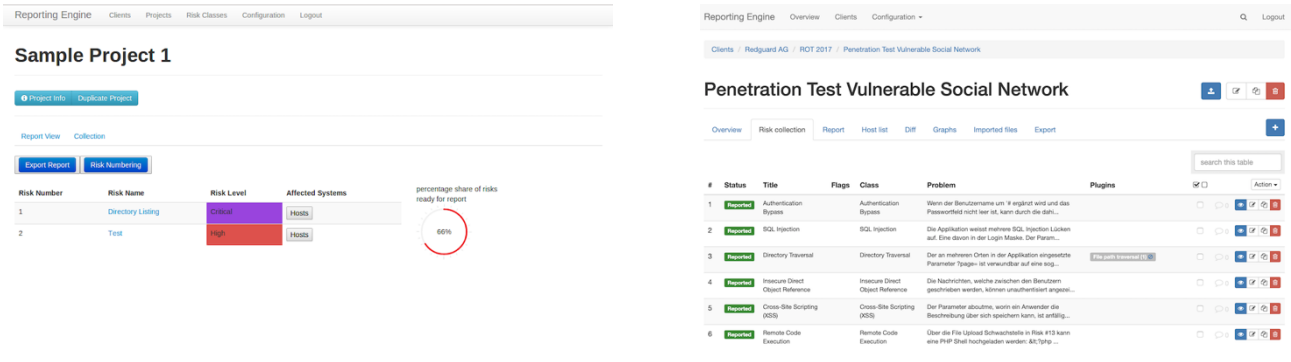
- Un rapport de test d'intrusion, **c'est une question de données, pas de texte**.
- Un test d'intrusion doit toujours être réalisé par au moins deux testeurs en sécurité. Tous les intervenants impliqués contribuent au rapport final, **la collaboration est donc essentielle**.
- Dans les projets de grande envergure, il y a souvent **beaucoup de découvertes et de résultats**. Il est dès lors difficile de les harmoniser et de garder une vue d'ensemble, une lecture attentive nécessite une navigation constante à travers l'ensemble du document.
- Chaque projet et donc chaque rapport est individuel, mais il y a souvent des trouvailles répétitives, comme les failles de type Cross-Site Scripting ou de type injection SQL. **Ces contenus doivent pouvoir être réutilisés**, et ce dans différents projets.
- Un test d'intrusion n'est pas un document « **isolé** », les risques et les vulnérabilités trouvés doivent pouvoir être transférés dans des applications existantes, tel un registre des risques ou un système de contrôle interne (SCI). Un service de test d'intrusion professionnel doit donc offrir la possibilité de **mettre à disposition les résultats dans différents formats**. Il peut s'agir de simples documents PDF, mais aussi de fichiers CSV, JSON, de listes Excel classiques ou dans tout autre format propriétaire défini par le client.
- La mise en forme aide à présenter les contenus de manière plus claire, mais **les formats utilisés doivent être utilisés de manière cohérente**.

Pour remédier aux lacunes et faiblesses d'un document Word, Redguard a commencé dès 2012 à générer des rapports de test d'intrusion basés sur les résultats des testeurs de sécurité à l'aide de quelques scripts Ruby. Pour être honnête, cela fonctionnait plutôt mal au départ ; nous avions des modèles basés sur XML dans lesquels nous remplissions nos résultats avec des éditeurs de texte, puis nous exécutions quelques scripts qui généraient d'abord un document HTML, puis avec un peu de JavaScript et de CSS un document PDF. Cela a certes fonctionné, mais le travail manuel dans les fichiers XML était plus important que si chaque testeur de sécurité avait écrit un document Word et que tout avait été assemblé à la fin.



Une découverte dans un de nos premiers rapports en 2012

Nous avons rapidement réalisé que l'idée de base était bonne, mais que la qualité de l'implémentation n'était pas encore au niveau que nous souhaitions, c'est le moins que l'on puisse dire. C'est à ce moment-là qu'une décision fondamentale a été prise au sein de Redguard : les rapports de test d'intrusion sont un élément si essentiel de nos services que nous voulions les créer dans une application dédiée. Nous avons ensuite parlé de cette idée avec notre actuel responsable des opérations, Dominique Meier, qui à l'époque était à la recherche d'un sujet pour sa thèse de bachelor. C'est également à ce moment-là que le terme Reporting Engine est apparu pour la première fois - un nom pas très créatif, mais qui correspondait exactement à ce que nous voulions. Dominique a pris les scripts Ruby et les modèles existants et les a transformés en une application web Ruby-on-Rails intuitive. Cette application permettait non seulement d'importer les résultats de différents outils de test de sécurité automatisés et de regrouper des résultats similaires, mais aussi, pour la première fois, **de mettre en place une véritable collaboration en temps réel entre les testeurs de sécurité impliqués dans un test d'intrusion**.



Première version du Reporting Engine (à gauche) comparée à la version actuelle (à droite).

Après l'achèvement de la thèse de bachelor de Dominique, le Reporting Engine est directement entré en production. Depuis lors, chaque rapport de test d'intrusion rédigé par les testeurs de sécurité de Redguard a été réalisé avec le

Reporting Engine et exporté à partir de celui-ci. La première version du moteur de rapports n'était bien sûr pas encore parfaite et, en plus de divers bugs, il manquait un grand nombre de fonctionnalités souhaitées par les testeurs de sécurité. Afin d'améliorer encore l'outil, Redguard a engagé un ingénieur logiciel spécialement pour le développement du moteur de rapports et celui-ci continue aujourd'hui encore à développer cet outil essentiel pour Redguard. Nous entendons souvent dire que c'est extrêmement cher pour améliorer l'outil de reporting des tests d'intrusion. Oui, mais cela en vaut vraiment la peine, surtout si l'on se base sur **les nombreux retours positifs des clients concernant la qualité de nos rapports**.

Voici quelques caractéristiques non exhaustives du moteur de reporting :

- Nous avons **des modèles de données pour tout** ce qui se doit figurer dans un rapport de test d'intrusion, par exemple des références automatisées aux CVEs ou aux cheat sheets OWASP. Nous pouvons ainsi automatiser les tâches de reporting et traiter les données de telle manière qu'elles puissent être interprétées et réutilisées au mieux par nos clients. Cela permet également d'analyser les risques en profondeur, par exemple la probabilité d'identifier une certaine classe de risque dans un périmètre de test défini, ce qui permet **d'évaluer les risques métiers**.

- **Tout est centralisé au même endroit** et tous les testeurs de sécurité travaillent simultanément avec la même **application web**. Dès qu'un testeur de sécurité saisit un nouveau risque, celui-ci est immédiatement disponible pour tous les testeurs de sécurité. Cela permet de **mettre en œuvre efficacement les workflows** tels que les processus de révision dans le moteur de rapports.

- Avec le Reporting Engine, il n'est plus nécessaire de fouiller dans les anciens rapports pour retrouver des risques déjà signalés et les réutiliser dans un rapport actuel. Grâce à notre backend basé sur Elasticsearch, nous avons la possibilité **d'effectuer une recherche en texte intégral** sur toutes les données qui ont été saisies.

- Les informations sensibles comme les noms de clients ou les adresses IP sont marquées comme telles et automatiquement supprimées si, par exemple, un risque est réutilisé comme modèle dans un autre projet. Nous pouvons ainsi garantir que **les informations sur les clients restent toujours séparées les unes des autres et ne se mélangent jamais**.

- L'interface utilisateur **est optimisée pour naviguer même dans des étendues de données gigantesques**. Lors d'analyses de vulnérabilité notamment, il peut y avoir quelques centaines de risques complètement différents.

- **Les risques similaires sont regroupés de manière entièrement automatisée** ou peuvent également être simplement regroupés manuellement.

- Nous gérons **une collection centrale de classes de risques** au sein du Reporting Engine, qui peuvent être utilisées comme modèles pour de nouveaux risques et minimiser ainsi le temps de traitement nécessaire.

- Le moteur de reporting comprend un grand nombre de **connecteurs pour les outils et plugin de sécurité courants**. Il est ainsi possible d'importer facilement des données d'outils tels que Nessus et Burp ou encore de scanner de sécurité d'images Docker comme Clair. De nouveaux outils ? Pas de problème. En moyenne, **de nouveaux modules d'importation peuvent être implémentés en l'espace de 30 minutes**.

- Grâce à l'utilisation **d'exports basée sur des plug-ins**, nous pouvons proposer à nos clients des rapports dans une multitude de formats, en particulier ceux qui sont adaptés à leurs besoins.

- **Tout est écrit en Markdown**, ce qui nous permet d'éviter les problèmes et les incohérences de formatage, comme les différentes tailles de police.

- **Le moteur de rapports prend en charge différentes langues** et nous permet ainsi de fournir à nos clients des rapports dans la langue qu'ils préfèrent.

Le moteur de rapports est devenu un outil indispensable pour nos testeurs de sécurité et, en comparaison directe avec les débuts avant que nous ne l'ayons, nous avons pu **réduire le temps de reporting de 50 à 60%**. Cela signifie que nous pouvons tester davantage avec le même temps de projet et que nos clients profitent donc aussi directement de notre moteur de rapports.

Tu es un testeur de sécurité qui doit encore écrire ses rapports dans Word ? Nous sommes toujours à la recherche de nouveaux testeurs de sécurité pour notre équipe et le Reporting Engine t'attend 😉.