Première présentation : OpenBSD, sécurité et contre-mesures

M. Matthieu HERRB (LAAS CNRS) a fait une présentation sur les différentes catégories de contre-mesures présentes dans OpenBSD afin de rendre plus complexes des attaques sur ce système d'exploitation.

La présentation a commencé par un bref rappel de l'historique et de la philosophie d'OpenBSD. Il s'agit d'un système d'exploitation dérivé de BSD 4.4 dont le développement traite en parallèle les aspects noyau, utilisateur (userland) et documentation. Le projet OpenBSD fait deux releases par an, de manière « obligatoire », tous les premiers mai et novembre. Les éventuels éléments non finalisés ou non stabilisés lors du gel d'une version (en phase de pré-diffusion de celle-ci) sont retirés, afin de garantir la stabilité de chaque version diffusée. Le travail étant axé sur une licence BSD, le code rattaché à la GPL est progressivement retiré (lorsque cela est possible).

L'optique de développement d'OpenBSD est, depuis 1996, qu'il soit « sûr par défaut ». Cela signifie notamment la volonté de production d'un code correct, des revues et relectures systématiques de code avant que celui-ci ne soit intégré dans le projet et, lors de la correction d'une erreur, la recherche explicite d'erreurs similaires dans l'ensemble du code.

OpenBSD a été amené à modifier quelques outils génériques, dont gcc, et utilise des outils tiers spécifiques (llvm, clang, parfait, etc.) pour l'audit de son code.

En outre, quelques techniques spécifiques ont été mises en oeuvre (strlcpy/strlcat, révocation de privilèges, cages, uid dédiés par service, etc.).

Débordements sur la pile et autres attaques classiques :

- stackgap : l'implantation en mémoire de la pile est décalée d'une taille aléatoire (mais toujours inférieure à une page)
- propolice : utilisation de canaris sur la pile, et réorganisation des variables locales des fonctions (lorsque celles-ci contiennent des tableaux) afin que les tableaux soient proches du canari
- W^X[1] : les pages en mémoire sont soit modifiables, soit exécutables, mais ne peuvent pas être les deux en même temps. La mise en oeuvre de cette technique a été plus ou moins difficile selon les architectures ; dans le pire des cas (lorsque diverses acrobaties techniques ont été nécessaires), la déperdition à l'exécution reste faible (inférieure à 3%)
- sources d'aléa (entropie et arc4random) utilisées systématiquement : modification de l'ordre de chargement des bibliothèques dynamiques ; déplacement aléatoire de l'adresse d'implantation en mémoire ; adresses renvoyées par mmap valides mais aléatoires ; malloc s'appuyant sur mmap s'il doit allouer moins d'une page mémoire, et sur une classique liste chaînée de blocs libres, avec choix aléatoire du bloc à renvoyer au programme.

1 Lire « W ou exclusif X »

Gestion et réduction des privilèges :

- les programmes ayant besoin de privilèges étendus les révoquent dès que possible
- notion de « moniteur root » paranoïaque dans ses actions, qui dispose toujours de droits étendus, lorsqu'il existe un besoin permanent de tels droits. Une douzaine de démons fonctionne ainsi (sshd, X, etc.)

Protections réseau :

- ajout d'aléas pour les données où une valeur arbitraire est possible, tout en garantissant leur non-répétition ainsi qu'un intervalle minimal entre deux valeurs, et en éliminant toute valeur « magique ». Ainsi, des attaques comme celle décrite par Paul Watson[2] sont déjouées grâce à des ports sources véritablement aléatoires et une légère entorse à TCP (en exigeant qu'un paquet RST acquitte l'octet placé à l'extrême droite de la fenêtre). De même, le DNS utilise des identificateurs pseudo-aléatoires et des ports source aléatoire depuis longtemps (avant la découverte de Dan Kaminsky[3]). La pile IP elle-même utilise en de nombreuses circonstances des valeurs aléatoires (IPID, numéro initial de séquence, ports source, etc.).
- packet filter (pf) dispose de la capacité d'injecter des valeurs aléatoires dans les paquets qui le traversent (estampilles temporelles, numéros initiaux de séquence, ID, etc.). Cela permet à un routeur OpenBSD d'ajouter ces aléas (et donc de bloquer des attaques réelles ou simplement hypothétiques) sur les paquets produits par d'autres systèmes d'exploitation.

Eléments encore manquants :

Il manque encore quelques éléments dans la boîte à outils mise en oeuvre par OpenBSD, dont par exemple :

- la protection des outils des utilisateurs (navigateur, messagerie, multimédia, etc.)
- la résolution d'un problème avec X (qui doit accéder au matériel) lié à certains composants Intel pouvant, du fait de failles, accéder à toute la mémoire du système (par transitivité, X peut donner accès à l'intégralité de la mémoire du système). Beaucoup de travaux sont menés sur ce point, dont notamment KMS/DRI

Seconde présentation : SCAP, OVAL

M. Benjamin MARANDEL (McAfee France) a fait une présentation sur SCAP et les sous-ensembles qui composent ce protocole.

SCAP (Security Content Automation Protocol) est un protocole du NIST[4] visant à homogénéiser les descriptions de vulnérabilités et de systèmes (logiciels, systèmes d'exploitation, etc.), les tests à mener pour vérifier la présence d'une vulnérabilité sur un outil particulier pour un système

2 Slipping in the Window, http://osvdb.org/ref/04/04030-SlippingInTheWindow_v1.0.doc
3 http://www.kb.cert.org/vuls/id/800113
4 http://scap.nist.org

d'exploitation donné, les contrôles à exécuter pour s'assurer de la bonne configuration d'un composant, etc. Il s'agit donc d'un cadre visant à regrouper et homogénéiser les différentes initiatives informationnelles relatives aux vulnérabilités, tests de configuration et/ou tests de vulnérabilités, descriptions de systèmes, etc.

SCAP repose sur six éléments :

1. OVAL (http://oval.mitre.org), Open Vulnerability and Assessment Language. Il s'agit d'une spécification XML permettant de décrire des tests sur des systèmes (vulnérabilités, configuration, mise en place de correctifs).
2. CVE (http://cve.mitre.org) : Common Vulnerability and Exposure, l'un des nombreux « dictionnaires » de vulnérabilités, et l'un des plus connus.
3. CCE (http://cce.mitre.org) : Common Configuration Enumeration, qui permet de décrire des points ou éléments de configuration.
4. CPE (http://cpe.mitre.org) : Common Platform Enumeration, pour la description des plate-formes techniques (systèmes et logiciels).
5. XCCDF (http://scap.nist.gov/specifications/xccdf/index.html) : eXtensible Configuration Checklist Description Format, pour décrire des tests des configuration techniques.
6. CVSS (http://www.first.org/cvss) : Common Vulnerabilities Scoring System, pour disposer d'une échelle homogène afin d'évaluer la criticité des vulnérabilités.

En regroupant ces différents éléments qui, tous, traitent de sécurité (mais chacun sous un angle particulier), SCAP vise à permettre de déterminer facilement comment tester/vérifier une vulnérabilité, quoi tester exactement, où faire les opérations, tout cela de manière standardisée et (en principe) inter-opérable. Un tel standard devrait permettre de réduire les coûts (efforts, durée, complexité) des politiques de gestion de risques, tout en accélérant les tests de conformité à certains standard.

Il existe déjà plusieurs outils respectant SCAP. La liste à jour se trouve sur le site du NIST : http://nvd.nist.gov/scapproducts.cfm. Cette liste est normalement vouée à s'étendre.

Quelques cas d'utilisation de SCAP ont été évoqués et discutés.

Très classiquement, la détection de vulnérabilités (au sens de contrôle de leur présence ou absence sur une certaine plate-forme) devrait être facilitée et améliorée grâce à des descriptions communes et publiques et des tests documentés et validés. Les tests pouvant être accompagnés de pré-conditions (sur les outils et leurs versions, ainsi que la configuration de ceux-ci, sur les plate-formes sous-jacentes ou les systèmes d'exploitation, etc.), cela permettra aux RSSI et ingénieurs sécurité de se concentrer sur les cas réels de vulnérabilités de leurs plate-formes, plutôt que sur des cas potentiels à vérifier par la suite.

SCAP devrait facilité la réunification et l'interfaçage de la gestion des équipements, des vulnérabilités, des configurations et de la sécurité.