Exemples d'attaques liées au réseau

Olivier ROUSSEL
olivier.roussel@univ-artois.fr

04/10/2016, version 2633

Attaques des protocoles de base

- usurpation d'adresse IP (IP spoofing)
  - prendre l'adresse IP d'une autre machine pour contourner les contrôles d'accès basés sur l'IP, ou générer un déni de service
  - *solution possible : filtrage des adresses sources, ne pas se contenter d'un contrôle sur l'adresse IP, surveiller les adresses MAC du réseau, bloquer l'accès aux adresses inconnues*
- usurpation d'adresse MAC
  - même principe que l'IP spoofing
  - *solution possible : ne pas se contenter d'un filtrage sur l'adresse MAC, 802.1X (authentification pour accès au LAN)*
- saturation des tables d'adresses MAC des switches
  - transformer un switch en hub, pouvoir observer le trafic de tout le réseau et le ralentir
  - *solution possible : limiter le nombre d'adresses MAC par port*

Attaques des protocoles de base

- utilisation de bugs ou de différences dans les implantations TCP/IP
  - ping of death, paquets IGMP : faire crasher le système d'exploitation
  - gestion différente de la fragmentation IP (peut servir à masquer une attaque)
- manipuler le protocole STP
  - forcer une nouvelle élection du switch racine, devenir switch racine et récupérer le trafic
  - *solution possible : bloquer les messages STP sur les ports qui ne relient pas des switches (portfast)*
- épuiser l'intervalle d'adresses IP distribuées par DHCP
  - envoyer de fausses requêtes DHCP
  - *solution possible : ne pas allouer de plages d'adresses*
- sortir du VLAN
  - utiliser un mauvais identifiant de VLAN
  - *solution possible : ne pas utiliser de VLAN taggé entre matériel non sécurisés, donner un numéro de VLAN fixe à chaque port*

Redirection du trafic

- rediriger le trafic du réseau local
  - envoyer de fausses requêtes DCHP ou des messages ICMP redirect pour router le trafic vers la machine du pirate, ou de faux messages de routage (RIP, OSPF, BGP). Mettre en plac un faux point d'accès WIFI
  - *solution possible : n'autoriser les réponses DHCP que sur certains ports des switchs, bloquer ICMP redirect (!), se prémunir contre les attaques MITM. Surveiller les réseaux WIFI*
- ARP spoofing
  - envoyer de fausses réponse ARP pour rediriger le trafic
  - *solution possible : utiliser des tables ARP statiques (!), surveiller les messages ARP et détecter les cas douteux, se prémunir contre les attaques MITM*
- brancher un matériel non autorisé sur le réseau
  - ce matériel peut servir à attaquer le réseau de l'intérieur
  - *solution possible : vérifier l'adresse MAC (!), authentification 802.1X*

Divers

- agir sur la résolution de noms
  - modifier le contenu d'un serveur DNS, ou envoyer de fausses réponses DNS
  - *solution possible : utiliser DNSSEC*
- packet sniffer
  - intercepter les communications (paquets) et les secrets éventuels
  - *solution possible : chiffrer avec SSL/TLS*
- network scanning
  - identifier les services disponibles sur un réseau, pour ensuite les attaquer
  - *solution possible : filtrage*
- vol de session
  - s'immiscer dans une communication TCP ou des échanges HTTP
  - *solution possible : utiliser de bons identifiants de session (aléatoire)*

Déni de service

But : saturer un système pour l'empêcher de fonctionner

- attaque smurf :
  - faire un ping vers une adresse de broadcast d'un gros réseau en prenant comme IP source celle de la victime
  - *solution possible : bloquer le ping vers le broadcast, bloquer les messages ICMP*
- SYN flood :
  - envoyer un maximum de demandes de connexions TCP sans gérer la suite de la communication
  - *solution possible : limiter le nombre de paquets SYN*
- slowloris :
  - ouvrir un maximum de connexions HTTP (ou TCP) en envoyant les données le plus lentement possible
  - *solution possible : augmenter le nombre de connexions possibles, imposer un débit minimal et un temps limite*

Déni de service distribués

But : saturer un système pour l'empêcher de fonctionner en envoyant des requêtes simultanément depuis de nombreux systèmes.

- les différents systèmes peuvent plus facilement saturer la bande passante de la cible
- les routeurs en amont du système peuvent aussi être saturés
- les sources multiples rendent plus difficile le filtrage
- *solution possible : avoir un FAI avec une infrastructure réseau ayant un débit suffisant, capable d'identifier automatiquement les flux anormaux et de filtrer le plus en amont possible*

Failles applicatives

But : profiter d'une faille applicative pour prendre le contrôle de la machine, ou provoquer un déni de service

- buffer overflow :
  - envoyer plus de données que prévu par le programme, et soit modifier la valeur de certaines variables, soit faire exécuter du code arbitraire
  - *solution possible : ne jamais dépasser la limite des tableaux, rendre la pile non exécutable, rendre aléatoire les adresses du programme*
- injection SQL :
  - transmettre un texte qui contient la fin de chaîne et des commandes SQL arbitraires
  - *solution possible : rendre inactif tout caractère spécial, vérifier les données utilisateur*
- et aussi
  - XSS
  - CSRF
  - vérification insuffisante des données utilisateurs

Virus, etc.

- virus
  - un programme qui se réplique de machine en machine en se cachant dans des programmes
  - *solution possible : anti-virus, ne pas lancer de programme/lire de documents dont l'origine est incertaine*
- cheval de Troie
  - un programme exécuté volontairement par un utilisateur, apparemment innocent mais en réalité malveillant
  - *solution possible : idem anti-virus*
- password cracking, default password
  - trouver les mots de passe de certains comptes
  - *solution possible : choisir des mots de passe suffisamment "aléatoires", changer immédiatement les mots de passe par défaut*

Ingénierie sociale

- phishing
  - présenter à l'utilisateur une fausse page web l'invitant à saisir ses identifiants
  - *solution possible : éduquer les utilisateurs, ne jamais cliquer sur des liens depuis des pages dont on ne connait pas l'origine*
- faux noms de domaines, typosquatting
  - fournir des liens avec des noms de domaine qui ressemblent à l'original mais sont différents (ex. omicron au lieu de o), utiliser des domaines qui correspondent à des fautes de frappe
  - *solution possible : éduquer les utilisateurs, ne jamais cliquer sur des liens depuis des pages dont on ne connait pas l'origine*
- manipulation de l'utilisateur
  - le convaincre de communiquer une information sensible (identifiants) ou de réaliser une action pour le pirate
  - *solution possible : éduquer les utilisateurs*

Remarques

- Toute machine connectée à un réseau présente un risque
- La sécurité physique doit être garantie
  - vol de la machine
  - démarrage depuis le réseau, l'USB ou le DVD
  - accès physique au disque dur
- Quand plusieurs machines ou systèmes d'exploitation cohabitent, la sécurité de votre système informatique est celle de son maillon le plus faible.
  - nécessité de mettre à jour toutes les machines
  - attention aux périphériques divers (imprimantes, caméra IP, ...)
  - attention aux vieux systèmes qui gèrent des installations spécifiques (contrôle d'accès, appareil de laboratoire, ...)

Le web

Interception de la communication

- Toute communication HTTP transite en clair sur le réseau, et peut donc être interceptée.
- Il est donc indispensable d'utiliser HTTPS pour toute information sensible.
- En HTTPS, il est indispensable de valider le certificat fourni par le serveur pour éviter toute interception par une attaque MITM. Cette validation est normalement de la responsabilité de l'utilisateur.

Validation des données utilisateurs

- Toute donnée transmise par l'utilisateur doit être validée, c'est à dire qu'il faut s'assurer qu'elle respecte la syntaxe attendue et reste dans les limites prévues par le programme.
- Toute donnée non validée doit être ignorée.
- La validation doit toujours se faire côté serveur, car la validation faite chez le client (par du code javascript par exemple) peut être contournée/désactivée.

Injections SQL

- On ne doit jamais construire une requête SQL en concaténant une chaîne représentant une commande avec une chaîne fournie par l'utilisateur (et qui n'a pas été correctement validée) .
- Exemple : "select champ from table where id=$id;". Ici, on pense que $id est un nombre. Mais si l'utilisateur transmet id="1; delete * from table;", la requête devient "select champ from table where id=1; delete * from table;".
- Il faut utiliser des requêtes préparées, où les valeurs fournies ne peuvent jamais être interprétées comme des commandes SQL.

Sécurisation des données

- Les données de l'utilisateur ne doivent jamais être accessibles par un pirate.
- Il faut contrôler les accès aux données.
- Il faut chiffrer les données sensibles (mot de passe) et ne jamais les transmettre par des moyens non sécurisés (mail).
- Il faut s'assurer que les données ne sont pas accessibles par un moyen détourné (exemple : accès direct au SGBD via un compte ayant un mot de passe faible, API REST).
- Attention aux documents (factures,...) accessibles via une URL avec un identifiant unique. Un pirate ne doit pas pouvoir deviner/intercepter cet identifiant. Il faut privilégier une solution avec authentification.

Rappels sur les cookies

- Les cookies sont des informations que le serveur stocke chez le client (navigateur) et qui sont renvoyées au serveur à chaque requête.
- Le serveur envoie un en-tête HTTP Set-Cookie: suivi de var=valeur et optionnellement d'une date d'expiration du cookie. On peut aussi préciser pour quel domaine ce cookie est valide (le serveur qui l'émet doit appartenir à ce domaine) et un chemin pour lequel ce cookie est valide. Exemple : `Set-Cookie: id=1234; Expires=Tue, 01 Jan 2030 00:00:00 GMT; Path=/; Domain=.dom.fr; Secure`
- Un cookie sans date d'expiration est temporaire (session cookie). S'il y a une date, il est considéré comme permanent (i.e. doit survivre à un arrêt du navigateur).

Rappels sur les cookies

- On peut spécifier Secure pour ne jamais envoyer le cookie sans passer par https.
- On peut spécifier HttpOnly pour ne pas rendre le cookie accessible aux scripts (js en particulier).
- Le navigateur doit enregistrer cette information, en l'associant au nom du serveur, et la renvoyer avec chaque requête adressée à ce serveur (ou au domaine s'il a été spécifié). Cela se fait avec l'en-tête HTTP `Cookie: id=1234`

Vol de session

- Le protocole HTTP est un protocole non connecté. Chaque requête au serveur est indépendante de la requête précédente.
- Pour relier les requêtes successives faites par un même utilisateur, et donc recréer une session, on doit ajouter à chaque requête un identifiant de session.
- L'identifiant de session doit être imprévisible par un attaquant (donc aléatoire et de taille suffisante pour ne pas permettre une recherche par force brute).
- L'identifiant ne doit pas pouvoir être intercepté, sinon le pirate accède directement à la session d'un tiers (vol de session)

Same Origin Policy

- Un script sur une page web ne peut accéder à une autre page web que si les 2 pages ont la même origine.
- L'origine est la combinaison du protocole, du nom de machine, et du numéro de port. Par exemple http://www.dom.fr:80
- Attention, cette restriction ne s'applique qu'aux scripts, pas aux balises HTML.
- Il peut y avoir des subtilités liés à certains cas spéciaux et implémentations différentes.

Cross Site Scripting (XSS)

- Le pirate place du code HTML/JavaScript dans un champ de formulaire qui sera affiché chez un autre client.
- Si le site ne désactive pas les balises HTML, le code injecté par le pirate sera actif chez le client.
- Ce code HTML peut contenir un script JavaScript qui accède au DOM (et/ou le modifie), par exemple pour lire les cookies du site et les transmettre au pirate.
- Cette attaque exploite la confiance d'un utilisateur pour un site

Cross Site Request Forgery (CSRF)

- Hypothèses : l'utilisateur est connecté au site de sa banque, et la banque propose une API pour envoyer un mandat .
- Le pirate incite l'utilisateur à ouvrir une page web qui contient un lien vers l'API de la banque.
- Quand l'utilisateur clique sur le lien, la requête est envoyée au site de la banque, avec les cookies d'autentification de la banque. L'ordre de mandat peut donc être effectué.
- Tout cela peut se faire sans intervention de l'utilisateur en utilisant du JavaScript.
- Cette attaque exploite la confiance d'un site pour un utilisateur.
- Exemple de parade : placer un champ caché avec une valeur non prévisible dans la page web transmis par la banque et vérifier la valeur de ce champ lors de chaque requête.

Same Origin Policy

- Un script sur une page web ne peut accéder à une autre page web que si les 2 pages ont la même origine.
- L'origine est la combinaison du protocole, du nom de machine, et du numéro de port. Par exemple http://www.dom.fr:80

Cross Origin Resource Sharing (CORS)

- Un script du site www.dom.fr veut télécharger des images depuis img.dom.fr. C'est normalement interdit par la "Same Origin Policy".
- un navigateur qui implèmente CORS va envoyer une requête HTTP OPTIONS à img.dom.fr en ajoutant l'en-tête HTTP `Origin: http://www.dom.fr`
- Si le serveur img.dom.fr souhaite accepter les requêtes depuis une page de www.dom.fr, il enverra un en-tête `Access-Control-Allow-Origin: http://www.dom.fr`. Le navigateur autorisera alors les requêtes à img.dom.fr par un script d'une page de www.dom.fr
- Si le serveur img.dom.fr ne souhaite pas accepter les requêtes depuis une page de www.dom.fr, il envoie une erreur.

Cross Origin Resource Sharing (CORS)

- Si un pirate met en place une page pirate.fr contenant un script cherchant à accéder à img.dom.fr, le navigateur enverra Origin: http://pirate.fr à img.dom.fr, qui enverra une erreur, et le navigateur s'en tiendra à la "Same Origin Policy".

Web sockets

- La Same Origin Policy ne s'applique pas aux web sockets. Il est donc nécessaire sur le serveur de valider l'origine de toute connexion web socket.