INSA – 5° R.T.

Sujet d'examen
26 janvier 2010

Sécurité des systèmes informatiques

2ème partie

Exercice 1 *(2 points)*
Une entreprise remarque que, statistiquement, elle souffre chaque année de cinq infections par des virus et de trois défigurations de son site web. La remise en état des machines après une infection par un virus nécessite deux jours de travail à l'administrateur, soit un coût total estimé à 2000 euros. Le site web peut être remis en état en quelques heures à partir des sauvegardes, soit un coût de 500 euros. La mise en place et la maintenance d'un produit antivirus et d'un système de protection du site web correspond à un coût annuel de 20 000 euros.

1. A partir des coûts ci-dessus, calculer la valeur du risque annuel dû aux virus et aux défigurations et juger l'utilité de la mise en place des mesures de sécurité énoncées.
2. Critiquer la manière dont le risque est calculé et les données limitées que l'on vous a fournies ; proposer une méthode plus adéquate.

Question 1:
En utilisant les informations fournies, on peut évaluer le coût annuel de traitement des infections virales à 10 jours de travail de l'administrateur, soit 10000 euros, et le coût de traitement des défigurations du site web à 1500 euros. Ceci conduit à un coût total de 11500 euros – ce qui semble justifier le choix d'accepter les risques associés à ces types d'attaques si le coût de protection par la mise en place de logiciels est supérieur (20000 euros).
Question 2:
Deux éléments au moins nécessaires à une évaluation plus complète du risque n'ont pas été pris en compte:

- *Tout d'abord, dans le cas des infections virales, le temps de travail de l'administrateur n'est pas la seule source de perte. En effet, pendant les 2 jours nécessaires au traitement des machines affectées par un virus, celles-ci sont également probablement indisponibles pour les utilisateurs (notamment pour empêcher la propagation du virus). Dans ce cas, ceci correspond à une indisponibilité du système d'information qui conduit aussi à un manque à gagner pour l'entreprise. Même si seulement une dizaine d'utilisateurs sont affectés, le coût de cette indisponibilité peut être bien plus important que celui de la mobilisation exceptionnelle de l'administrateur. Si on trouve un serveur parmi les machines infectées, son indisponibilité peut avoir un large impact (mais il peut être traité en priorité par l'administrateur). Enfin, si l'infection virale est généralisé et affecte tous les utilisateurs, le manque à gagner associé à un arrêt quasi complet de l'entreprise (à part l'administrateur occupé à effectuer les désinfections) devient alors certainement très largement supérieur aux 10000 euros estimés.*

- *En ce qui concerne les défigurations du site web, un autre manque à gagner non estimé dans notre calcul peut être associé à l'indisponibilité du service auprès d'éventuels clients de l'entreprise. Mais il y a également dans ce cas un important risque d'image à prendre en compte. L'impact de ce dernier risque est difficile à quantifier, mais on peut très bien comprendre que ce type d'évènement, arrivant plusieurs fois par an à l'entreprise, conduira par exemple l'ensemble de ses clients à avoir une très mauvaise opinion de sa gestion de la sécurité informatique. L'impact financier sur son activité sera peut-être mesurable au bout de quelques années, mais peut-être aussi irrémédiable. En fait, certaines catégories de risque se prêtent assez mal à une évaluation financière et il faut aussi avoir une appréciation qualitative pour savoir si un risque est acceptable ou inacceptable.*

Exercice 2 *(1 point)*
Face à une infection virale :

1. Quel est le rôle d'un système de diffusion de correctifs de sécurité ?
2. Quel est le rôle d'un logiciel antivirus ?

Question 1:
Un système de diffusion de correctifs de sécurité va permettre de corriger les failles de sécurité exploitées par un virus. Il va donc permettre d'empêcher sa propagation dans le système d'information ou de la stopper vers de nouvelles machines. C'est donc principalement un mécanisme de prévention des menaces.
Question 2:
Un logiciel antivirus va viser le virus lui-même et permettre : de le détecter à son arrivée sur une machine (vulnérable) pour l'empêcher de s'installer ; il va aussi permettre d'examiner les machines pour retirer un virus qui aurait réussi à s'y installer. C'est donc principalement un mécanisme d'élimination des menaces.

Exercice 3 *(3 points)*
Une vulnérabilité récente découverte dans le protocole TLS a été présentée comme suit (*Jake Edge*, LWN-2009-11-18, http://lwn.net/Articles/361697/) :
« Transport Layer Security (TLS), and its predecessor Secure Sockets Layer (SSL), are commonly used protocols for encrypting internet traffic, so TLS vulnerabilities can potentially affect a wide range of internet services. A recently disclosed flaw in the TLS protocol—though there is some dispute whether TLS is at fault—allows an "injected plaintext" attack against an encrypted session. This allows a "man in the middle" (MITM) attacker to prefix a victim's request with their own data, which gets interpreted by the server as if it came from the victim. *[…]*
TLS allows clients and servers to renegotiate various session parameters within the TLS connection. When the renegotiation is done, however, TLS applications still accept data that came in before the renegotiation as if it were in the new security context. That hole allows a MITM attack. By arranging that the last data received is from the attacker, then causing a renegotiation with the victim, the attack effectively prepends the attacker's payload to the victim's request. *[...]*
To attack a web-based application, the attacker typically would send their prefix to the server, then cause the renegotiation to occur. That renegotiation would actually be done between the victim's client and the server (with the MITM attacker just proxying the traffic). Due to the bug, the server would process the prefix in the new security context that gets established via the renegotiation. So, neither the client nor the server have any idea that this has occurred, and the attacker gets to insert his payload into the the client's secure session.

Eric Rescorla is one of those working on a long-term fix, but he also has a fairly straightforward example of the plaintext injection:

E.g., the attacker would send:

```
GET /pizza?toppings=pepperoni;address=attackersaddress HTTP/1.1
X-Ignore-This:
```

And leave the last line empty without a carriage return line feed. Then when the client makes his own request

```
GET /pizza?toppings=sausage;address=victimssaddress HTTP/1.1
Cookie: victimscookie
```

the two requests get glued together into:

```
GET /pizza?toppings=pepperoni;address=attackersaddress HTTP/1.1
X-Ignore-This: GET /pizza?toppings=sausage;address=victimssaddress HTTP/1.1
Cookie: victimscookie
```

[...] »

Après avoir examiné ces informations et sachant que la faille concernée (CVE-2009-3555) affecte toutes les versions existantes du protocole SSL/TLS (SSL 2.0, SSL 3.0, TLS 1.0, TLS 1.1, TLS 1.2) :

1. Expliquez le résultat attendu par l'attaquant de cet exemple.
2. Envisagez (sommairement) des modifications du mécanisme de renégociation permettant d'éviter cette attaque.
3. Quelle va être la principale difficulté pour la mise en œuvre de ce type de protection ?

Question 1:
Étant donné la forme finale de la requête reçue par le serveur et le sens que l'on peut facilement déduire du marqueur « X-Ignore-This » dans le flux HTTP, on peut supposer que l'attaquant entend altérer la commande d'une pizza à la saucisse par sa victime pour se faire livrer une pizza aux pepperoni directement à sa propre adresse – tout en laissant le soin de régler l'addition à la victime bien sûr.
Question 2:
Afin d'éviter ce type d'attaque, il va falloir faire évoluer les mécanismes de renégociation de TLS. On peut envisager de ne plus accepter des données émises antérieurement à une renégociation (mais ceci est probablement malcommode au niveau de l'utilisation du protocole). On peut aussi envisager d'introduire dans les échanges de la renégociation de nouveaux éléments (comme un numéro de séquence) liés cryptographiquement aux informations d'authentification de l'ensemble de la session. Ceci permettra d'éviter que des renégociations soient effectuées à l'intérieur d'une même session par des identités différentes.
Question 3:
La principale difficulté est un problème de mise en œuvre. On parle ici d'une évolution d'un protocole de communication normalisé largement répandu (SSL/TLS est à la base de https par exemple). On ne peut pas de manière réaliste supposer que tous les logiciels utilisant ce protocole vont pouvoir évoluer de manière simultanée pour mettre en œuvre la nouvelle version du protocole.
On va donc devoir choisir entre 2 options.

- *Soit on introduira des mécanismes de compatibilité entre les différentes versions du protocole pour permettre son déploiement progressif sans perturbation, mais dans ce cas, la pression pour le déploiement de la nouvelle version sera moindre, et la vulnérabilité persistera tant que les anciennes versions pourront être utilisées. (Il est bien évident qu'un attaquant cherchera alors à faire basculer le serveur sur les anciennes versions avant de lancer son attaque.)*

- *Soit on paramètrera la nouvelle version pour ne pas accepter les clients utilisant l'ancienne ce qui permettra de faire clairement la distinction entre les logiciels vulnérables et ceux ne l'étant pas. Mais dans ce cas, le déploiement de la version sécurisé risque d'être moins bien accepté en raison des perturbations de fonctionnement qu'elle va induire.*

Exercice 4 *(4 points)*

Voici 3 exemples de méthode utilisables par un utilisateur pour choisir un mot de passe :

1. Utilisation d'un mot de passe de 8 symboles choisis au hasard parmi les symboles alphanumériques. On suppose (comme le font certains outils de génération automatique de mots de passe complexes) que le mot de passe contient exactement 1 lettre majuscule et 1 chiffre, les autres symboles étant des lettres minuscules.
   Exemple : `wei0Oone`, `Yei5rauk` ou `roh3Paht`
2. Utilisation de 2 mots du dictionnaire français courant accolés ou *éventuellement* séparés par un des caractères spéciaux choisis parmi : `,;:!?.+-*/<>=`
   Exemples : `LUNE;MIEL` ou `ALLEZBLEUS`
3. Utilisation de la première lettre d'une phrase comptant au moins dix mots. On fera l'hypothèse que seules des minuscules sont utilisées.
   Exemple : `oflhqsdmsu` (2ème phrase ci-dessus)

Question 1 : Comparez ces méthodes en évaluant *quantitativement* le nombre de mots de passe différents associé à chaque méthode, en vous servant si besoin des informations indiquées ci-dessous. Il n'est pas indispensable de calculer de manière exacte la taille de l'espace considéré, mais essayez d'obtenir un ordre de grandeur *justifié* du nombre de mots de passe possibles (ou un minorant).

Question 2 : Ensuite, en vous appuyant sur les informations expérimentales fournies ci-dessous, calculez la durée espérée de résistance d'un mot de passe de type 1, 2 ou 3 face à un outil d'attaque par dictionnaire[1], suite à un vol de la forme chiffrée du mot de passe[2]. Considérez : un système Unix utilisant un chiffrement DES classique, un système Windows utilisant NT LM (DES) et un système OpenBSD utilisant (une variante de) Blowfish.

[1] Une attaque consistant à essayer systématiquement les mots d'un dictionnaire ou des combinaisons simples de ces mots entre eux ou avec des symboles courants.

[2] Contenue dans /etc/passwd ou /etc/shadow sur un système Unix, ou dans la base SAM d'une machine Windows, ou transitant sur le réseau pour certaines applications (VNC, etc.).

Quelques information utiles :

- On pourra considérer qu'un lexique de français courant compte environ 4000 mots[3] (méthode 2).
- Pour plus de réalisme, on pourra considérer (méthode 3) qu'une phrase est composée à 50% de « mots outils » qui ne représentent que 0,5% du lexique total (soit 20 mots au lieu de 4000), mais surtout qui ne correspondent qu'à 25% des lettres de l'alphabet (pour leur première lettre), soit seulement 6 lettres. Pour simplifier, on pourra considérer que la première lettre des autres mots correspond de manière équiprobable à l'ensemble des 26 lettres de l'alphabet.
- Le plus sûr est souvent d'évaluer expérimentalement le nombre de combinaisons par seconde (c/s) qu'une machine peut essayer[4]. Voici, sur une machine récente (Core2 Quad 2,4GHz) les performances obtenues par un outil librement disponible :

```
ortalo@mist:~$ john -test
Created directory: /home/ortalo/.john
Benchmarking: Traditional DES [64/64 BS MMX]... DONE
Many salts:  1026K c/s real, 1026K c/s virtual

Benchmarking: OpenBSD Blowfish (x32) [32/32]... DONE
Raw:  405 c/s real, 405 c/s virtual

Benchmarking: NT LM DES [64/64 BS MMX]... DONE
Raw:  8184K c/s real, 8184K c/s virtual
```

Pour les 3 méthodes, le nombre de mots de passe possibles est le suivant :

- *(chiffre.position)(majuscule.position)(minuscules)*
  $(10.8).(26.7).26^6 = 4\,497\,813\,698\,560 \cong 4{,}5.10^{12}$ *possibilités*
- *On a* 4000^2 *possibilités pour le choix des 2 mots, et ensuite (13+1) possibilités pour choisir, soit un des 13 symboles, soit rien du tout, entre ou après les 2 mots ; ce qui donne au total :* $4000^2 \times (13+1) = 224\,000\,000 \simeq 2{,}2.10^8$ *possibilités.*
- *Dans le troisième cas, on peut considérer qu'un caractère sur deux du mot de passe (par exemple les caractères pairs) est choisi parmi seulement 6 lettres de l'alphabet, et non 26 comme les autres. Soit au total :* $6^5 \times 26^5 = 92\,389\,579\,776 = 9{,}2.10^{10}$

En divisant le nombre total de mots de passe possibles par la vitesse d'essai de l'outil, on obtient alors les résultats suivants (valeurs approchées minorées).

	Cas 1	*Cas 2*	*Cas 3*
DES	$4{,}4.10^6$ *s*	*218 s*	*90 224 s*
NT LM	$5{,}5.10^5$ *s*	*27 s*	*11 289 s*
Blowfish	$1{,}1.10^{10}$ *s*	$5{,}5.10^5$	$2{,}3.10^8$ *s*

soit

[3] Le lexique Dubois-Buyse des mots français courants (enfants entre 0 et 16 ans) compte 3726 mots.

[4] La quasi-totalité des systèmes d'exploitation utilisent un « grain de sel » (*salt*) aléatoire pour ralentir les attaques par dictionnaire (c'est une valeur concaténée au mot de passe de l'utilisateur qui multiplie immédiatement le nombre de combinaisons à essayer pour une attaque par dictionnaire). Voir : crypt(3). Le temps nécessaire pour chiffrer et comparer un mot du dictionnaire à la valeur dérobée dépend bien sûr du type d'algorithme (DES, MD5, MD4, Blowfish, etc.), mais dépend aussi fortement de la longueur du « *salt* » (10 bits, 12 bits, ou plus).

	Cas 1	*Cas 2*	*Cas 3*
DES	*50 jours*	*3,6 min.*	*1 jour*
NT LM	*6,3 jours*	*27 s*	*3,1 h*
Blowfish	*352 ans*	*6,4 jours*	*7,2 ans*

En conclusion, comme on peut le voir, la méthode 2 reste largement vulnérable à ce type d'attaque, surtout dans les cas où l'algorithme de stockage du mot de passe sur le système d'exploitation utilise des techniques « anciennes ». Les méthodes 1 et 3 semblent fournir un meilleur niveau de sécurité, essentiellement dû à l'augmentation de la longueur du mot de passe (et au fait qu'on n'utilise pas directement les mots d'un dictionnaire).
Indépendamment de la méthode choisie, on constate aussi l'effet des efforts de conception au niveau du système d'exploitation: le choix d'une méthode de stockage spécifiquement conçue pour résister à ce type d'attaque dans le cas d'OpenBSD permet d'offrir un niveau de résistance bien plus élevé que sur les autres systèmes même en présence d'utilisateurs peu avertis.
Dans tous les cas, compte tenu de l'efficacité de cette attaque dans la pratique, il faut souligner l'importance de protéger les serveurs contenant la forme chiffrée des mots de passe (utilisée en entrée pour l'outil d'attaque) : par exemple les contrôleurs de domaine, ou les gestionnaires Kerberos ou LDAP ou même les postes de travail isolés (le mot de passe d'administration local d'une machine étant souvent « similaires » à des mots de passe plus privilégiés).