



Cours Sécurité et cryptographie

Chapitre 3: Vulnérabilités des logiciels - logiciels malveillants

Plan



- Partie 1: Vulnérabilités des logiciels
 - Contexte
 - OWASP Top 10 Application Security Risks
 - CWE/SANS Top 25 Most Dangerous Software Errors
 - Exemples de vulnérabilités

- Partie 2: logiciels malveillants
 - Définitions et principales caractéristiques
 - Exemples



Vulnérabilités des Logiciels

Contexte

- Vulnérabilités logicielles:
 - Communes à plusieurs langages **ou**
 - Spécifiques à un langage particulier

- L'exploitation d'une vulnérabilité nécessite:
 - La compréhension du logiciel vulnérable,
 - La détermination
 - de la plateforme,
 - du système d'exploitation,
 - des standards adoptés,
 - des protocoles de communication, ...
 - Atteinte de la partie vulnérable
 - Par exemple: depuis les données acceptées en entrée.

OWASP Top 10 (2010)

Source : https://www.owasp.org/index.php/Top_10_2010-Main

- 1) Injection
- 2) Cross Site Scripting (XSS)
- 3) Broken Authentication and Session Management
- 4) Insecure Direct Object References
- 5) Cross Site Request Forgery (CSRF)
- 6) Security Misconfiguration
- 7) Insecure Cryptographic Storage
- 8) Failure to Restrict URL Access
- 9) Insufficient Transport Layer Protection
- 10) Unvalidated Redirects and Forwards



2011 CWE/ SANS Top 25 (1/3)



Source: <http://cwe.mitre.org/top25/#CWE-89>



Rank	Score	ID	Name
[1]	93.8	CWE-89	Improper Neutralization of Special Elements used in an SQL Command (' SQL Injection ')
[2]	83.3	CWE-78	Improper Neutralization of Special Elements used in an OS Command (' OS Command Injection ')
[3]	79.0	CWE-120	Buffer Copy without Checking Size of Input (' Classic Buffer Overflow ')
[4]	77.7	CWE-79	Improper Neutralization of Input During Web Page Generation (' Cross-site Scripting ')
[5]	76.9	CWE-306	Missing Authentication for Critical Function
[6]	76.8	CWE-862	Missing Authorization
[7]	75.0	CWE-798	Use of Hard-coded Credentials
[8]	75.0	CWE-311	Missing Encryption of Sensitive Data

2011 CWE/ SANS Top 25 (2/3)



Source: <http://cwe.mitre.org/top25/#CWE-89>



Rank	Score	ID	Name
[9]	74.0	CWE-434	Unrestricted Upload of File with Dangerous Type
[10]	73.8	CWE-807	Reliance on Untrusted Inputs in a Security Decision
[11]	73.1	CWE-250	Execution with Unnecessary Privileges
[12]	70.1	CWE-352	Cross-Site Request Forgery (CSRF)
[13]	69.3	CWE-22	Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')
[14]	68.5	CWE-494	Download of Code Without Integrity Check
[15]	67.8	CWE-863	Incorrect Authorization
[16]	66.0	CWE-829	Inclusion of Functionality from Untrusted Control Sphere

2011 CWE/ SANS Top 25 (3/3)



Source: <http://cwe.mitre.org/top25/#CWE-89>



Rank	Score	ID	Name
[17]	65.5	CWE-732	Incorrect Permission Assignment for Critical Resource
[18]	64.6	CWE-676	Use of Potentially Dangerous Function
[19]	64.1	CWE-327	Use of a Broken or Risky Cryptographic Algorithm
[20]	62.4	CWE-131	Incorrect Calculation of Buffer Size
[21]	61.5	CWE-307	Improper Restriction of Excessive Authentication Attempts
[22]	61.1	CWE-601	URL Redirection to Untrusted Site (' Open Redirect ')
[23]	61.0	CWE-134	Uncontrolled Format String
[24]	60.3	CWE-190	Integer Overflow or Wraparound
[25]	59.9	CWE-759	Use of a One-Way Hash without a Salt

Buffer overflow (overrun)

- Un programme permettant d'écrire des données au-delà de la limite d'un tableau:
 - Les données mises dans un buffer excède sa taille
 - Mettre des données dans un emplacement mémoire extérieur au buffer
 - Copie d'un buffer sans tester la taille de ce dernier (classic buffer overflow)
- Risques:
 - Meilleur cas: arrêt brutal de l'application.
 - Pire cas:
 - Exécution d'un code injecté durant le débordement de la zone mémoire
 - Prise de contrôle du système
- Langages:
 - Principalement: C et C++
 - D'autres langages peuvent aussi être vulnérables: Java, C# et VB

Buffer overflow: Exemples (langage C)

■ *Exemple 1:*

```
Char name[20];  
printf ("Enter your name: ");  
scanf ("%s", name);
```

➔ Pb: pas de restriction sur la taille du nom saisie

■ *Exemple 2:*

```
void copy (char *input)  
{ char buf[16];  
strcpy (buf, input);...}
```

➔ Pas de contrôle sur la taille de *input*

Buffer overflow (overrun)

■ *Exemple 3:*

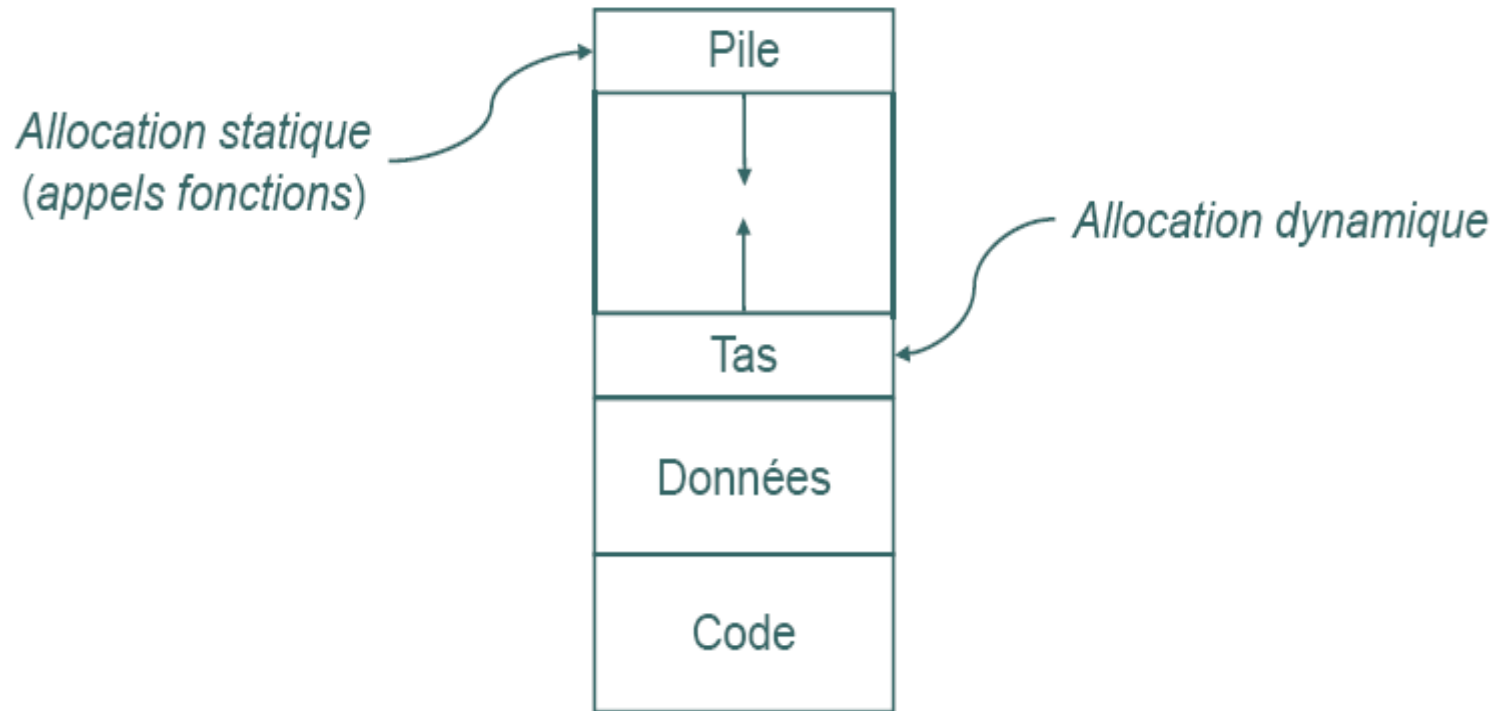
```
char dest[20], src[20];  
char prefix[] = "http://";  
strcpy (dest, prefix);  
strncat (dest, src, sizeof (dest));  
→ au lieu de  
strncat (dest, src, sizeof(dest)-sizeof(prefix));
```

■ *Exemple 4:*

```
char buf[20], data[32];  
strncpy (buf, data, strlen (data));  
→ au lieu de  
strncpy (buf, data, strlen (buf))
```

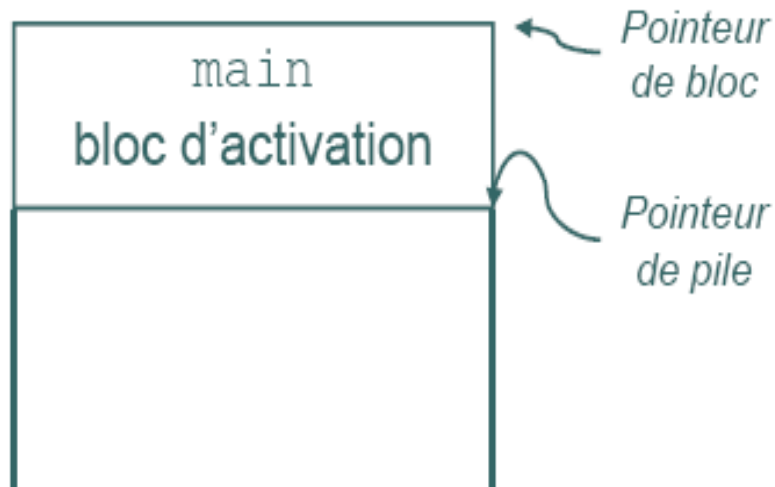
Buffer overflow: exploitation

- 2 classes d'exploitations:
 - Les débordements sur la pile (*Stack overflow – Stack smashing*).
 - Tableaux statiques
 - Les débordements sur le tas (*Heap overflow*).
 - Tableaux dynamiques

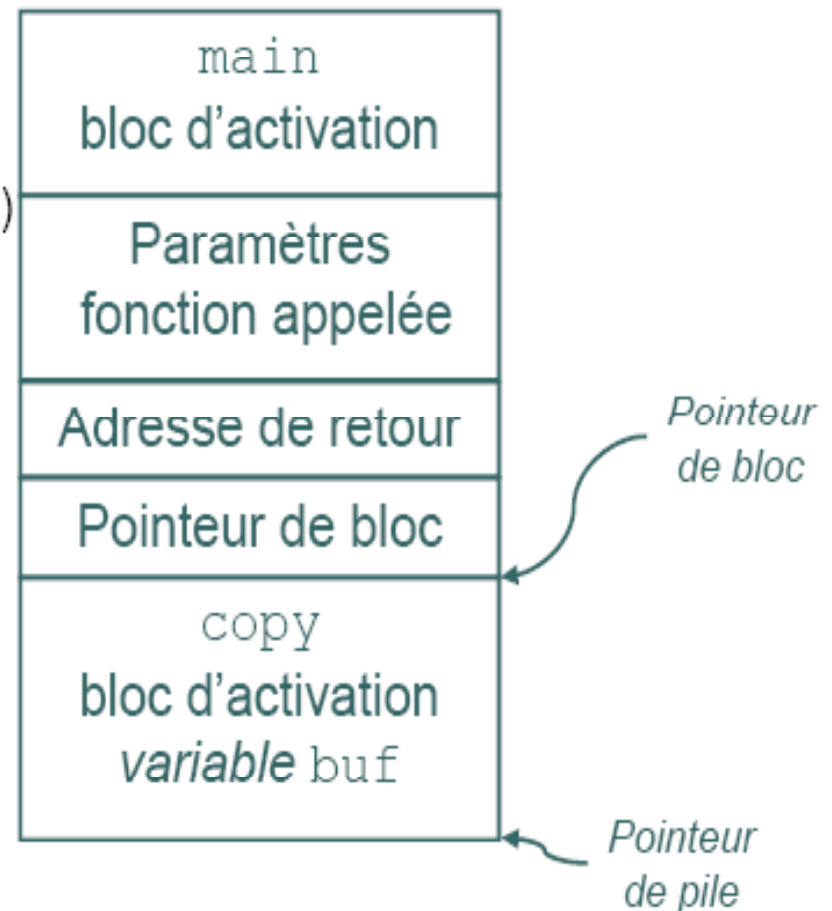


Stack overflow: exploitation

```
void copy (char *in)
{ char buf[16];
  strcpy (buf, input);}
int main (int argc, char *argv[])
{ copy (argv[1]);}
```



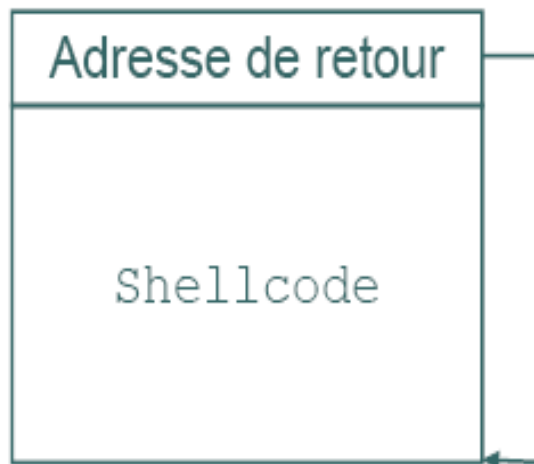
Avant l'appel



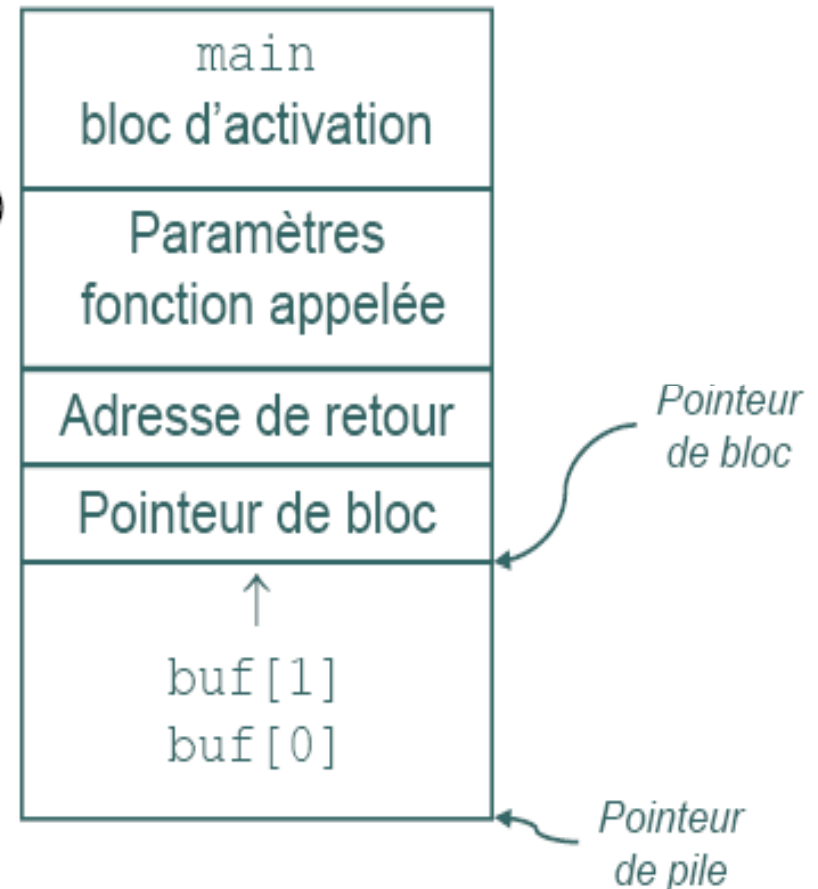
Après l'appel

Stack overflow: exploitation

```
void copy (char *in)
{ char buf[16];
  strcpy (buf, input);}
int main (int argc, char *argv[])
{ copy (argv[1]);}
```



Après le débordement



Avant le débordement

Buffer overflow: vulnérabilités

- CVE-2002-0649 – Utilisé par le ver **Slammer**.
 - Nombreux débordements de tableau (SQL Server 2000 Resolution Service) permettant à un attaquant distant de causer un **déni de service ou d'exécuter un code malicieux**.

- CVE-2003-0533 – Utilisé par le ver **Sasser**.
 - Débordement de tableau (Active Directory du Local Security Authority Subsystem Service (LSASS)) permettant à un attaquant distant **d'exécuter un code malicieux**.

- CVE-2010-2212
 - Débordement de tableau (Adobe Reader et Acrobat 9.x avant 9.3.3, et 8.x avant 8.2.3 pour Windows et Mac OS X) permettant à un attaquant distant de causer un **déni de service ou d'exécuter un code malicieux**.

SQL injection

■ Description

- De nombreuses applications utilisent des informations provenant d'un utilisateur pour construire leurs requêtes SQL. (nom, login, password...)
- Les données **ordinaires** fournies par les utilisateurs peuvent être interprété comme des requêtes SQL: présence de mauvais caractères tels que : ' ;)"#|

■ Risques:

- Contourner les vérifications de sécurité
- Obtenir des informations confidentielles.
- Modifier la BD,
- Exécuter des commandes systèmes

■ Types

- Injections SQL standards
- Injections SQL de requêtes d'union
- Injections SQL aveugles

Détection de la présence de vulnérabilité

- Comprendre quand l'application se connecte à un serveur de BD:
 - Formulaire d'authentification:
 - Il y a des chances que les informations d'identification sont comparées à une BD qui contient tous les comptes des utilisateurs
 - Moteurs de recherche:
 - les chaînes présentées par l'utilisateur peuvent être utilisées dans une requête SQL qui extrait tous les documents pertinents à partir d'une BD
 - Sites de Commerce électronique:
 - Les produits et leurs caractéristiques (prix, description, disponibilité, ...) sont très susceptibles d'être stockés dans une BD

Détection de la présence de vulnérabilité

- Stratégie:
 - Tester chaque champs séparément en utilisant différents chaines et récupérer les messages d'erreurs générées afin de les analyser.
 - ➔ Permet d'identifier avec précision les paramètres qui sont **vulnérables**

- Test de base:
 - l'ajout de mauvais caractères tels que : ' ;)"#|
 - (') : utilisée dans SQL comme une marque de fin de chaîne de caractères
 - (;): utilisé pour marquer la fin d'une instruction SQL.
 - (/*) et (*): balises de commentaires
 - En l'absence de filtres, ces caractères peuvent mener à l'exécution de requêtes incorrectes et/ou générer des erreurs

- Commencer par les injections standards

Injection SQL standard: exemples

■ Exemple 1:

- SQL permet l'exécution de **commandes shell**

```
SELECT ITEM,PRICE FROM PRODUCT WHERE  
ITEM_CATEGORY='$USER_INPUT' ORDER BY PRICE
```

- Si l'utilisateur fournit `$USER_INPUT = ';' exec master..xp_cmdshell 'dir' --`

- Le résultat sera:

1. `SELECT ITEM,PRICE FROM PRODUCT WHERE ITEM_CATEGORY='';`
2. `exec master..xp_cmdshell 'dir' --` (exécution de la cmde dir)
3. `'ORDER BY PRICE` (considéré comme commentaire)

Injection SQL standard: exemples

■ Exemple 2:

- Soit le script où \$login et \$password résultent des champs du formulaire d'authentification (page html, php, asp, cgi...)

```
$login = Request.Form("login")
```

```
$password = Request.Form("password")
```

```
SELECT field FROM users WHERE Login=$login AND Password=$password
```

→ Le script ne vérifie pas la présence de mauvais caractères tels que : ' ;)'#|

1) Si l'utilisateur fournit 'or'=' pour les deux variables, le résultat sera:

```
SELECT * FROM users WHERE Login=" OR "=" AND Password=" OR "'
```

→ 'or'="" retourne toujours VRAI

→ Si aucun autre contrôle n'est effectué → accès sous l'identité du premier utilisateur inscrit dans la BD, le plus souvent, **l'administrateur.**

Injection SQL standard: exemples

- 2) Gagner les droit d'accès d'un utilisateur particulier connaissant son nom de login ou son adresse email (soit user1) : saisir **user1** pour \$login et '**or**'="" pour \$password

La requête sera:

```
SELECT * FROM users WHERE Login='user1' AND Password= " OR "=
```

- 3) Ignorer une partie de la requête en utilisant /* et /* qui sont des balises de commentaire: saisir '/* pour \$login et /* **OR** '= pour \$password

La requête sera:

```
SELECT * FROM users WHERE Login = '/*' AND Password = '/* OR '=
```

Qui est equivalente à:

```
SELECT * FROM users WHERE Login = '' OR '' = ''
```

Injection SQL standard: exemple

■ Exemple 3 (PHP):

```
$ID = $_COOKIE["MID"];  
MYSQL_QUERY("SELECT MESSAGEID, SUBJECT FROM MESSAGES WHERE  
MESSAGEID = '$ID'");
```

- Si l'utilisateur modifie le COOKIE \$id = 1432' or '1' = '1'
- Le résultat sera:

```
SELECT MessageID, Subject FROM messages WHERE MessageID = '1432' or '1' = '1'
```

- ➔ Ce qui permet de récupérer tout les messages (1432 et tout les autres)
- ➔ Solution: \$id = intval(\$_COOKIE["mid"]);

Injection SQL standard: exemples

■ Exemple C#

- Chercher les items (**itemname**) dont le nom coïncide avec la chaîne saisie par l'utilisateur authentifié (**userName**)

```
string userName = ctx.getAuthenticatedUserName();  
string query = "SELECT * FROM items WHERE owner = '" +  
userName + "' AND itemname = '" + ItemName.Text + "'";
```

- La requête que ce code cherche à exécuter est

```
SELECT * FROM items WHERE owner = <userName> AND itemname =  
<itemName>;
```

- Si l'utilisateur wiley fournit name' OR 'a'='a' comme itemName, la requête sera:

```
SELECT * FROM items WHERE owner = 'wiley' AND itemname =  
'name' OR 'a'='a';
```

- Ce qui est équivalent à SELECT * FROM items; car 'a'='a' est toujours vrai

OS Command injection

■ Description

- Utilisation de données provenant d'un usager pour construire les commandes soumises au système d'exploitation.
- Les données utilisateurs peuvent contenir des éléments pouvant dévier la commande de son objectif

■ Exemple (php)

- Prendre le nom d'un utilisateur puis lister le contenu de son répertoire home

```
$userName = $_POST["user"];  
$command = 'ls -l /home/' . $userName;  
system($command);
```

-

- Si l'utilisateur fournit ;rm -rf / la commande sera ls -l /home;/rm -rf /
→ Permet de supprimer tout le système de fichier (la racine /)

Logiciels malveillants

Logiciels malveillants (malware)

- Développés pour des fins malicieuses sous diverses formes
 - Virus, Ver, Chevaux de Troie, Backdoors, Spywares, adware ...

- Exploitent
 - Les vulnérabilités logicielles: l'intervalle de temps entre la découverte d'une vulnérabilité et la disponibilité du remède.
 - La naïveté des usagers.

- Varient selon
 - Le mode d'exécution
 - Le mode de propagation
 - L'effet malicieux
 - ...

Virus

- **Forme:**
 - s'attache à un autre logiciel ou document
- **Mode d'exécution:**
 - exécuté suite à l'exécution du code hôte.
- **Propagation:**
 - transfert du programme / document hôte d'un système à un autre.
- **Reproduction:**
 - se reproduisent au sein du nœud infecté
 - peuvent infecter d'autres logiciels/documents
 - À l'aide des usagers
 - Grace à des vulnérabilités logicielles
- **Exemples:**
 - *ILoveyou, Melissa,*

Ver

- Semblable au virus (ver = virus réseau)
- Forme:
 - autonome (stand alone)
- Mode d'exécution:
 - **Automatique** ou par l'intervention de l'**usager** (ingénierie sociale)
 - Peut modifier l'OS hôte pour être lancé automatiquement
- Propagation:
 - **par le réseau** vers d'autres ordinateurs vulnérables.
 - Balayage de connexion TCP, courrier, messagerie instantanée (Obtient les informations réseau à partir du poste infecté)
- Reproduction:
 - Par lui-même (généralement grâce à une vulnérabilité logicielle)
- Exemples:
 - *Moris Worm, Slammer, Sasser, CodeRed, Blaster, ...*

Cheval de troie

- **Forme:**
 - souvent attachés manuellement au logiciel hôte (jeu, utilitaire...)
- **Mode d'exécution:**
 - exécutés en faisant partie d'un autre programme
- **Propagation :**
 - téléchargement de logiciels hôtes: inciter les utilisateurs à les télécharger (Ingénierie sociale)
- **Reproduction:**
 - Ne peuvent ni se reproduire ni infecter d'autres logiciels
- **Exemples:**
 - *Sub7, Back Orifice*, faux antivirus

Spyware (logiciel espion)

■ Forme:

- Tout logiciel qui contient un programme-espion
- emploie en arrière-plan la connexion Internet de l'utilisateur
- recueillir et transmettre des données personnelles (intérêts, habitudes de navigation..) à une régie publicitaire.

■ Mode d'exécution:

- Installé avec un logiciel populaire, un outil *shareware*, un faux *anti-spyware...etc*

■ Propagation:

- Ne se propage pas automatiquement.

■ Reproduction:

- Ne se reproduit pas.
- Ne nécessite pas de programme hôte à infecter.

Adware (logiciel publicitaire)

- **Forme:**
 - Logiciel gratuit affichant, lors de son utilisation, des annonces publicitaires menant à des sites commerciaux.
- **Propagation:**
 - Ne se propage pas automatiquement.
- **Reproduction:**
 - Ne se reproduit pas.
 - Ne nécessite pas de programme hôte à infecter.
- **Fonctions:**
 - Affiche des fenêtres publicitaires (pop-up)
 - Vole les informations personnelles (stockées dans l'ordinateur infecté).
 - Enregistre les habitudes de navigation web
 - Déroute certaines requêtes HTTP vers des sites commerciaux
 - ...

Conclusions



- Nécessité de bien comprendre le fonctionnement des logiciels malveillant:
 - Comment se reproduisent-ils?
 - Comment se propagent-ils?
 - Comment s'exécutent-ils?
 - Que font-ils exactement?

- Pour:
 - Déterminer les menaces qu'ils représentent
 - Evaluer les risques correspondants.
 - Mettre en place les mécanismes de protection adéquats
 - Déterminer l'efficacité des divers moyens de protection