# Recherche : Les Exploits

*Problématique : Fléau de l’informatique, les Exploits pourront-ils être éradiqués ?*

*Plan :*

1. *Introduction*
2. *Présentation*
   1. *Définitions*
   2. *Généralités*
   3. *Les différents types d’Exploit*

*2.3.x Exploit x : contexte, causes et conséquences*

1. *Expérimentation* 
   1. *Création d’un programme*
   2. *Mise en place d’une faille*
   3. *Utilisation d’un Exploit*
2. *Création d’un Exploit de test*
   1. *Les différentes étapes de construction*
   2. *Tester son Exploit*
3. *Les différentes protections*
   1. *Améliorer son code*
   2. *Les logiciels existant*

*5.2.x Logiciel x : présentation*

1. *Conclusion*

0-day : programme qui exploite une faille encore inconnue qui n'a pas encore été publié.

bugtraq c'est à dire des sites chargés de répertorier, l'ensemble des failles de sécurité connues

materiel.net

LamerZ

script kiddies :

Audits

MD5

Cascading Style Sheets

Javascript

Cooking

Pishing

buffer : (un buffer est un zone mémoire temporaire utilisée par une application)

# Introduction

L’expansion de l’informatique conduit de plus en plus de gens à développer leurs propres programmes, que ça soit pour une entreprise ou pour une utilisation personnel (ex : site web). Cette expansion est aussi une porte ouverte au piratage, de plus en plus courant en raison du nombre de développeurs qui augmente chaque année. Les Exploits, faisant partie de nos jours des logiciels malveillants les plus utilisés par les pirates, sont de plus en plus virulent.

Ce thème est très intéressant car il nous permettra, en tant que développeur, de comprendre les failles que l’on peut générer en programmant nos logiciels et comment y remédier. La sécurité logicielle étant une compétence de choix au vue des constatations dites précédemment. Il nous amènera également à nous poser la problématique suivante :

**Comment protéger nos programmes des Exploits ?**

Dans une première partie, nous verrons ce que sont réellement les exploits, leurs objectifs, leurs fonctionnement général. Comme pour les virus (Worm, chevaux de Troie etc.) il existe plusieurs types d’exploits. Par conséquent, nous aborderons les différences entre chacun, ainsi que leurs caractéristiques, de plus il sera expliqué pour chacun d’entre eux, les causes, conséquences et contexte dans lequel ils se situent.

Nous tenterons dans une seconde partie d’expérimenter le concept de « faille logicielle » en créant un programme simple et en y insérant l’une des failles qui sera détaillée dans la partie précédente. Nous pourrons donc par la suite tester un Exploit spécifique à la faille choisie.

Un développeur spécialisé dans la sécurité logicielle doit savoir exploiter les failles visibles d’un programme afin de pouvoir y remédier. Nous expliquerons donc dans cette troisième partie les différentes étapes de construction d’un Exploit  qui nous permettrons par la suite de créer le nôtre puis ensuite le tester.

Nous verrons dans une dernière partie des méthodes appropriées pour protéger son programme contre les Exploits. Nous expliquerons dans un premier temps les erreurs à ne pas commettre dans un code source, afin d’éviter l’ouverture d’une faille. Puis nous présenterons les différents logiciels permettant de lutter contre le piratage.

La résultante de cette dernière partie nous permettra de conclure si la protection de nos programmes contre les Exploits est actuellement envisageable où bien si celle-ci ne dépend que de la capacité du développeur à anticiper lors de son implémentation les différentes failles que son programme pourrait engendrer.

# Présentation

## Définition

Un Exploit est un programme informatique mettant en œuvre l’exploitation  d'une vulnérabilité lié à un logiciel. Chaque exploit est spécifique à une version d'une application car il permet d'en exploiter les failles. Ils sont utilisé la plus part du temps pour les raisons suivantes :

* Augmentation des privilèges : les Exploits les plus redoutables permettent de prendre le contrôle sur les programmes exécutés avec les privilèges d'administrateur (*root* sous les systèmes de type UNIX) ;
* Provocation d'une erreur système : certains Exploits ont pour objectif la saturation d'un programme informatique afin de le faire « planter ».

La plupart du temps les exploits sont écrits en langage C ou en Perl. Ils peuvent toutefois être écrits avec tout langage pour lequel il existe un interpréteur sur la machine cible. Le pirate utilisant un exploit doit ainsi posséder une connaissance minimale du système cible et des bases en programmation pour arriver à ses fins.

Afin de pouvoir l'utiliser, le pirate doit la plupart du temps le compiler sur la machine cible. Si l'exécution réussit le pirate peut, selon le rôle de l'exploit, obtenir un accès à l'interpréteur de commandes ([SHELL](http://www.commentcamarche.net/contents/unixx/unix-shell.php3)) de la machine distante.

## Généralités

L’exploit se voit attribué deux fonctions. Tout d’abord, il doit provoquer la faille de sécurité, par exemple pour une faille de type de buffer Overflow l'exploit devra envoyer le nombre de données adéquates afin que le dépassement de tampons se produise. Le Pirate écrivant un exploit se base le plus souvent sur les données techniques d'une faille de sécurité publié par des professionnels de la sécurité. Par exemple la découverte d'un dépassement de tampon permettant l'exécution de code arbitraire est un début pour le codage d'un exploit. Et c'est seulement après que l'exploit est réalisé, car les exploits 0-day sont très rares. Mais les experts de la sécurité publient aussi leur propre code source d'exploit suite à la découverte d'une faille de sécurité.

Toutefois certains exploit ne sont pas rendus publique, et reste au sein de ces professionnels de la sécurité qui travaillent le plus souvent en collaboration avec les grandes firmes de logiciels informatique. Il appartient à l'équipe de professionnels de publier ou non ce code malicieux, de le rendre public dans des Bugtraq.

Ce qui amène à se poser la question du full disclosure, c'est-à-dire : Doit on publier toutes les failles de sécurité, tous les codes écrits par les experts de la sécurité, les rendre visible au grand publique, et donc aux pirates, ou alors les laisser à des professionnels de la sécurité, et de ce fait laisser ces connaissances qu'à une seul catégorie de personne. Car même si les exploits sont dangereux entre de mauvaises mains (LamerZ, script kiddies, etc.), ils permettent néanmoins de faire des audits sur un système, et de savoir si celui-ci est vulnérable à cette faille de sécurité.

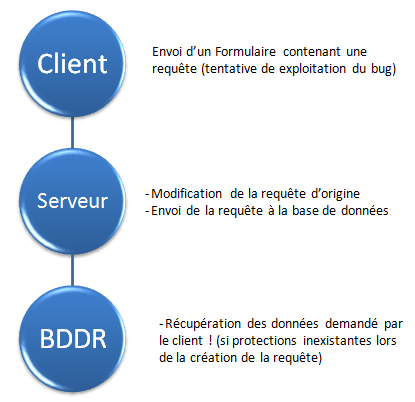
La deuxième fonction de l'exploit pour la majorité des cas consistera à faire injecter un code malicieux à la cible appelé. C'est lui qui permettra de prendre le contrôle de la machine attaqué. Le plus souvent un shellcode, va lancer un shell (C'est à dire un interpréteur de commandes: cmd.exe, sh, bash), qui va forcer l'ordinateur à se connecter au pc de l'attaquant en liant un shell lors de la connexion. Il va faire exécuter tout code qui pourrait viser à compromettre l'ordinateur cible, à en prendre le contrôle. Un shellcode doit être le plus petit possible, pour pouvoir s'exécuter en mémoire à l'endroit voulu lors de l'écrasement des données en mémoire (dans le cas d’un BoF).

## Les différents types d’Exploits

### Injection de commandes SQL

Les Blind SQL injections sont des attaques par injection de commandes SQL. Elles visent les sites web s'appuyant sur des bases de données relationnelles. Dans ce type de sites, des paramètres sont passés à la base de données sous forme d'une [requête SQL](http://www.commentcamarche.net/contents/sql/sqlintro.php3). Ainsi, si le concepteur n'effectue aucun contrôle sur les paramètres passés dans cette requête, il est possible à un pirate de modifier la requête afin d'accéder à l'ensemble de la base de données, voire à en modifier le contenu. Les injections SQL font partie des failles Web redoutables, puisqu'elles s'exploitent côté serveur

En effet, certains caractères permettent d'enchaîner plusieurs requêtes SQL ou bien ignorer la suite de la requête. Ainsi, en insérant ce type de caractères dans la requête, un pirate peut potentiellement exécuter la requête de son choix.



***Fonctionnement****:*

Considérons un site web qui dispose d'un système permettant aux utilisateurs possédant un nom d'utilisateur et un mot de passe valides de se connecter (site web marchand, forum, etc.). La base de données possède une table avec deux champs : login et password.

Supposons qu’un utilisateur quelconque veuille se connecter à son compte. La requête transmise à la base de données serait de la forme suivante :

SELECT utilisateurs WHERE login = ‘Chauvelin’ AND password = ‘philippe’;

Imaginons à présent que le script PHP exécutant cette requête ne vérifie pas les données entrantes pour garantir sa sécurité. Un [pirate informatique](http://fr.wikipedia.org/wiki/Pirate_informatique) pourrait alors fournir les informations suivantes :

* Utilisateur : Chauvelin' /\*
* Mot de passe : n'importe lequel

La requête devient :

SELECT utilisateurs WHERE login = 'Chauvelin' /\* ' AND password = ‘philippe’;

En effet, les caractères /\*(ou les caractères -- selon la base de données) marquent le début d'un [commentaire](http://fr.wikipedia.org/wiki/Commentaire_%28informatique%29) en SQL. La requête est donc équivalente à :

SELECT utilisateurs WHERE login = ‘Chauvelin’;

Le pirate peut alors se connecter sous l'utilisateur Chauvelin avec n'importe quel mot de passe. Il s'agit d'une injection de SQL réussie, car le pirate est parvenu à injecter les caractères qu'il voulait pour modifier le comportement de la requête.

Supposons maintenant que le pirate veuille non pas tromper le [script](http://fr.wikipedia.org/wiki/Script) SQL sur le nom d'utilisateur, mais sur le mot de passe. Il pourra alors [injecter](http://fr.wikipedia.org/w/index.php?title=Injecter&action=edit&redlink=1) le code suivant :

* Utilisateur : Chauvelin
* Mot de passe : ' or 1=1/\*

L'apostrophe indique la fin de la zone de frappe de l'utilisateur, le code "or 1=1" demande au script si 1=1 est vrai, or c'est toujours le cas, et /\* indique le début d'un commentaire.

La requête devient alors :

SELECT utilisateurs WHERE login = 'Chauvelin' AND password = ' ' OR 1=1 /\* ‘philippe';

Ce qui donne finalement :

SELECT utilisateurs WHERE login = 'Chauvelin' AND password = ' ' OR 1=1;

Ainsi, une fois la requête envoyée à la base de données, le script php verra que le mot de passe ne correspond pas au login mais que 1=1 (qui est toujours vrai), par conséquent il autorisera l’accès avec ce login.

***Conclusion* :**

L’injection de commandes SQL appelée aussi *Blind SQL Injection*, est donc un Exploits extrêmement dangereux pour tous les sites disposant d’espaces privés, comme par exemple les sites WEB marchands où chaque client possède son espace personnel avec ses coordonnées. Cet exploit est d’autant plus inquiétant si les coordonnées bancaires sont enregistrées sur le compte client, ce qui est le cas sur certain site marchand (materiel.net, corrigé depuis 1an).

***Nota*** : Cette faille a été testé (voir Annexe 1) localement avec easy php v1.6, v1.8 et v5.3.2 en PHP v4 et v5.

### Cross-Site Scripting (XSS)

Les exploits de type XSS sont des attaques visant les sites web qui affichent dynamiquement du contenu coté client et ce sans effectuer de contrôle des informations saisies par les utilisateurs. Les exploits Cross-Site Scripting consistent donc à forcer un site web à afficher du code HTML ou des scripts non autorisé et saisis par les utilisateurs. Ce type d’exploit est très similaire aux *Blind SQL injection*, à la différence qu’ici le XSS n’agit pas sur le serveur mais sur le client.

Il arrive souvent que les sites web affichent des messages d'information reprenant un paramètre précisé par l'utilisateur comme par exemple l’affichage de la page « pages d'erreur 404 ». Certains sites modifient le comportement du site web, afin d'afficher un message d'erreur personnalisée lorsque la page demandée par le visiteur n'existe pas. Parfois la page générée dynamiquement affiche le nom de la page demandée. Appelons *http://site.vulnerable* un site possédant une telle faille. L'appel de l'URL *http://site.vulnerable/page-inexistante* correspondant à une page n'existant pas provoquera l'affichage d'un message d'erreur indiquant que la page « page-inexistante » n'existe pas. Il est ainsi possible de faire afficher ce que l'on souhaite au site web en remplaçant « page-inexistante » par toute autre chaîne de caractère.

Ainsi, si aucun contrôle n'est effectué sur le contenu fourni par l'utilisateur, il est possible d'afficher du code HTML arbitraire sur une page web, afin d'en changer l'aspect, le contenu ou bien le comportement.

De plus, la plupart des navigateurs sont dotés de la capacité d'interpréter des scripts contenus dans les pages web, écrits dans différents langages, tel que [JavaScript](http://www.commentcamarche.net/contents/javascript/jsintro.php3), [VB Script](http://www.commentcamarche.net/contents/vbscript/vbsintro.php3), [Java](http://www.commentcamarche.net/contents/java/java.php3), ActiveX ou Flash. Les balises HTML suivantes permettent ainsi d'incorporer des scripts exécutables dans une page web : <SCRIPT>, <OBJECT>, <APPLET>, and <EMBED>.

Il est ainsi possible à un pirate d'injecter du code arbitraire dans la page web, afin que celui-ci soit exécuté sur le poste de l'utilisateur dans le contexte de sécurité du site vulnérable. Pour ce faire, il lui suffit de remplacer la valeur du texte destiné à être affiché par un script, afin que celui s'affiche dans la page web. Pour peu que le navigateur de l'utilisateur soit configuré pour exécuter de tels scripts, le code malicieux a accès à l'ensemble des données partagées par la page web de l'utilisateur et le serveur (cookies, champs de formulaires, etc.).

### Buffer Overflow (BoF)

Un buffer overflow est une attaque très efficace et assez compliquée à réaliser. Elle vise à exploiter une faille, une faiblesse dans une application (type browser, logiciel de mail, etc.) pour exécuter un code arbitraire qui compromettra la cible (acquisition des droits administrateur, etc.).

Le fonctionnement général d'un buffer overflow est de faire « crasher » un programme en écrivant dans un buffer plus de données qu'il ne peut en contenir, dans le but d'écraser des parties du code de l'application et d'injecter des données utiles pour exploiter le crash de l'application.

L'intérêt de ce type d'attaque est qu'il ne nécessite pas le plus souvent d'accès au système, ou alors un accès restreint suffit. D'un autre côté, il reste difficile à mettre en œuvre car il requiert des connaissances avancées en programmation; de plus, bien que les nouvelles failles soient largement publiées sur le web, les codes ne sont pas ou peu portables. Une attaque par buffer overflow signifie en général que l'on a affaire à des attaquants doués, pratiquant couramment la programmation plutôt qu'à des scripts kiddies.

Conclusion

Dans le domaine de la sécurité informatique, un exploit est un programme permettant à un individu d'exploiter une faille de sécurité informatique dans un système d'exploitation ou un logiciel que ce soit à distance (remote exploit) ou sur la machine sur laquelle cet exploit est exécuté (local exploit), ceci, afin de prendre le contrôle d'un ordinateur, ou encore d’accéder à des données privées.

Nous avons vu dans ce mémoire les différents types d’exploits existant, leurs fonctionnement, le contexte dans lequel ils s’appliquent et comment y remédier.

/\*

Les différents exploits

\*/

En effet, il existe des logiciels permettant d’éviter ce genre de désagrément ; comme par exemple

/\*

Logiciel x : faille x : contexte

\*/