

Conférences de l'ASIQ
10 février 2010

Victrix
4 secteurs d'intervention
Privilégiés

Sécurité
Solutions Applicatives
Solutions d'Infrastructure
Réseaux & Télécommunication

Patrick Chevalier
CISSP, CISA, CSSLP, GIAC
GSEC, CEH, SEC+

Conseiller senior en sécurité
pchevalier@victrix.ca

Les failles de logique dans les applications Web

Identification et exploitation

- **Patrick Chevalier**, CISSP, CISA, CSSLP, GIAC, CEH
- Conseiller en sécurité de l'information
- Plus de dix (10) ans d'expérience en sécurité
- Membre de l'OWASP et de l'ISACA
- Deux (2) principaux domaines de spécialisation :
 - Audit et tests d'intrusion (applications, systèmes, infra)
 - Sécurité applicative et développement sécuritaire
- Formateur en sécurité applicative

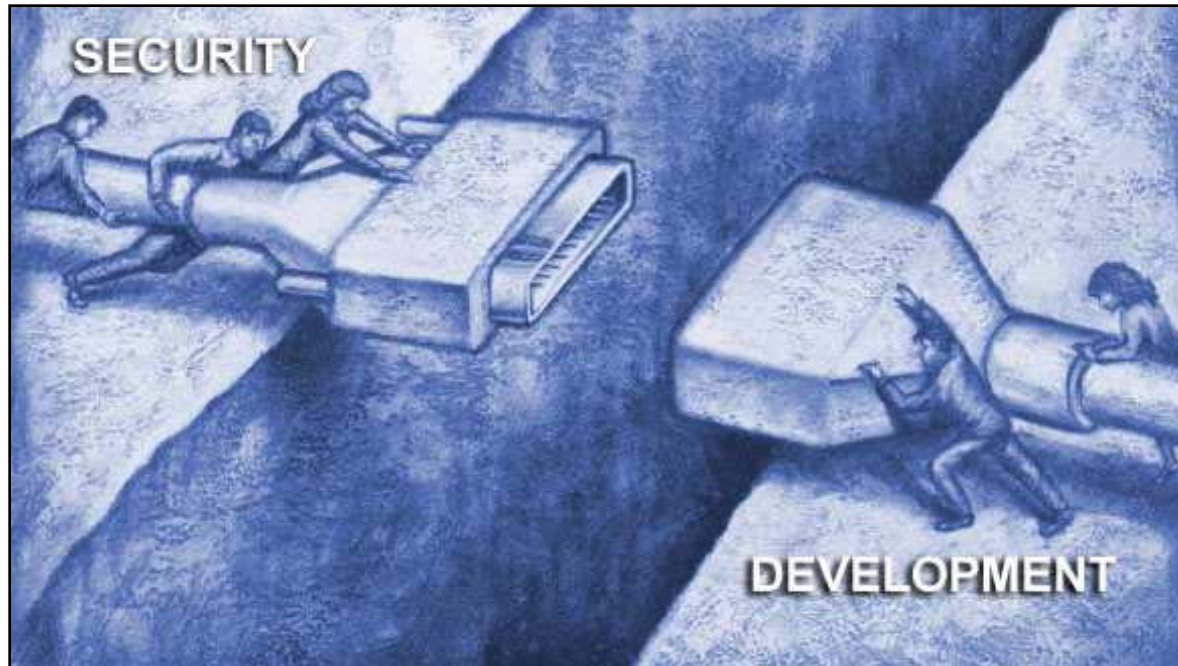
- Objectifs de la présentation
- Introduction et mise en contexte
- Qu'est-ce que la logique d'affaires ?
- Failles techniques vs. Failles de logique
- Exemples et explications
- Réduction des risques
- Période de questions

Objectifs de la présentation

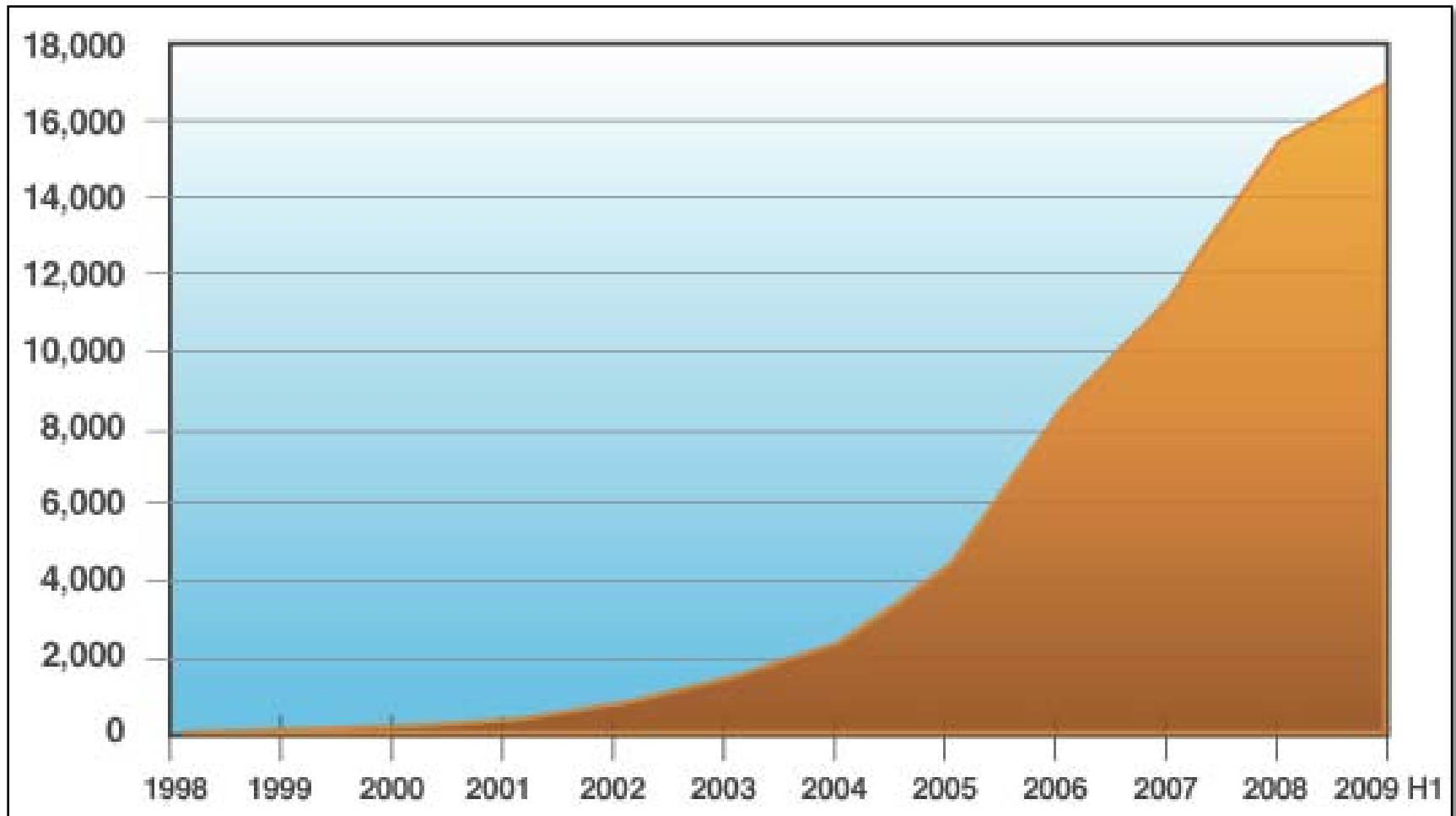
- Vous sensibiliser aux problèmes affectant la sécurité des applications et des sites Web
- Démontrer les différences fondamentales entre les « failles techniques » et les « failles de logique » à l'aide d'exemple concret
- Fournir des pistes de solutions et des recommandations générales permettant de minimiser les risques

Le « application security gap »

Constat : Très peu de professionnels de la sécurité possèdent des connaissances en développement et inversement, très peu de développeurs possèdent des connaissances en sécurité.

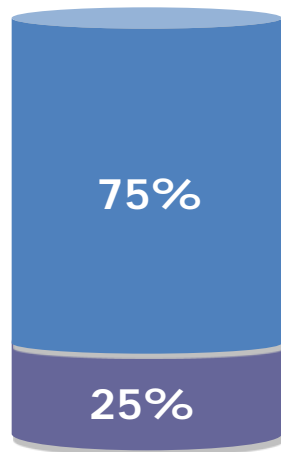


Vulnérabilités au niveau des applications Web



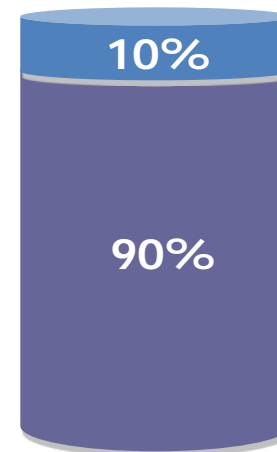
Risques

% des attaques



Budget

% du montant



"75% of All Attacks on Information Security are Directed to the Web Application Layer"
(Source : Gartner)

Le talon d'Achille de la sécurité

InformationWeek

Web Applications: Achilles' Heel Of Corporate Security

Feb 2009

Custom-built software is more likely to garner an online attack and less likely to be disclosed in bug reports, IBM reveals.

Last year, 55% of all the computer security vulnerabilities disclosed affected Web applications, and 74% of these had no patch.

So says IBM in its 2008 X-Force Trend and Risk report, released Monday, which paints a dire picture of online computer security.

More Security Insights
White Papers
Simplifying Network Security with a Single Source Provider
Federal CTO Agenda: The Industry's Advice to President Obama
Webcasts
Gone in 6.0 Seconds: Protecting Laptops and Data from Theft
Why Bad Security Breaches Keep Happening To Good Organizations
Reports
Cybersecurity
Balancing Act
Take It From The Top
Videos

SonicWall VP touts rapid growth of networking and security sectors as driving strong activity at Interop. "Certain types of corporate applications, namely custom-built software like Web applications, remain a highly profitable and inexpensive target for criminal attackers," the report states. "The sheer number of new vulnerabilities, the majority of which have no available patch, coupled with the hundreds of thousands of custom Web applications that are also vulnerable (but never subject to a vulnerability disclosure, much less a patch), continue to be the Achilles' heel of corporate security."

The risk IBM (NYSE: IBM) sees is that as legitimate sites become compromised, customer trust becomes collateral damage.

Qu'est-ce que la logique d'affaires ?

La « **logique d'affaires** » est un terme généralement utilisé pour décrire les aspects fonctionnels (règles d'affaires, politiques d'affaires et « workflows ») qui gèrent les échanges d'informations entre une source de données (ex: base de données, annuaire d'entreprise) et l'interface utilisateur.

Source: http://en.wikipedia.org/wiki/Business_logic

- Toute application ou site Web est potentiellement vulnérable
 - ...d'une façon qui lui est propre !
 - ...le nombre de variations est infini !
- Souvent présentes au niveau de fonctionnalités critiques, ex:
 - Authentification et autorisation,
 - Gestion de session et récupération de mots de passe
 - Paiement et transaction en ligne

- Avec la complexification des applications, les failles de ce type sont de plus en plus fréquentes
 - « Web 2.0 », fonctionnalités AJAX, etc.
- Autres problèmes...
 - Les outils d'analyse de vulnérabilités sont incapables de les identifier
 - Les systèmes de détection d'intrusion (IDS) ne les détectent pas
 - Les pare-feu applicatifs (WAF) ne les bloquent pas

Faille technique vs. Faille de logique

24 classes de vulnérabilités du WASC

Analyse manuelle

Failles de logique

- **Authentification**
 - Brute-Force
 - Authentification insuffisante
 - Énumération et force brute
- **Autorisation et contrôle d'accès**
 - Prédiction de session
 - Autorisation insuffisante
 - Expiration de session insuffisante
 - Fixation de session
- **Logique applicative**
 - Abus de fonctionnalités
 - Déni de service
 - « Antiautomatisation » insuffisante
 - Processus de validation insuffisant

Analyse automatisée (scanneur)

Failles techniques

- **Exécution de commande**
 - Débordement de Tampon
 - « Format String » attack
 - LDAP Injection
 - Exécution de commande OS
 - Injection SQL
 - « Server-Side include » (SSI)
 - Injection Xpath
- **Divulgence d'information**
 - Indexation de répertoire
 - Fuite d'information
 - « Directory Traversal »
 - Location de ressource prédictible
- **Côté-Client**
 - Spoofing de contenu
 - Cross-site Scripting

Faille technique vs. Faille de logique

Faille « technique »	Faille « logique »
Requêtes malformées	Requête d'apparence légitime
Entrant utilisateur illégal	Entrant utilisateur légitime
Souvent une seule requête	Généralement plusieurs requêtes
Modification d'une fonctionnalité	Abus d'une fonctionnalité
Dompage direct à l'application	Dompage direct à l'organisation
Dompage indirect à l'organisation	Dompage indirect à l'application

Les failles de logique (2)

- Difficile à identifier lors de l'assurance qualité
 - traditionnellement, l'assurance qualité valide le comportement « normal » d'utilisateur et de l'application
- Nécessite une compréhension intime du fonctionnement et de la logique de l'application
 - Contrairement aux failles techniques, la majorité des failles de logiques sont hors de portée des outils d'analyses automatisés.
 - L'outil automatisé n'est pas en mesure de comprendre la logique d'une application, il ne voit qu'une simple série de formulaires et de paramètres.
- Ne nécessite pas nécessairement une compréhension des aspects technologiques
 - Souvent, une simple permutation ou modification manuelle de la valeur d'une variable clé est nécessaire
- L'exploitation nécessite généralement d'automatiser certaines requêtes ou processus

Flaw In Sears Website Left Database Open To Attack

Business-logic flaw in Sears.com Web application could have let hackers brute-force attack the retailer's gift card database

Sep 01, 2009 | 03:49 PM

A newly discovered vulnerability on Sears.com could have allowed attackers to raid the retail giant's gift card database.

Alex Firmani, owner of Merge Design and a researcher, this week revealed a major security hole on Sears.com that could allow an attacker to easily steal valid gift cards -- a heist he estimates could be worth millions of dollars. Firmani says he alerted Sears about the flaw, and that Sears has since "plugged" the hole by removing the feature that let customers verify and check their gift-card balances.

The vulnerability was a business logic flaw in a Web application that handles gift card account inquiries; Firmani was able to stage a brute-force attack that could grab all valid, active Sears and Kmart gift cards from the company's database.

Firmani says the site wasn't auditing verification requests, which allowed him to verify gift card and PIN combinations using a homegrown PHP script that automatically submitted the requests. "I wrote a PHP script to hammer their verification server. It happily replied with thousands of verification responses per minute," he says.

The Sears application relied on client-side cookies to halt brute-force verification attempts, which Firmani says wasn't effective. "They should know where the verification requests come from, log them all, and be able to disable the verifications when they have a malicious attack," he says. "It doesn't appear to me that they had any server-side control over how many verifications were done."

...brute-force
attack the
retailer's
database...

...Heist could be
worth millions of
dollars...

...Business logic
flaw in a web
application that
handles gift cards
accounts...

...wrote a script to
hammer their
verification
server...

WIRED

Man Allegedly Bilks E-trade, Schwab of \$50,000 by Collecting Lots of Free 'Micro-Deposits'

By [Kevin Poulsen](#) May 27, 2008

A California man has been indicted for an inventive scheme that allegedly siphoned \$50,000 from online brokerage houses E-trade and Schwab.com in six months — a few pennies at a time.

Michael Largent, 22, of Plumas Lake, California, allegedly exploited a loophole in a common procedure both companies follow when a customer links his brokerage account to a bank account for the first time. To verify that the account number and routing information is correct, the brokerages automatically send small "micro-deposits" of between two cents to one dollar to the account, and ask the customer to verify that they've received it.

Michael Largent allegedly used a script to open 58,000 online brokerage accounts in the names of cartoon characters, and other aliases.

Hank Hill courtesy Fox Broadcasting Largent allegedly used an automated script to open 58,000 online brokerage accounts, linking each of them to a handful of online bank accounts, and accumulating thousands of dollars in micro-deposits.

I know it's only May, but I think the competition for Threat Level's Caper of the Year award is over.

Largent's script allegedly used fake names, addresses and Social Security numbers for the brokerage accounts. Largent allegedly favored cartoon characters for the names, including Johnny Blaze, King of the Hill patriarch Hank Hill, and Rusty Shackelford. That last name is doubly-fake — it's the alias commonly used by the paranoid exterminator Dale Gribble on King of the Hill.

...siphoned
50,000\$ from
online brokerage
houses...

...used a script to
open 58,000
accounts...

...favored
cartoon
characters for the
names...

- **Client:** Courtier en assurances
- **Catégorie:** Abus de fonctionnalité / Fuite d'information
- Le formulaire d'authentification de l'Extranet utilisé par un grand nombre de représentants renvoie un message différent en fonction de différentes conditions:
 - Utilisateur connu: « Votre mot de passe est incorrect »
 - Utilisateur inconnu: « Nom d'utilisateur invalide »
- Des variations étaient également présentes dans le format de la page HTML.

intruder attack 3

attack save view

request	payload	status	error	timeo...	length	valid
1	0667	200	<input type="checkbox"/>	<input type="checkbox"/>	18147	<input type="checkbox"/>
2	0586	302	<input type="checkbox"/>	<input type="checkbox"/>	9719	<input type="checkbox"/>
3	0745	302	<input type="checkbox"/>	<input type="checkbox"/>	9719	<input type="checkbox"/>
4	0923	200	<input type="checkbox"/>	<input type="checkbox"/>	20420	<input type="checkbox"/>
5	0874	302	<input type="checkbox"/>	<input type="checkbox"/>	9719	<input type="checkbox"/>
6	0502	302	<input type="checkbox"/>	<input type="checkbox"/>	9719	<input type="checkbox"/>
7	0439	200	<input type="checkbox"/>	<input type="checkbox"/>	20527	<input type="checkbox"/>
8	0816	302	<input type="checkbox"/>	<input type="checkbox"/>	9719	<input type="checkbox"/>
9	0636	302	<input type="checkbox"/>	<input type="checkbox"/>	9719	<input type="checkbox"/>
10	0385	302	<input type="checkbox"/>	<input type="checkbox"/>	9719	<input type="checkbox"/>
11	0055	200	<input type="checkbox"/>	<input type="checkbox"/>	19385	<input type="checkbox"/>
12	0662	302	<input type="checkbox"/>	<input type="checkbox"/>	9719	<input type="checkbox"/>
13	0672	200	<input type="checkbox"/>	<input type="checkbox"/>	17605	<input type="checkbox"/>
14	0932	302	<input type="checkbox"/>	<input type="checkbox"/>	9719	<input type="checkbox"/>
15	0983	302	<input type="checkbox"/>	<input type="checkbox"/>	9719	<input type="checkbox"/>
16	0815	302	<input type="checkbox"/>	<input type="checkbox"/>	9719	<input type="checkbox"/>
17	0505	302	<input type="checkbox"/>	<input type="checkbox"/>	9719	<input type="checkbox"/>
18	0163	200	<input type="checkbox"/>	<input type="checkbox"/>	18317	<input type="checkbox"/>
19	0886	302	<input type="checkbox"/>	<input type="checkbox"/>	9719	<input type="checkbox"/>
20	0370	302	<input type="checkbox"/>	<input type="checkbox"/>	9719	<input type="checkbox"/>
21	0981	302	<input type="checkbox"/>	<input type="checkbox"/>	9719	<input type="checkbox"/>
22	0452	302	<input type="checkbox"/>	<input type="checkbox"/>	9719	<input type="checkbox"/>
23	0527	200	<input type="checkbox"/>	<input type="checkbox"/>	17843	<input type="checkbox"/>
24	0207	200	<input type="checkbox"/>	<input type="checkbox"/>	9719	<input type="checkbox"/>
25	0961	302	<input type="checkbox"/>	<input type="checkbox"/>	9719	<input type="checkbox"/>
26	0312	302	<input type="checkbox"/>	<input type="checkbox"/>	9719	<input type="checkbox"/>
27	0551	302	<input type="checkbox"/>	<input type="checkbox"/>	9719	<input type="checkbox"/>
28	0348	200	<input type="checkbox"/>	<input type="checkbox"/>	21778	<input type="checkbox"/>
29	0859	302	<input type="checkbox"/>	<input type="checkbox"/>	9719	<input type="checkbox"/>
30	0618	200	<input type="checkbox"/>	<input type="checkbox"/>	29585	<input type="checkbox"/>
31	0886	302	<input type="checkbox"/>	<input type="checkbox"/>	9719	<input type="checkbox"/>
32	0618	302	<input type="checkbox"/>	<input type="checkbox"/>	9719	<input type="checkbox"/>
33	0256	302	<input type="checkbox"/>	<input type="checkbox"/>	9719	<input type="checkbox"/>
34	0414	302	<input type="checkbox"/>	<input type="checkbox"/>	9719	<input type="checkbox"/>
35	0243	302	<input type="checkbox"/>	<input type="checkbox"/>	9719	<input type="checkbox"/>

finished

En analysant les réponses retournées par le serveur Web, l'attaquant est en mesure d'utiliser cette information afin d'énumérer les comptes utilisateurs valides présents au niveau de l'application

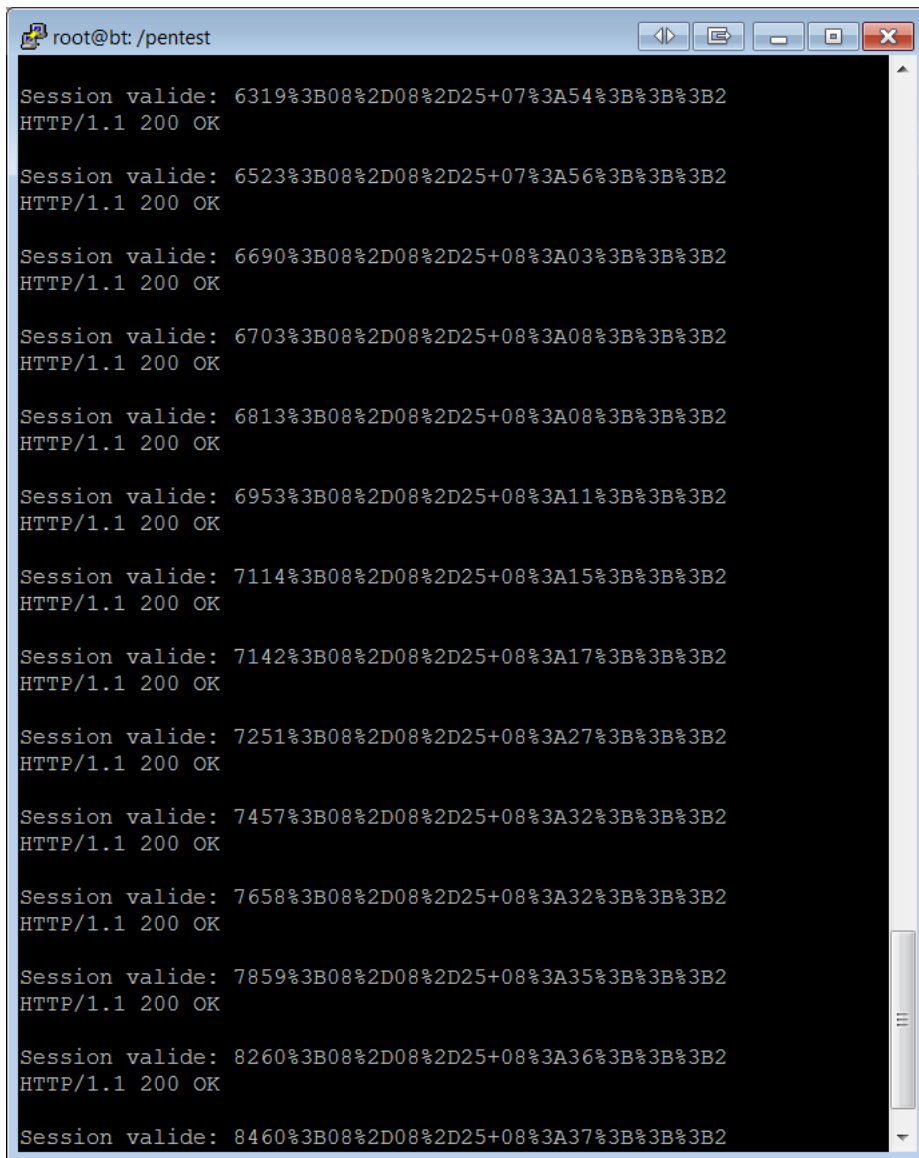
Identification de plus de **300 comptes** utilisateurs

Une simple attaque de dictionnaire sur les comptes identifiés nous a permis d'**accéder à plus de 50 comptes.**

- **Client:** Ministère
- **Catégorie:** Abus de la fonctionnalité (gestion des sessions)
- L'application web utilise un algorithme de génération de session utilisateur développé à l'interne, un jeton de session type est représenté comme suit:
 - **11330%3B08%2D08%2D25+16%3A12%3B%3B%3B2**
- Notez la présence de caractères hexadécimaux (indiqué par %) qui une fois décodés, nous ont permis d'obtenir ce résultat :
 - **11330;08-08-25+16:12;;;2**

- Le jeton de session est constitué de trois parties distinctes:
 1. Séquence numérique incrémentielle correspondant au numéro de session
 2. Date et heure auxquelles fut émis le jeton de session
 3. Valeur numérique inconnue pouvant prendre une valeur limitée
- Côté serveur, les sessions sont configurées pour expirer après un délai de 48 heures.
- Un grand nombre de sessions utilisateurs sont initiées entre 8:00 et 9:00 AM...

Génération des cookies et validation à l'aide d'un script automatisé (Perl LWP):



```
root@bt: /pentest
Session valide: 6319%3B08%2D08%2D25+07%3A54%3B%3B%3B2
HTTP/1.1 200 OK
Session valide: 6523%3B08%2D08%2D25+07%3A56%3B%3B%3B2
HTTP/1.1 200 OK
Session valide: 6690%3B08%2D08%2D25+08%3A03%3B%3B%3B2
HTTP/1.1 200 OK
Session valide: 6703%3B08%2D08%2D25+08%3A08%3B%3B%3B2
HTTP/1.1 200 OK
Session valide: 6813%3B08%2D08%2D25+08%3A08%3B%3B%3B2
HTTP/1.1 200 OK
Session valide: 6953%3B08%2D08%2D25+08%3A11%3B%3B%3B2
HTTP/1.1 200 OK
Session valide: 7114%3B08%2D08%2D25+08%3A15%3B%3B%3B2
HTTP/1.1 200 OK
Session valide: 7142%3B08%2D08%2D25+08%3A17%3B%3B%3B2
HTTP/1.1 200 OK
Session valide: 7251%3B08%2D08%2D25+08%3A27%3B%3B%3B2
HTTP/1.1 200 OK
Session valide: 7457%3B08%2D08%2D25+08%3A32%3B%3B%3B2
HTTP/1.1 200 OK
Session valide: 7658%3B08%2D08%2D25+08%3A32%3B%3B%3B2
HTTP/1.1 200 OK
Session valide: 7859%3B08%2D08%2D25+08%3A35%3B%3B%3B2
HTTP/1.1 200 OK
Session valide: 8260%3B08%2D08%2D25+08%3A36%3B%3B%3B2
HTTP/1.1 200 OK
Session valide: 8460%3B08%2D08%2D25+08%3A37%3B%3B%3B2
```

À partir de ces informations, l'attaquant est en mesure de créer une liste de valeurs de cookie potentiellement attribué à d'autres utilisateurs au cours d'une période fixe.

Utilisation d'un script automatisé afin de vérifier la validité des cookies en analysant les réponses retournées par le serveur Web (Code 200 pour une session valide et Code 302 pour une session invalide).

Cette opération nous a permis d'accéder à **plus de 10 sessions** utilisateurs.

- **Client:** Institution financière
- **Catégorie:** Abus de fonctionnalité (récupération de mot de passe)
- Formulaire de changement de mot de passe classique, les administrateurs disposaient également d'une fonctionnalité leur permettant d'effectuer un reset de tout les comptes utilisateurs, et ce, sans disposer du mot de passe.
- Les deux fonctionnalités étaient présentes au niveau du même script, voici le pseudo-code:

```
String UserPassword = request.getParameter("UserPassword");  
if (UserPassword == null) {  
    message("Aucun mot de passe spécifié, accès administrateur");  
    return true;  
}  
else {  
    message("Verification du mot de passe existant");  
    ...  
}
```

- Requête de changement de mot de passe typique:

request			
raw	params	headers	hex
GET request to /intranet/password-reset.jsp			
type	name	value	
cookie	PREF	ID=94f4390237c0db14::LD=fr:CR=2	
body	UserName	test_victrix	
body	UserPassword	Old1234	
body	NewPassword	New1234	
body	ConfirmPassword	New1234	

- L'attaquant est en mesure d'intercepter la requête et de la modifier manuellement... il suffit de supprimer le paramètre:

request			
raw	params	headers	hex
GET request to /intranet/password-reset.jsp			
type	name	value	
cookie	PREF	ID=94f4390237c0db14::LD=fr:CR=2	
body	UserName	Administrateur	
body	NewPassword	New1234	
body	ConfirmPassword	New1234	

- Aucun contrôle ne valide si l'utilisateur effectuant la requête dispose réellement des privilèges administrateurs

- **Client:** Détaillant en ligne
- **Catégorie:** Abus de fonctionnalité (processus de paiement)
- Site web transactionnel, offre la possibilité d'effectuer des commandes en ligne, opération effectuée en quatre étapes:
 1. Parcoure du catalogue et ajout d'items au panier de l'utilisateur
 2. Confirmation des items au panier (type et quantité)
 3. Saisie des informations de paiements (numéro de carte de crédit)
 4. Saisie de l'adresse de livraison et confirmation de la commande
- **L'hypothèse:** Les utilisateurs sont forcés de respecter l'ordre logique des étapes proposées par le site web.

- **L'attaque:** Puisqu'il contrôle chaque requête effectuée à l'application, l'utilisateur est en mesure de passer directement de l'étape #2 à l'étape #4...
 1. Parcoure du catalogue et ajout d'items au panier de l'utilisateur
 2. Confirmation des items au panier (type et quantité)
 3. Saisie des informations de paiements (numéro de carte de crédit)
 4. Saisie de l'adresse de livraison et confirmation de la commande

Votre commande a bien été enregistrée.

Merci d'avoir choisi XXX pour l'achat de vos XXX professionnels et techniques. Votre commande a été enregistrée sous le numéro XXX.

Un message e-mail de confirmation a aussi été envoyé ...

- Autres exemples:
 - Modifier son ID utilisateur permet d'effectuer des actions sous une autre identité
 - Contournement des contrôles d'accès en appelant directement les fonctionnalités JavaScript (AJAX)
 - Modifier le ID de facture permet de consulter les données des autres utilisateurs
 - Modifier l'en-tête « Referer » permet de contourner l'authentification et d'accéder directement à la section « privée » d'un site
 - Modifier manuellement le montant d'un item permet d'obtenir un rabais

- L'identification des failles de logique n'est pas une tâche simple
 - Contrairement aux failles techniques, il n'y a pas de « patterns » spécifiques
 - Les outils d'analyses automatisés sont pratiquement inefficaces
 - Nécessite une compréhension intime du fonctionnement de l'application et de la logique d'affaires
- Les contrôles de sécurité traditionnels sont inefficaces
- L'exploitation des failles de logique a souvent un impact de sécurité très important, voir critique

Identifier les failles de logique

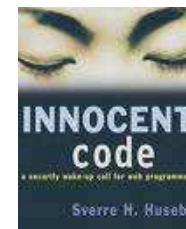
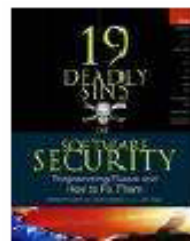
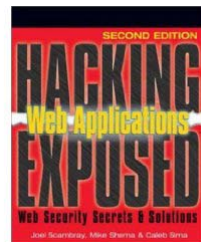
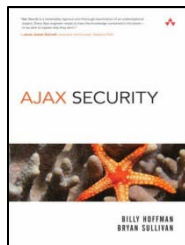
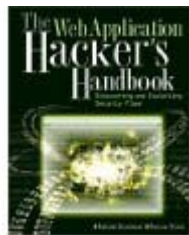
- **Utilisation du « OWASP Testing Guide » (OWASP-BL-001), ex:**
- **Usurper d'autres utilisateurs** (Modification manuelle de paramètres spécifiques (ID d'utilisateur, nom d'utilisateur, ID de session, chemin d'accès, valeur des cookies, question secrète, etc.)
- **Modifier des valeurs spécifiques** (Modification manuelle de paramètres spécifiques (Prix de produit, carte de crédit de test, numéro de compte, date d'expiration, etc.)
- **Inverser certains champs** (Compte utilisateur, adresse courriel, date de transfert, compte d'origine et de destination.
- **Obtenir des informations sensibles** (Chemin d'accès spécifique, ID de session, numéro de requête, numéro de transaction, etc.
- **Contourner les mécanismes d'authentification et d'autorisation** (Requête directe aux fonctionnalités, exploitation des requêtes asynchrones, modification des en-têtes « Referer », manipulation des cookies, etc

Recommandations pour prévenir les failles de logiques

- Utiliser des méthodes éprouvées pour les fonctionnalités sensibles (ex: authentification, autorisation, chiffrement, etc.)
- Valider l'interaction humaine et utiliser des mécanismes anti-automatisation (ex: utilisation de « CAPTCHA »)
- Ne jamais faire confiance aux données provenant des utilisateurs
- Adhérer aux principes de « défense en profondeur » (ex: validation à différent niveau)
- Éviter l'utilisation de séquences incrémentielles (paramètres, sessions, URL's, # de commande, etc.)
- Éviter la sécurité par l'obscurité
- Assurez-vous que les numéros générés aléatoirement le soient vraiment
- Bien évaluer les implications de sécurité avant d'ajouter ou modifier une composante applicative

- Formation et sensibilisation des développeurs aux problèmes de sécurité !
- Tester, valider, vérifier !
 - Avant la mise en production
 - Annuellement
 - Lors de modifications majeures
- Intégration de la sécurité aux différentes étapes du cycle de développement, ex:
 - Lors du design
 - Lors du développement
 - Lors de la maintenance évolutive
- Intégration de critères de sécurité au niveau des appels d'offres (responsabiliser les fournisseurs)

- OWASP - Business logic vulnerability
 - http://www.owasp.org/index.php/Business_logic_vulnerability
- OWASP - Testing for business logic (OWASP-BL-001)
 - [http://www.owasp.org/index.php/Testing_for_business_logic_\(OWASP-BL-001\)](http://www.owasp.org/index.php/Testing_for_business_logic_(OWASP-BL-001))
- WhiteHatSec - Seven Business Logic Flaws That Put Your Website At Risk
 - http://www.whitehatsec.com/home/assets/WP_bizlogic092407.pdf
- ACSAC - Designing a Secure Framework Method for Business Application Logic Integrity in e-Commerce Systems
 - <http://www.acsac.org/2008/program/wip/>



Merci !

- *Patrick Chevalier*
- 418.573.7979
- pchevalier@victrix.ca



VICTR*i*X