

Web (In)Security

Patrick Chevalier

CISA, CISSP, GIAC, MCSE, CEH

pchevalier@infostrategique.com

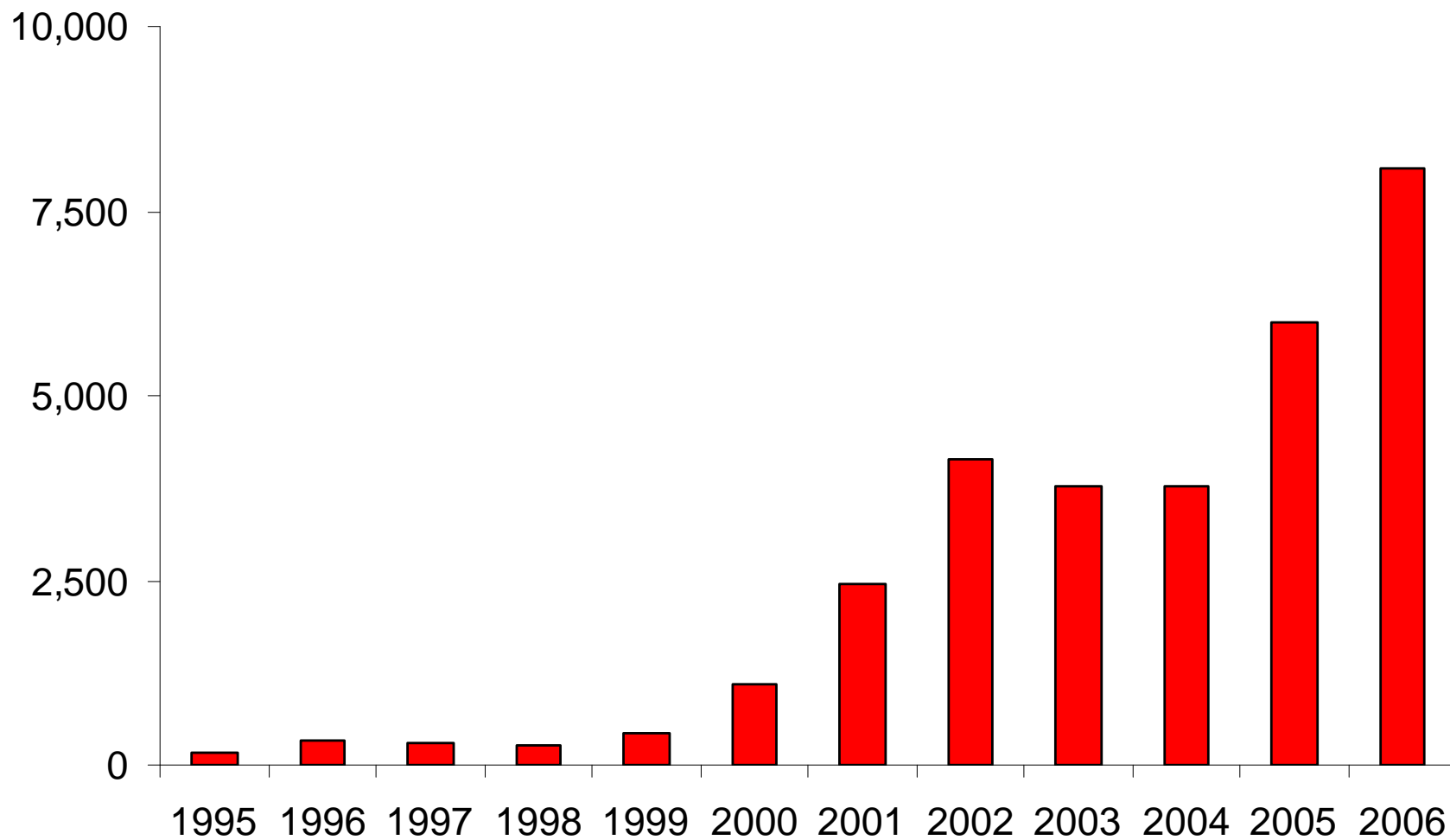
Objectifs de cette présentation

- Vous **sensibiliser** aux problèmes affectant la sécurité Web
- Donner un aperçu des différentes menaces autant du côté serveur que du côté client
- Offrir des pistes de solution
- **Piquer votre curiosité !**

Agenda

- Évolution des vulnérabilités et des menaces.
- Actualité et sécurité Web
- Présentation de différents vers Web (Santy, Samy, Yamaner)
- Comment minimiser les risques ?
 - Côté serveur
 - Côté client
- OWASP et autres ressources
- Période de questions !

Vulnérabilités: 1995 à 2006



Source: SANS Institute

Vulnérabilités: Tendances fin 2006

- **12%** plus de vulnérabilités dans la 2^{ème} moitié de 2006 comparativement à la 1^{re} moitié de 2006 et plus que jamais auparavant
- **66%** des vulnérabilités identifiées concernaient des applications Web
- **79%** des vulnérabilités de la 2^{ème} moitié de 2006 étaient considérées comme facilement exploitables
- **77%** de ces vulnérabilités affectaient des applications Web (**61%** du total), contre **7%** pour les serveurs

Vulnérabilités: Tendances fin 2006 (2)

- **94%** des vulnérabilités considérées comme facilement exploitables étaient de type « remote »
- **68%** des vulnérabilités n'ont pas été reconnues par le vendeur concerné, une hausse de **61%** comparativement à la 1^{re} moitié de 2006
- **54** vulnérabilités affectant Microsoft Internet Explorer ont été identifiées, **40** pour les fureteurs Mozilla, **4** pour Apple Safari et également **4** pour Opera
- **10** vulnérabilités dites "0day" dans les logiciels de Microsoft, dont **6** dans la suite Office

Vulnérabilités: Tendances fin 2006 (3)

- **25%** des exploits ont été mis en circulation moins d'une journée après la publication de la vulnérabilité.
- **31%** ont été mis en circulation dans des délais de un à six jours après la publication de la vulnérabilité.
- Le délai moyen pour le développement d'un exploit était de **5 jours** alors que la moyenne pour le développement d'un correctif était de **52 jours**.

Breaking News: Cenzic Q1 2007 !

- **1,561** vulnérabilités uniques dans le 1^{er} quart de 2007
- **67%** des vulnérabilités affectaient les serveurs web, les applications web et les fureteurs
- Les applications écrites en PHP constituaient **30%** du total
- **63%** des vulnérabilités peuvent être classifiées dans 4 catégories: File inclusion, SQL Injection, XSS, Directory traversal
- **71%** des vulnérabilités étaient classifiées comme facilement ou très facilement exploitables.
- **19%** des vulnérabilités comportaient du XSS

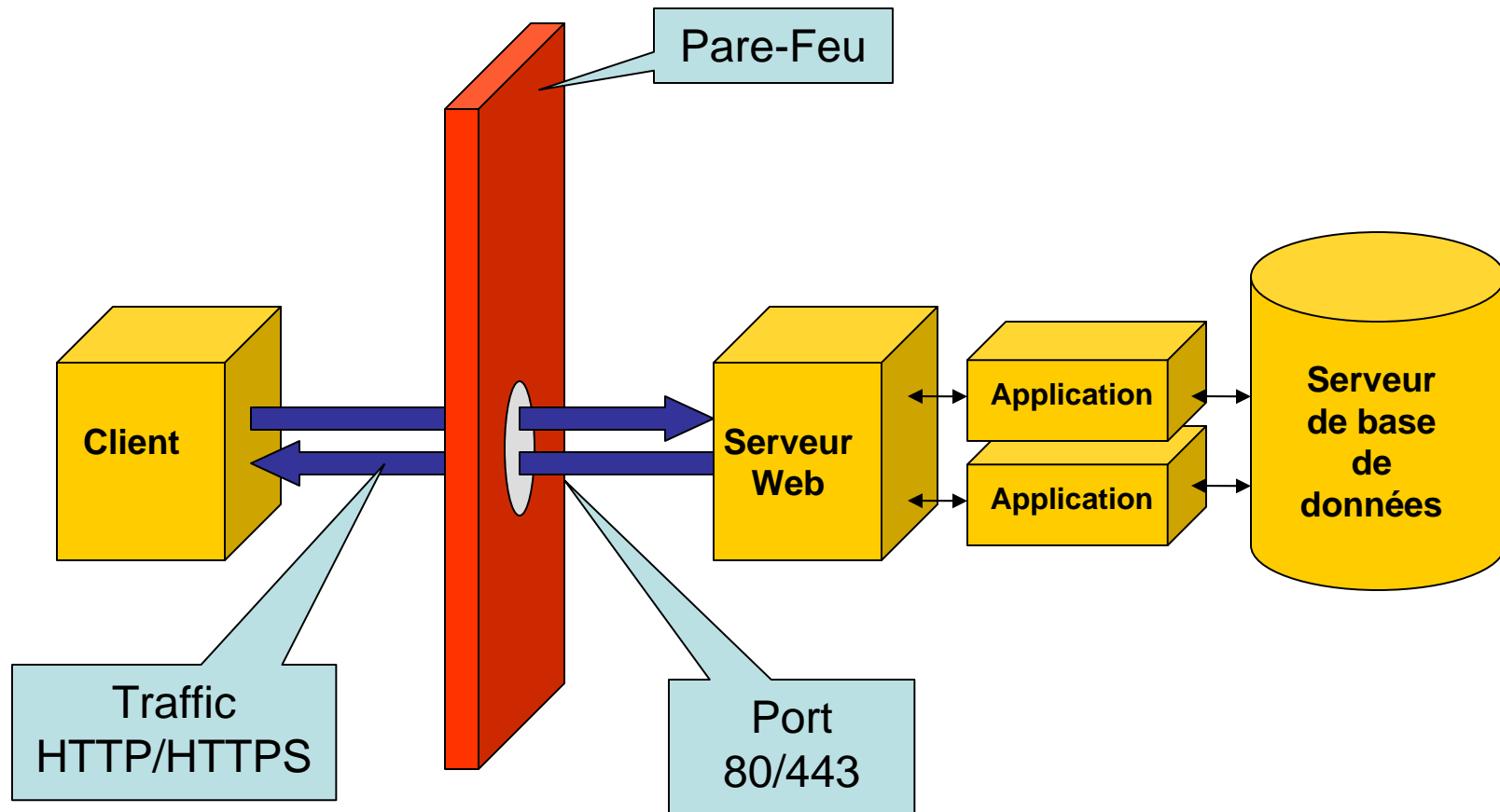
Constats et observations

- Augmentation constante du nombre de vulnérabilité
- Les attaquants focus de plus en plus sur les applications Web:
 - Exploitation généralement très facile
 - Le bassin de cibles potentielles est énorme
- Les attaquants sont à la recherche de cibles faciles:
 - Approche traditionnelle: choisir une cible, trouver des vulnérabilités
 - Approche « Google Hacking »: choisir une vulnérabilité, trouver des cibles
- La commercialisation des activités malicieuses s'amplifie (Spam, Botnet, Fishing, Fraud, Malware, etc)
- Pas de « silver bullet » pour se protéger contre les menaces Web

5 mythes de la sécurité Web

- Nous utilisons un certificat SSL 128 bits
- Notre analyseur de vulnérabilité réseau n'a rien trouvé
- Notre analyseur de vulnérabilité Web n'a rien trouvé
- Nous effectuons des revues de sécurité annuelles
- Notre pare-feu protège notre site web

Défense périmétrique et Sécurité Web



Top 10 CVE (MITRE)

- 01 - Cross-Site-Scripting (XSS)
- 02 - SQL Injection
- 03 - PHP remote file inclusion
- 04 - Buffer Overflow
- 05 - Directory traversal
- 06 - Information Leak
- 07 - DoS caused by malformed input
- 08 - Integer Overflow
- 09 - Issue with Permissions
- 10 - Format String

Top 10 OWASP (Edition 2007)

- A01 - Cross Site Scripting (XSS)
- A02 - Injection Flaws
- A03 - Malicious File Execution
- A04 - Insecure Direct Object Reference
- A05 - Cross Site Request Forgery (CSRF)
- A06 - Information Leakage and Improper Error Handling
- A07 - Broken Authentication and Session Management
- A08 - Insecure Cryptographic Storage
- A09 - Insecure Communications
- A10 - Failure to Restrict URL Access

Évolution des menaces Web

- 1998 - Manipulation de champ « hidden »
- 1999 - Failles d'encodage dans IIS / ASP (ex: Double Byte Bug)
- 2000 - Injection SQL sur MSSQL
- 2000 - Faille Unicode/UTF-8 sur IIS (Nimda)
- 2001 - Injection SQL sur Oracle, MySQL, DB2 etc.
- 2001 - Premier proxy applicatif
- 2002 - « Fuzzing » d'application Web
- 2003 - Injection SQL Aveugle
- 2004 - Injection LDAP et XML, Santy
- 2005 - Samy
- 2006 - Yamaner, Quickspace
- 2007 - Balayage de ports JavaScript et Keylogger AJAX

Aggregators Wikis
Folksonomy User Centered Joy of Use
Blogs Participation Six Degrees Usability Widgets
Pagerank XFN Social Software FOAF Browser
Recommendation Sharing Collaboration Perpetual Beta Simplicity AJAX
Videocasting Podcasting

Web 2.0 Design

CSS Pay Per Click

UMTS Mobility Atom XHTML SVG Ruby on Rails VC Trust Affiliation

OpenAPIs RSS Semantic Web Standards SEO Economy

OpenID Remixability REST Standardization The Long Tail

DataDriven Accessibility XML

Microformats Syndication

Modularity SOAP

JavaScript in web browsers is new security weak spot Ajax fingered as main culprit

By Phil Manchester Published Thursday 17th May 2007 08:02 GMT

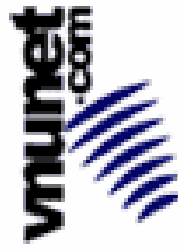
The growing use of JavaScript in web browsers is the new security weak spot, says Brian Chess, chief scientist and founder of US security software specialist [Fortify Software](#).

Specifically, the use of Ajax techniques to build Web 2.0 applications makes enterprise applications more vulnerable.

"It is really hard to see the difference between what Ajax is supposed to do and what is an attack from hijacking JavaScript," Chess says. "Potentially it provides a bridge between external internet applications and internal intranet applications behind the firewall."

This week, Fortify announced the latest version of its Secure Coding Rulepacks aimed specifically at the JavaScript hijacking Chess refers to. Fortify's recent research into source code [vulnerabilities](#) highlights what Chess sees as the next major wave of hacking activity.

"We think the problem is being seriously underestimated because of the surge in Web 2.0 applications. People don't want to talk about it in case it spoils the party - in fact, some have criticised us and said we are ruining Ajax by highlighting its vulnerabilities. But there are some potentially dangerous scenarios that could spin off JavaScript hacks."



Google warns of web malware epidemic

One in ten sites hosting code that attacks browsers

Iain Thomson, vmunet.com 14 May 2007

A study released today by **Google** has warned of "very high levels" of malware being hosted on websites.

In a year-long scan of over 4.5 million sites the Google team found code on 450,000 pages that could inject malware onto users' PCs via improperly patched browsers.

A further 700,000 sites hosted similar code that, while not necessarily malicious, could harm the security of the PC viewing the page.

"In most cases, a successful exploit results in the automatic installation of a malware binary, also called drive-by download," said the five-member team which wrote the Ghost in the Browser paper.

"The installed malware often enables an adversary to gain control over the compromised system and can be used to steal sensitive information such as banking passwords, to send out spam or to install more malicious executables over time."

Popular sites highly vulnerable to attack

By Erik Larkin, PC World 20 April 2007

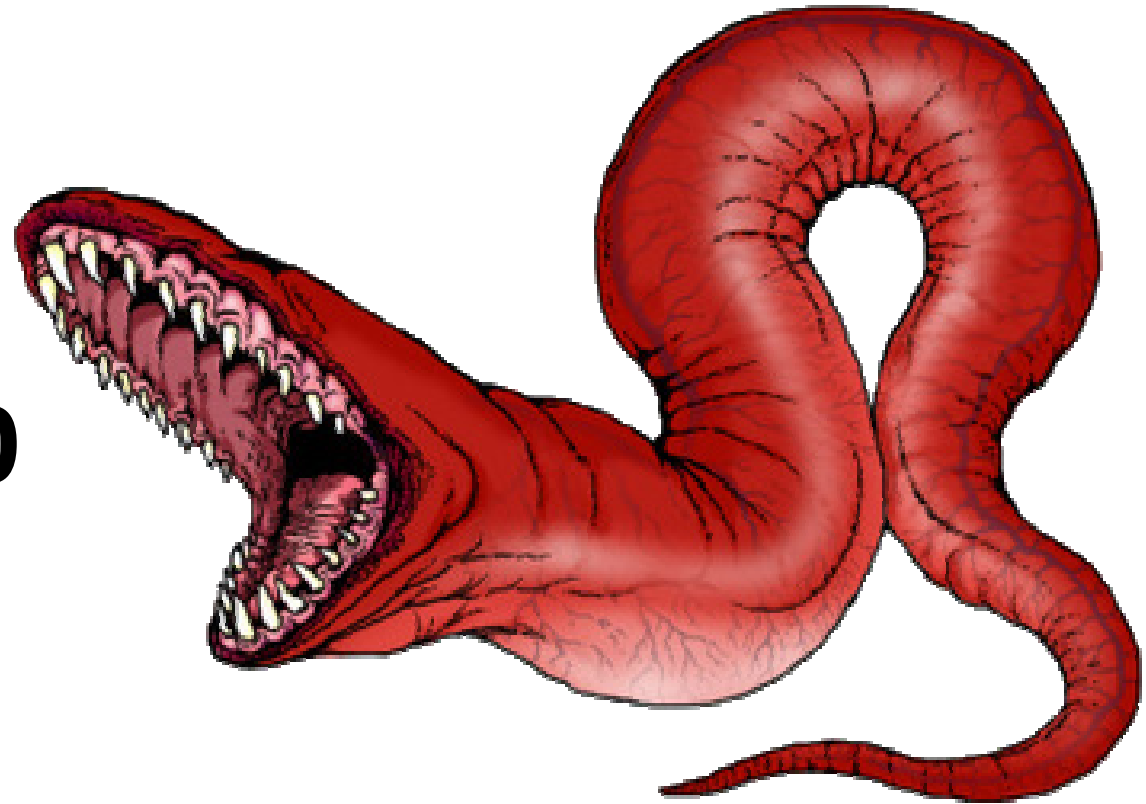
Eight out of ten websites contain flaws that can allow attackers to steal customer data, create phishing exploits, or craft a variety of other attacks, a security company said.

WhiteHat Security regularly scans hundreds of "very popular, very high-traffic sites" for its online business customers, said Jeremiah Grossman, the company's founder. "More than likely, you have shopped there, or bank there," he said. Thirty percent of scanned sites contain an urgent vulnerability, such as one that allows direct access to a company database with customer information, he said.

Two out of three scanned sites have one or more cross-site scripting (XSS) flaws, which take advantage of problems with sites' programming and are increasingly used in phishing attacks. A recent eBay scam used a now-fixed XSS hole on the auction site to direct anyone who clicked on a phony car auction to a phishing site.

About a third of scanned sites are at risk for some sort of information leakage, which often means the providing of programming data about the site that can facilitate an attack. And about one out of four sites allows content spoofing, another phishing risk, according to WhiteHat's vulnerability report.

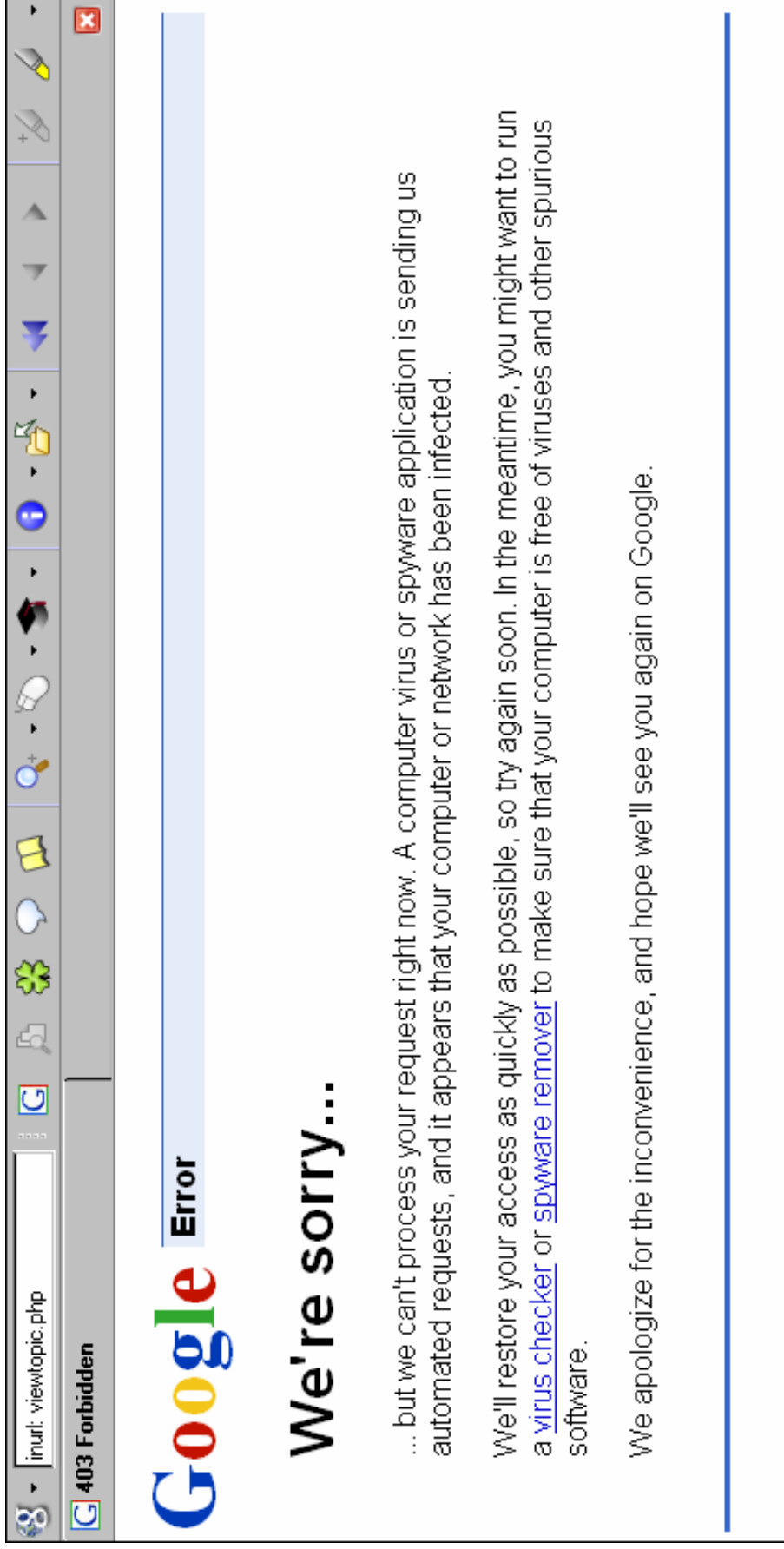
Ajax HTTP
WWW **Web 2.0**
Port 80 JavaScript
XMLHttpRequest



Santy



- 21 Décembre 2004: Cible phpBB >2.0.11
- Utilise la librairie LWP de Perl
- **Vecteur d'attaque:**
 - Vulnérabilité d'injection de script de phpBB (BID 10701) permet d'obtenir un accès sur le serveur cible
- **Vecteur de propagation:**
 - Utilise l'engin de recherche de Google pour générer une liste de cible potentielle, utilise le query « viewtopic.php ».
 - Exploite la vulnérabilité d'injection de script.
 - Se copie dans un fichier nommé « m1h020f ».
- **Payload:**
 - Écrase les fichiers de type .asp .htm .jsp .php .phtm .shtm avec le texte suivant:
This site is defaced!!!
NeverEverNoSanity WebWorm generation X
- Pour stopper la propagation, Google ont bloqué les requêtes similaires à celles utilisées par Santy...



Last week, [Google](#) was able to shut down Santy.A, but new variants from Santy.B to Santy.E have used [AOL](#) and [Yahoo](#) to spread.

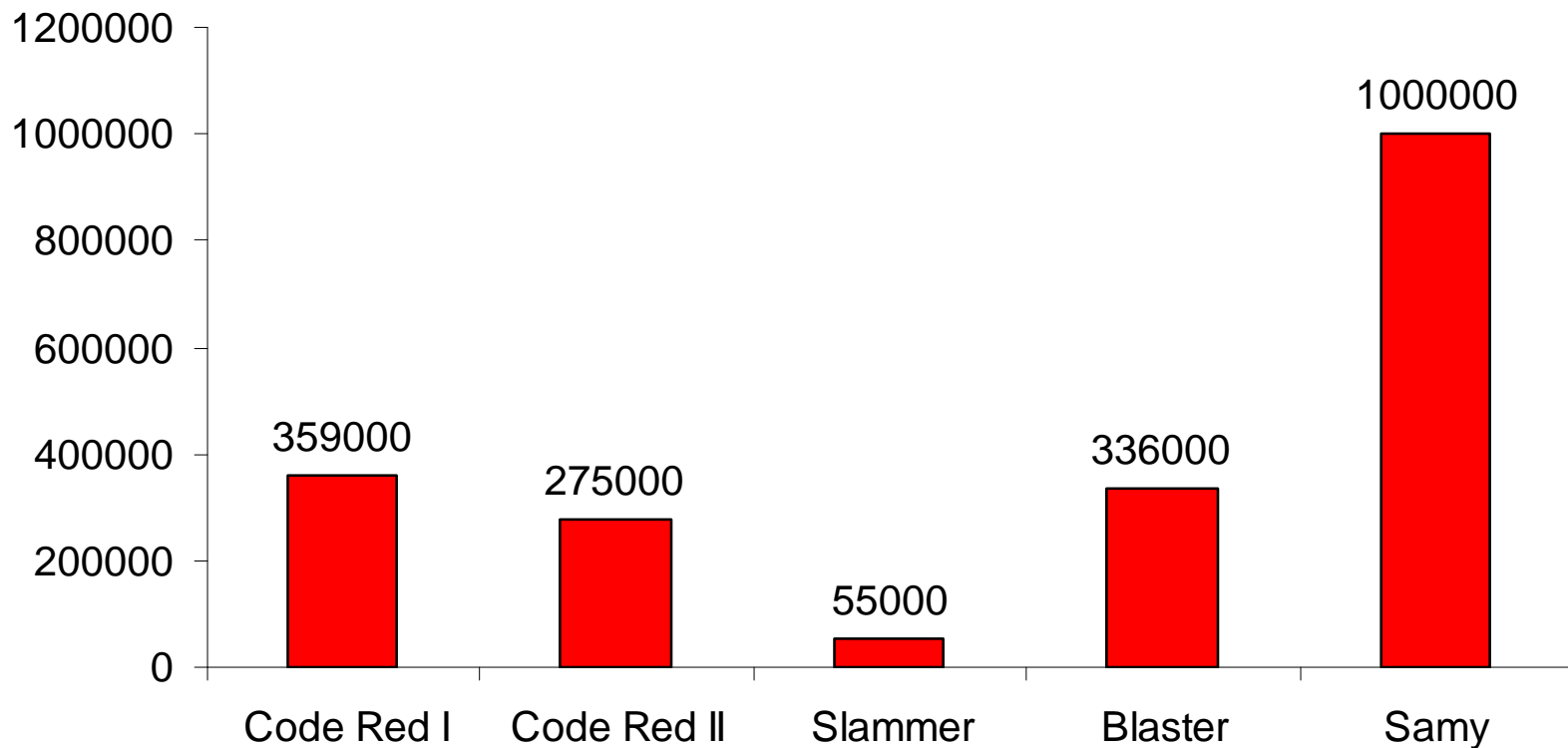
Samy



- 13 Octobre 2005: A infecté le site MySpace.com
- JavaScript avec AJAX
- **Vecteur d'attaque:**
 - Une vulnérabilité XSS permettait d'ajouter une balise <script> dans un profile usager.
- **Vecteur de propagation:**
 - À utilisé AJAX pour injecter le virus dans le profil d'un utilisateur visitant une page infectée.
- **Payload**
 - Forçait un usager à ajouter « Samy » (le créateur) à sa liste d'amis.
 - Utilisait AJAX pour ajouter la string « Samy is my hero » au profil de sa victime.
- Plus de 1,000,000 infections en 24h !
- MySpace.com ont dû fermer leur service pour effectuer un nettoyage.
- Voir <http://namb.la/popular/tech.html> pour l'ensemble des détails techniques.

Comparaison ...

Première 24h de propagation



Yamaner



- 12 Juin 2006: A infecté le site Yahoo! Mail
- **Vecteur d'attaque:**
 - Vulnérabilité dans l'utilisation de la fonction javascript « onload() ».
- **Vecteur de propagation:**
 - Envoie un email avec comme titre « New Graphic Site » avec une adresse source forgée.
 - Lorsque le email est ouvert, il exploite la vulnérabilité et exécute le javascript
 - Lorsqu'exécuté il envoie une copie de lui même à la liste de contacts de l'utilisateur infecté
- **Payload**
 - Envoie la liste des email amassés vers le site <http://www.av3.net>
- Avait pour objectif ultime de collecter les adresses @yahoo.com et @yahoogroups.com aux fin de spam
- Impossible d'estimer combien d'adresses email ont été volées par les spammeurs

Constats

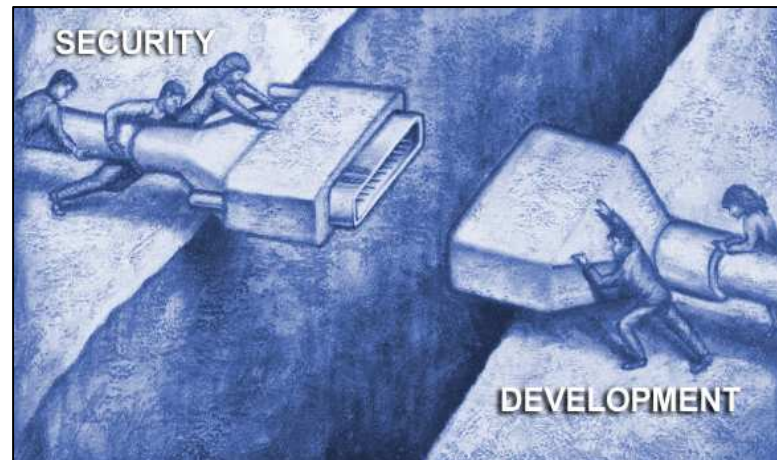
- Santy et Samy était principalement des création de type "preuve de concept"
- Leur payloads était bénin
- Yamaner à été le premier vers Web avec un payload "malicieux" mais dont l'impact fut très limité
- Voici un exemple de type « worst case scenario » ...

Hypothèse

- Utilise la librairie LWP de Perl
- **Vecteur d'attaque:**
 - Plusieurs vulnérabilités de type « SQL Injection » dans différentes applications Web.
- **Vecteur de propagation:**
 - Utilisation de plusieurs engins de recherche pour identifier des sites vulnérables aux failles d'injection SQL.
 - Mutation de la requête de recherche pour éviter un blocage:
 - allinurl: => inurl:
 - Ajout de mots ignorés (the,in,of,at,a,an)
 - Algorithme de génération de mots, ou utilisation de /usr/local/dict
 - Changement de l'ordre des paramètres.
- **Payload**
 - Vulnérabilité connue = application connue = schéma de base de données connu:
 - Publie les tables de mots de passe sur des mailing-lists, blogs ou forums
 - Ou insère du « garbage » dans les bases de données
 - OU ... 100,000 DROP TABLE`s !

Le « application security gap »

Très peu de professionnels de la sécurité possèdent des connaissances en développement et très peu de développeurs possèdent des connaissances en sécurité.



Principes importants

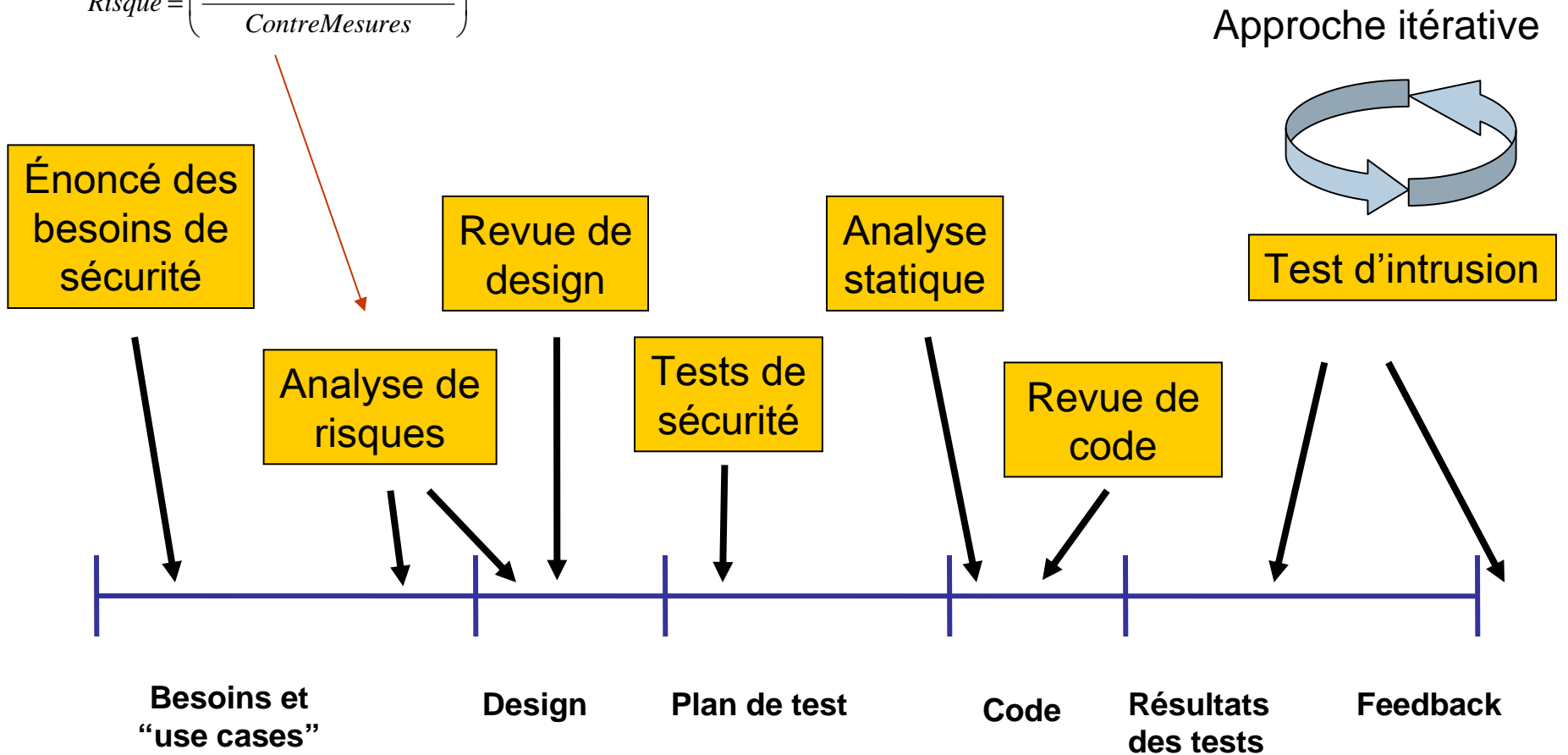
- L'application doit se défendre elle-même !
- Ne jamais faire confiance aux données provenant des usagers.
- Le concept de « périmètre » est dépassé en matière de sécurité Web.

"100 fois plus dispendieux de corriger les problèmes de sécurité à la production qu'au design"

(Source: IBM System Sciences Institute)

Sécurité et SDLC

$$Risque = \left(\frac{Vulnérabilité \times Menace}{ContreMesures} \right)$$



Balayage automatisé ?

- Il est généralement accepté qu'un balayage automatisé identifie environ **30%** des vulnérabilités applicatives.
- Un scanneur automatisé n'est pas en mesure d'identifier les failles logiques.
 - Généralement les vulnérabilités les plus critiques...
 - Authentification, autorisation, gestion des sessions, etc.
- Les résultats sont difficiles à interpréter pour plusieurs professionnels de la sécurité réseau.
- Les vulnérabilités identifiées doivent être corrigées, il ne s'agit pas simplement d'appliquer une mise à jour.
- Les balayages automatisés peuvent apporter un faux sentiment de sécurité à une organisation.

Meilleures pratiques côté serveur

- La majeure partie de la sécurité applicative consiste à valider les entrants des usagers.
- Ne **JAMAIS** faire confiance aux données provenant des usagers.
- Il est essentiel de **tout valider** car tout peut être modifié:
 - Les champs de type « hidden »
 - Cookies
 - Paramètres de requêtes GET et POST
 - Headers HTTP
- La validation doit être mise en place à différents niveaux:
 - La validation côté client (Javascript) ne peut être prise au sérieux.
 - La validation côté serveur est la seule valable, elle doit idéalement être mise en place au niveau de l'application et au niveau de la base de données
- Sensibilisation des développeurs aux problèmes de sécurité !

OWASP ?

- L'OWASP (Open Web Application Security Project) est un effort communautaire destiné à aider les organisations à comprendre et améliorer la sécurité de leurs applications et services Web. Plusieurs ressources disponibles:
 - **Publications ,articles, standards, ex:**
 - OWASP Guide to Building Secure Web Applications
 - OWASP Testing Guide
 - OWASP Code Review Project (in development)
 - OWASP PHP Project (in development)
 - OWASP Java Project
 - OWASP .NET Project
 - **Outils:**
 - WebGoat
 - WebScarab
 - **Mailing-lists**

Meilleures pratiques côté client

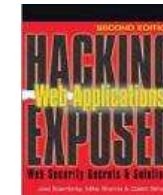
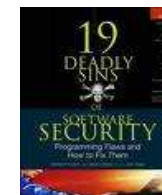
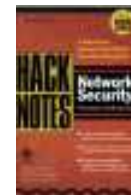
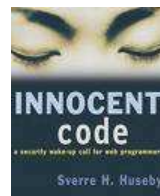
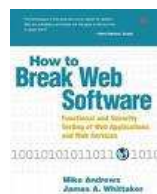
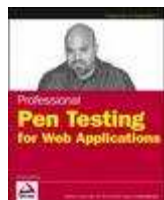
- **Sensibilisation des usagers !**
- Application rapide des correctifs du système d'exploitation (Utilisation du ZERT en cas de vulnérabilité 0day)
- Mise en place et entretien des logiciels de protection courants: anti-spyware, anti-virus, HIPS, etc
- Utilisation de mécanismes de défense périmétrique pour bloquer l'accès aux sites Web non-autorisés
- Mise à jour des logiciels tel que: Microsoft Office, Windows Media Player, Acrobat Reader, WinAMP, QuickTime, etc.
- Désactivation de JavaScript/ActiveX (extension NoScript de Firefox ou model de zone de IE)
- Veille technologique sur l'apparition de nouvelles menaces

Hot Stuff !

- Début le 1^{er} Juin du 'Month of Search Engines Bugs' (MOSEB)
- Nouveau livre de Syngress, Cross Site Scripting Attacks: Xss Exploits and Defense
- Network Computing's évalue différents analyseurs de vulnérabilités Web (Acunetix, Cenzic, N-Stalker, SPI Dynamics, Syhunt, Watchfire, etc)
- Résultats du « WASC Open Proxy Honeypot »
- Annonce sur l'arrivée d'une certification OWASP, au même niveau que PCI
- Lancement du « Google Online Security Blog »
- Publication par Cenzic du « Application Security Trends Report Q1 2007 »

Ressources

- Open Web Application Security Project (OWASP)
 - <http://www.owasp.org>
- Web Application Security Consortium (WASC)
 - <http://www.webappsec.org>
- ModSecurity
 - <http://www.modsecurity.org>
- CGISecurity
 - <http://www.cgisecurity.com>



INFORMATIQUE



IS

STRATÉGIQUE

Questions ???