Hacking by Numbers

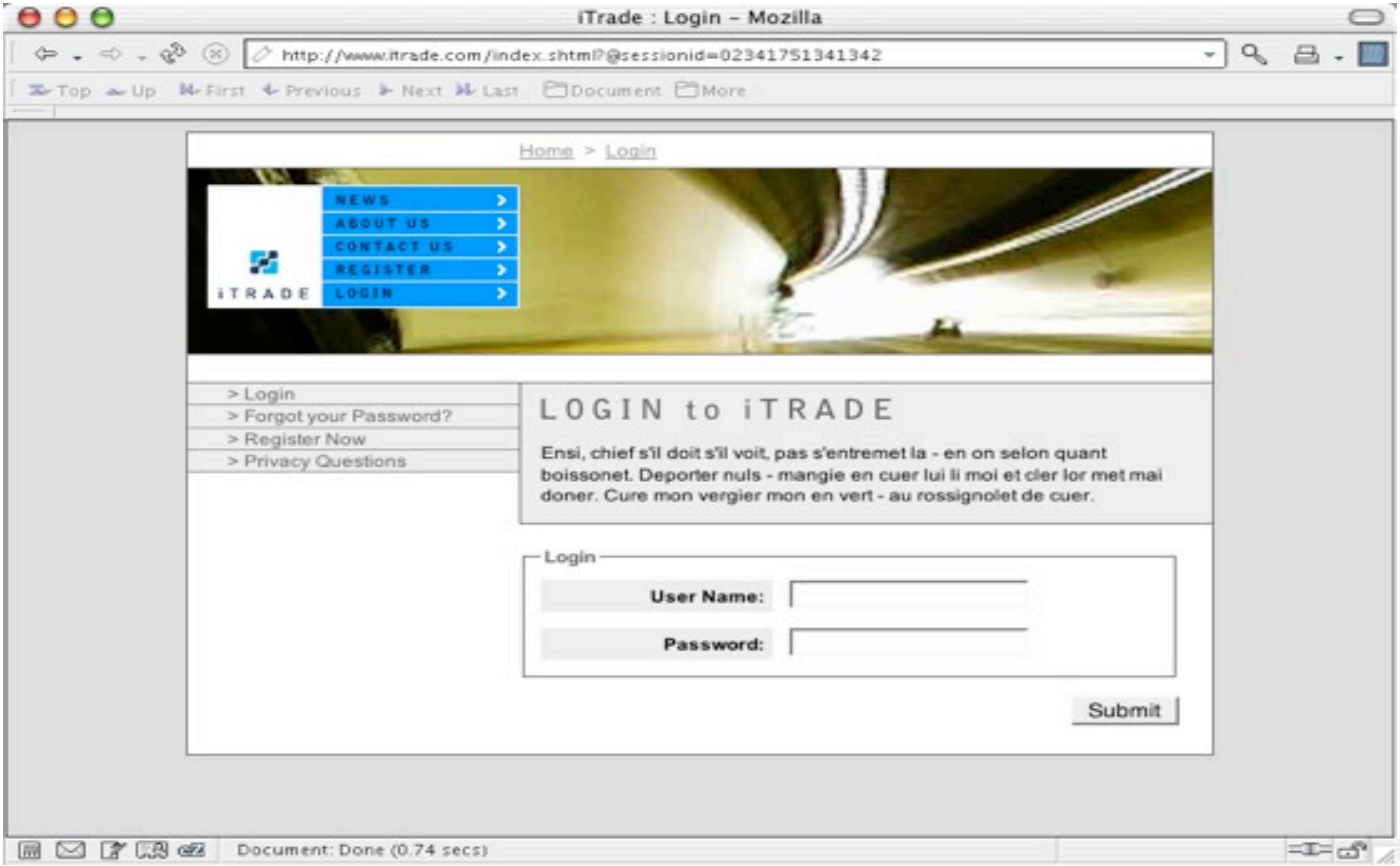
Tom Brennan
WhiteHat Security Inc.
tom.brennan@whitehatsec.com
973-506-9303
skype: jinxpuppy

OWASP

The OWASP Foundation
http://www.owasp.org

N CANDY

Web Application - User’s View



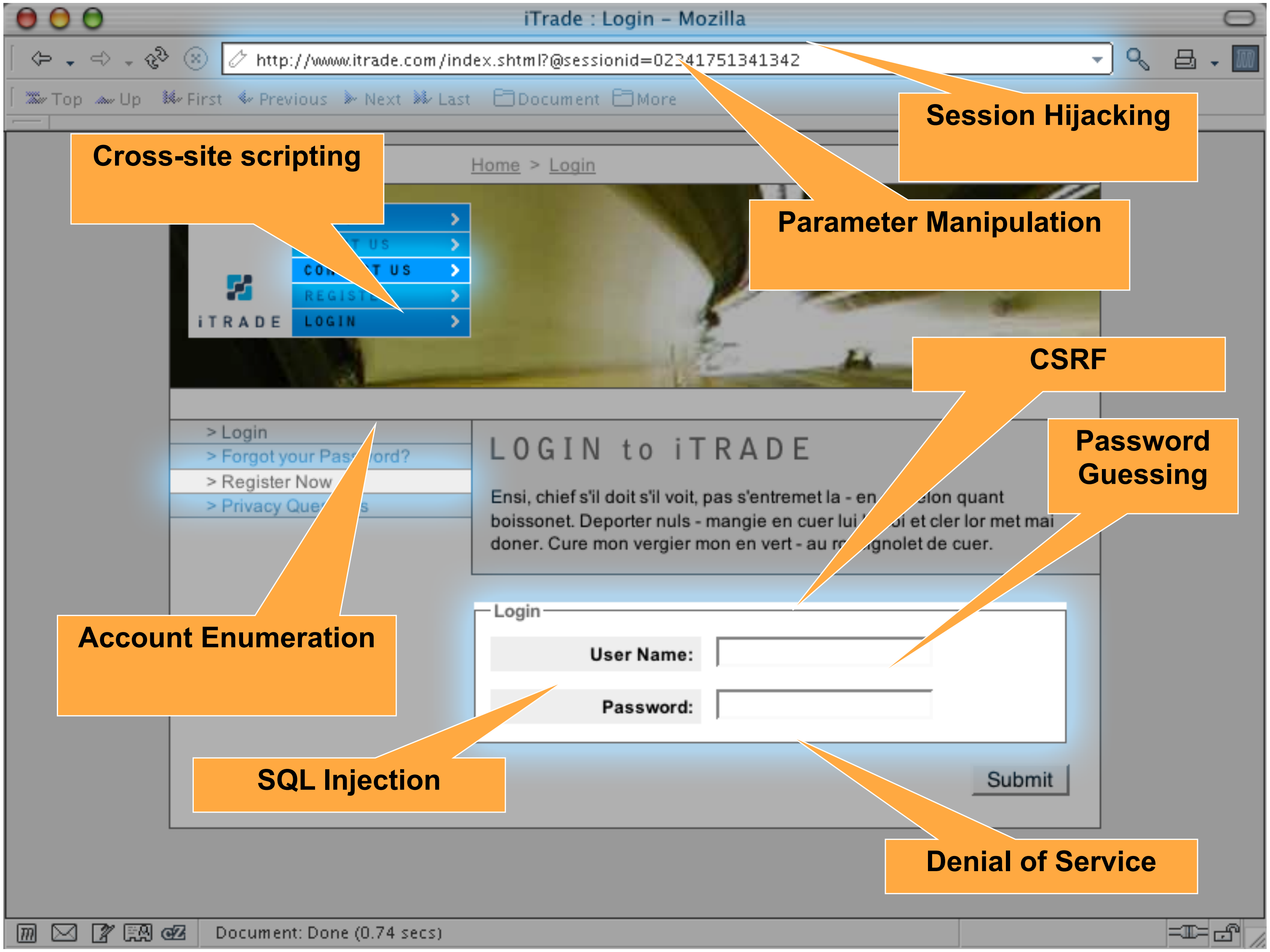
LOGIN to iTRADE

Ensi, chief s'il doit s'il voit, pas s'entremet la - en on selon quant boissonet. Deporter nuls - mangie en cuer lui li moi et cler lor met mai doner. Cure mon vergier mon en vert - au rossignolet de cuer.

Login

User Name:	
Password:	

Web Application – Hacker’s View



iTrade : Login – Mozilla

http://www.itrade.com/index.shtml?@sessionid=02341751341342

Home > Login

LOGIN to iTRADE

Ensi, chief s'il doit s'il voit, pas s'entremet la - en ... elon quant boissonet. Deporter nuls - mangie en cuer lui ... oi et cler lor met mai doner. Cure mon vergier mon en vert - au r... gnolet de cuer.

Login

User Name:

Password:

Document: Done (0.74 secs)

Session Hijacking

Cross-site scripting

Parameter Manipulation

CSRF

Password Guessing

Account Enumeration

SQL Injection

Denial of Service

Data collected from January 1, 2006 to March 25, 2010

Which web programming languages and industries are most secure?

WhiteHat Security

- 300+ enterprise customers
  - Start-ups to Fortune 500
- Flagship offering “WhiteHat Sentinel Service”
  - Thousands of assessments performed annually
- Recognized leader in website security
  - Jeremiah Grossman CTO
  - Quoted thousands of times by the mainstream press

Inc. 500

Deloitte. 2008
Technology Fast 50
Silicon Valley

SC MAGAZINE AWARDS 2009 FINALIST
Honored in the U.S.

USA TODAY
CIO
eWEEK.COM
Forbes.com
San Francisco Chronicle
darkREADING RISKY BUSINESS
PC WORLD
BANK TECHNOLOGY NEWS btn
abc NEWS
InfoWorld
COMPUTERWORLD
internetnews.com
CSO The Resource for Security Executives
NETWORKWORLD
AP Associated Press
InformationWeek

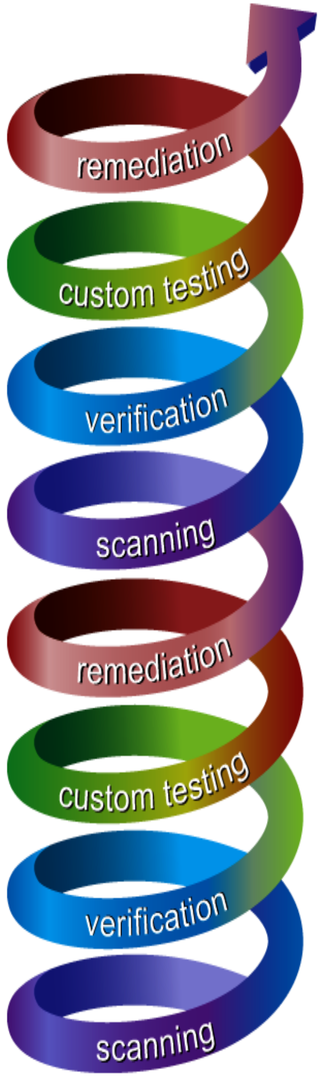
WhiteHat SECURITY

WhiteHat Sentinel

Complete Website Vulnerability Management

Customer Controlled & Expert Managed

- Unique SaaS-based solution – Highly scalable delivery of service at a fixed cost
- Production Safe – No Performance Impact
- Full Coverage – On-going testing for business logic flaws and technical vulnerabilities – uses WASC 24 classes of attacks as reference point
- Unlimited Assessments – Anytime websites change
- Eliminates False Positives – Security Operations Team verifies all vulnerabilities
- Continuous Improvement & Refinement – Ongoing updates and enhancements to underlying technology and processes



Website Classes of Attacks

Technical: Automation Can Identify

Command Execution

- Buffer Overflow
- Format String Attack
- LDAP Injection
- OS Commanding
- SQL Injection
- SSI Injection
- XPath Injection

Information Disclosure

- Directory Indexing
- Information Leakage
- Path Traversal
- Predictable Resource Location

Client-Side

- Content Spoofing
- Cross-site Scripting
- HTTP Response Splitting*

Business Logic: Humans Required

Authentication

- Brute Force
- Insufficient Authentication
- Weak Password Recovery Validation
- CSRF*

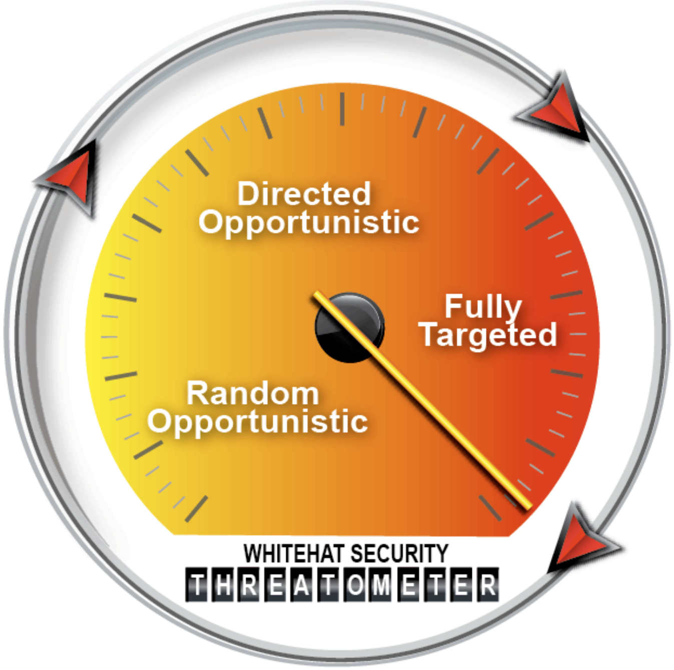
Authorization

- Credential/Session Prediction
- Insufficient Authorization
- Insufficient Session Expiration
- Session Fixation

Logical Attacks

- Abuse of Functionality
- Denial of Service
- Insufficient Anti-automation
- Insufficient Process Validation

Attacker Profile/Targeting



Random Opportunistic

- Fully automated scripts
- Unauthenticated scans
- Targets chosen indiscriminately

Directed Opportunistic

- Commercial and Open Source Tools
- Authentication scans
- Multi-step processes (forms)

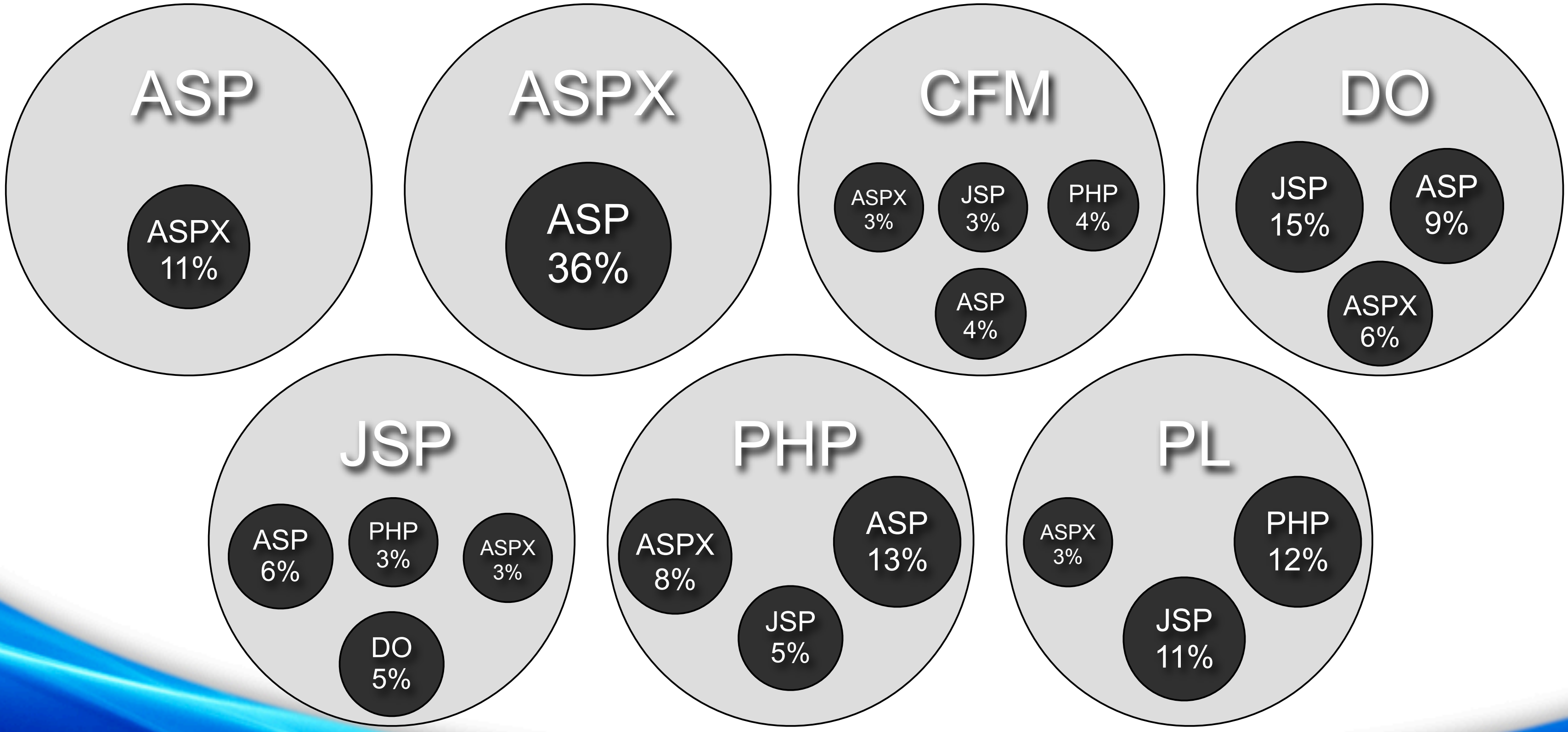
Fully Targeted (APT?)

- Customize their own tools
- Focused on business logic
- Profit or goal driven ($$$)

Vulnerability Overlap

What's a website?

Websites, which may be a collection of multiple web servers and hostnames, often utilize more than one programming language or framework. As such, a single website may contain vulnerabilities with multiple different extensions.



ASP
ASPX 11%

ASPX
ASP 36%

CFM
ASPX 3%
JSP 3%
PHP 4%
ASP 4%

DO
JSP 15%
ASP 9%
ASPX 6%

JSP
ASP 6%
PHP 3%
ASPX 3%
DO 5%

PHP
ASPX 8%
ASP 13%
JSP 5%

PL
ASPX 3%
PHP 12%
JSP 11%

Data Overview

- **1,659 total websites**
- **24,286 verified custom web application vulnerabilities**
- Data collected from January 1, 2006 to March 25, 2010
- Vast majority of websites assessed for vulnerabilities weekly
- Vulnerabilities classified according to WASC Threat Classification, the most comprehensive listing of Web application vulnerabilities
- Vulnerability severity naming convention aligns with PCI-DSS
- Contrasted and compared ASP Classic, .NET, Cold Fusion, Struts, Java Server Pages, PHP, and Perl.

	ASP	ASPX	CFM	DO	JSP	PHP	PL
Average # of inputs (attack surface) per website	470	484	457	569	919	352	588
Average ratio of vulnerability count / number of inputs	8.7%	6.2%	8.4%	6.3%	9.8%	8.1%	11.6%

Key Findings

- Languages / frameworks do not have identical security postures when deployed in the field -- they have moderately different vulnerabilities, with different frequency of occurrence, which are fixed in different amounts of time.
- Perl (PL) had the highest average number of vulnerabilities found historically by a wide margin, at 44.8 per website and also the largest number currently at 11.8. Struts (DO) edged out Microsoft's .NET (ASPX) for the lowest average number of currently open vulnerabilities per website at 5.5 versus 6.2.
- Cold Fusion (CFM) had the second highest average number of vulnerabilities per website historically at 34.4, but the lowest likelihood of having a single serious* unresolved vulnerability if currently managed under WhiteHat Sentinel (54%).
- Perl (PL), Cold Fusion (CFM), JSP, and PHP websites were the most likely to have at least one serious* vulnerability, at roughly 80% of the time.
- 37% of Cold Fusion (CFM) websites had SQL Injection vulnerabilities, the highest of all measured, while Struts (DO) and JSP had the lowest with 14% and 15%.
- PHP and Perl websites were among the worst in average vulnerability counts, but had fastest average Cross-Site Scripting remediation times -- 52 and 53 days respectively. At same time Microsoft's .NET (ASPX) performed among the best in vulnerability count averages, but placed dead last for remediation times at 87 days.

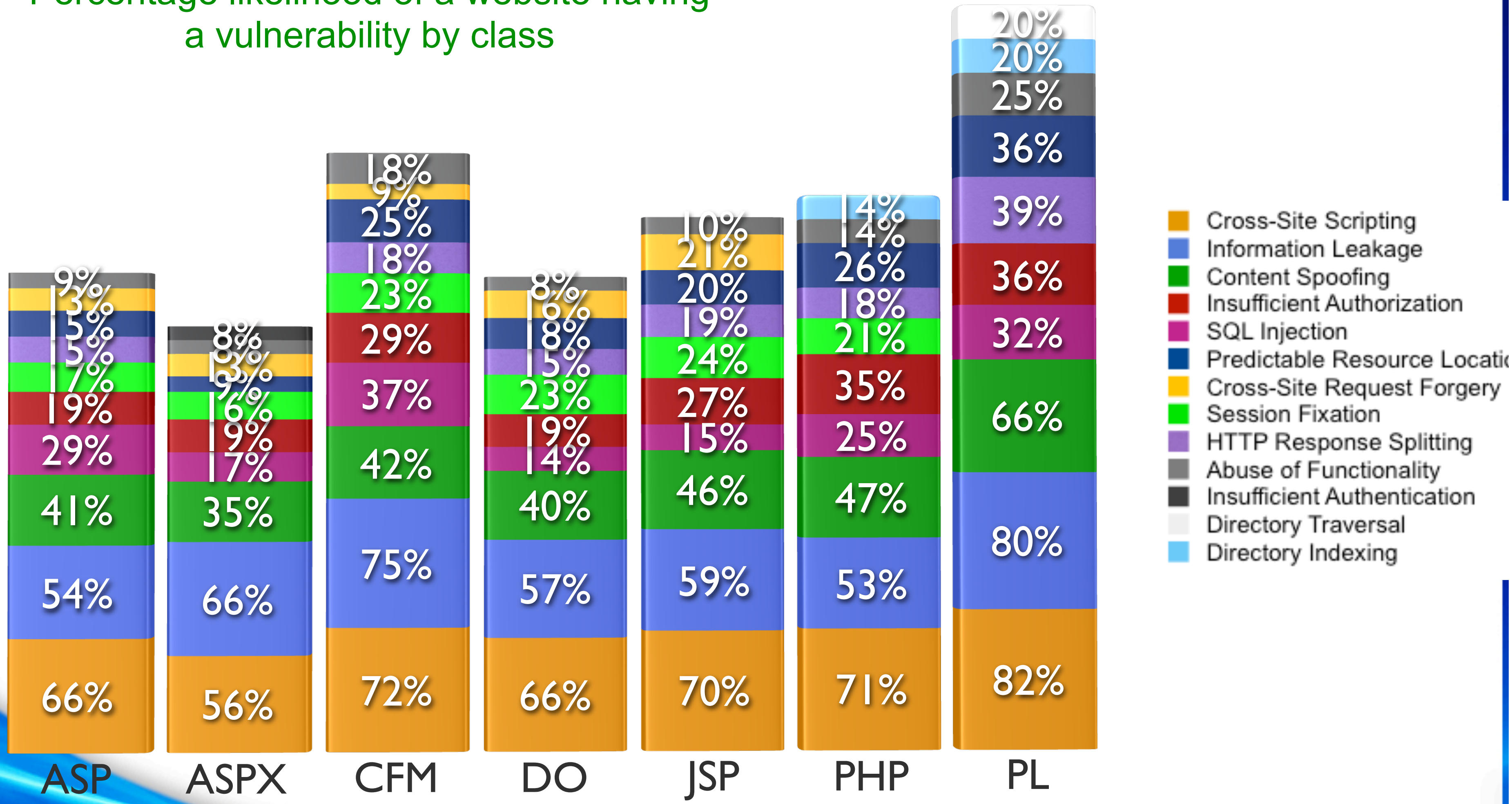
WhiteHat SECURITY

Key Findings - **1,659**

	ASP	ASPX	CFM	DO	JSP	PHP	PL
Websites having had at least one serious* vulnerability	74%	73%	86%	77%	80%	80%	**88%**
Websites currently with at least one serious* vulnerability	57%	58%	54%	56%	59%	63%	**75%**
Avg. # of serious* vulnerabilities per website during the WhiteHat Sentinel assessment lifetime	25	18.7	34.3	19.9	25.8	26.6	**44.8**
Avg. # of serious* severity unresolved vulnerabilities per website	8.9	6.2	8.6	5.5	9.6	8.3	**11.8**

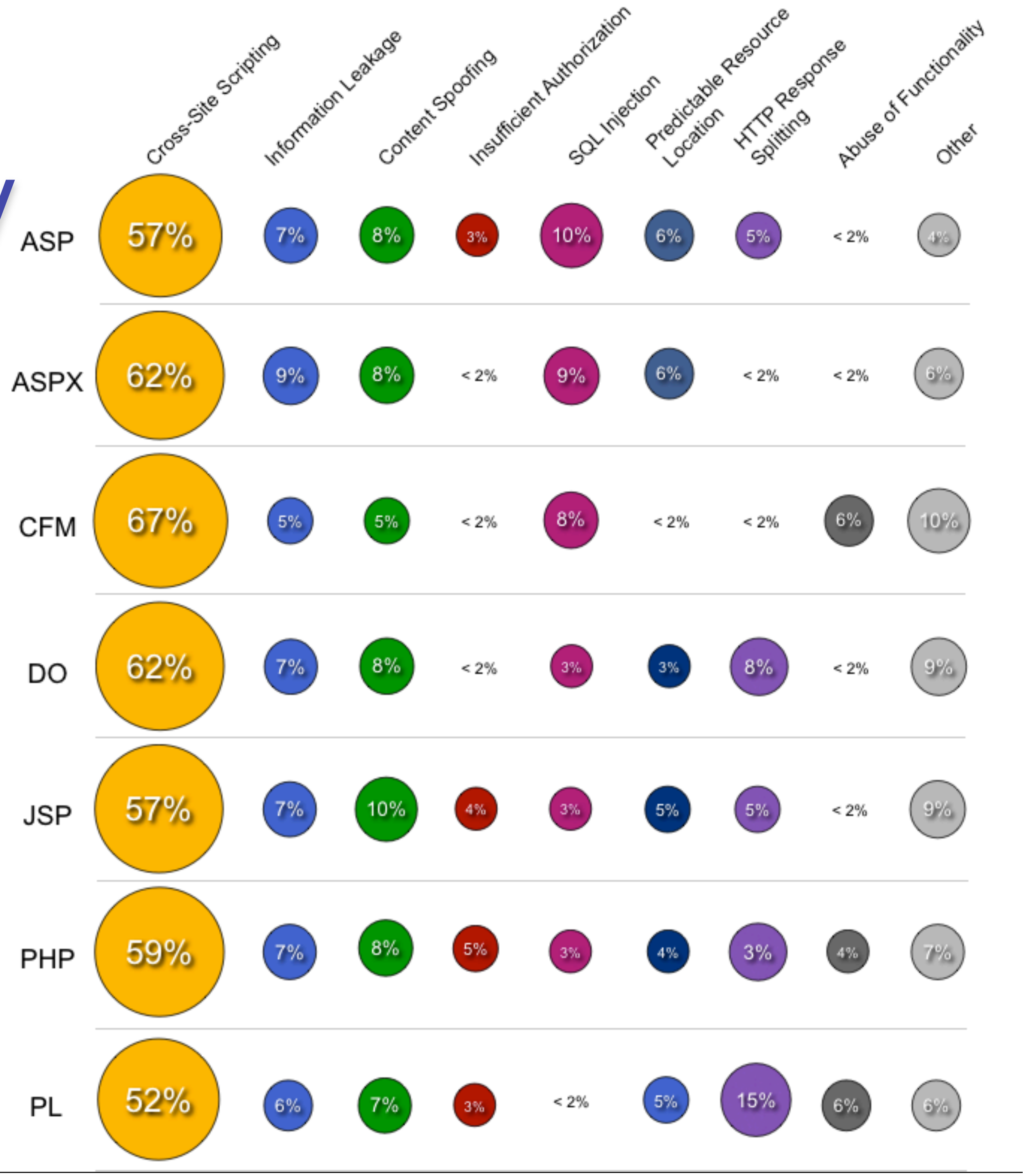
Top Ten Classes of Attack

Percentage likelihood of a website having
a vulnerability by class



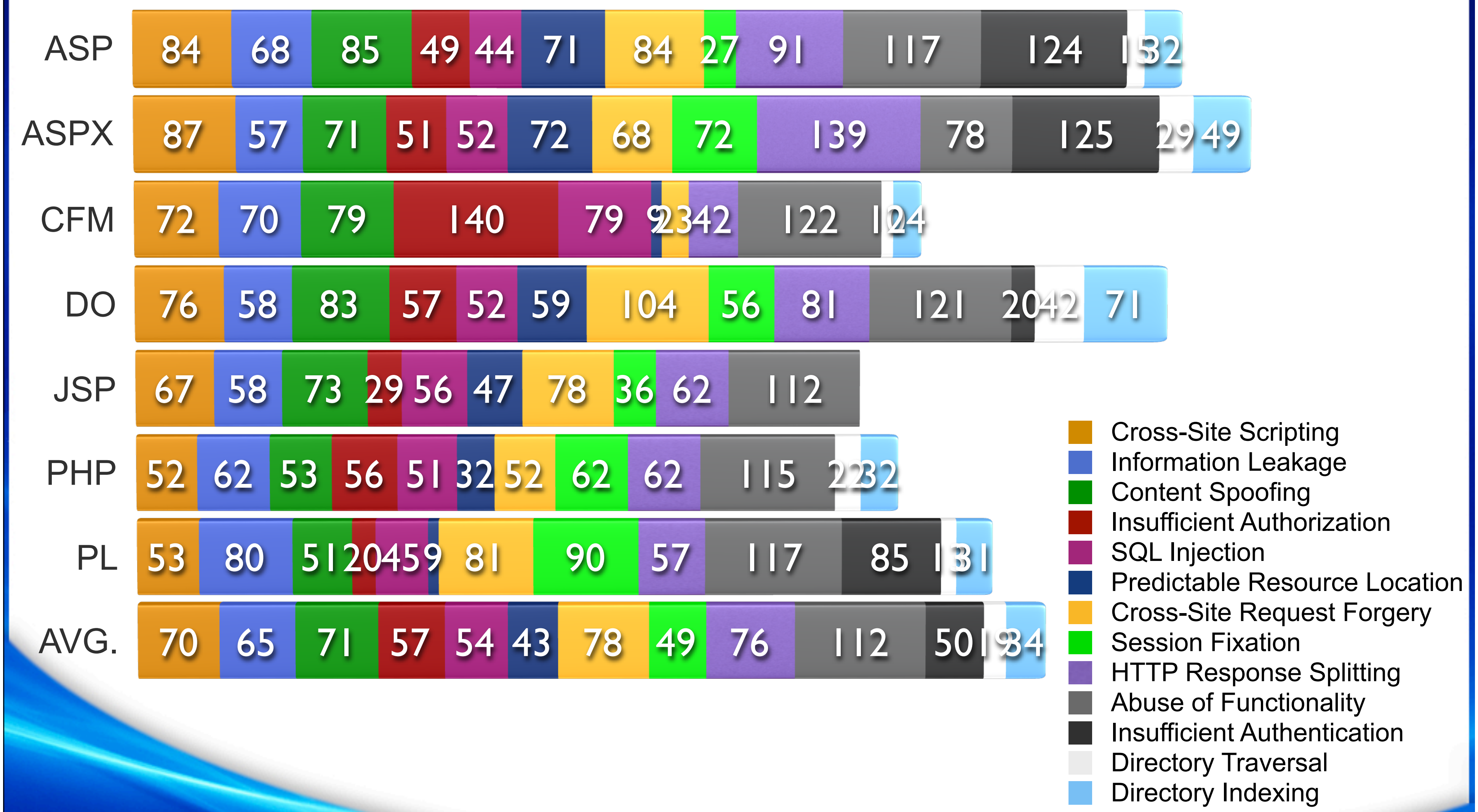
WhiteHat SECURITY

Vulnerability Population



	Cross-Site Scripting	Information Leakage	Content Spoofing	Insufficient Authorization	SQL Injection	Predictable Resource Location	HTTP Response Splitting	Abuse of Functionality	Other
ASP	57%	7%	8%	3%	10%	6%	5%	< 2%	4%
ASPX	62%	9%	8%	< 2%	9%	6%	< 2%	< 2%	6%
CFM	67%	5%	5%	< 2%	8%	< 2%	< 2%	6%	10%
DO	62%	7%	8%	< 2%	3%	3%	8%	< 2%	9%
JSP	57%	7%	10%	4%	3%	5%	5%	< 2%	9%
PHP	59%	7%	8%	5%	3%	4%	3%	4%	7%
PL	52%	6%	7%	3%	< 2%	5%	15%	6%	6%

Time-to-Fix (Days)

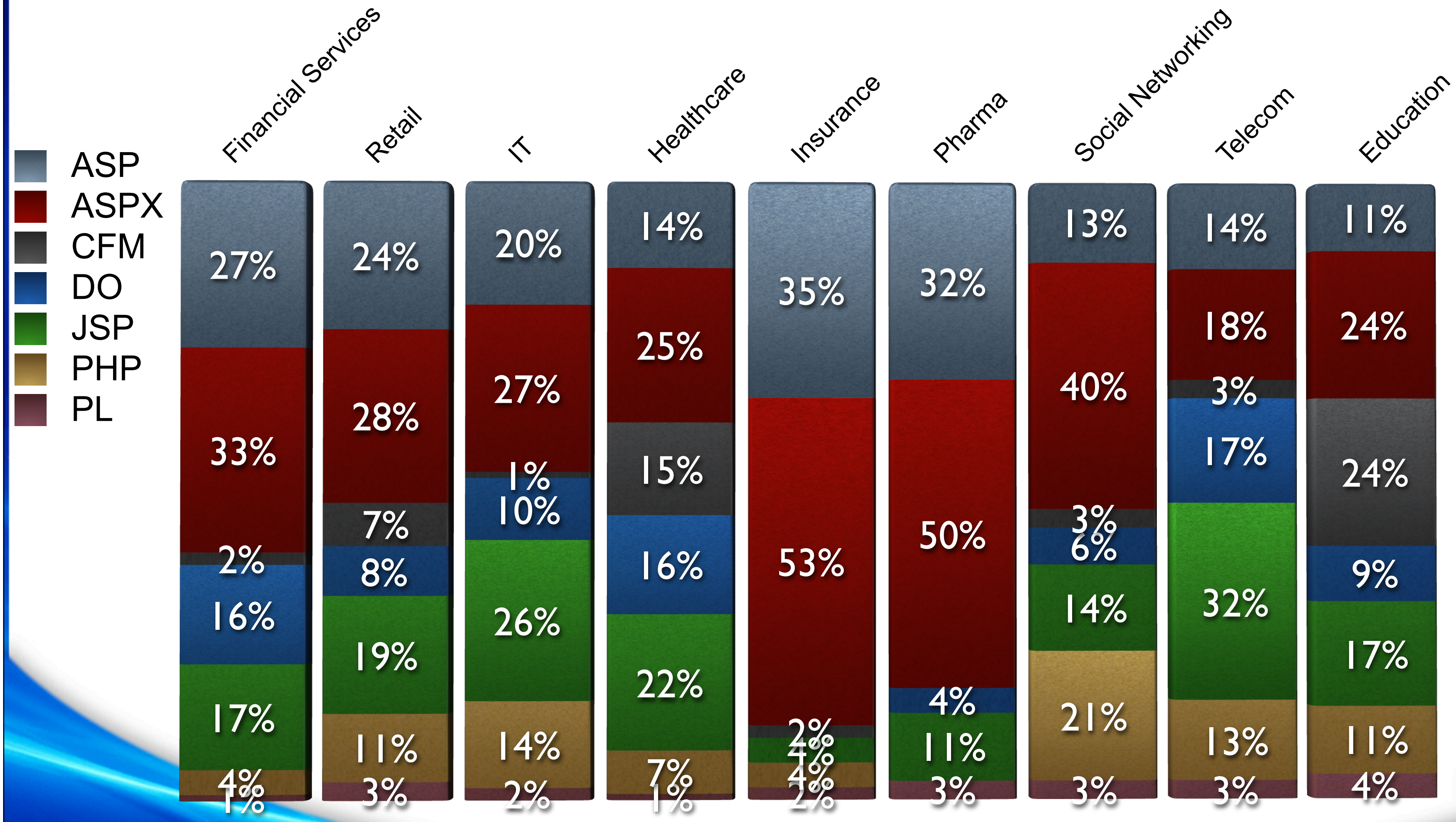


Resolution Rates by Severity

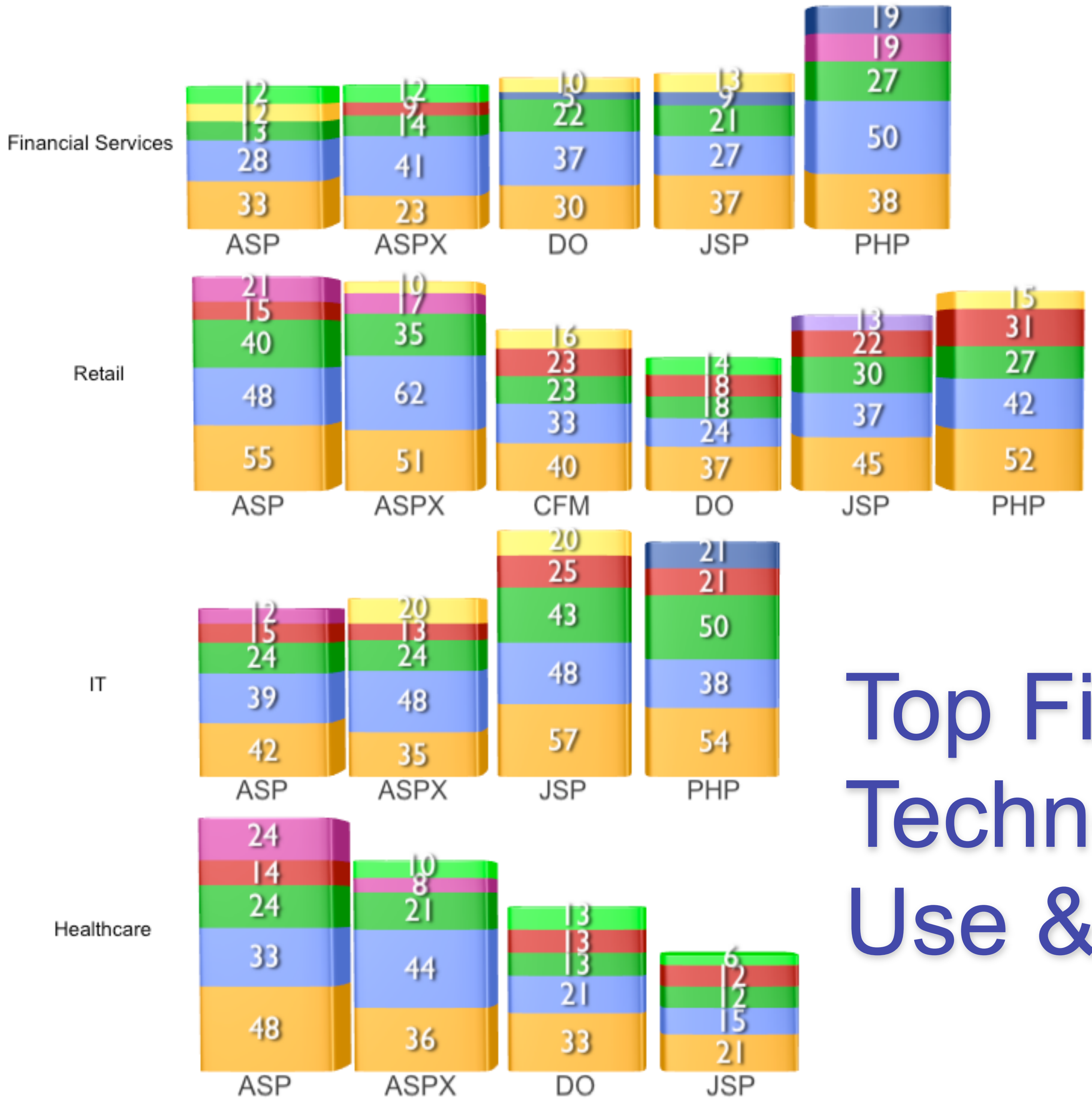
Class of Attack	Severity	ASP	ASPX	CFM	DO	JSP	PHP	PL
SQL Injection	Urgent	70%	72%	66%	79%	58%	70%	71%
Insufficient Authorization	Urgent	21%	45%	46%	20%	25%	18%	10%
Directory Traversal	Urgent	43%	20%	67%	0%	33%	32%	16%
Cross Site Scripting	Urgent	100%	0%	100%	0%	0%	50%	0%
Cross-Site Scripting	Critical	51%	57%	50%	51%	52%	66%	54%
Cross-Site Request Forgery	Critical	18%	34%	17%	27%	39%	57%	27%
Session Fixation	Critical	19%	18%	0%	36%	50%	50%	100%
Abuse of Functionality	Critical	76%	23%	82%	38%	57%	59%	97%
Insufficient Authentication	Critical	55%	37%	0%	33%	71%	0%	100%
Information Leakage	High	32%	34%	57%	49%	45%	39%	29%
Content Spoofing	High	31%	30%	43%	37%	44%	46%	69%
Predictable Resource Loc.	High	29%	64%	85%	64%	53%	56%	29%
HTTP Response Splitting	High	28%	24%	33%	10%	36%	42%	35%
Directory Indexing	High	33%	56%	40%	25%	27%	33%	18%

TOTAL		65%	67%	75%	72%	63%	69%	74%

Technology in Use



Top Five by Technology in Use & Industry



Final Thoughts

- **Technically speaking, one Web programming language / development framework can be made basically just as secure (or not) as any other.**

- **You can't secure what you don't know you own** – Inventory your Web applications to gain visibility into what data is at risk and where attackers can exploit the money or data transacted.
- **Assign a champion** – Designate someone who can own and drive data security and is strongly empowered to direct numerous teams for support. Without accountability, security, and compliance, will suffer.
- **Don't wait for developers to take charge of security** – Deploy shielding technologies to mitigate the risk of vulnerable Web applications.
- **Shift budget from infrastructure to Web application security** – With the proper resource allocation, corporate risk can be dramatically reduced.
- **Community -** get personally involved in supporting OWASP, leverage the projects that can help you **Open Software Assurance Maturity Model (Open SAMM)**

Thank You

Tom Brennan
tom.brennan@whitehatsec.com

Full Report Available

https://www.whitehatsec.com/home/assets/WPstats_spring10_9thFR.pdf