VERACODE

Data Mining a Mountain of Vulnerabilities

Chris Wysopal

HITB Kuala Lumpur– October 10, 2012

10 Biggest Breaches of 2011

Organization	Cause	Application Vulnerability Details
Comodo	Application Vulnerability	SQL Injection
RSA	Spearphishing + Application Vulnerability	Memory corruption
Epsilon	Unknown Hack	
Wordpress	Unknown Hack	
Sony Playstation Network	Application Vulnerability	Unknown
Citibank	Application Vulnerability	Authorization Error
Cyworld	Unknown Hack	
Tricare	Lost media	
Facebook	Social Engineering	
Steam	Unknown Hack	

Why so many application related breaches?

Question:

Who would release a product riddled with security problems simply to make money?

Answer:

Pretty much every vendor out there.

- Andrew Hay, Senior Security Analyst

451 Research

Building a Secure Application

- ✓Even educated developers make mistakes
- ✓It is difficult but easier than in the past
- ✓Automation can detect and point to about 2/3 of the top vulnerability categories
- ✓It's a dereliction of duty to not perform adequate security testing before shipping

Waterholing trend

- ✓Attackers increasing vectors for breaching perimeter security:
  - ✓Bribe insider
  - ✓Removeable media (USB. The floppy is back)
  - ✓Email attachment
  - ✓Compromised website: the waterhole.

- ✓RSA recently reported on VOHO campagin
- ✓Could waterholes overtake spearphishing?

So let’s mine some data!

The Data Set

- ✓ Applications from over 300 commercial and US government customers
- ✓ **Scanned** 9,910 applications over past 18 months
- ✓ Ranged in size from 100KB to 6GB
- ✓ Software was pre-release and in production
- ✓ Internally built, outsourced, open source, and commercial ISV code

Application Metadata

- Industry vertical
- Application supplier (internal, third-party, etc.)
- Application type
- Assurance level
- Language
- Platform

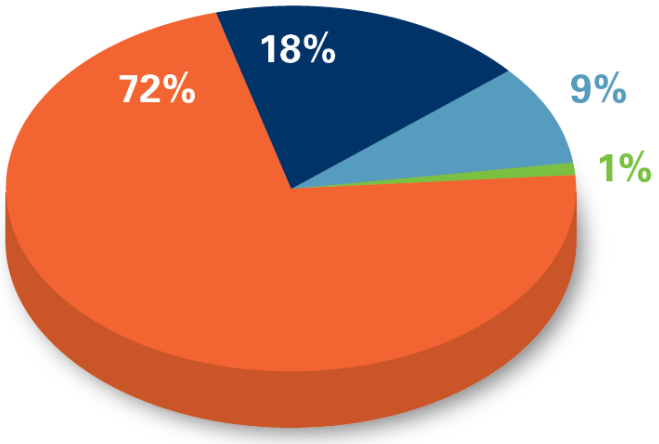
Scan Data

- Scan number
- Scan date
- Lines of code
- Flaw type

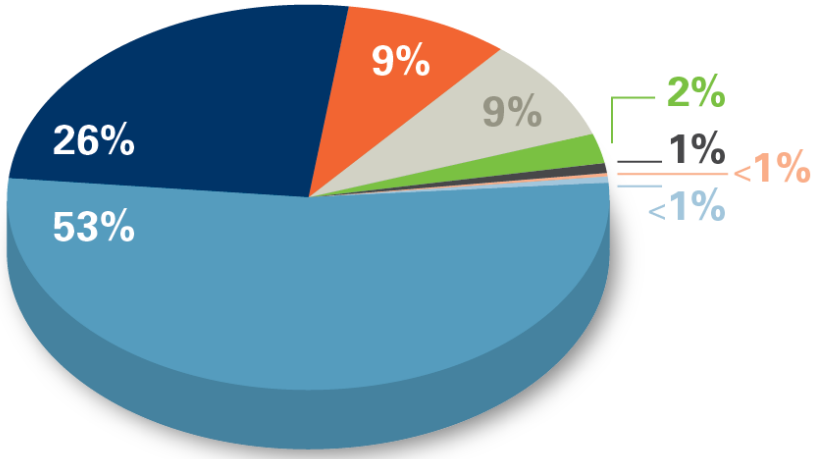
Application Security Metrics

- Flaw counts
- Flaw percentages
- Application count
- Risk-adjusted rating
- First scan acceptance rate
- Time between scans
- Days to remediation
- Scans to remediation
- CWE/SANS Top25 (pass/fail)
- OWASP Top Ten (pass/fail)
- Custom policies

Applications by Supplier Type

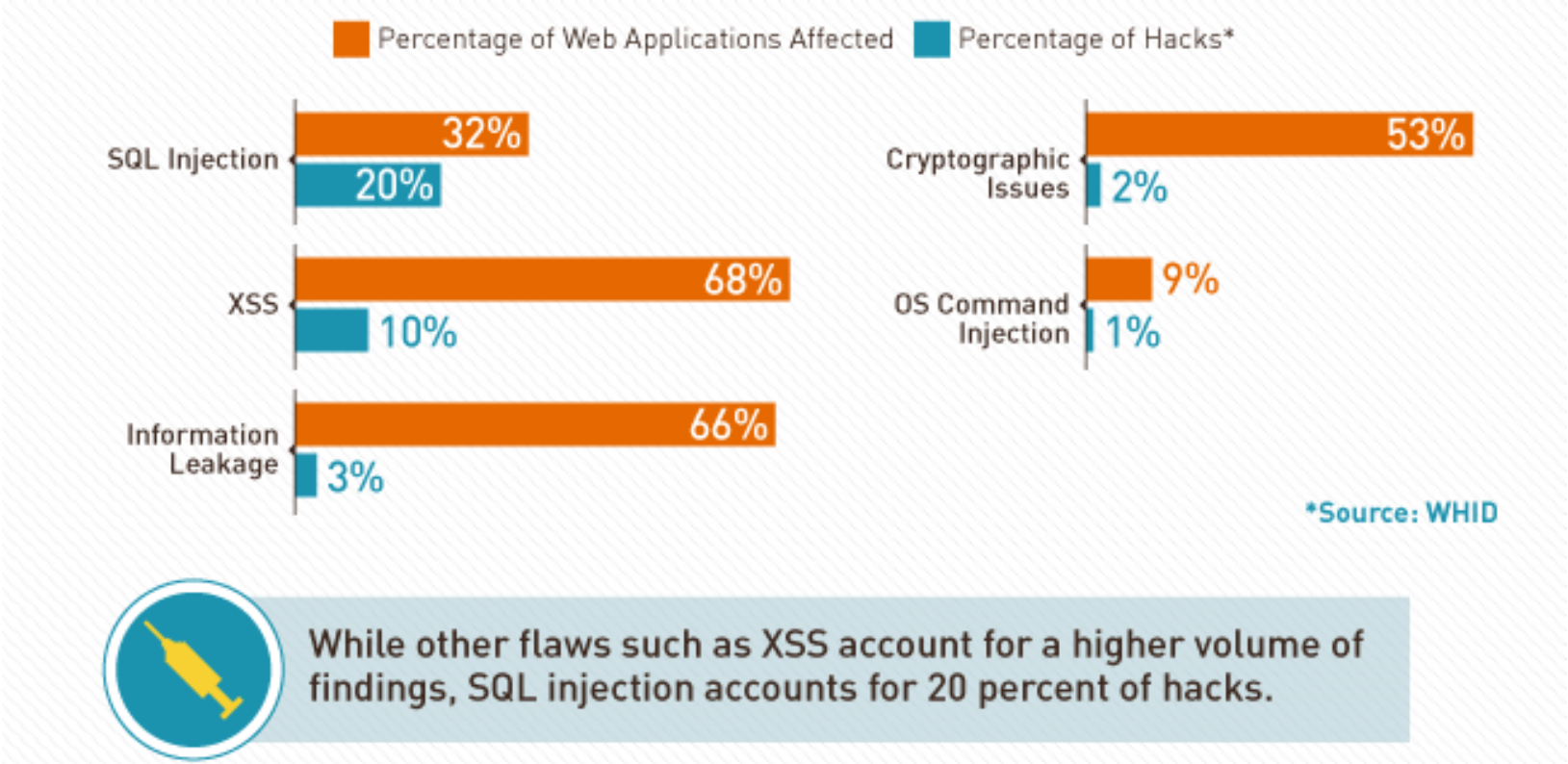


Applications by Language Family

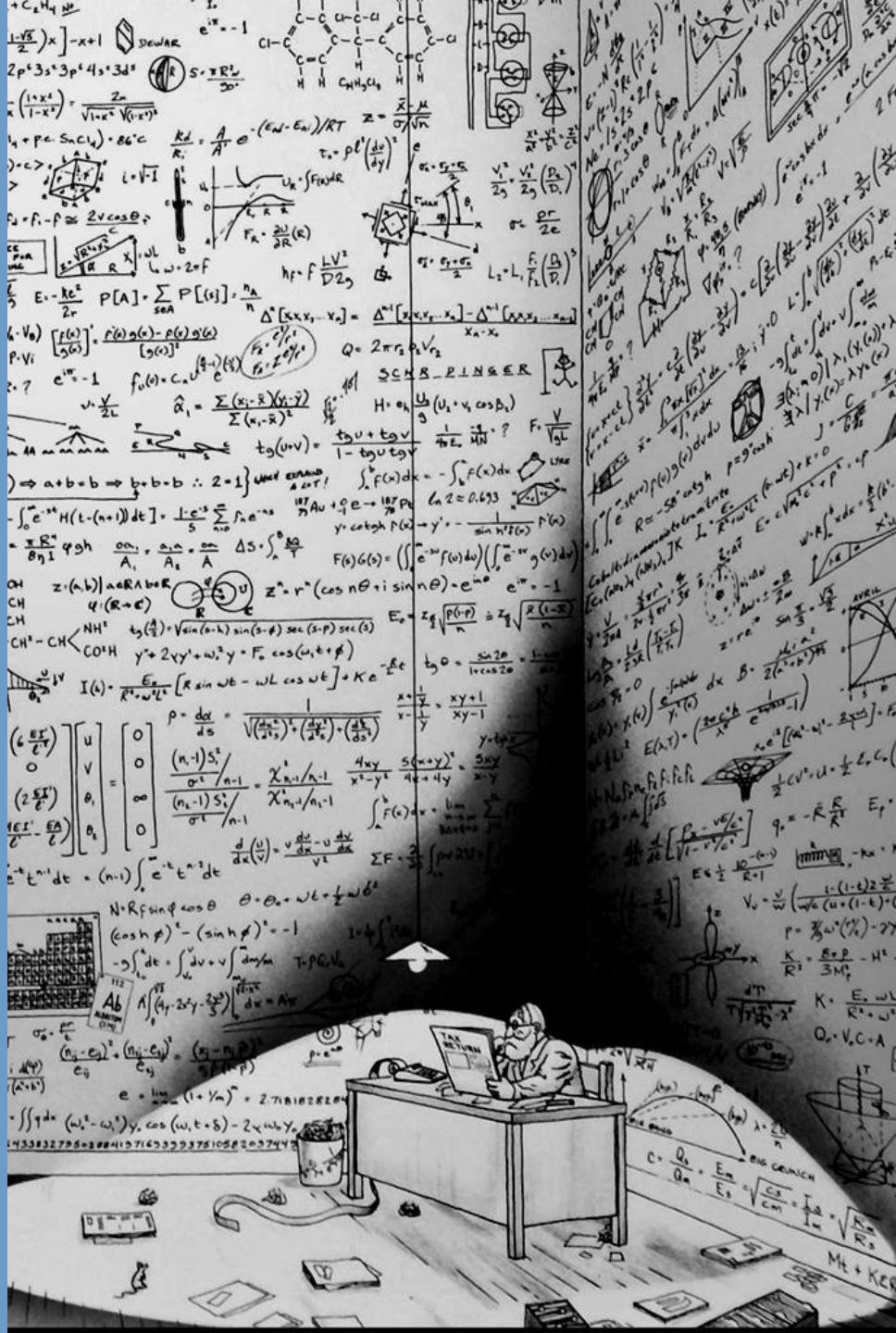


The latent Vulnerabilies vs. The Attacks

Top 5 Attacked Web Application Vulnerabilities

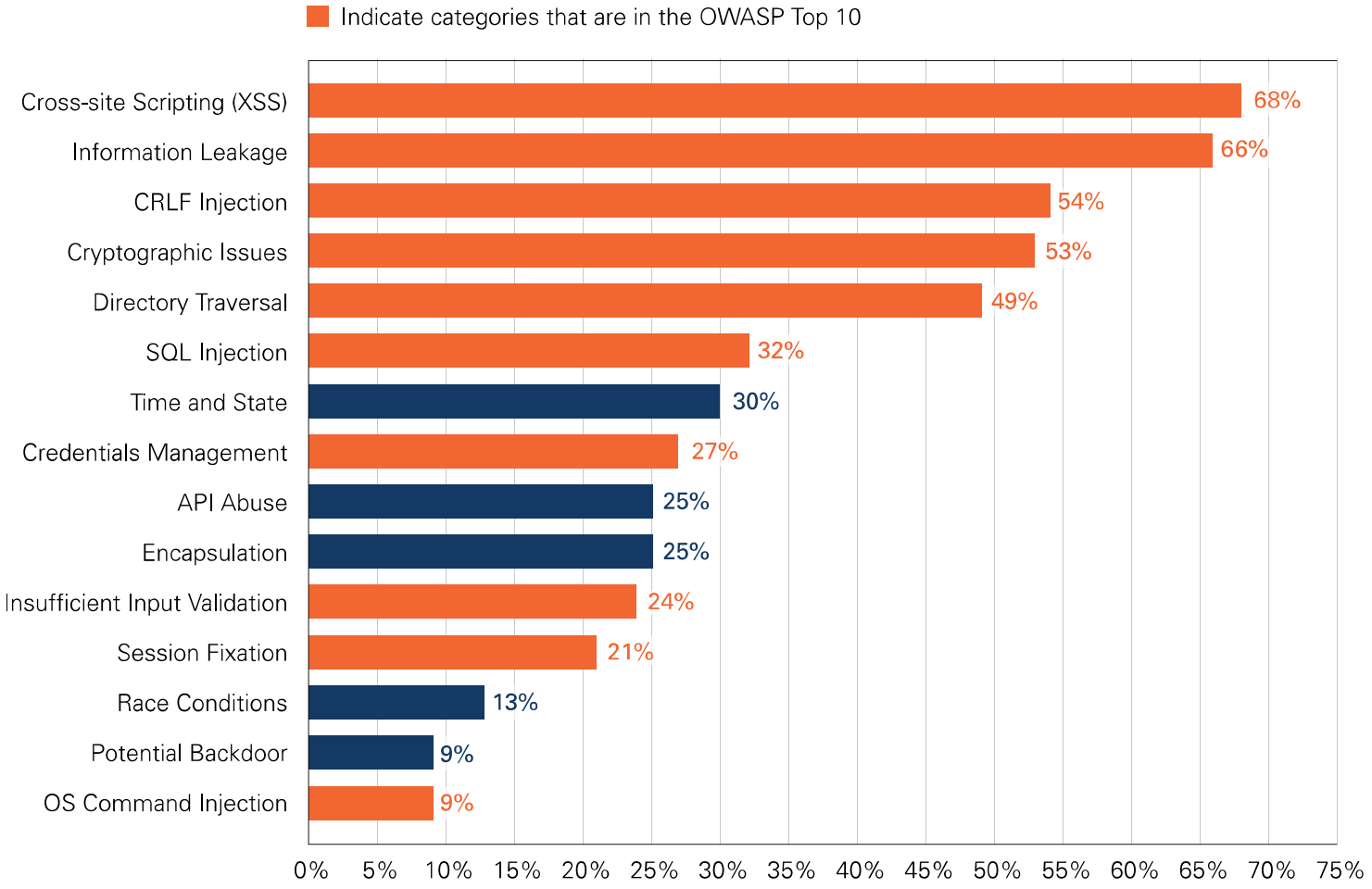


Let's take a closer look at the numbers

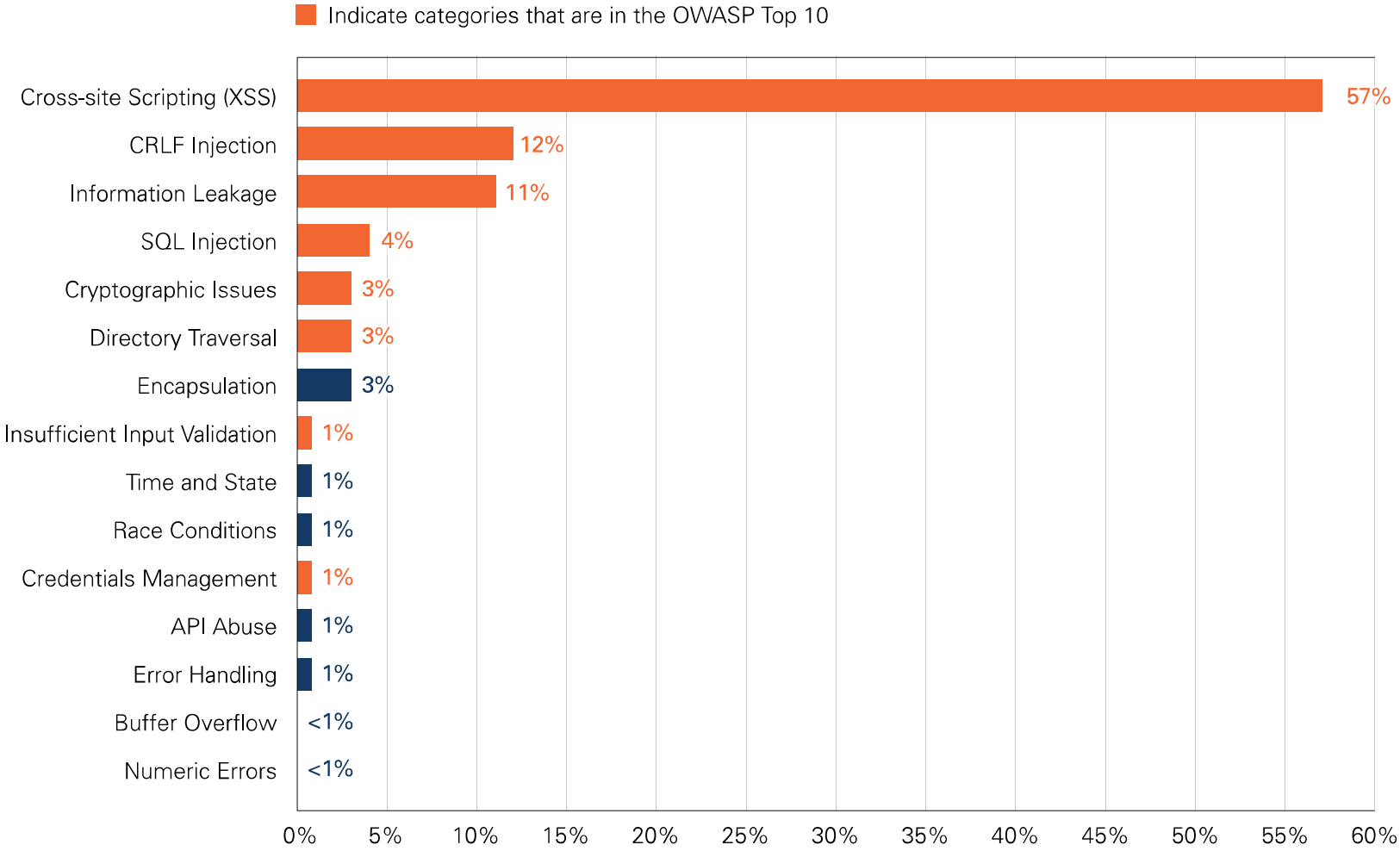


Top Vulnerability Categories

(Percent of Applications Affected for Web Applications)

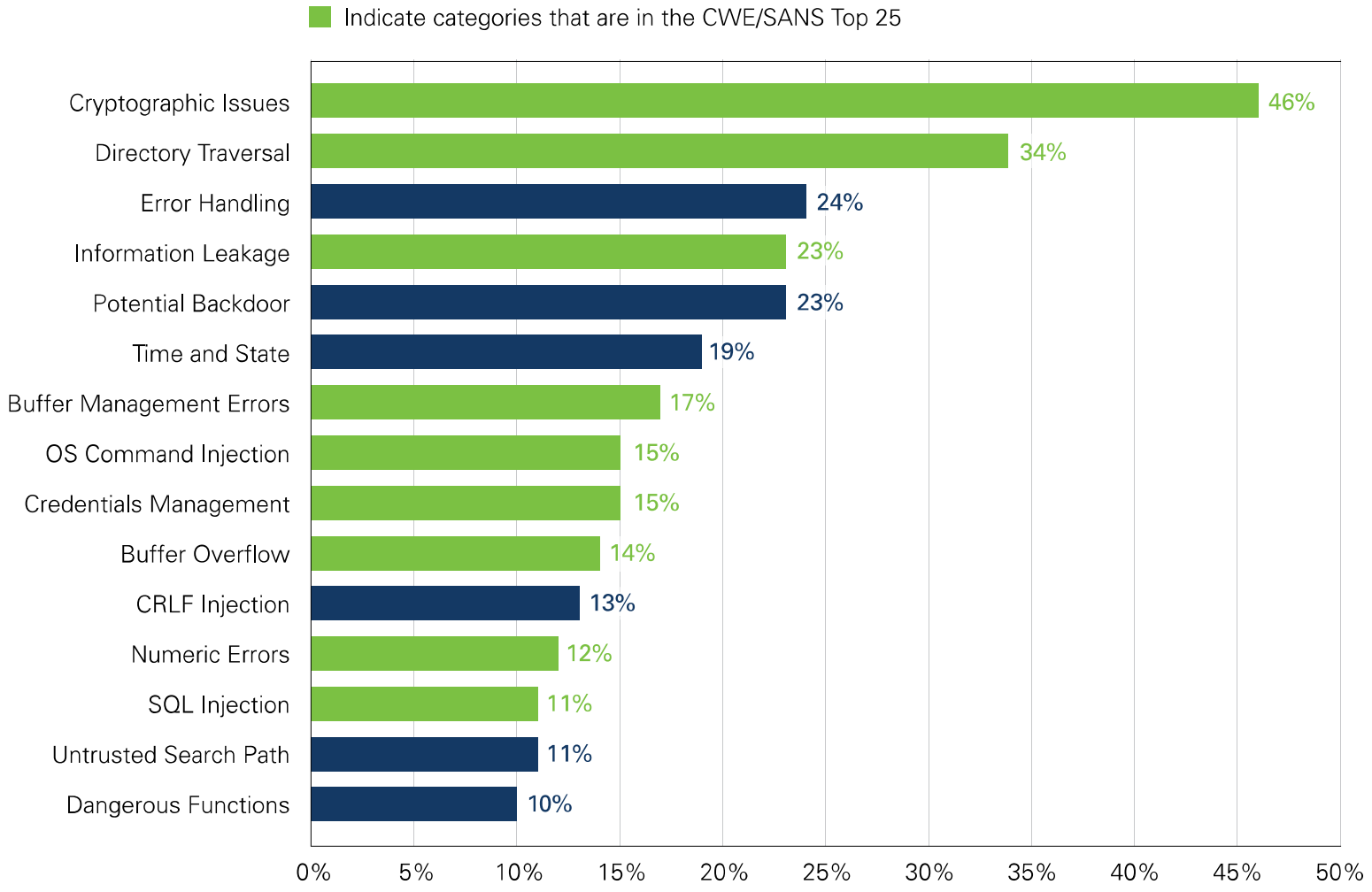


Top Vulnerability Categories
(Overall Prevalence for Web Applications)



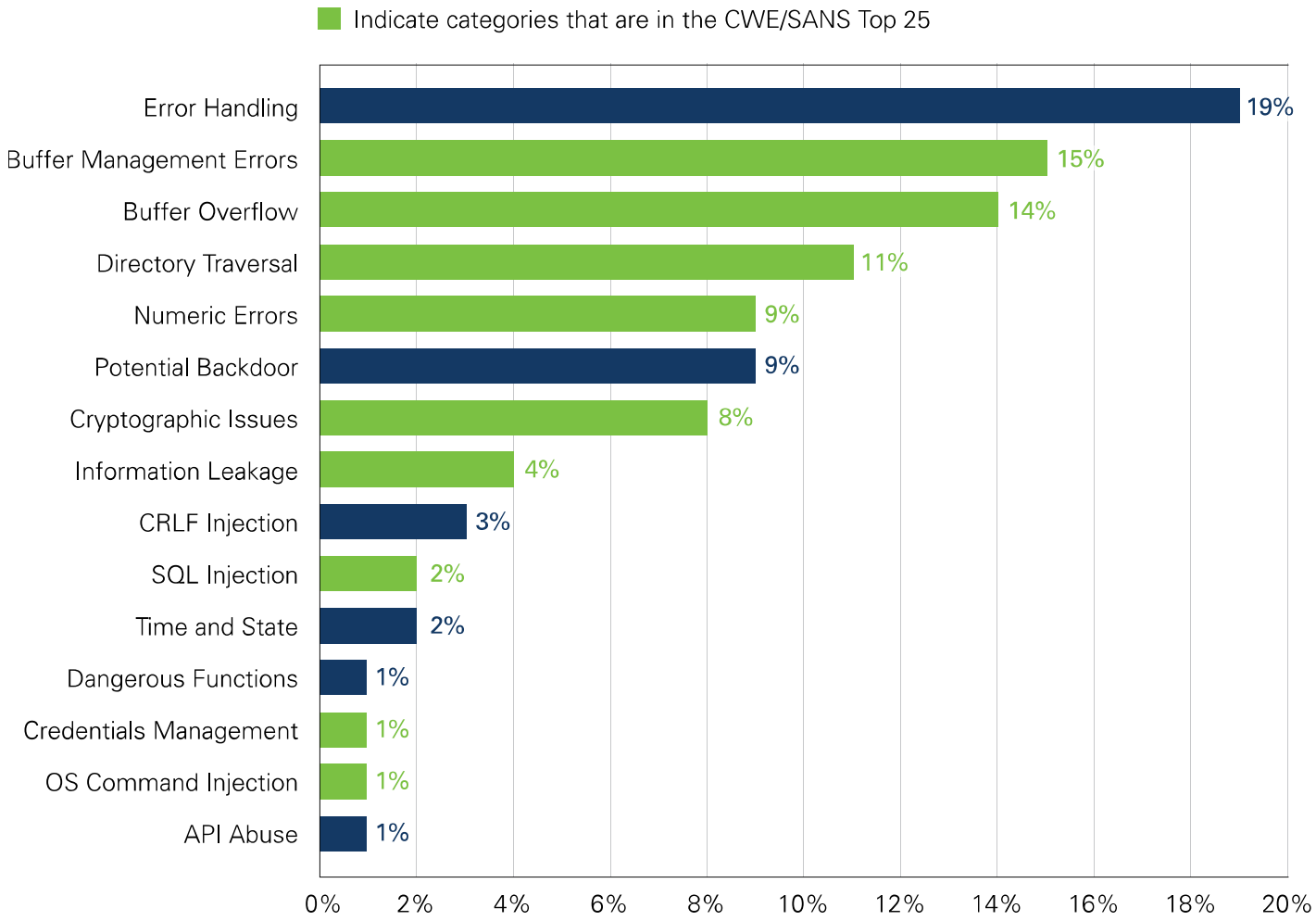
Top Vulnerability Categories

(Percentage of Applications Affected for Non-Web Applications)

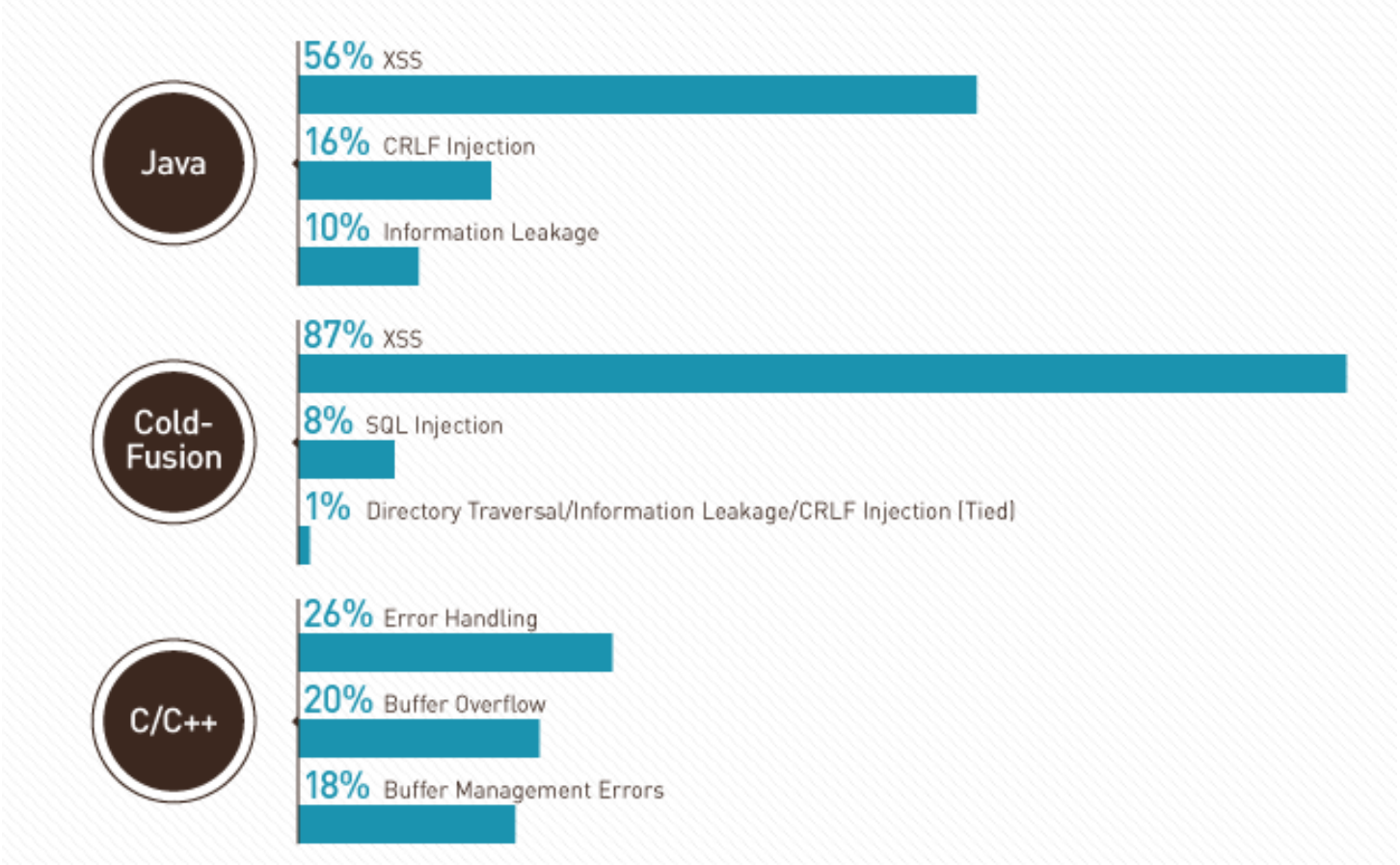


Top Vulnerability Categories

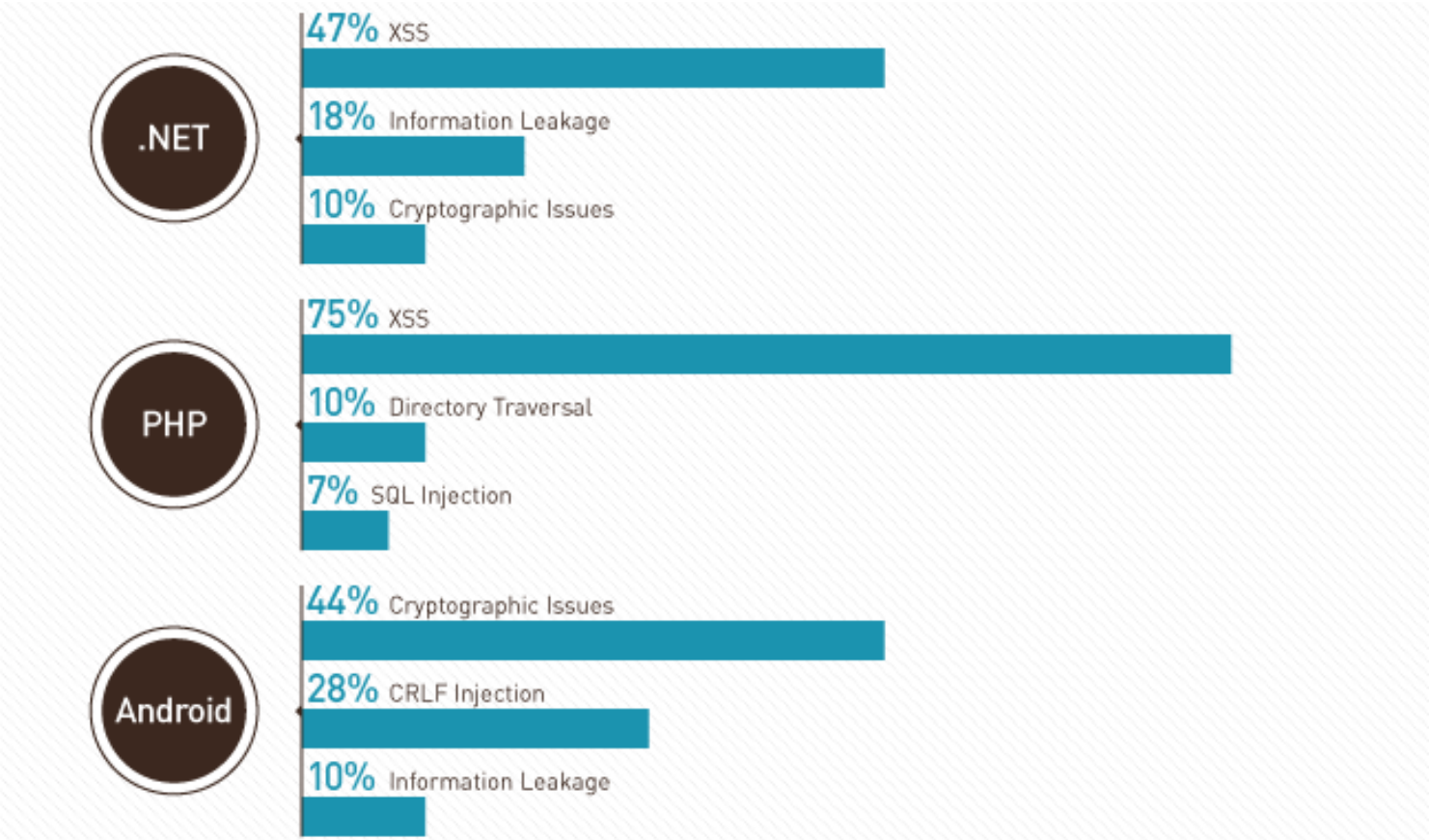
(Overall Prevalence for Non-Web Applications)



Top 3 Vulnerabilities by Language



Top 3 Vulnerabilities by Language



Different developers deliver different vulns

Vulnerability Distribution by Supplier

Internally Developed		Commercial		Open Source		Outsourced*	
Cross-site Scripting (XSS)	58%	Cross-site Scripting (XSS)	44%	Cross-site Scripting (XSS)	41%	CRLF Injection	47%
CRLF Injection	12%	Information Leakage	11%	Directory Traversal	13%	Cross-site Scripting (XSS)	28%
Information Leakage	10%	CRLF Injection	8%	Information Leakage	13%	Information Leakage	6%
SQL Injection	4%	Directory Traversal	6%	CRLF Injection	11%	Encapsulation	6%
Cryptographic Issues	3%	Error Handling	5%	Cryptographic Issues	8%	Cryptographic Issues	5%
Encapsulation	3%	Cryptographic Issues	5%	SQL Injection	3%	Credentials Mgmt	3%
Directory Traversal	3%	Buffer Mgmt Errors	4%	Error Handling	2%	Directory Traversal	2%
Insufficient Input Validation	1%	Buffer Overflow	3%	Time and State	2%	API Abuse	1%
Time and State	1%	Potential Backdoor	3%	API Abuse	2%	Time and State	1%
Race Conditions	1%	SQL Injection	3%	Insufficient Input Validation	1%	Insufficient Input Validation	1%

Table 2: Vulnerability Distribution by Supplier

(*Small sample size)*

Different industries accept different vulns

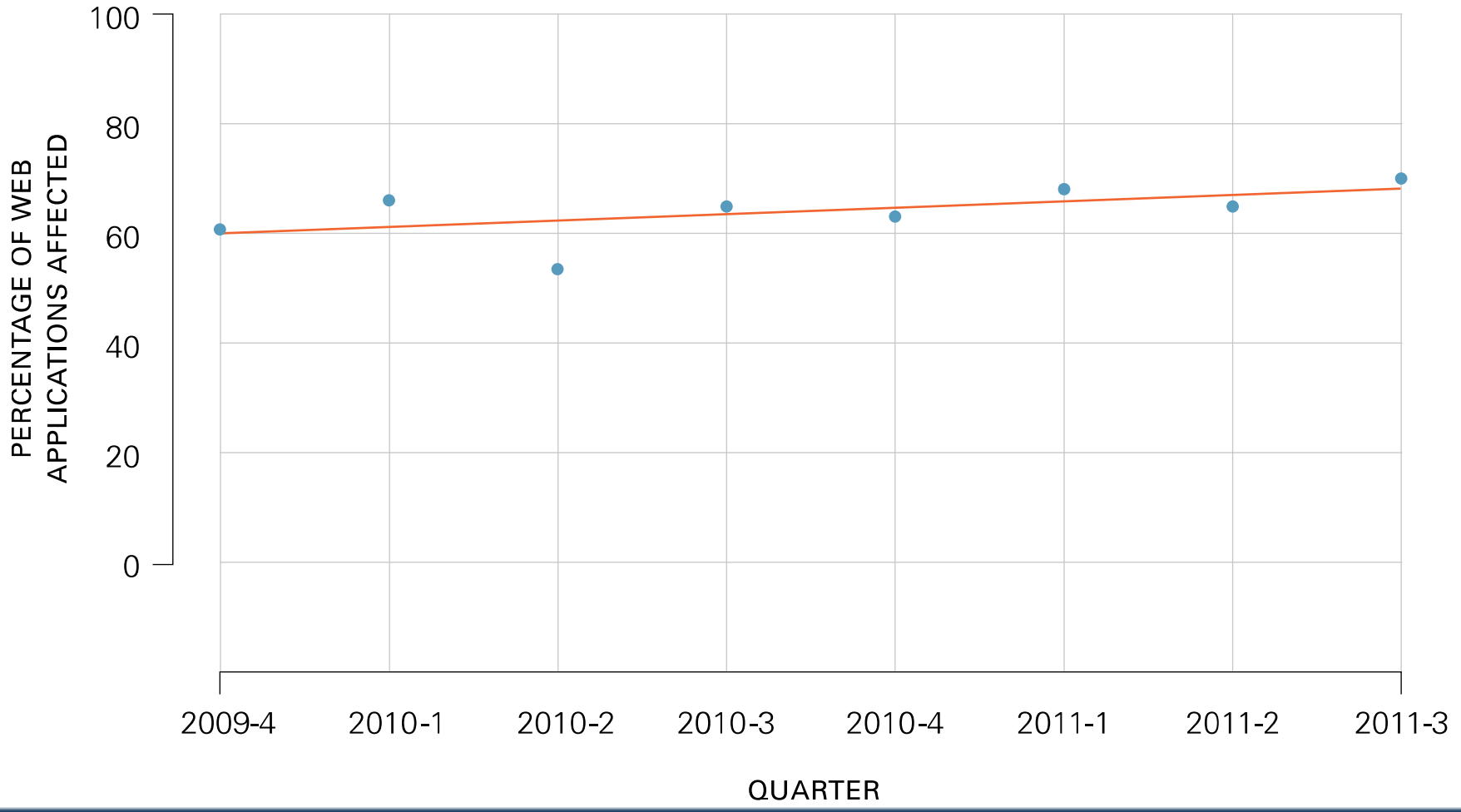
Vulnerability distribution by industry

Government ▾		Finance		Software	
Cross-site Scripting (XSS)	75%	Information Leakage	68%	Cryptographic Issues	59%
Information Leakage	66%	Cross-site Scripting (XSS)	67%	Information Leakage	59%
SQL Injection	40%	Cryptographic Issues	53%	Cross-site Scripting (XSS)	55%
Cryptographic Issues	35%	CRLF Injection	51%	CRLF Injection	54%
Directory Traversal	31%	Directory Traversal	47%	Directory Traversal	54%
Insufficient Input Validation	27%	Insufficient Input Validation	30%	Time and State	39%
CRLF Injection	27%	SQL Injection	29%	Credentials Mgmt	31%
OS Command Injection	19%	Time and State	28%	SQL Injection	30%
Time and State	18%	API Abuse	26%	API Abuse	25%
Credentials Mgmt	16%	Encapsulation	25%	Encapsulation	23%
API Abuse	14%	Credentials Mgmt	24%	Session Fixation	18%
Potential Backdoor	12%	Session Fixation	19%	OS Command Injection	14%
Session Fixation	11%	Race Conditions	13%	Potential Backdoor	14%
Encapsulation	11%	Potential Backdoor	10%	Race Conditions	13%
Untrusted Search Path	3%	OS Command Injection	6%	Insufficient Input Validation	13%

Are DEVEL0PERs making any progress at eradicating cross-site scripting or sql injection?

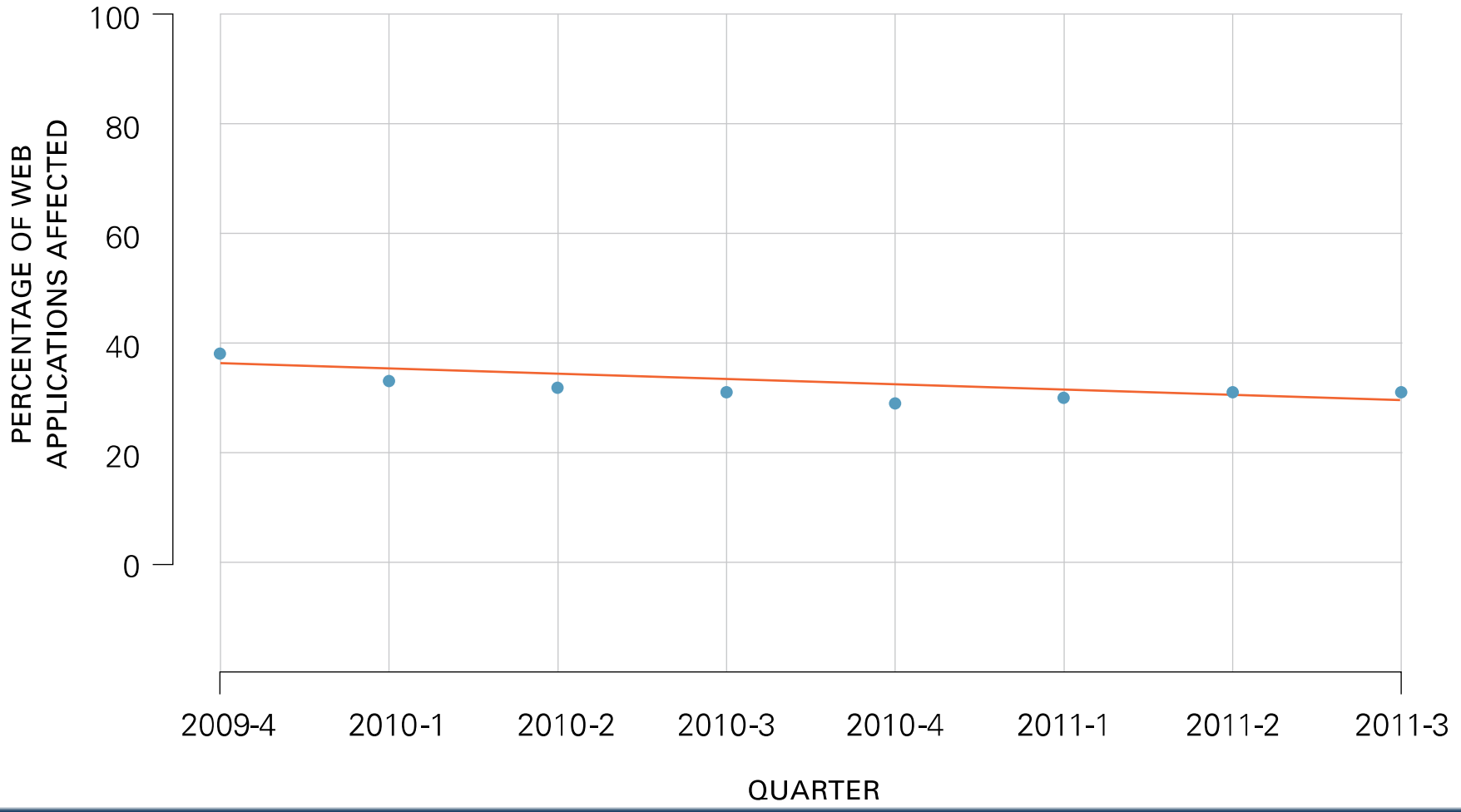
Quarterly Trend for XSS

pvalue = 0.124: Statistically, the trend is flat.



Quarterly Trend for SQL Injection

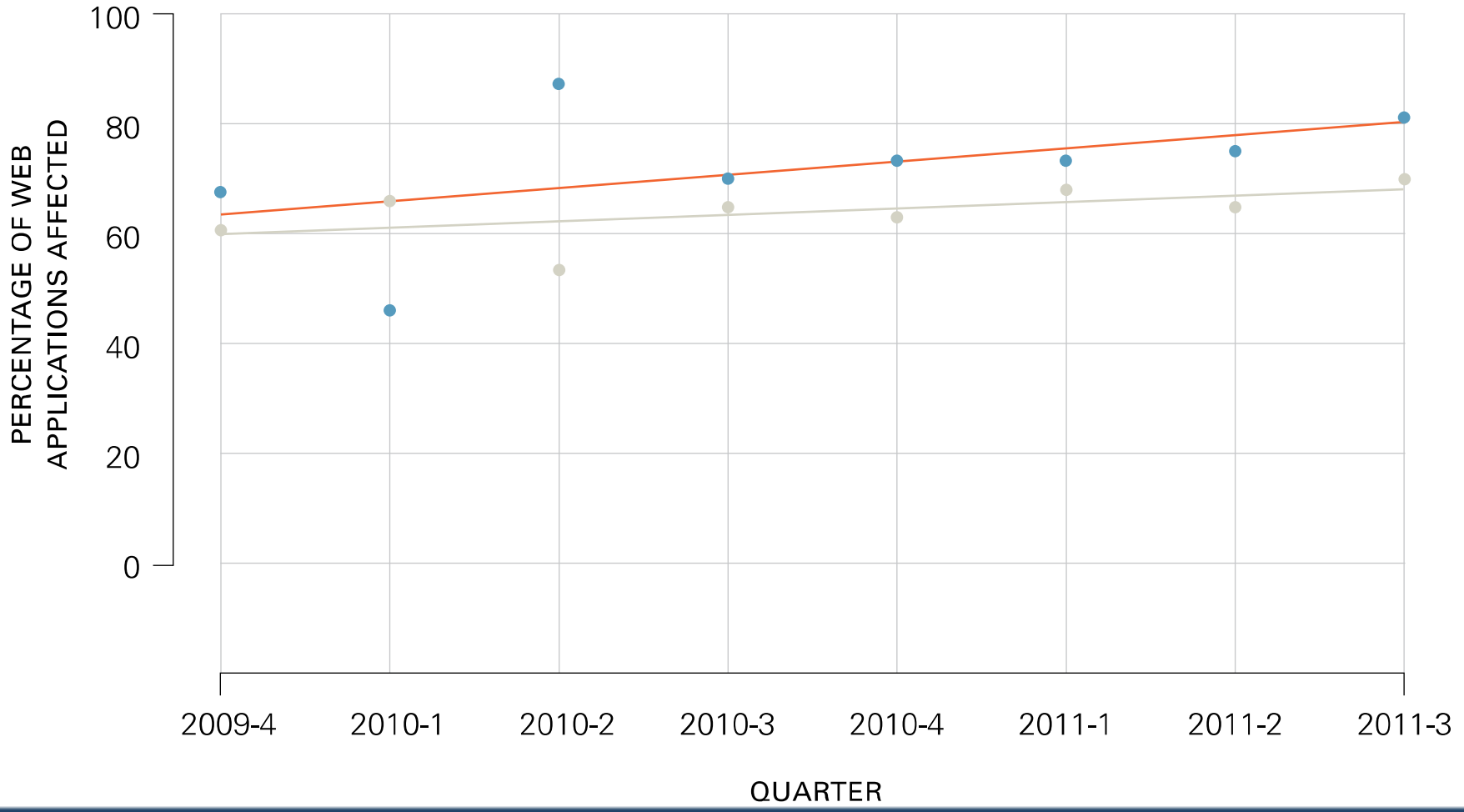
pvalue = 0.048: Statistically, the trend is down.



Dare we ask, How is the U.S. government sector doing?

Quarterly Trend for XSS in Government Web Applications

pvalue = 0.215: Statistically, the trend is flat.



What percentage of WEB applications fail OWASP TOP TEN?

a) 34%

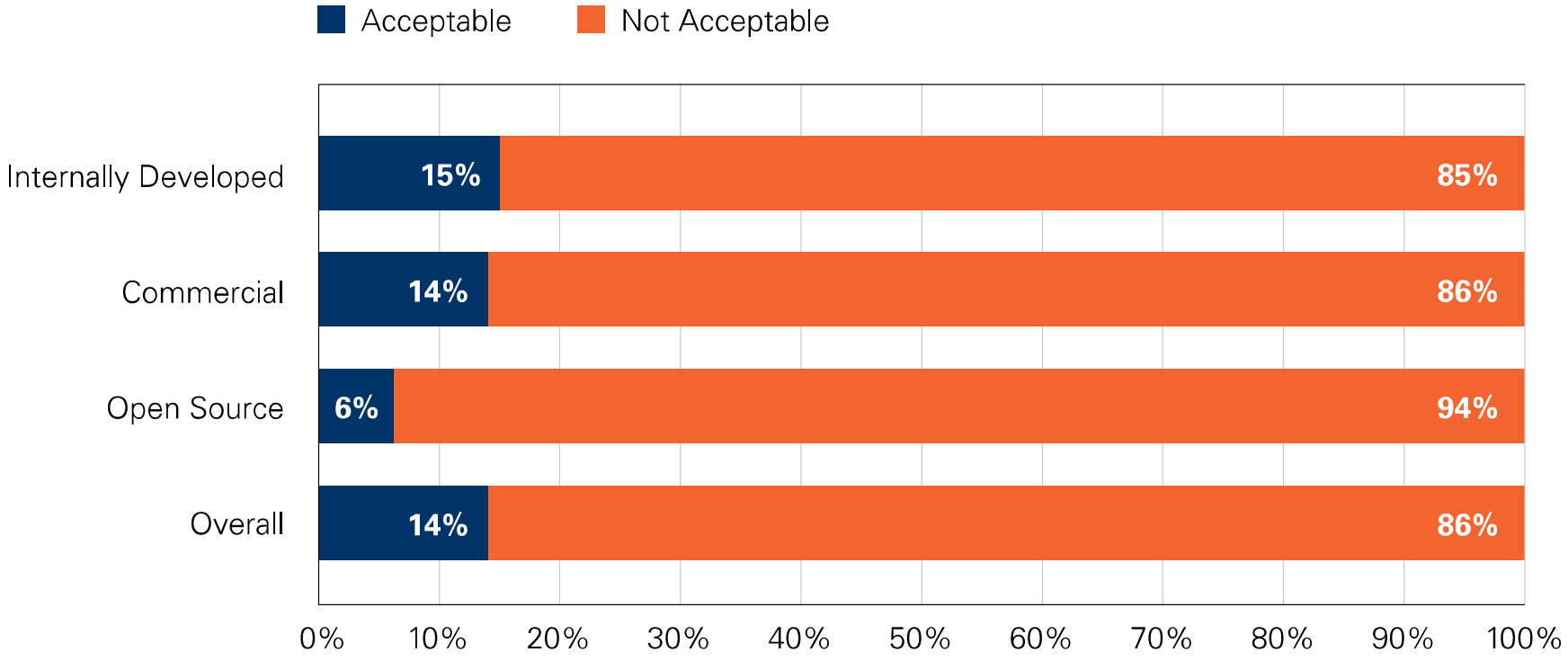
b) 57%

c) 86%

d) 99%

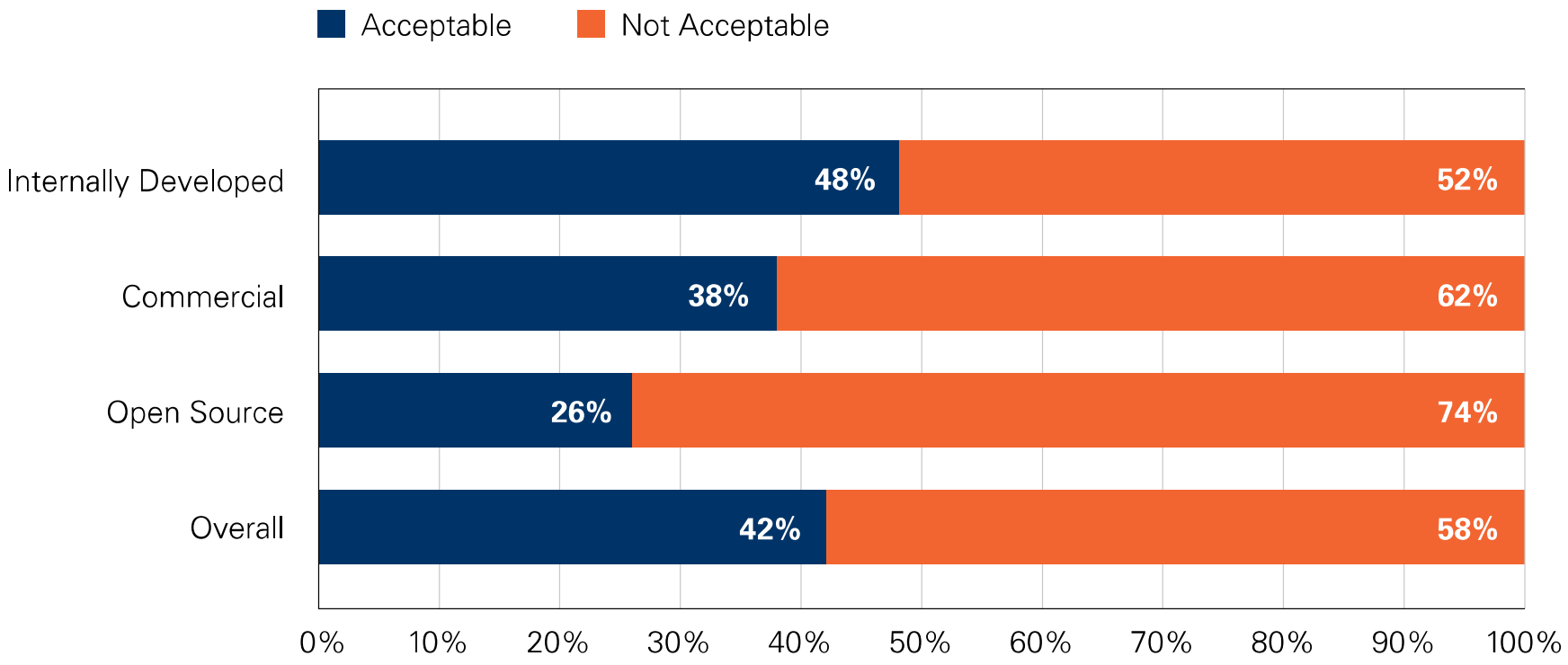
OWASP Top 10 Compliance by Supplier on First Submission

(Web Applications)



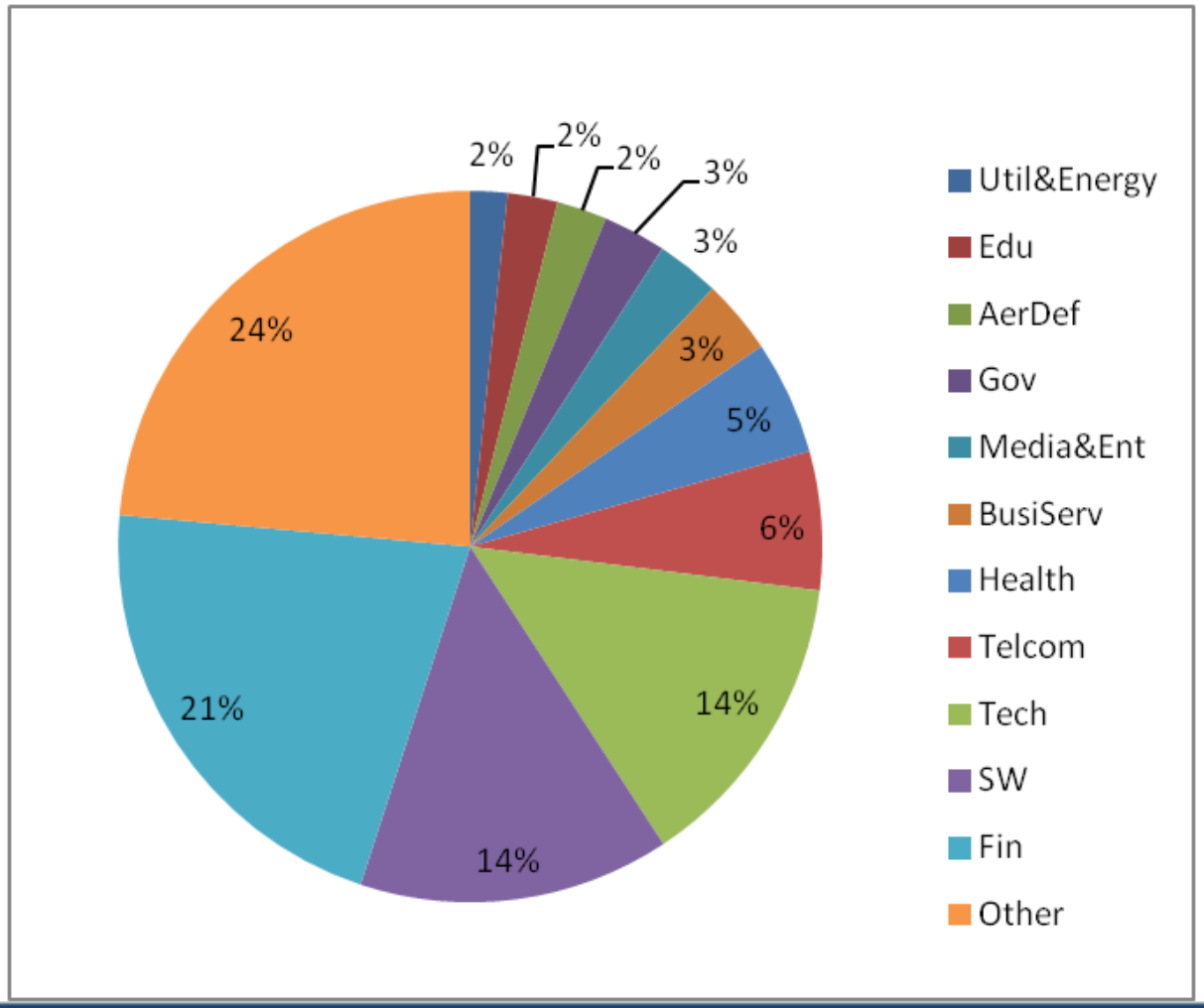
CWE/SANS Top 25 Compliance by Supplier on First Submission

(Non-Web Applications)

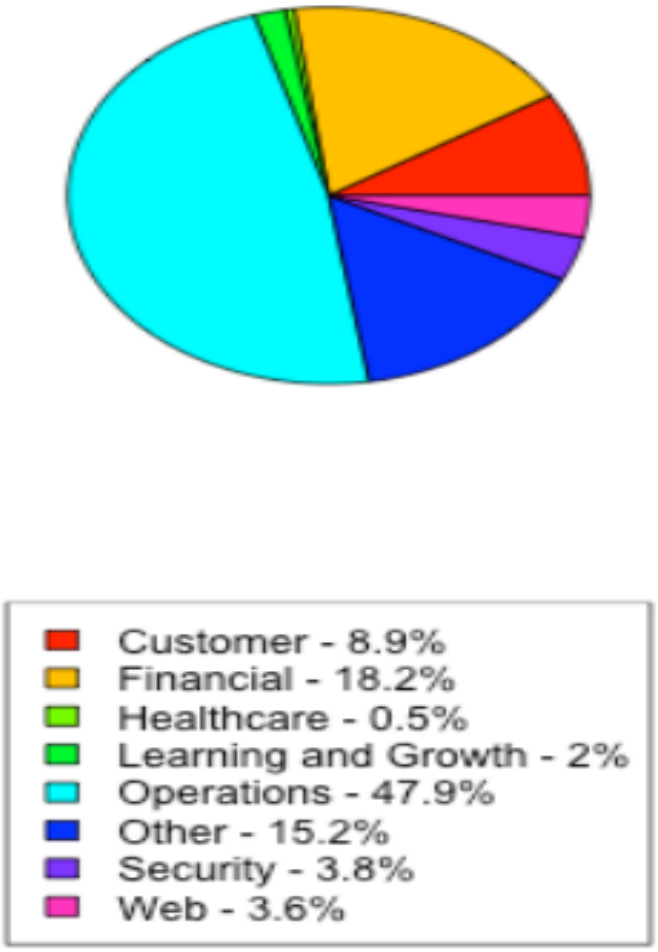


Who is holding their software vendors accountable?

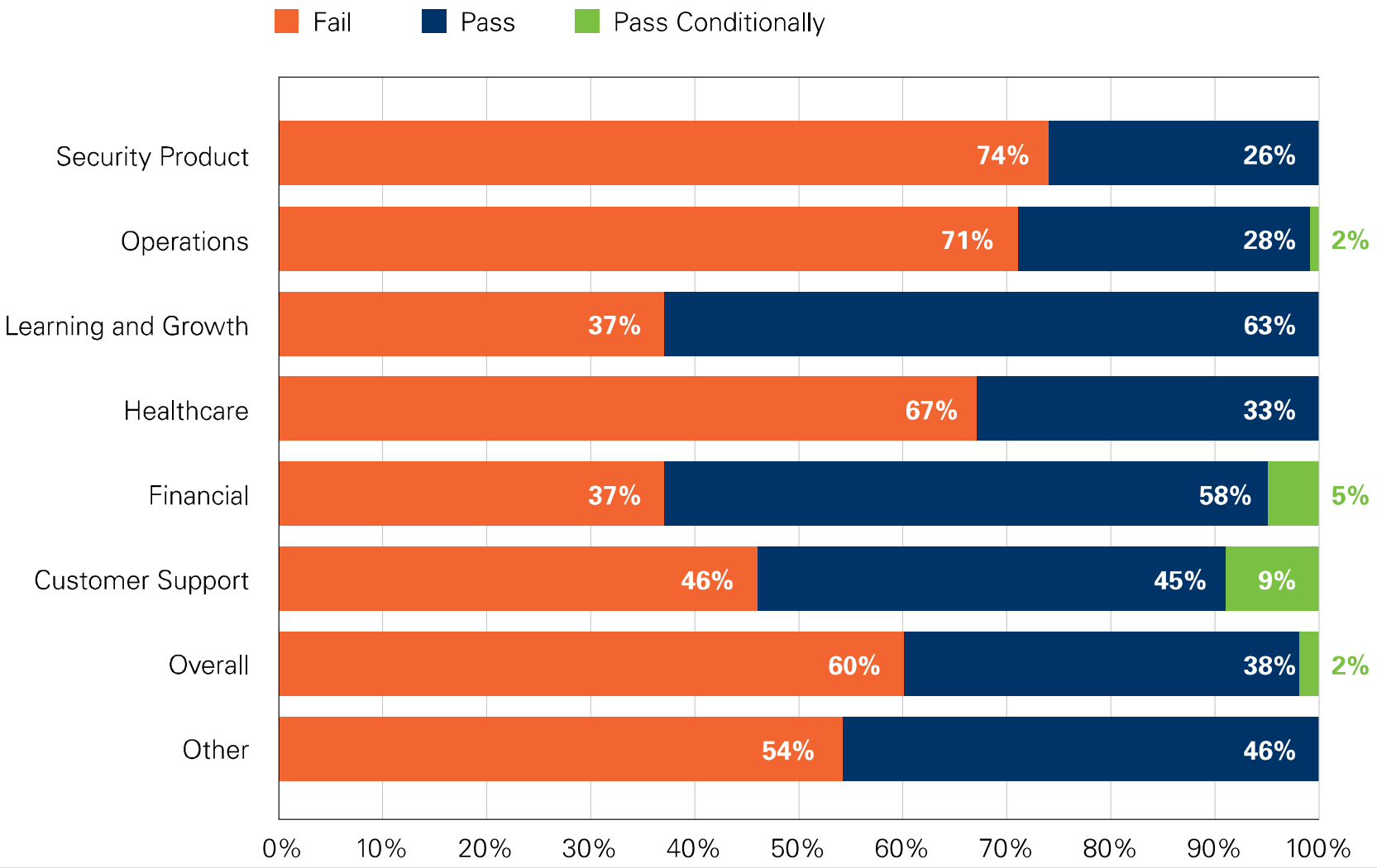
Enterprise Industries



3rd Party Application Purpose



Performance Against Enterprise Policy by Application Purpose



No discernable difference in security quality score on first submission of applications from public and private software companies. Contrary to expectation!

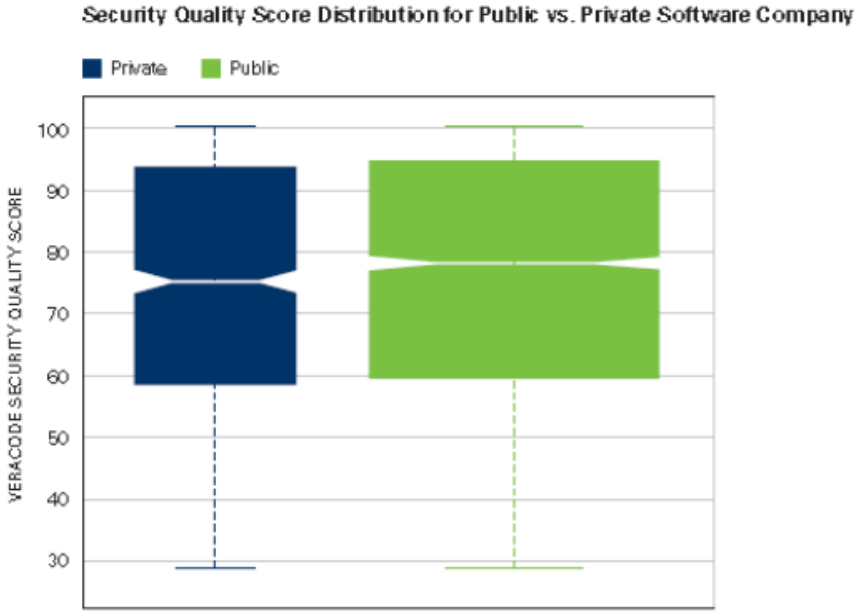


Figure 25: Security Quality Score Distribution for Public vs. Private Software Company

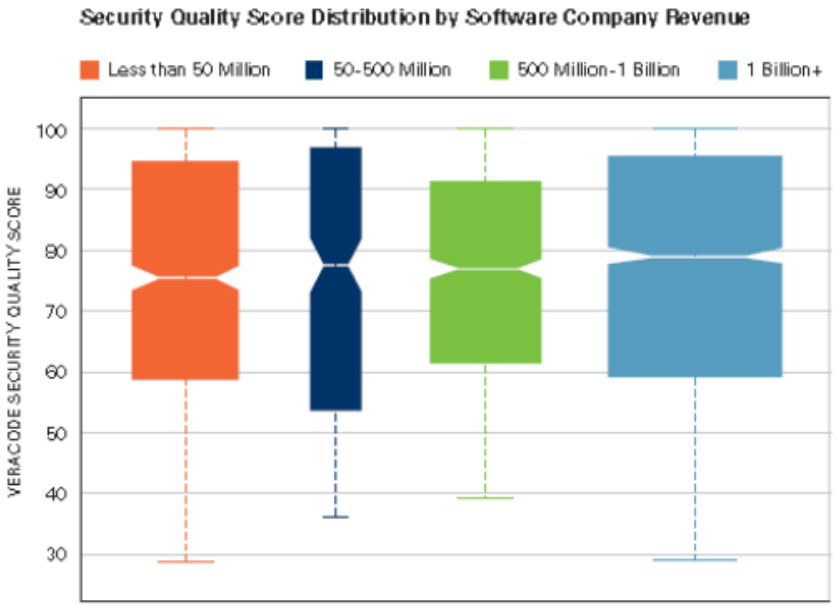
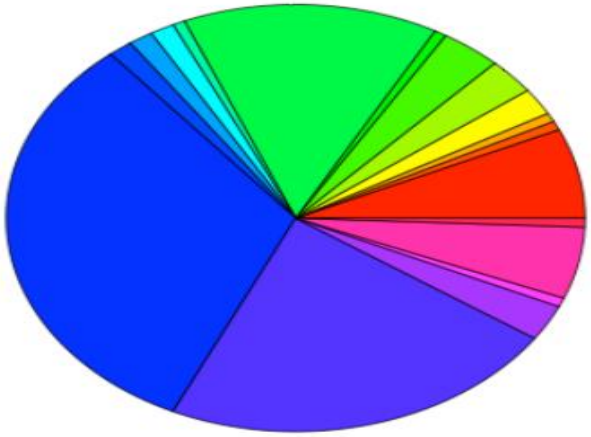


Figure 26: Security Quality Score Distribution by Software Company Revenue

No discernable difference in security quality score on first submission of applications from software companies in different revenue brackets.

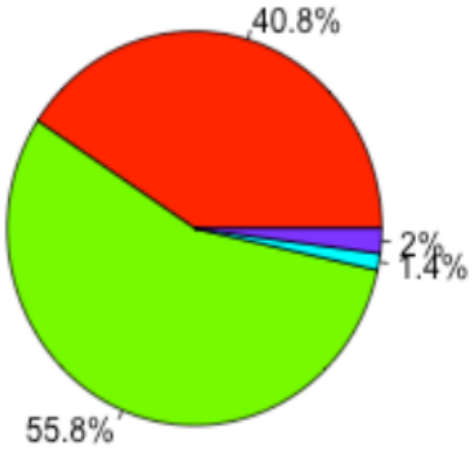
So I hear you can run applications on smart phones?

Distribution by industry



- bank - 6.8%
- busiserv - 0.7%
- comm - 0.7%
- comphard - 2%
- compserv - 2.7%
- compsoft - 3.4%
- engine - 0.7%
- finserv - 14.3%
- food - 0.7%
- health - 1.4%
- hospity - 1.4%
- ins - 1.4%
- media - 32%
- other - 22.4%
- retail - 2.7%
- secprodserv - 0.7%
- tech - 5.4%
- teleserv - 0.7%

Distribution by supplier type



- Commercial - 40.8%
- Internally Developed - 55.8%
- Open Source - 1.4%
- Outsourced - 2%

Percentage of Android Apps Affected

	AffectedAppVerPct
Cryptographic Issues	68.5
CRLF Injection	47.2
Information Leakage	39.1
Time and State	27.9
SQL Injection	23.4
Directory Traversal	10.7
Cross-Site Scripting (XSS)	6.1
Authorization Issues	5.6
Credentials Management	5.1

Percentage of iOS Apps Affected

Language	FlawCat	AffectedAppVerPct
iOS	Error Handling	81.0
iOS	Cryptographic Issues	67.2
iOS	Information Leakage	54.4
iOS	Buffer Management Errors	29.9
iOS	Code Quality	27.7
iOS	Directory Traversal	23.7
iOS	Credentials Management	14.6
iOS	Numeric Errors	10.2
iOS	Buffer Overflow	4.7

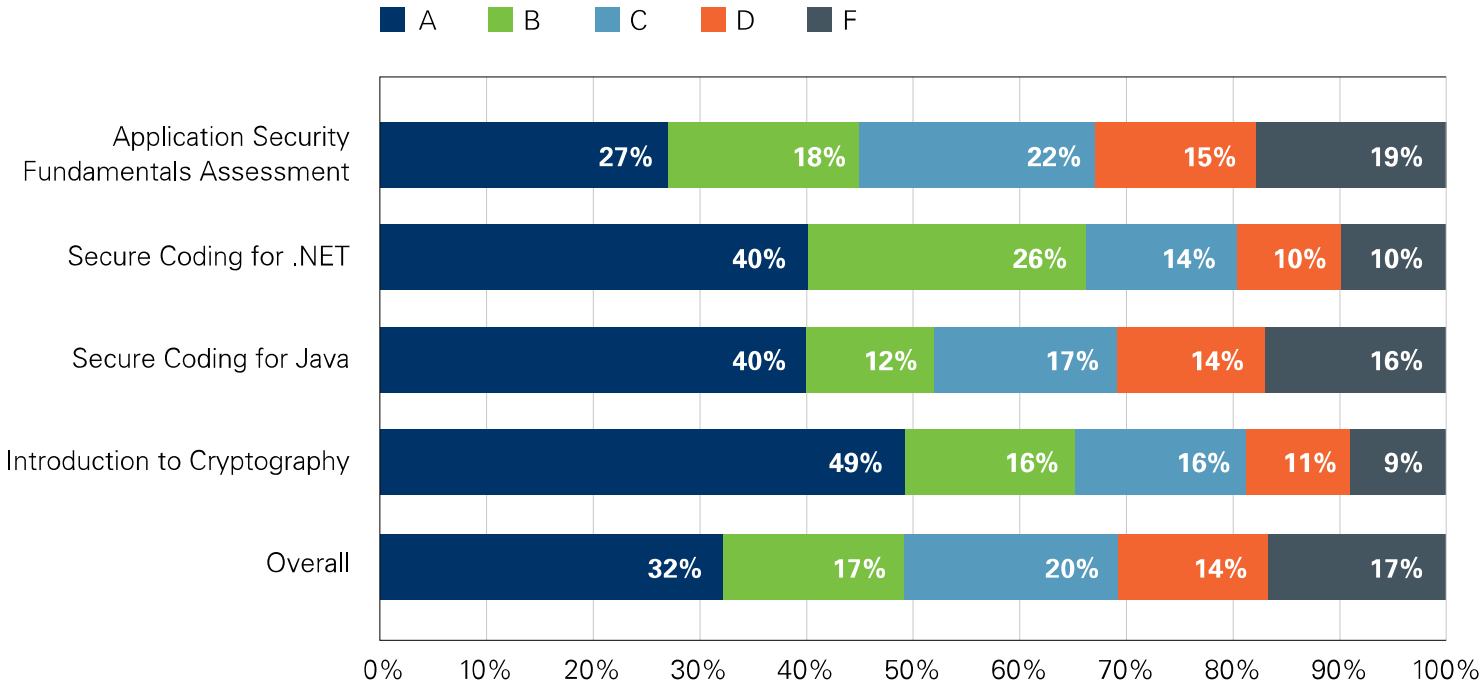
When given an exam on application security fundamentals, over half of developers…

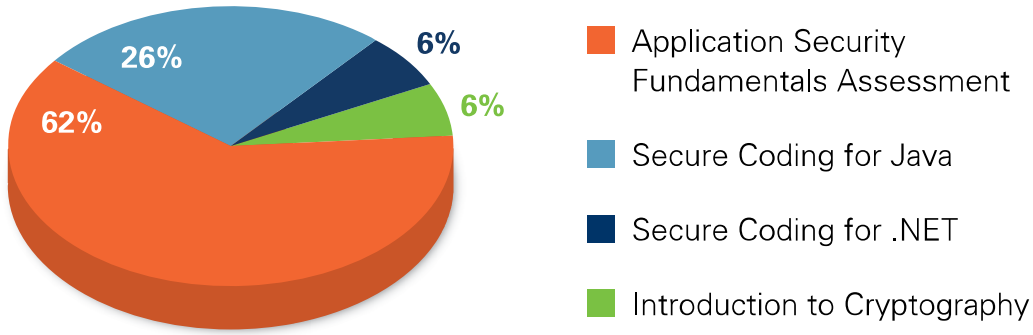
a) Receive an A

b) Receive a B or worse

c) Receive a C or worse

d) Fail (receive a D or F)


A
B
C
D
F
Application Security Fundamentals Assessment
Secure Coding for .NET
Secure Coding for Java
Introduction to Cryptography
Overall
27%
18%
22%
15%
19%
40%
26%
14%
10%
10%
40%
12%
17%
14%
16%
49%
16%
16%
11%
9%
32%
17%
20%
14%
17%
0%
10%
20%
30%
40%
50%
60%
70%
80%
90%
100%


26%
6%
6%
62%
Application Security Fundamentals Assessment
Secure Coding for Java
Secure Coding for .NET
Introduction to Cryptography

Questions?

Chris Wysopal
cwysopal@veracode.com

@weldpond