Exploring the Relationship Between Web Application Development Tools and Security

Matthew Finifter and David Wagner
University of California, Berkeley

It’s a great time to be a developer!

Languages

PHP	Java	Ruby
Perl	Python	Scala
Haskell	Cold Fusion	…

It’s a great time to be a developer!

Languages

PHP	JAVA	RUBY
PERL	PYTHON	SCALA
HASKELL	COLD FUSION	…

Frameworks

Yii, ASP.NET, Zend, Struts, Django, Snap, GWT, RoR, Mason, Sinatra, CakePHP, Fusebox, Catalyst, Spring, Grails, Dancer, CodeIgniter, Tapestry, Pyjamas, Symfony

It’s a great time to be a developer!

Languages

PHP	Java	Ruby
Perl	Python	Scala
Haskell	Cold Fusion	…

- Object Relational Model (ORM) Framework
- Templating Language
- Libraries
- Vulnerability Remediation Tools or Services

Frameworks

Yii, ASP.NET, Zend, Struts, Django, Snap, GWT, RoR, Mason, Sinatra, CakePHP, Fusebox, Catalyst, Spring, Grails, Dancer, CodeIgniter, Tapestry, Pyjamas, Symfony

- Client-side framework
- Meta-framework
- Content Management System (CMS)

Choice is great, but…

- How should a developer or project manager choose?
- Is there any observable difference between different tools we might choose?
- What should you optimize for?
- How will you know you’ve made the right choices?
- We need meaningful comparisons between tools so that developers can make informed decisions.

Talk Outline

- Introduction
- Goals
- Methodology
- Results
- Conclusion and Future Work

Goals

- Encourage future work in this problem space
- Introduce methodology for evaluating differences between tools
- Evaluate **security** differences between different tools
  - Programming Language
  - Web Application Development Framework
  - Process for Finding Vulnerabilities

Methodology

- Secondary data set from [Prechelt 2010]
- Different groups of developers use different tools to implement the same functionality
- Control for differences in specifications, human variability
- Measure the security of the developed programs
  - Black-box penetration testing (Burp Suite Pro)
  - Manual security review
- Use statistical hypothesis testing to look for associations

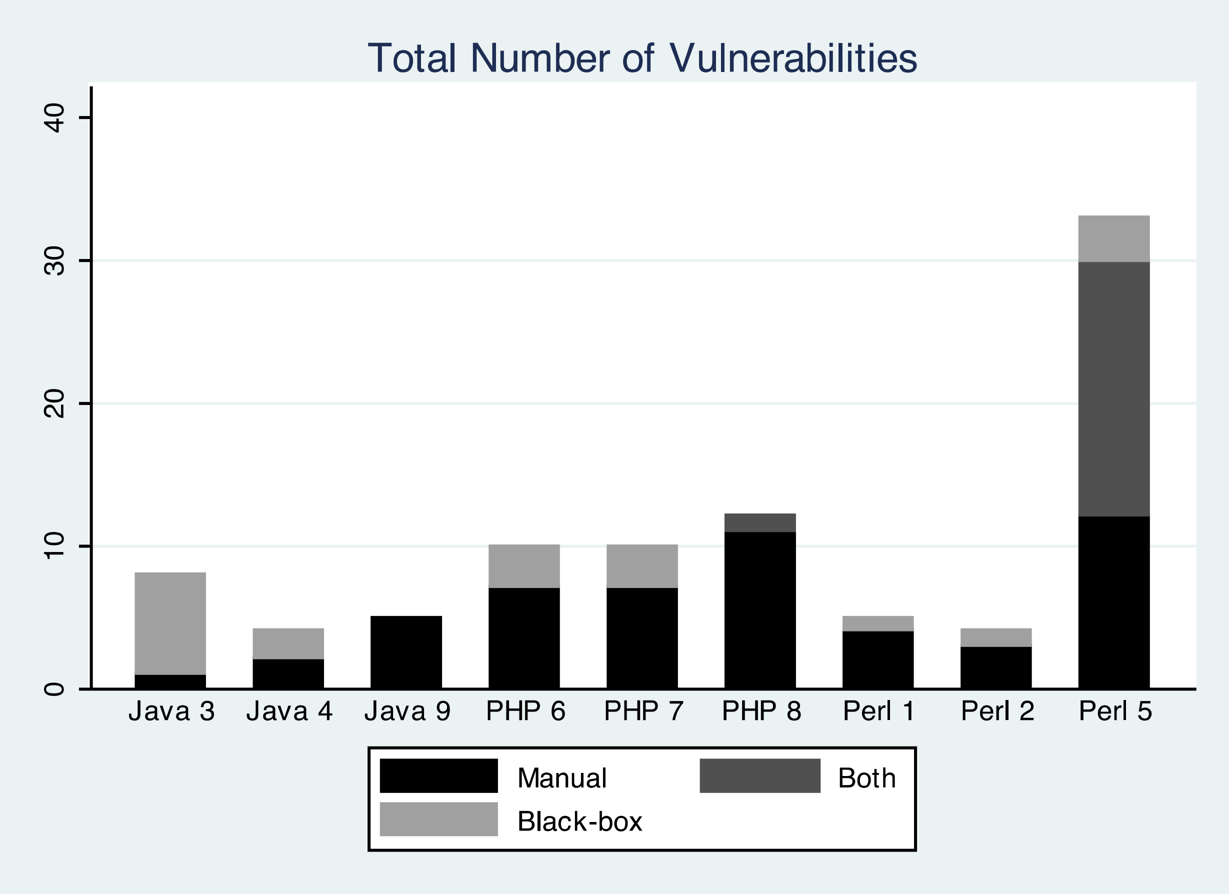
Limitations

- Experimental design
- Only one security reviewer (me)
- Application not necessarily representative
- Small sample size
- … and more (see the paper)

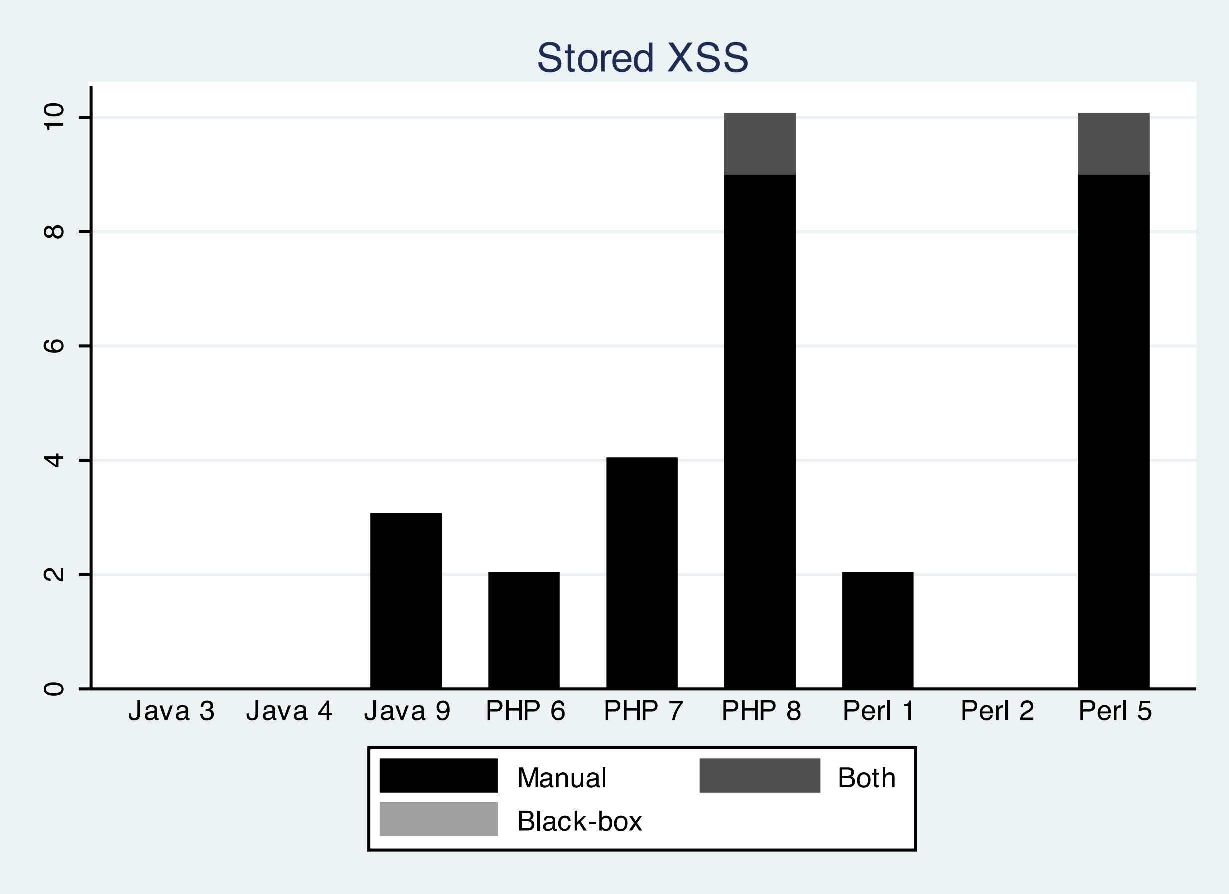
Programming Language

- 3 Java teams, 3 Perl teams, 3 PHP teams
- Look for association between programming language and:
  - Total number of vulnerabilities found in the implementation
  - Number of vulnerabilities for each vulnerability class
- Main conclusion: 9 samples is too few to find these associations.
  - Maybe there is no association
  - Maybe we need more data

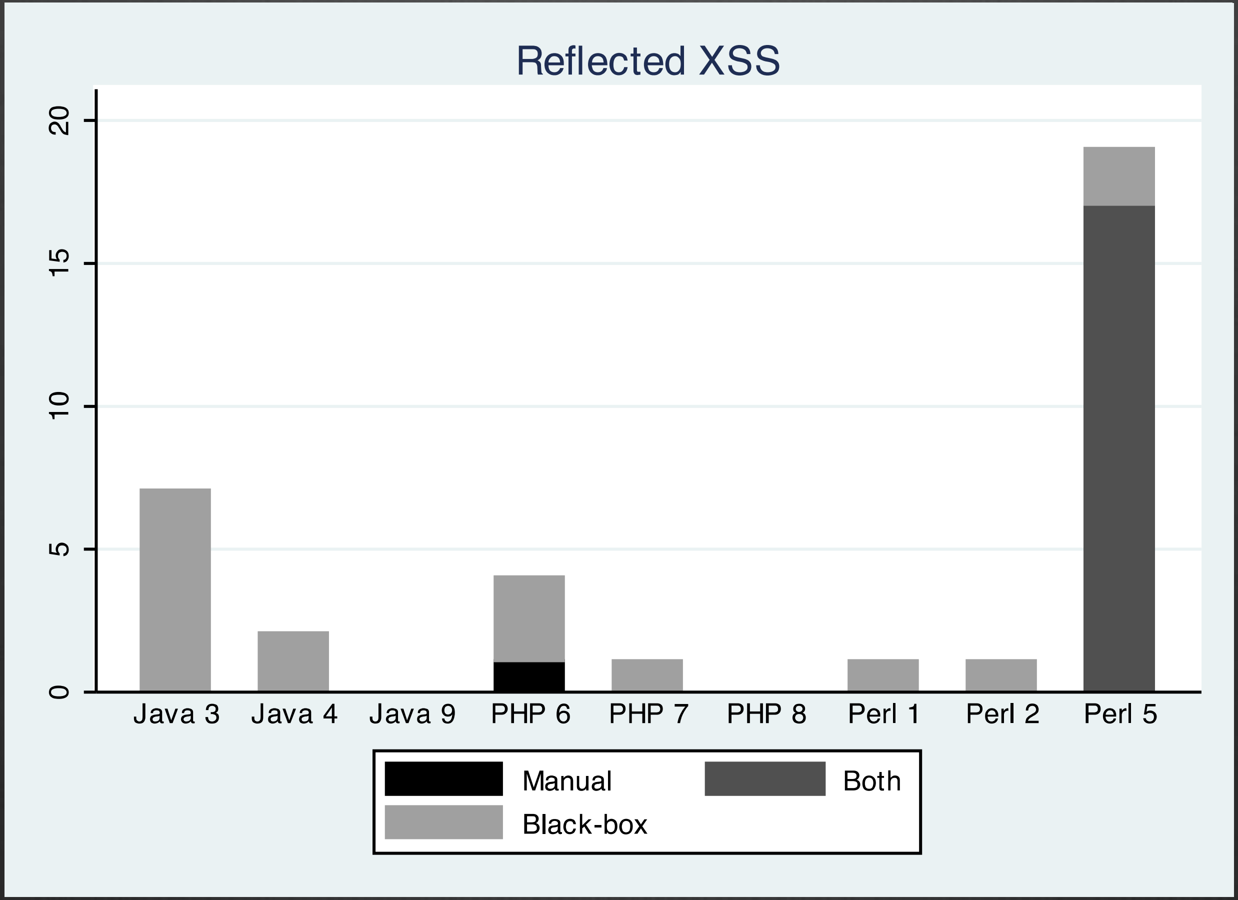
Results: Total Vulnerabilities



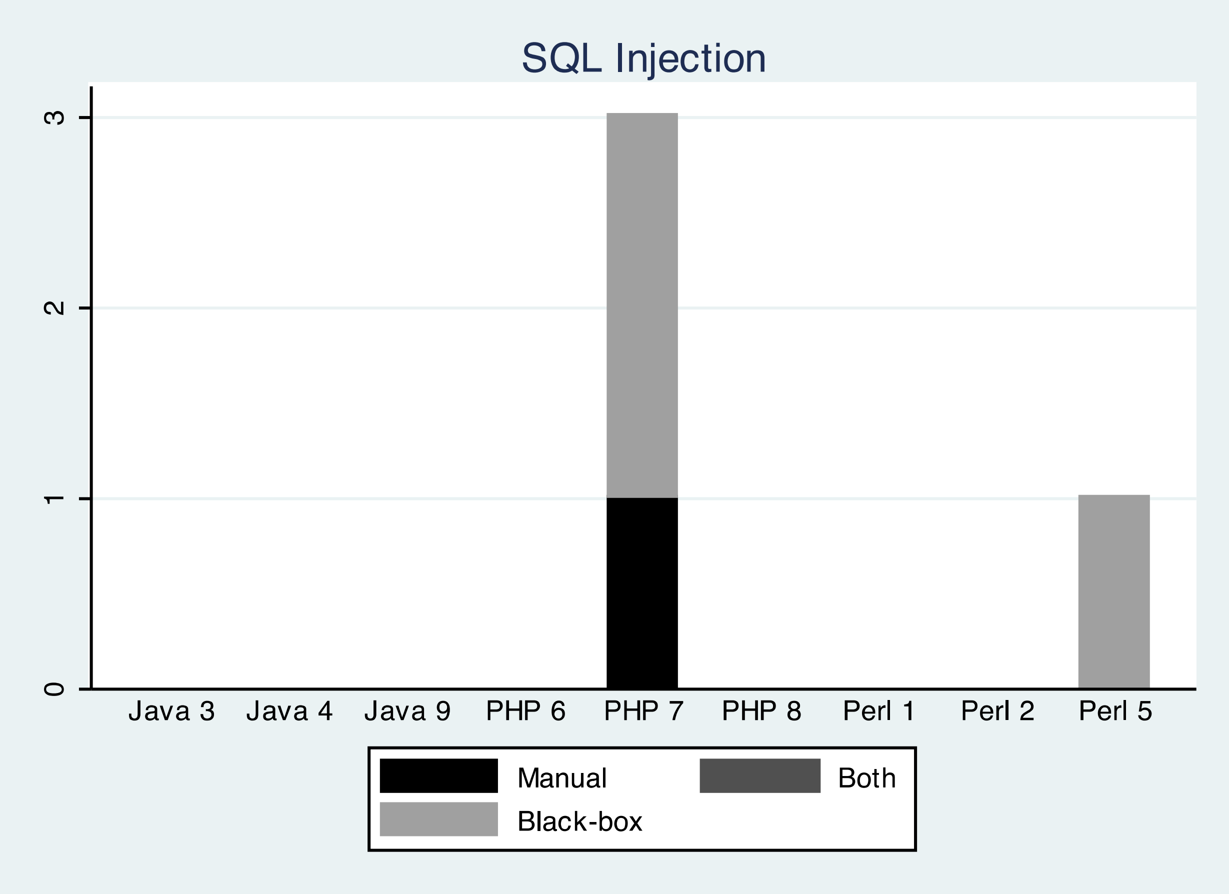
Results: Stored XSS



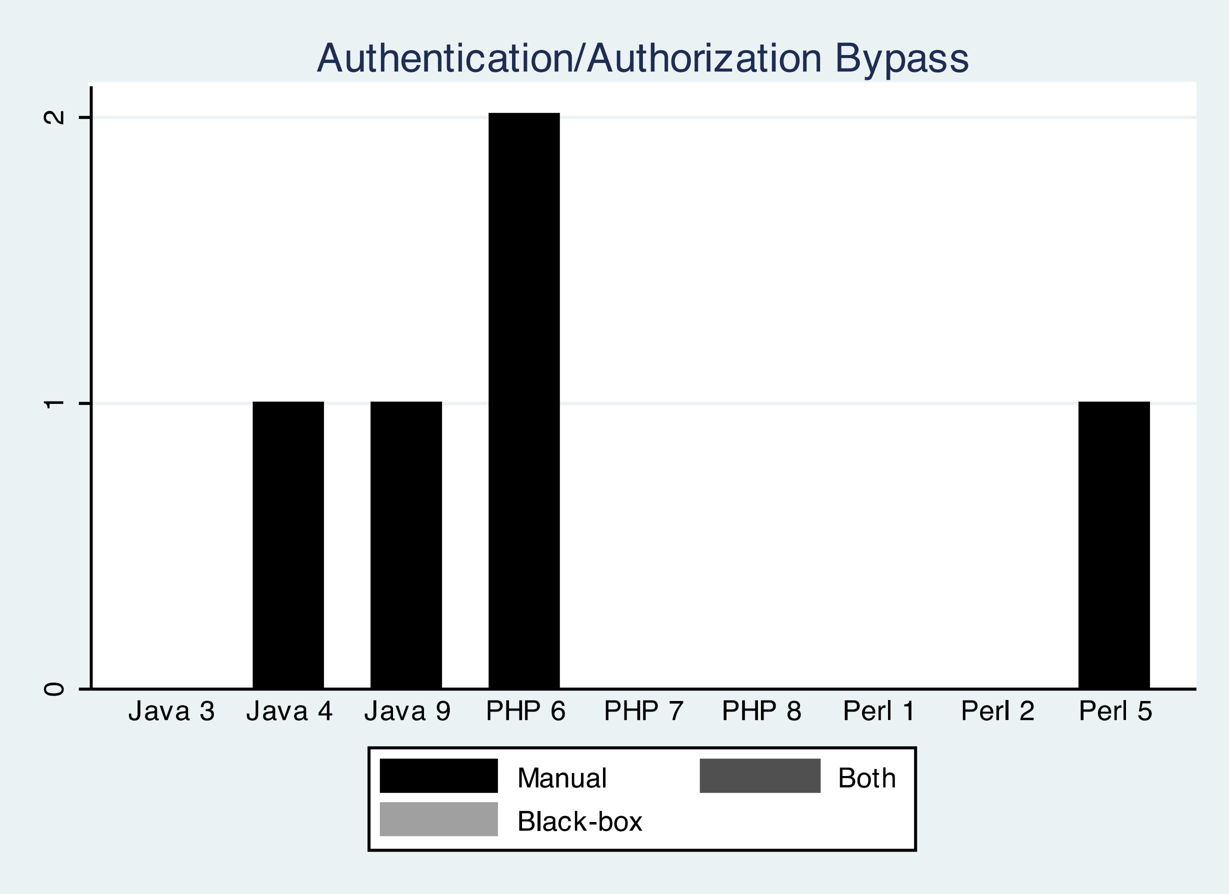
Results: Reflected XSS



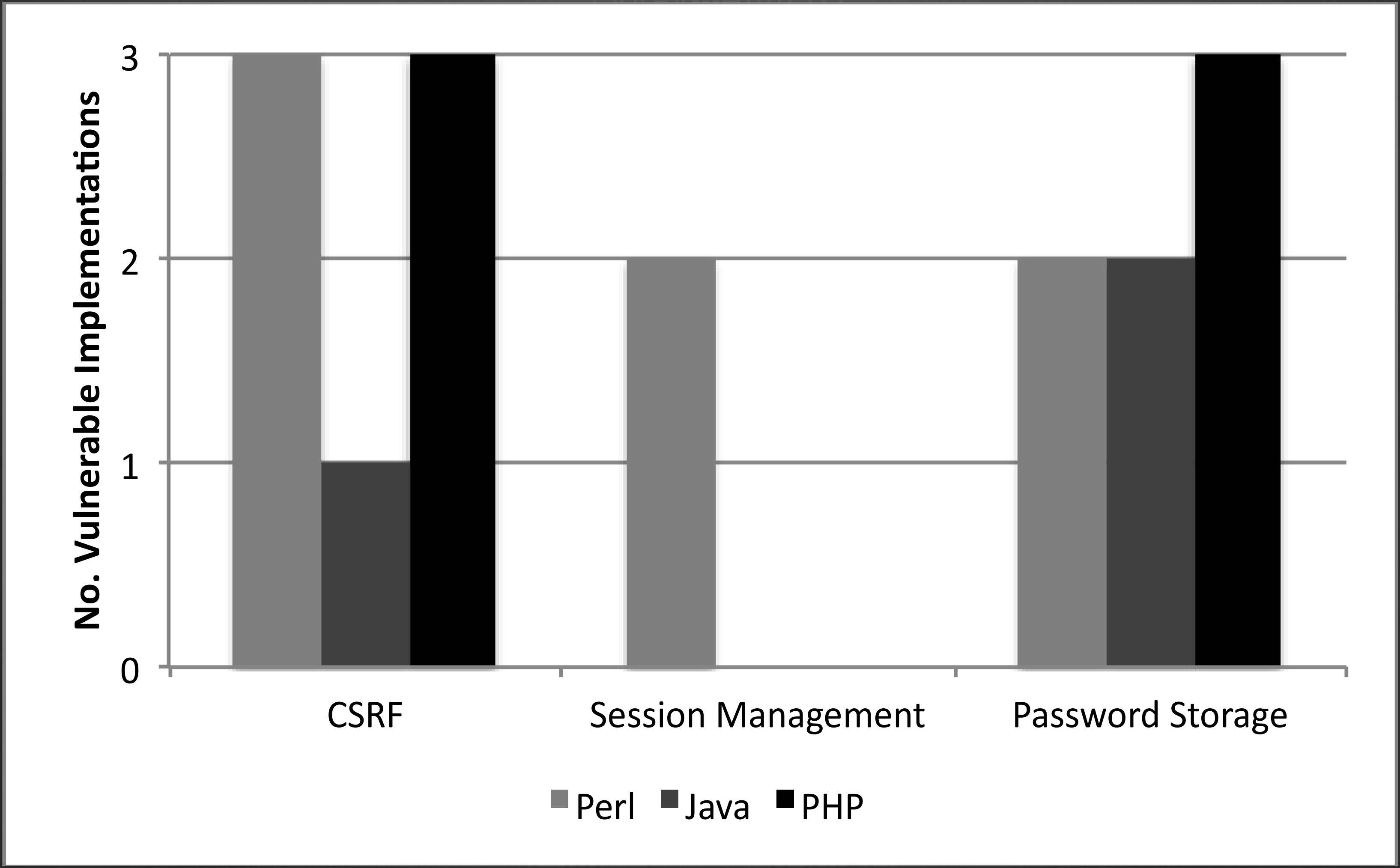
Results: SQL Injection



Results: Auth. Bypass



Results: "Binary" Vulnerabilities



Framework Support

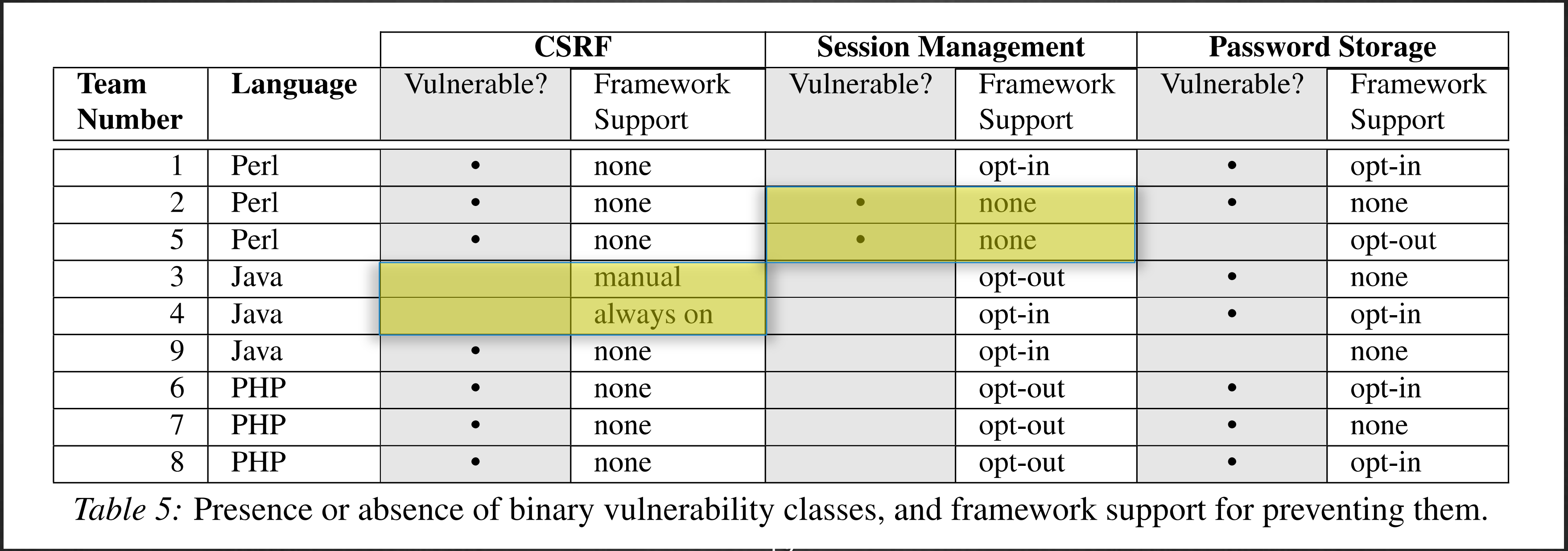
- Different frameworks offer different features
- Taxonomy of framework support
  - None
  - Manual
  - Opt-in
  - Opt-out
  - Always on

Framework Support

- Labeled each (team number, vulnerability class) with a framework support level
- E.g., “team 4 had always-on CSRF protection”
- This data set allows us to consider association between level of framework support and vulnerabilities.
- In other words, does a higher level of framework support help?

Framework Support

- No associations found for XSS, SQL injection, auth. bypass, or secure password storage.
- Statistically significant associations found for CSRF and session management.



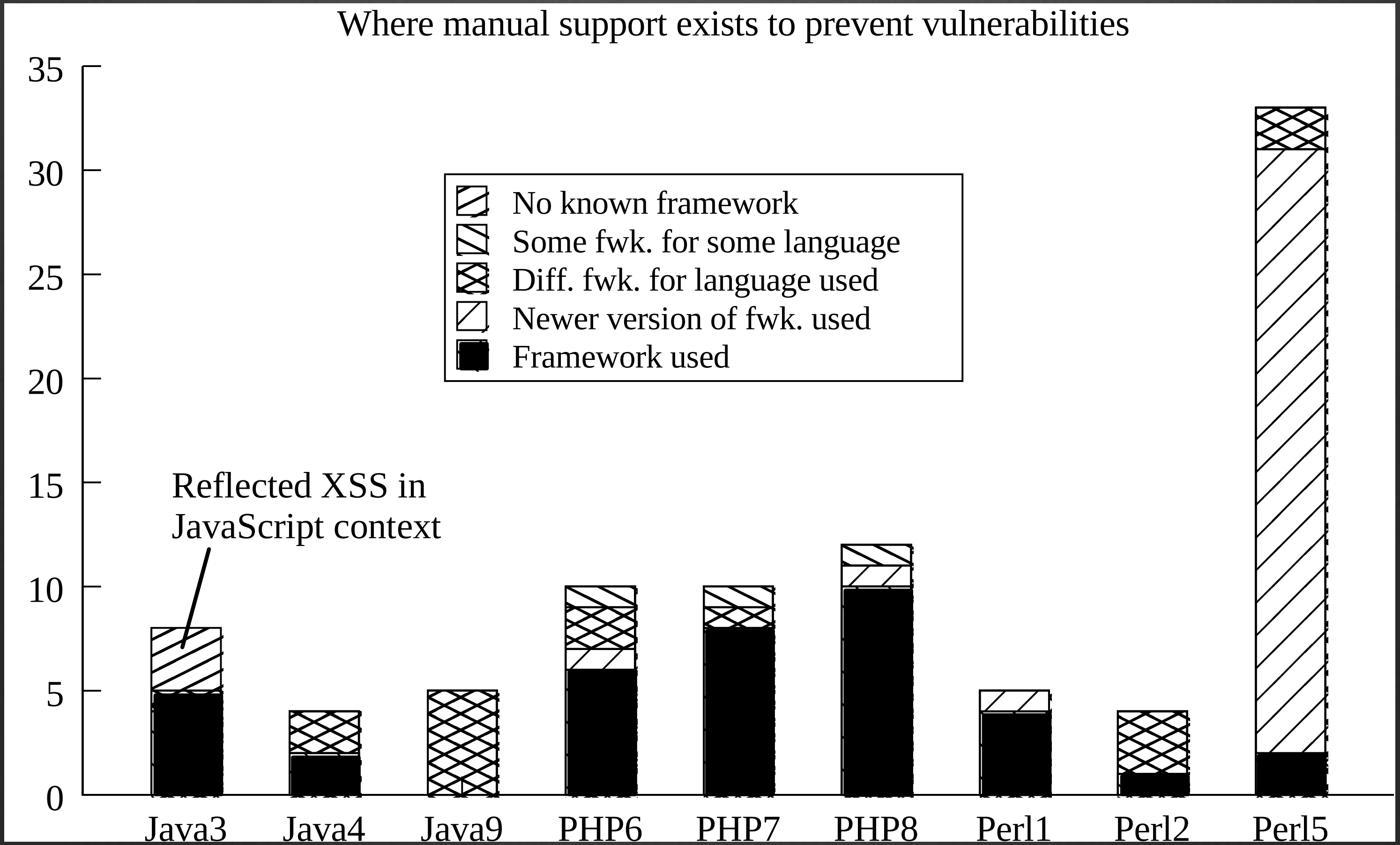
		CSRF		Session Management		Password Storage	
Team Number	**Language**	Vulnerable?	Framework Support	Vulnerable?	Framework Support	Vulnerable?	Framework Support
1	Perl	•	none		opt-in	•	opt-in
2	Perl	•	none	•	none	•	none
5	Perl	•	none	•	none		opt-out
3	Java		manual		opt-out	•	none
4	Java		always on		opt-in	•	opt-in
9	Java	•	none		opt-in		none
6	PHP	•	none		opt-out	•	opt-in
7	PHP	•	none		opt-out	•	none
8	PHP	•	none		opt-out	•	opt-in

Table 5: Presence or absence of binary vulnerability classes, and framework support for preventing them.

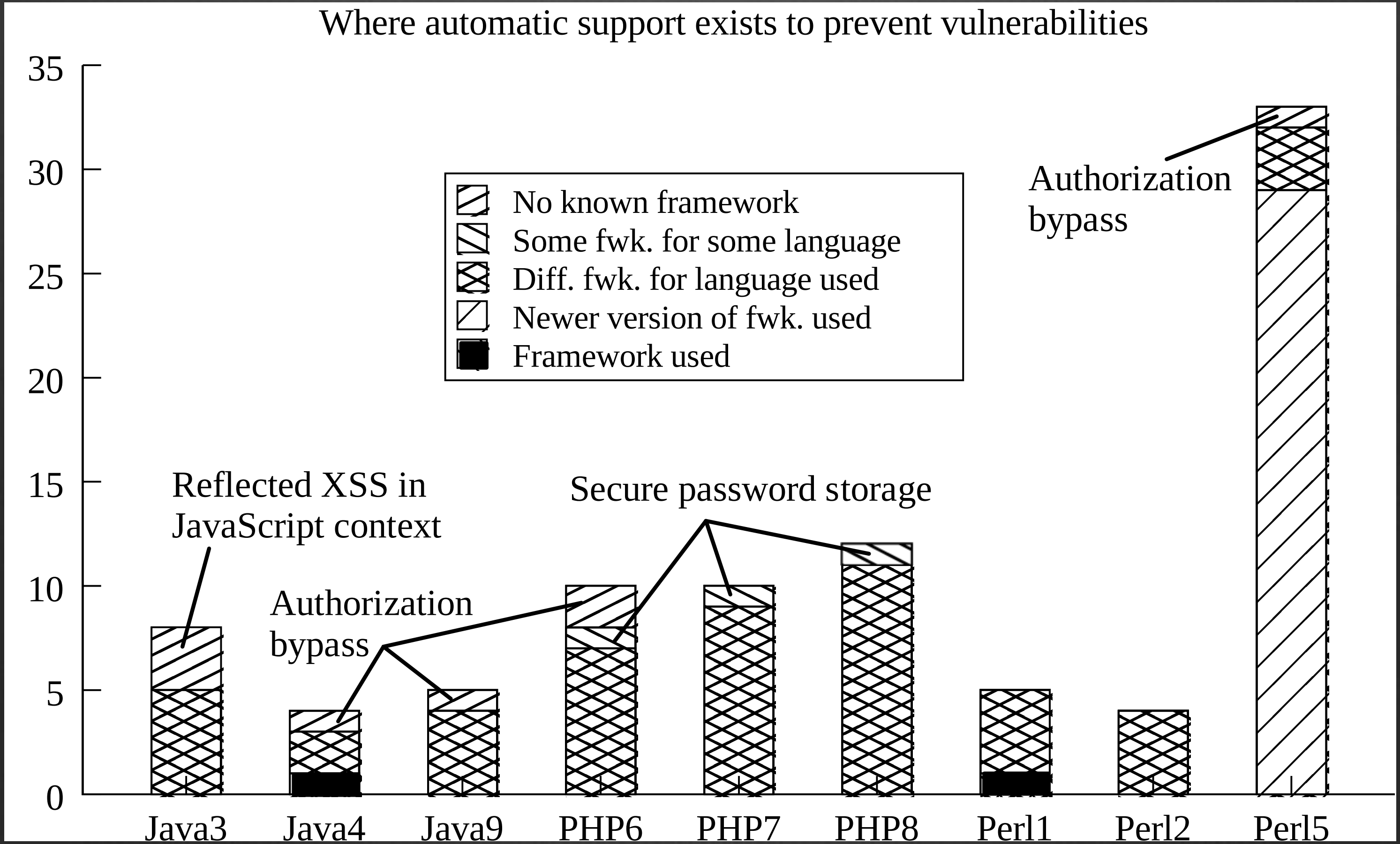
Individual Vulnerability Data

- More data to shed light on frameworks
- *How far away* from chosen tools to find framework support?
    - Framework used
    - Newer version of framework used
    - Another framework for language used
    - Some framework for some language
    - No known support
- For both automatic and manual framework support

Individual Vulnerability Data (Manual Support)



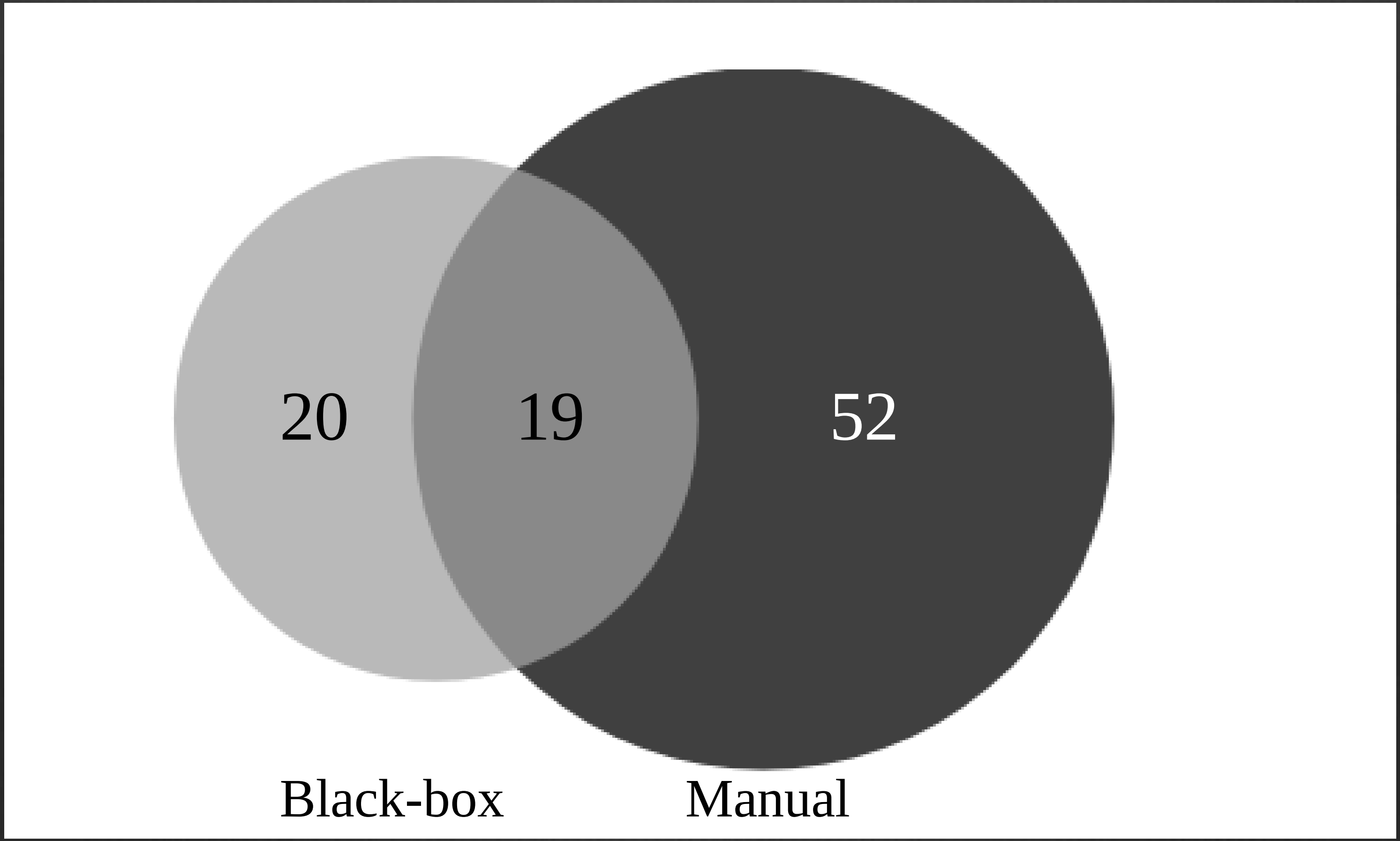
Individual Vulnerability Data (Automatic Support)



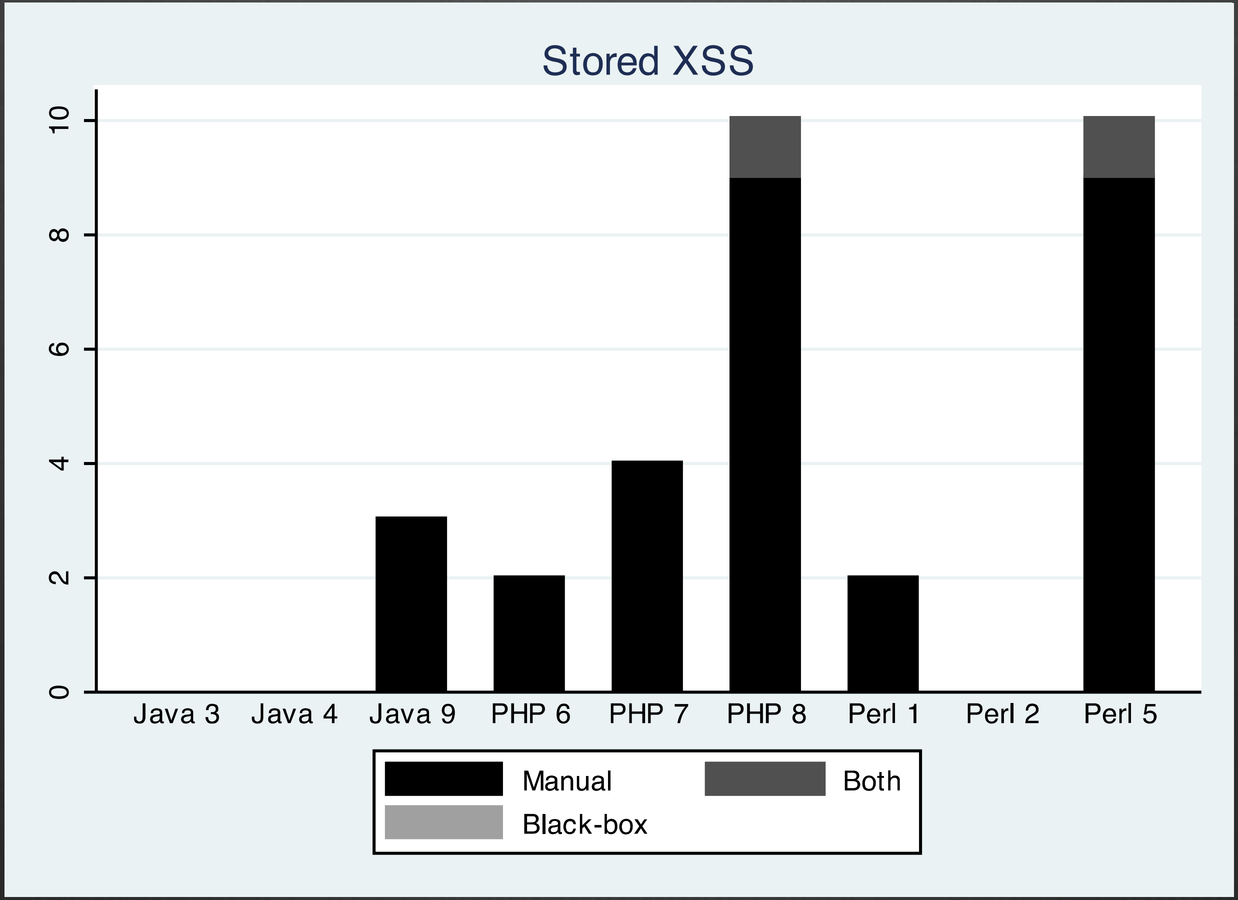
Method of Finding Vulnerabilities

- Automated black-box penetration testing
- Manual source code review

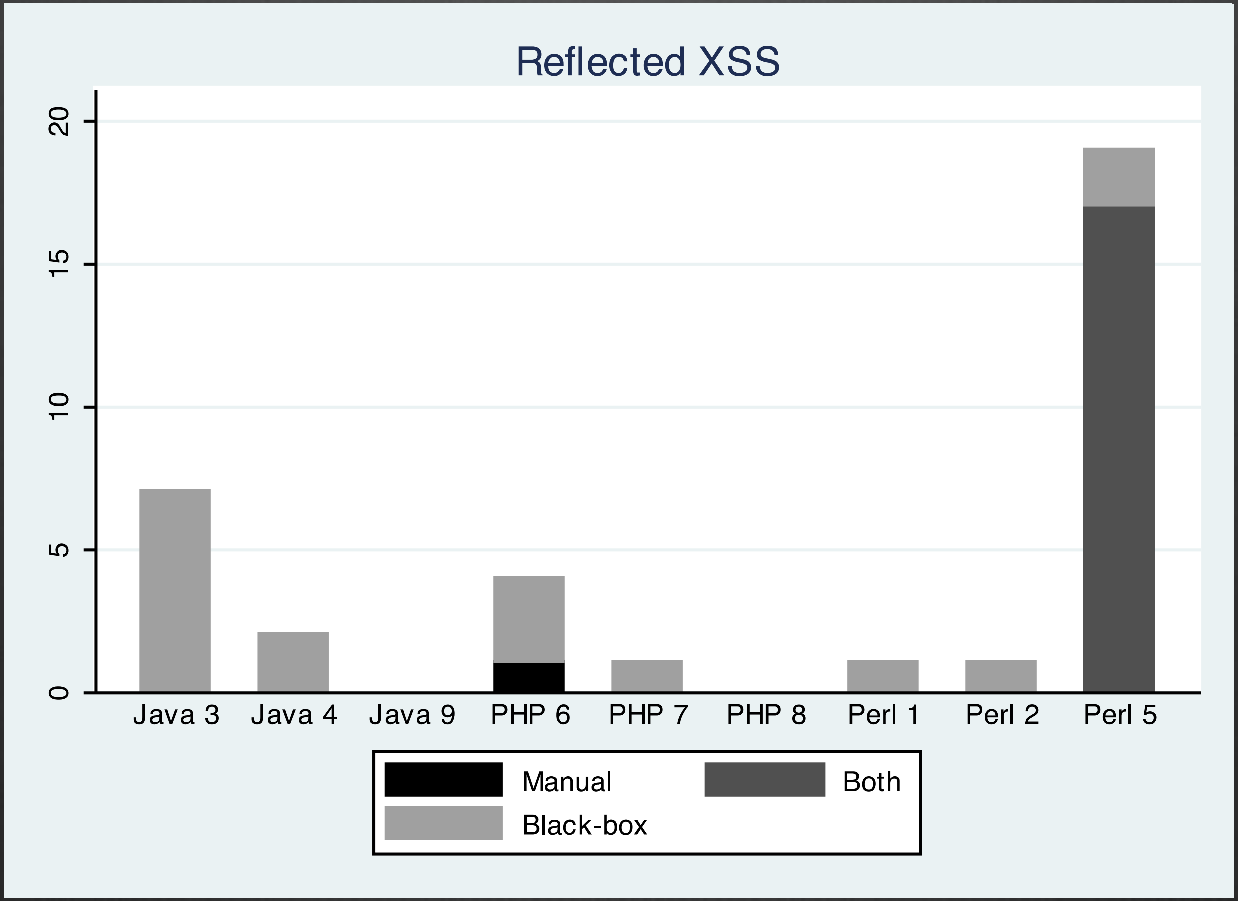
Method of Finding Vulnerabilities



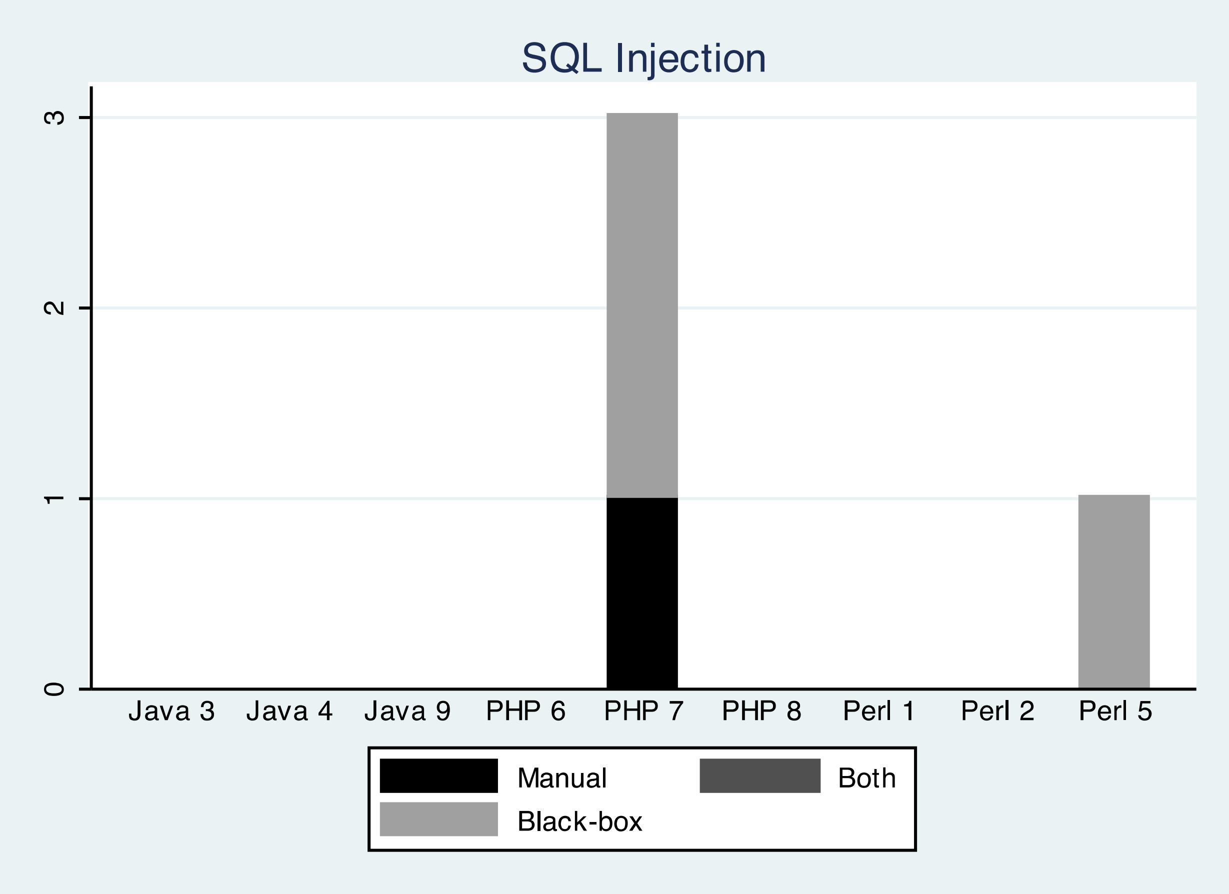
Results: Stored XSS



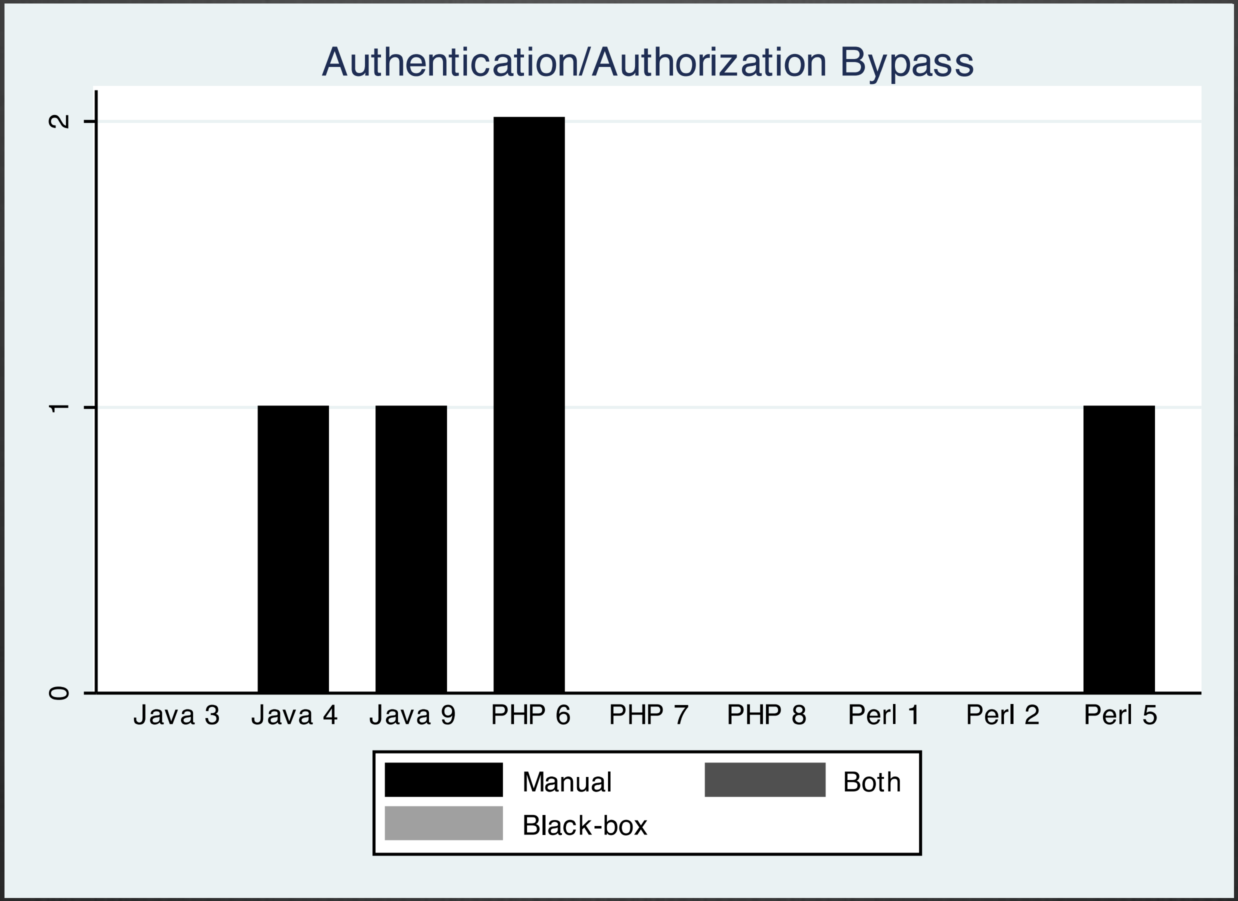
Results: Reflected XSS



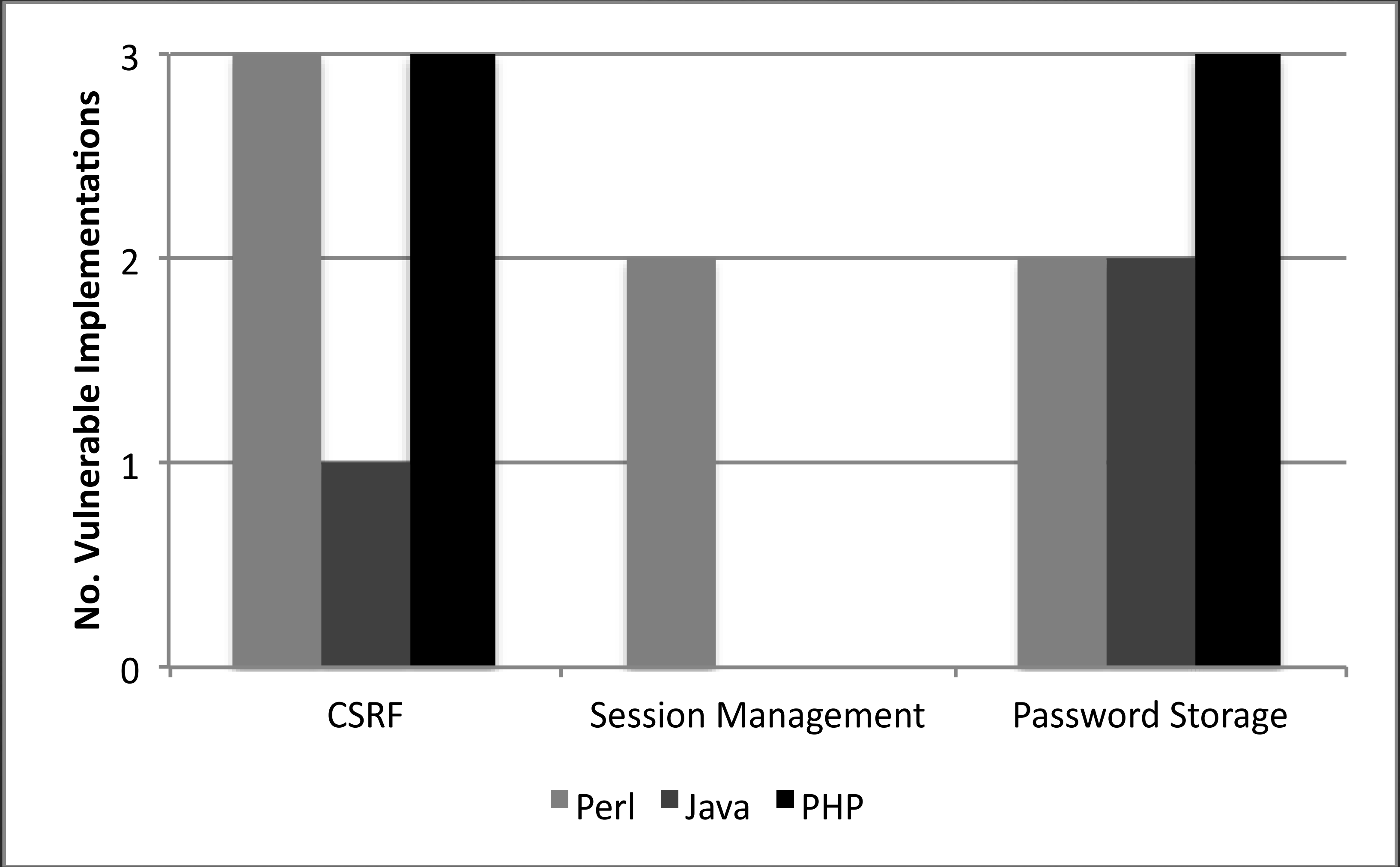
Results: SQL Injection



Results: Auth. Bypass



Results: “Binary” Vulnerabilities



Related Work

- Bau et al. *State of the Art: Automated Black-box Web Application Vulnerability Testing.*

- Doupé et al. *Why Johnny Can't Pentest: An Analysis of Black-Box Web Vulnerability Scanners.*

- Prechelt et al. *Plat_Forms: A Web Development Platform Comparison by an Exploratory Experiment Searching for Emergent Platform Properties.*

- Wagner et al. *Comparing Bug Finding Tools with Reviews and Tests.*

- Walden et al. *Java vs. PHP: Security Implications of Language Choice for Web Applications.*

- *WhiteHat Website Security Statistic Report, 9th Edition.*

Conclusion

- We should quantify our tools along various dimensions
- This study started (but did not finish!) that task for *security*
- Language, framework, vulnerability-finding method

Conclusion

- Web security is still hard; each implementation had at least one vulnerability.
- Level of framework support appears to influence security
- Manual framework support is ineffective
- Manual code review more effective than black-box testing
  - But they are complementary.
  - And they perform differently for different vulnerability classes

Future Work

- Gathering and analyzing larger data sets
- Other dimensions: reliability, performance, maintainability, etc.
- Deeper understanding of *why* some tools fare better than others
- Not just web applications!

Thank you!

Matthew Finifter

finifter@cs.berkeley.edu