INVITED CONTENT

Editor: **Giuliano Antoniol**
Polytechnique Montréal

Editor: **Steve Counsell**
Brunel University

Editor: **Phillip Laplante**
Pennsylvania State University

Web App Security

A Comparison and Categorization of Testing Frameworks

Satish M. Srinivasan and Raghvinder S. Sangwan

CONCOMITANT WITH the demand for web apps that provide convenient access to information and services, a dramatic rise has occurred in vulnerabilities that inevitably put web app users at risk.[1] To mitigate these vulnerabilities, software security practices recommend testing. However, web app developers often face challenges in using the many available security-testing frameworks, owing to those frameworks' inherent complexity and the lack of proper documentation. No up-to-date criteria exist that can help practitioners and organizations select an appropriate framework. Consequently, numerous vulnerabilities go undetected in the final product, creating a potential for major attacks.

To help practitioners select the right framework, we classified 26 frameworks, using 27 criteria.

Classification Criteria

We systematically assembled the criteria from sources including "Web Security Testing Approaches: Comparison Framework,"[2] "Automatic Testing of Program Security Vulnerabilities,"[3] and the available documentation for each framework. Five criteria were particularly significant:

- the types of attacks a framework covers;
- its testing approach (black box, white box, or both);
- whether it's fully automated, semiautomated, or manual;
- whether it supports penetration testing for PCI (Payment Card Industry) compliance; and
- whether it mitigates false positives.

We also wanted to know

- the language in which a framework was developed;
- whether a framework was open source;
- whether it was platform independent; and
- whether it supported testing of web apps, mobile apps, and services.

Once we established our criteria, we assembled a tidy dataset containing the 26 security frameworks categorized along these dimensions for further analysis. All the frameworks we analyzed had data for at least 19 of the criteria. Owing to space limitations, we discuss the five most significant criteria next.

TABLE 1

The top 10 web app security-testing frameworks.*

Framework	Attacks covered	Testing approach	Functional nature	Penetration testing for Payment Card Industry compliance	Mitigates false positives
OWASP Enterprise Security API	• Injection • Broken authentication and session management • Cross-site scripting (XSS) • Insecure direct object references • Cross-site request forgery (CSRF)	N/A	Automated	Yes	Yes
Python–Django-based framework	• Injection • XSS • CSRF • Clickjacking	Black box and white box	Automated	Yes	Yes
w3af	• Injection • Broken authentication and session management • XSS • Security misconfiguration	Black box	Automated and manual	Yes	Yes
Hdiv	• Injection • Broken authentication and session management • XSS • Insecure direct object references • Security misconfiguration • Sensitive data exposure • Missing function level access control • CSRF • Using components with known vulnerabilities • Unvalidated redirects and forwards	Black box and white box	Automated	No	No
Spring Security	• Broken authentication and session management • Missing function level access control • CSRF • Clickjacking • Authorization • Session fixation	Black box and white box	Semiautomated	Yes	Yes
Samurai Web Testing Framework	• Injection • XSS • Security misconfiguration • Remote file inclusion • Other common vulnerabilities	Black box	Automated	Yes	No
SPIKE Proxy	• Injection • XSS • Directory traversal	N/A	Automated	Yes	No
Yii	• Injection • Broken authentication and session management • XSS • CSRF • Authorization	Black box and white box	Semiautomated with the help of Gii	No	N/A
JBoss or JBossSX	• Broken authentication and session management • Authorization	Black box and white box	Automated	Yes	N/A
YSO Mobile Security Framework	• Detect insecure permissions and configurations • Detect insecure code	Black box and white box	Automated	Yes	N/A

** N/A indicates that no data was available.*

The Framework Landscape

Using the Apriori algorithm, we performed association-rule mining on the 14 frameworks that covered at least three of the top 10 attacks identified by OWASP (Open Web Application Security Project).[4] Using a minimum support of 0.65 and confidence of 0.5, we inferred rules that led to three observations. First, when a framework addresses authentication, it also addresses authorization. Second, when a framework addresses insecure direct object references, it also addresses authorization. Finally, frameworks that specifically address authorization are less likely to address cross-site request forgery (CSRF).

The first observation is straightforward because authentication and authorization go hand in hand. The second observation makes sense because attackers can exploit insecure direct object references to bypass the authorization mechanism to access resources in a system that otherwise wouldn't be accessible. The third observation was surprising because CSRF occurs when no proper authorization checks are in place. This observation might suggest that authorization mechanisms still aren't mature enough to handle CSRF, which is a new breed of attack and comes in many flavors.

To uncover these attacks, seven frameworks use black-box testing, three use white-box testing, and 10 use both. Seventeen frameworks are fully automated, six are semiautomatic, and three are manual. Only 12 frameworks support penetration testing. Only four could mitigate false positives: OWASP Enterprise Security API, the Python–Django-based framework, Spring Security, and w3af.

On the basis of the five criteria, we determined the top 10 frameworks. Table 1 shows the results; it lists both the attacks in the OWASP top-10 list and other types of attacks. The more types of attacks a framework covers, the better it is. If two frameworks cover the same number of attack types, the framework that covers attacks higher on the OWASP top-10 list is better. Frameworks that provide both black-box and white-box testing received more weight in our ranking. Similarly, frameworks that uncover attacks automatically, support penetration testing, and mitigate false positives are better than those that lack any of those features. A framework's lack of sufficient documentation or guidance regarding a criterion counted against that framework. All the frameworks except one (the Python–Django-based framework) are open source.

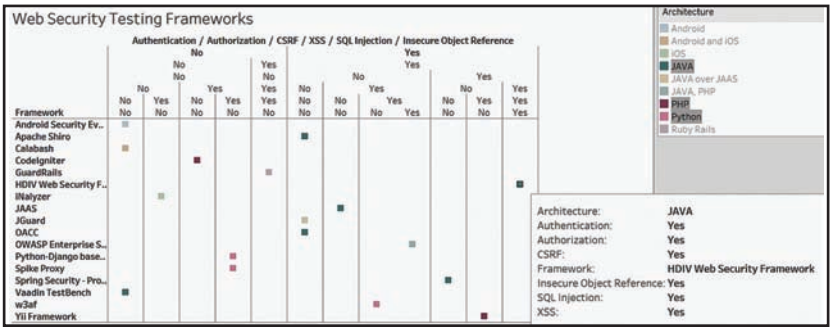

FIGURE 1. The Tableau output for a user query for a list of suitable Java, PHP, or Python-based testing frameworks that cover attacks including injection, cross-site scripting, insecure direct object references, and cross-site request forgery. The results indicate that the best framework would be Hdiv.

Using the Dataset

Our dataset is publically available at https://sites.google.com/site/psuwebframeworksteam/to-dos/websecurityappframework. We're working on two interfaces for it. One of them, which is already available, lets expert users add information on existing or new frameworks. Before we add this information to the dataset, it goes through a formal review to ensure its integrity.

The other interface, which is under development, will let users query our dataset to retrieve information about frameworks to aid their decision making. Users will also be able to visualize the results. For instance, Figure 1 shows the output for a query to obtain a list of suitable Java, PHP, or Python-based frameworks that cover attacks including injection, cross-site scripting, insecure direct object references, and CSRF. The results indicate that the best framework would be Hdiv.

Recent breaches have compromised personal medical and financial data and sensitive government and political information, making cybersecurity threats a serious issue. Security can no longer be an afterthought but a quality that must be proactively designed into systems. Our dataset is a step in this direction, helping to aggregate and maintain current information on web security-testing frameworks and providing a valuable service to those interested in using these frameworks

to design and develop secure web apps and services.

References

1. G. Erdogan, "Security Testing of Web Based Applications," master's thesis, Dept. Computer and Information Science, Norwegian Univ. of Science and Technology, 2009; www.diva-portal.org/smash/get/diva2:348920/FULLTEXT01.pdf.
2. F.T. Alssir and M. Ahmed, "Web Security Testing Approaches: Comparison Framework," *Proc. 2nd Int'l Congress Computer Applications and Computational Science*, 2011, pp. 163–169.
3. H. Shahriar and M. Zulkernine, "Automatic Testing of Program Security Vulnerabilities," *Proc. 33rd Ann. IEEE Int'l Computer Software and Applications Conf.* (COMPSAC 09), vol. 2, 2009, pp. 550–555.
4. "Top 10 2013-Top 10," OWASP Foundation, 2013; www.owasp.org/index.php/Top_10_2013-Top_10.

SATISH M. SRINIVASAN is an assistant professor of information science at Penn State University. Contact him at sus64@psu.edu.

RAGHVINDER S. SANGWAN is an associate professor of software engineering at Penn State University. Contact him at rsangwan@psu.edu.