

Jing Song U65712216

Professor Osama Alshaykh

EC601 Product Design ECE

Mini Project 2 Cyber Security Report

October 20, 2019

Pros and Cons of Open-Source Security Framework and Its Applicable Usage

In 2008, Apache Shiro, a Java open-source security framework was born. “Shiro” means “castle” in Japanese, and the framework was built to provide a “simple but powerful” way to secure the web applications. We could now use Apache Shiro API to realize functions like authentication, authorization, cryptography, session management, etc. for our own applications. These four features are the main usages that developers demand from such a security framework. Authentication makes sure that certain functions are only accessible for login users. Authorization allows developers to control user access. Cryptography could hide personal and sensitive data from the public, and session management allows each user to have his or her own “time sensitive state” (Hazlewood). Apache Shiro is just one of the many open-source security frameworks available on the Internet that allow developers to secure their applications. These security tools do make life much easier for developers, but we should still be aware of some drawbacks they will possibly bring us.

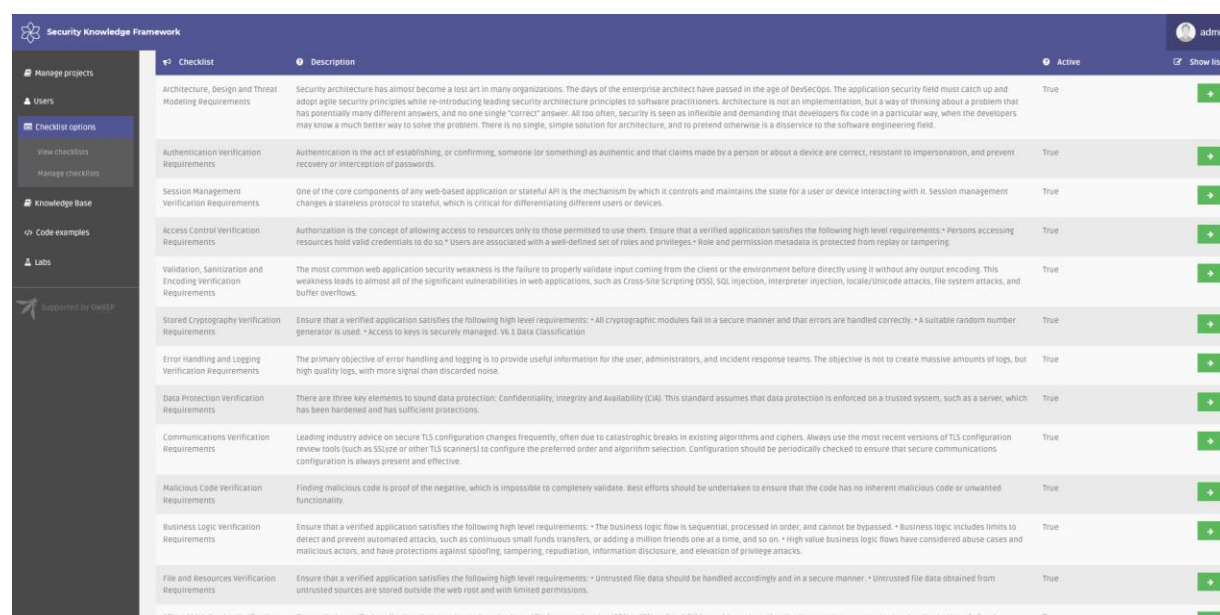
OWASP (the Open Web Application Security Project) is an organization and a community

where lots of security developers gather to work on projects that improve application security. The organization is non-profit and it allow any individual or organization to access it (“Main Page”). The people in this community has already developed a quite mature security framework and it is licensed under the “Creative Commons Attribution-ShareAlike 3.0 license” (“OWASP Open Cyber Security Framework Project”). Open-source security frameworks like the one provided by OWASP saves tons of time for developers. Instead of figuring out a complicated security implementation on their own, developers now could adopt these frameworks into their own softwares, sometimes even without any cost. Developing such framework in an open community also makes it easier to find out bugs and fix them. Since “developers care about their reputations”, they would work very diligently to find potential problems in their code and show off the perfect code (“Three Myths Debunked About Open Source Software Security.”). People who are paid to develop such a security framework has an incentive to find all flaws in the code, while those who work on non-profit projects may still be self-motivated to perfect their code. Either way makes open-source security framework better to adapt into softwares.

Even though many developers are working on fixing bugs in these open-source security frameworks, problems may still arise from all different aspects. For instance, fixing security problems require expertise in this area, and not all developers master such expertise. Those who do not may even give misleading ideas, so that the bugs would be not fixed or could be even worse. Besides, open-source security framework is open to the public and many companies or organizations may have already been using it. It is possible that people with bad intentions find a bug in the open-source code and use it to hack the existing security system. A

short time frame between one finding a bug and another person finding a bug could lead to unexpected results. An open-source security framework has to balance the publicity of code and being really secure.

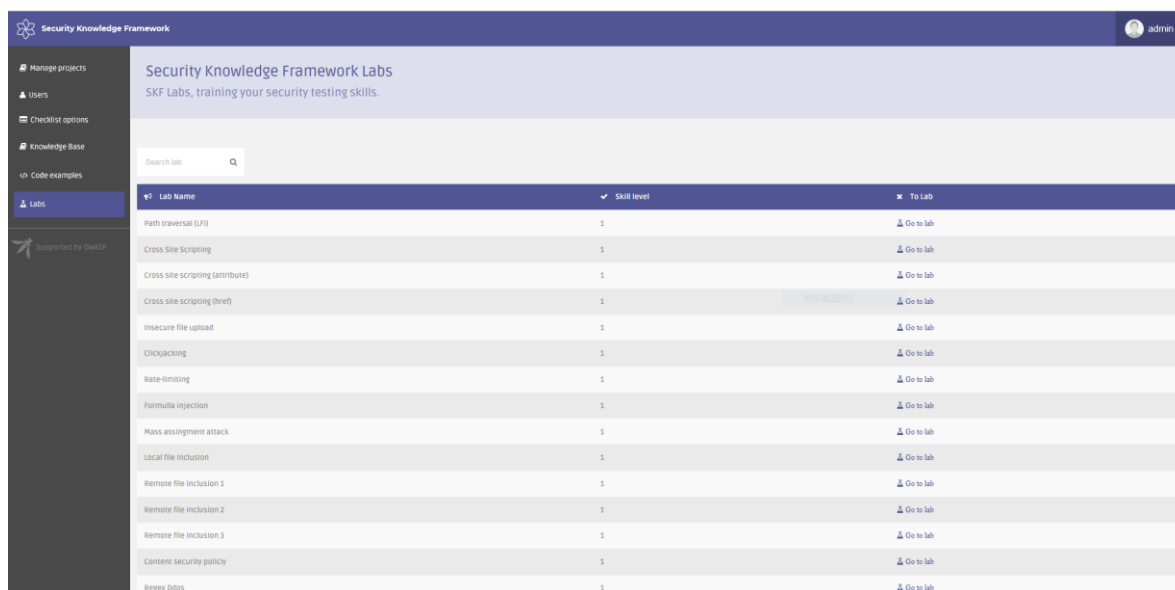
OWASP also provides a security knowledge framework as a guideline for developers to implement the security requirements for their softwares. The “checklist” option of the framework looks like this on the website:



Checklist	Description	Active	Show list
Architecture, Design and Threat Modeling Requirements	Security architecture has almost become a lost art in many organizations. The days of the enterprise architect have passed in the age of DevSecOps. The application security field must catch up and adopt agile security principles while re-introducing leading security architecture principles to software practitioners. Architecture is not an implementation, but a way of thinking about a problem that has potentially many different answers, and no one single "correct" answer. All too often, security is seen as inflexible and demanding that developers fix code in a particular way, when the developers may know a much better way to solve the problem. There is no single, simple solution for architecture, and to pretend otherwise is a disservice to the software engineering field.	True	+
Authentication Verification Requirements	Authentication is the act of establishing, or confirming, someone (or something) as authentic and that claims made by a person or about a device are correct, resistant to impersonation, and prevent recovery or interception of passwords.	True	+
Session Management Verification Requirements	One of the core components of any web-based application or stateful API is the mechanism by which it controls and maintains the state for a user or device interacting with it. Session management changes a stateless protocol to stateful, which is critical for differentiating different users or devices.	True	+
Access Control Verification Requirements	Authorization is the concept of allowing access to resources only to those permitted to use them. Ensure that a verified application satisfies the following high level requirements: * Persons accessing resources hold valid credentials to do so. * Users are associated with a well-defined set of roles and privileges. * Rule and permission metadata is protected from replay or tampering.	True	+
Validation, Sanitization and Encoding Verification Requirements	The most common web application security weakness is the failure to properly validate input coming from the client or the environment before directly using it without any output encoding. This weakness leads to almost all of the significant vulnerabilities in web applications, such as Cross-Site Scripting (XSS), SQL injection, interpreter injection, locale/unicode attacks, file system attacks, and buffer overflows.	True	+
Stored Cryptography Verification Requirements	Ensure that a verified application satisfies the following high level requirements: * All cryptographic modules fail in a secure manner and that errors are handled correctly. * A suitable random number generator is used. * Access to keys is securely managed. * V6.1 Data Classification	True	+
Error Handling and Logging Verification Requirements	The primary objective of error handling and logging is to provide useful information for the user, administrators, and incident response teams. The objective is not to create massive amounts of logs, but high quality logs, with more signal than discarded noise.	True	+
Data Protection Verification Requirements	There are three key elements to sound data protection: Confidentiality, Integrity and Availability (CIA). This standard assumes that data protection is enforced on a trusted system, such as a server, which has been hardened and has sufficient protections.	True	+
Communications Verification Requirements	Leading industry advice on secure TLS configuration changes frequently, often due to catastrophic breaks in existing algorithms and ciphers. Always use the most recent versions of TLS configuration review tools (such as SSLuze or other TLS scanners) to configure the preferred order and algorithm selection. Configuration should be periodically checked to ensure that secure communications configuration is always present and effective.	True	+
Malicious Code Verification Requirements	Finding malicious code is proof of the negative, which is impossible to completely validate. Best efforts should be undertaken to ensure that the code has no inherent malicious code or unwanted functionality.	True	+
Business Logic Verification Requirements	Ensure that a verified application satisfies the following high level requirements: * The business logic flow is sequential, processed in order, and cannot be bypassed. * Business logic includes limits to detect and prevent automated attacks, such as continuous small funds transfers, or adding a million friends one at a time, and so on. * High value business logic flows have considered abuse cases and malicious actors, and have protections against spoofing, tampering, repudiation, information disclosure, and elevation of privilege attacks.	True	+
File and Resources Verification Requirements	Ensure that a verified application satisfies the following high level requirements: * Untrusted file data should be handled accordingly and in a secure manner. * Untrusted file data obtained from untrusted sources are stored outside the web root and with limited permissions.	True	+
API and Web Service Verification	Ensure that a verified application that uses trusted service layer APIs (commonly using JSON or XML or GraphQL) has: * Adequate authentication, session management and authorization of all web	True	+

Screenshot from <https://demo.securityknowledgeframework.org/checklist>

The knowledge framework has a “lab” option for developers to try out several security testings on their own. Such an interactive approach would be very useful for beginners in cybersecurity and I would recommend anyone who is interested to try out this page.



Lab Name	Skill level	To Lab
Path traversal (LFI)	1	Go to lab
Cross Site Scripting	1	Go to lab
Cross site scripting (attribute)	1	Go to lab
Cross site scripting (href)	1	Go to lab
Insecure file upload	1	Go to lab
Clickjacking	1	Go to lab
Rate-limiting	1	Go to lab
Formula injection	1	Go to lab
Mass assignment attack	1	Go to lab
Local file inclusion	1	Go to lab
Remote file inclusion 1	1	Go to lab
Remote file inclusion 2	1	Go to lab
Remote file inclusion 3	1	Go to lab
Content security policy	1	Go to lab
Regex DDoS	1	Go to lab

Screenshot from <https://demo.securityknowledgeframework.org/labs>

Overall, open-source security framework is a good resource for developers if they need some guidelines on the security implementation or would like to save time and cost. However, these open-source security framework does not guarantee 100% security. Cyber security is such a big topic and we could not expect to solve all security problems with simply adopting the open-source security frameworks.

Work Cited

Hazlewood, Les. "Application Security With Apache Shiro." *InfoQ*, InfoQ, 14 Mar. 2011,

<https://www.infoq.com/articles/apache-shiro/>.

"Main Page." *OWASP*, https://www.owasp.org/index.php/Main_Page.

"OWASP Open Cyber Security Framework Project." *OWASP*,

https://www.owasp.org/index.php/OWASP_Open_Cyber_Security_Framework_Project.

"Three Myths Debunked About Open Source Software Security." *RubyGarage*,

<https://rubygarage.org/blog/open-source-software-security>.

Summary of Yi-Wei Chen's Report(<https://github.com/ThomasChen1997/EC601-Mini-Project-2/blob/master/Mini-project%202.pdf>):

Yi-Wei Chen describes in details the purpose of cyberattacks and how different groups of people could defend themselves from cyberattacks.

Purposes of Cyberattacks:

- (1) Damage sensitive info
- (2) Extorting money
- (3) Interrupt business activities

How to protect oneself from cyberattacks:

- (1) Users: strong password, phishing emails, back-up data
- (2) Organizations/ Companies: reliable security framework
- (3) Technology: protect endpoint devices, networks, cloud

He also talks about the four most common types of attacks:

Four types of attacks:

- (1) Ransomware: malicious software asking for ransom payment
- (2) Malware: unauthorized access to damage the computer
- (3) Social Engineering: tricks users to reveal sensitive information
- (4) (most common) Phishing: fraud emails

Then he introduces the open-source security framework: Clusterfuzz, which is very helpful for software companies to locate security issues in their software products. Yi-Wei gives a list of some nice features of Clusterfuzz:

1. Highly scalable. Google's internal instance runs on over 25,000 machines.
2. Accurate deduplication of crashes.
3. Fully automatic bug filing and closing for issue trackers (Monorail only for now).
4. Testcase minimization.
5. Regression finding through bisection.
6. Statistics for analyzing fuzzer performance, and crash rates.
7. Easy-to-use web interface for management and viewing crashes.
8. Firebase authentication.
9. Support for coverage guided fuzzing (e.g. libFuzzer and AFL) and blackbox fuzzing.
10. Learners can get the new updates, versions or suggestions from Github.
11. It is operated in Python which contains lots of library to use.

Clusterfuzz has two main components, one is called App Engine, which provides a web user interface for users to access the stats and info on crashes. The other part is a pool of bots, these bots run preemptible (shuts down the machine) and non-preemptible tasks (does not shut down the machine). Users could choose between the different types of tasks, depending on their needs.

What I learned from Yi-Wei's paper is that our reports could be read by anyone from all different backgrounds, so it is always necessary to introduce the topic and include some basic concepts of cybersecurity in the report, so that our reader could understand the concepts better and follow closely the further discussion of cybersecurity in the report. Besides, I did not list enough detailed features of the open-source cybersecurity frameworks in my report and I should not have missed writing about them.