ESAPI Cheat Sheet

From OWASP

<UNDER CONSTRUCTION>

Introduction

This article provides some simple rules of thumb to keep in mind when implementing an Enterprise Security API (ESAPI) for your application(s) and/or your organization. Although ESAPI from a certain perspective is "just a set of interfaces", things aren't actually quite so simple as that.

1) For an existing web application currently using a MVC framework (like Spring or Struts) are we today (9th Jan 2009) officially recommending that this web application development team adds OWASP's ESAPI.jar to the list of 'external' APIs (i.e. libs) they use, support and maintain?

- Yes. ESAPI is not a framework – it’s a set of foundational security controls so that developers don’t have to keep remaking the same mistakes over and over. There are several security controls in ESAPI which are not provided by those frameworks. Of course, it’s always best if developers can work within their framework. If your framework already provides a control already, great. If your framework wants to use ESAPI and hide it from the developer, great. In some cases, the framework might want to expose ESAPI, and that’s fine too. Sometimes, you may want to integrate ESAPI into your framework’s existing security control. Generally this isn’t too hard. If you want to use Struts Validators, for example, you can write a custom Struts validator that delegates to an ESAPI validator. This allows most developers to use the Struts pattern as normal, but you get the advantage of canonicalization and intrusion detection under the hood.

2) When adopting the OWASP ESAPI's J2EE implementation, is ESAPI.jar ALL they need to add? or are there other dependencies (i.e. jars) that also need to be added, supported and maintained? (for example on the 'Dependencies' section of the ESAPI Java EE page (i.e. Tab) it seems to imply that there are other *.jars needed)

- The ESAPI *interfaces* have no dependencies. However, the reference implementation, like most Java libraries these days, has several dependencies managed by Maven.

3) Where can I find detailed information about each of the 9 Security Controls that ESAPI.jar currently supports: 1) Authentication, 2) Access control, 3) Input validation, 4) Output encoding/escaping, 5) Cryptography, 6) Error handling and logging, 7) Communication security, 8) HTTP security and 9) Security configuration? (I took this list of controls from the Introduction to ESAPI pdf)

- There are a number of documents at various stages of development linked on the OWASP Wiki. The JavaDoc is extensive and pretty much up to date. You can also check out the draft “ESAPI Book” which needs to be updated but you may find helpful.

4) When adopting EASPI.jar, are we recommending that the developers should adopt or retrofit their existing code on the areas affected by those 9 Security Controls? (i.e. code related to: Authentication, Access control, Input validation, Output encoding/escaping, Cryptography, Error handling and logging, Communication security, HTTP security and Security configuration)

- One of the specific use cases for ESAPI is to enable remediation of known security problems. You can use ESAPI to patch a single vulnerability, as part of a standard coding pattern, or as part of your framework hidden from developers. You can use the ESAPI controls in many different ways. I’m hopeful that creating security building blocks that we might enable the creation of new and better security models.

5) Should we recommend the adoption of ALL 9 Security Controls? or are there some controls that are not ready today (9 Jan 2009) for production environments and should not be recommended? (for example is the 'Authentication' control as mature as the 'Error handling and logging' control?)

- In working with a number of companies on the implementation of *their* Enterprise Security API, it’s clear that the best approach is to phase in controls over time. Most organizations are starting with validation, canonicalization, and encoding. The error handling, logging, encryption, randomizer, intrusion detector, security configuration, and http utilities don’t require much customization and are easily integrated. The authenticator requires some customization to tie it into your identity store. The access controller requires even more customization, as it varies so widely from organization to organization.

6) Are there commercial (i.e. paid) support services available for the companies who want to add ESAPI.jar to they application?

- Aspect Security provides paid support for ESAPI users, including training, integration, and customization. All improvements are contributed back to

the ESAPI project as open source enhancements. Anyone interested can contact me directly.

7) What is the version of ESAPI.jar that we should recommend? the version 1.4 (which looks like a stable release) or the version 2.0 rc4 (which looks like it is a Release Candidate)

- I recommend moving to the ESAPI 2.0 branch. It's our fourth release candidate and contains significant improvements over the 1.4 branch. Note that if you are using Java 1.4, then you'll have to stay with the ESAPI 1.4 branch since ESAPI 2.0 uses newer language features. We've started the process of back-porting the 2.0 improvements to 1.4 since so many ESAPI users are still on Java 1.4.

8) Where can I find the documentation of where and how ESAPI should be used? More importantly, where can I find the information of how it CAN NOT or SHOULD NOT be used (i.e. the cases where even when the EASPI.jar are used, the application is still vulnerable)

- I think the decision about which security controls cannot or should not be used is requires quite a bit more context than is possible generically. Can you find documentation on the web about where deadbolt locks cannot or should not be used? We'll try to give people reasonable guidance about when and how to use the controls in ESAPI.

9) If there list of companies that have currently added ESAPI.jar to their applications and have deployed it? (i.e. real world usage of EASPI)

- http://www.owasp.org/index.php /Category:OWASP_Enterprise_Security_API#tab=Contributors.2FUsers. There are many others that I'm not at liberty to disclose.

10) Has the recommended ESAPI.jar (1.4 or 2.0 rc4) been through a security review? and if so where can I read its report?

- Yes. A major systems integrator did a line-by-line code review of both version 1.4 and 2.0. They found a few issues and made a few requests for enhancement, which we have done. They are now approved to use ESAPI on all their internal applications. I will ask them again if it is okay to share their report.

11) when Jim says "... you can build a new secure app without an ESAPI. But libs like OWASP ESAPI will get you there faster and cheaper....", do we have peer-reviewed data that suports this claim?

- Nope. Jim is speaking from personal experience building enterprise

applications both with and without ESAPI. I propose a controlled double-blind experiment where we take say 30 student programmers and have them all build the same web application (with features that should get some security). We give half of them ESAPI and let them go. Then we evaluate the results. Anyone willing to help get this done?

12) Is there a roadmap or how-to for companies that wish to adopt ESAPI.jar on an a) new application or b) existing real-world application'?

- I think the ESAPI Book or perhaps the ESAPI SwingSet (demo app) are good places to start.

13) If a development team decides to use (for example) Spring and ESAPI together in their (new or existing) application, what are the recommended 'parts' from each of those APIs (Spring and EASPI) that the developers should be using? (for example: a) use Encoding from ESAPI, b) use Authentication from Spring, c) use Authorization from ESAPI, d) use Error Handling from Spring, e) use Logging from ESAPI, etc…)

- We don’t have a guideline written for how to integrate ESAPI and the frameworks. Both libraries have been used successfully in combination with ESAPI since both support extension and ESAPI is not a framework. However, you’ll probably have to make some architectural choices and do some integration with your environment. Hopefully, we’ll get to write some adapters to help with this process soon. If you’re using ESAPI and these frameworks, please let us know so that we can learn from your experiences and share with others.

Related Articles

Other Articles in the OWASP Prevention Cheat Sheet Series

- Authentication Cheat Sheet
- Cross-Site Request Forgery (CSRF) Prevention Cheat Sheet
- Forgot Password Cheat Sheet
- Cryptographic Storage Cheat Sheet
- SQL Injection Prevention Cheat Sheet
- Transport Layer Protection Cheat Sheet
- XSS (Cross Site Scripting) Prevention Cheat Sheet
- DOM based XSS Prevention Cheat Sheet

Retrieved from "http://www.owasp.org/index.php/ESAPI_Cheat_Sheet"

Categories: OWASP Application Security Verification Standard Project | OWASP Enterprise Security API | How To | Cheatsheets

- Powered by MediaWiki