ASPECT SECURITY
application security specialists

Establishing Your Organization's Enterprise Security API

Jeff Williams
Aspect Security CEO
jeff.williams@aspectsecurity.com

OWASP Chair
jeff.williams@owasp.org

The Challenge…

- Your organization has hundreds of applications
- Every one of them needs:
  - Authentication, access control, input validation, encoding, encryption, logging, error handling, etc…
- You can use these building blocks:
  - Log4j, Reform, ACEGI, Struts, Stinger, Spring, Validator, Jasypt, JCE, JAAS, Cryptix, BouncyCastle, Anti-XSS, xml-dsig, xml-enc, lots lots more….

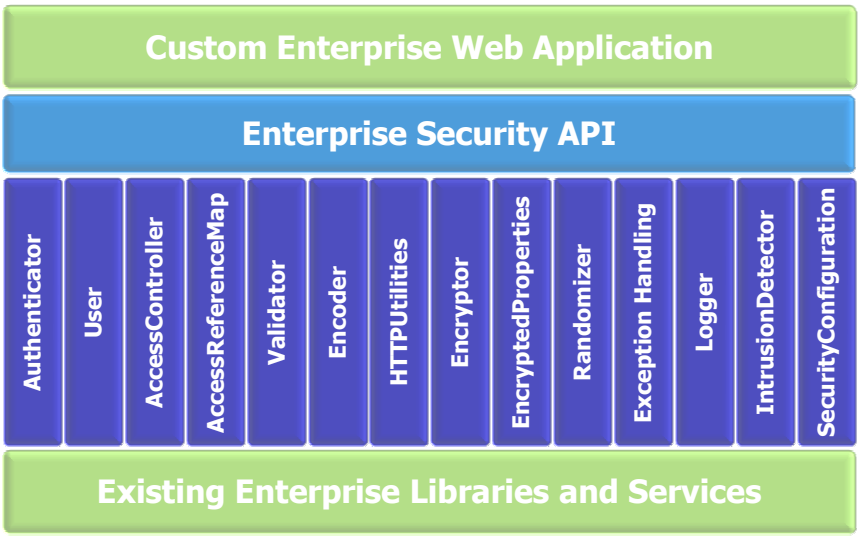
Approach

- Using security controls is different from building
  - All the security guidelines, courses, tutorials, websites, books, etc… are all mixed up because everyone builds their own controls
- Most developers shouldn't build security controls
  - When to use a control
  - How to use a control
  - Why to use a control (maybe)
- Most enterprises need the same set of calls

Design

- Only include methods that…
  - Are useful in a large percentage of applications
  - Focus on the most risky areas
- Designed to be simple to understand and use
  - Interfaces with concrete reference implementation
  - Full documentation and usage examples
- Same basic API across common platforms
  - Java EE, .NET, PHP, others?

Architecture Overview



Benefits of Integration

- **One library for each function doesn't work!**

ESAPI Feature	Benefits
Unified error handling	Comprehensive security logging and intrusion detection
Strong cryptography	Creating passwords, tokens, random filenames, keys, etc..
Identity everywhere	Simplifies API, enables access control, logging, and intrusion detection
Centralized configuration	One place to set all security relevant parameters securely
Simple consistent API	Developers actually do the security checks consistently

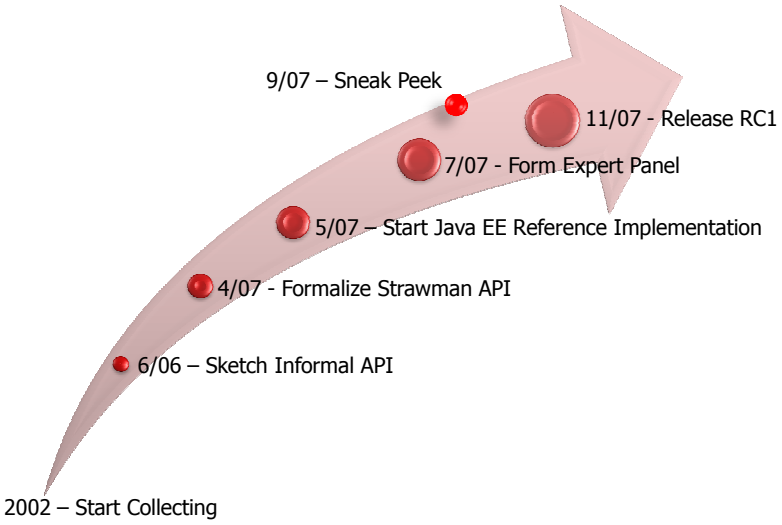
Customizing

- Your ESAPI Implementation
  - Wrap your existing libraries and services
  - Extend and customize your ESAPI implementation
  - Fill in gaps with the reference implementation
- Your Coding Guideline
  - Tailor the ESAPI coding guidelines
  - Retrofit ESAPI patterns to existing code

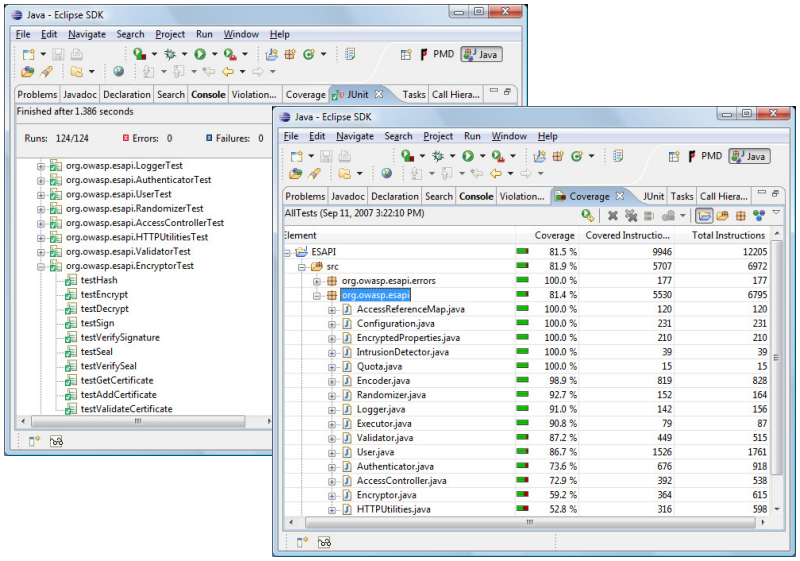
Frameworks and ESAPI

- ESAPI is NOT a framework
  - Just a collection of security functions, not "lock in"
- Frameworks already have some security
  - Controls are frequently missing, incomplete, or wrong
- ESAPI Framework Integration Project
  - We'll share best practices for integrating
  - Hopefully, framework teams like Struts adopt ESAPI

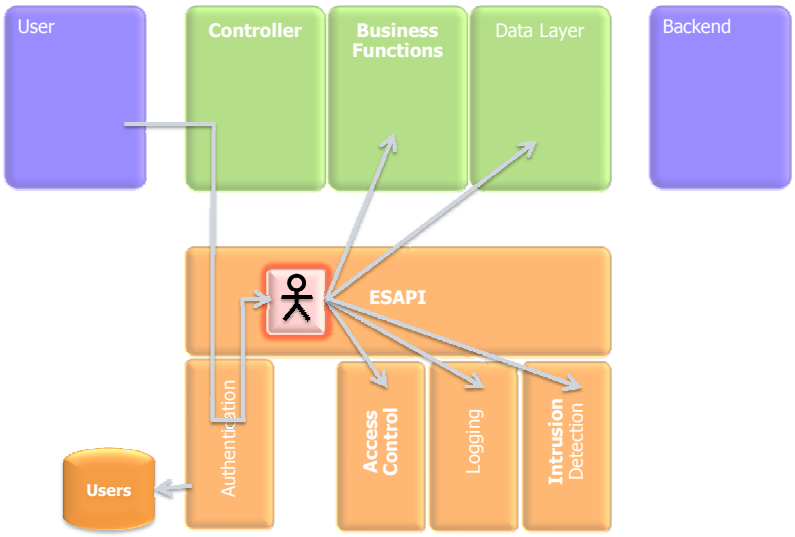
Project Plan and Status



Quality



Handling Authentication and Identity



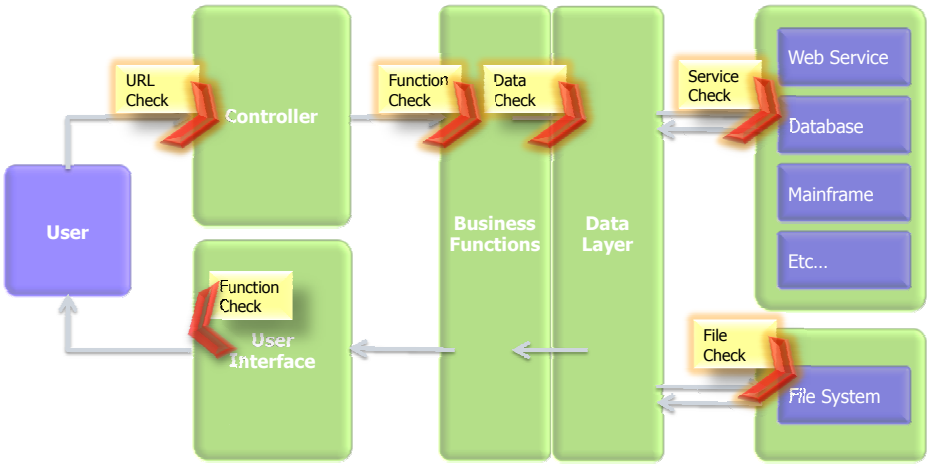
Authenticator

- Key Methods
  - createUser(accountName, pass1, pass2)
  - generateStrongPassword()
  - getCurrentUser()
  - login(request, response)
  - verifyAccountNameStrength(acctName)
  - verifyPasswordStrength(newPass, oldPass)
- Use threadlocal variable to store current User
- Automatically change session on login and logout
- Main program to set initial accounts

User

- **Key Methods**
  - changePassword(old, new1, new2)
  - disable() enable()
  - getAccountName() getScreenName()
  - getCSRFToken()
  - getLastFailedLoginTime() getLastLoginTime()
  - getRoles() isInRole(role)
  - isEnabled() isExpired() isLocked()
  - loginWithPassword(password, request, response)
  - logout(request, response)
  - resetCSRFToken() resetPassword()
  - verifyCSRFToken(token)

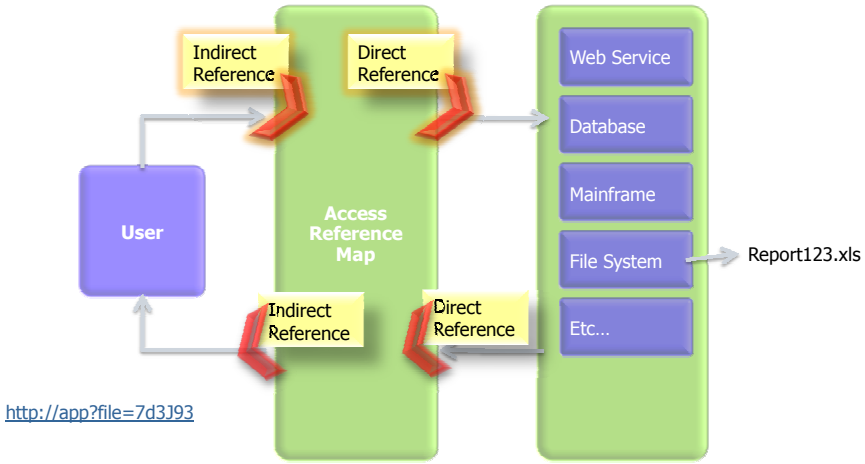
Enforcing Access Control



AccessController

- **Key Methods**
  - isAuthorizedForData(key)
  - isAuthorizedForFile(filepath)
  - isAuthorizedForFunction(functionName)
  - isAuthorizedForService(serviceName)
  - isAuthorizedForURL(url)
- **Reference Implementation (not required)**
  - /admin/* | admin | allow | admin access to /admin
  - /* | any | deny | default deny rule

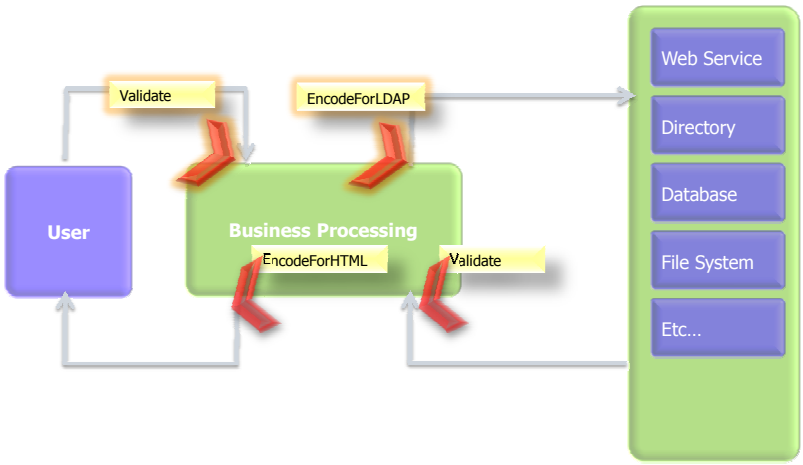
Handling Direct Object References



AccessReferenceMap

- **Key Methods**
  - getDirectReference(indirectReference)
  - getIndirectReference(directReference)
  - iterator()
  - update(directReferences)
- **Example**
  - http://www.ibank.com?file=report123.xls
  - http://www.ibank.com?file=a3nr38

Validating and Encoding Untrusted Input



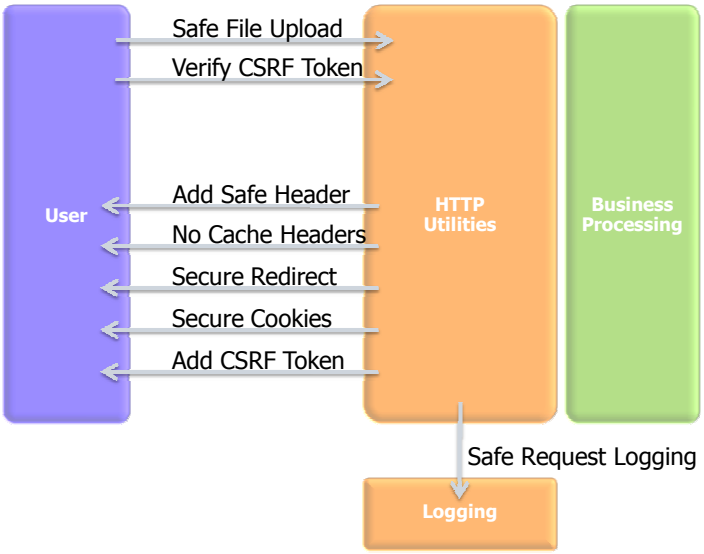
Validator

- Key Methods
  - canonicalize(input)
  - isValidFileUpload(filepath, filename, content)
  - isValidHTTPRequest (request)
  - isValidCreditCard(input)
  - isValid***** (input)
  - isValidRedirectLocation(location)
  - isValidSafeHTML(input)
  - safeReadLine(inputStream, maxchars)
- Canonicalization is really important
- Global validation of HTTP requests

Encoder

- Key Methods
  - encodeForBase64(input)
  - encodeForDN(input)
  - encodeForHTML(input)
  - encodeForHTMLAttribute(input)
  - …, encodeForJavascript, encodeForLDAP, ~~encodeForSQL~~, encodeForURL, encodeForVBScript, encodeForXML, encodeForXMLAttribute, encodeForXPath
- Function names help tell developer when to use
- Some of these are quite hard

Enhancing HTTP



HTTPUtilities

- **Key Methods**
  - addCSRFToken(href)
  - addSafeHeader(header, value, response)
  - changeSessionIdentifier(request)
  - getFileUploads(request, tempDir, finalDir)
  - killCookie(name, request, response)
  - sendRedirect(href)
  - setCookie(name, value, age, domain, path, response)
  - setNoCacheHeaders(response)
- **Safer ways of dealing with HTTP, secure cookies**

Encryptor

- Key Methods
  - decrypt(ciphertext)
  - encrypt(plaintext)
  - hash(plaintext, salt)
  - loadCertificateFromFile(file)
  - getTimeStamp()
  - seal(data, expiration) verifySeal(seal, data)
  - sign(data) verifySignature(signature, data)
- Simple master key in configuration
- Minimal certificate support

EncryptedProperties

- Key Methods
  - getProperty(key)
  - setProperty(key, value)
  - keySet()
  - load(inputStream)
  - store(outputStream, comments)
- Simple protected storage for configuration data
- Main program to preload encrypted data!

Randomizer

- Key Methods
  - getRandomInteger(min, max)
  - getRandomReal(min, max)
  - getRandomString(length, characterSet)
- Several pre-defined character sets
  - Lowers, uppers, digits, specials, letters, alphanumerics, password, etc…

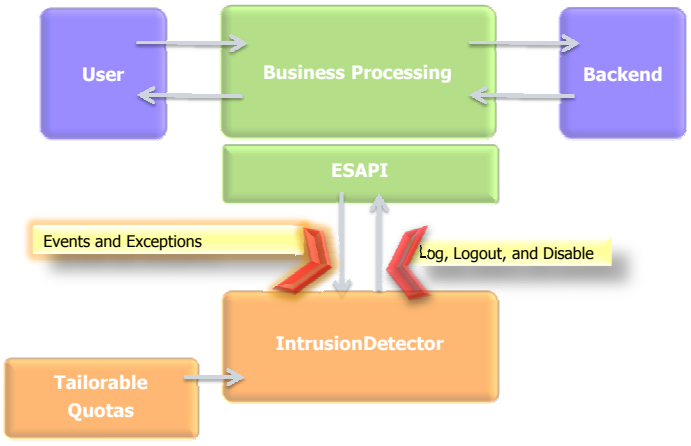
Exception Handling

- EnterpriseSecurityException
  - AccessControlException
  - AuthenticationException
  - AvailabilityException
  - CertificateException
  - EncodingException
  - EncryptionException
  - ExecutorException
  - IntrusionException
  - ValidationException
- Allows a sensible security exception framework

Logger

- **Key Methods**
  - getLogger(applicationName,moduleName)
  - formatHttpRequestForLog(request, sensitiveList)
  - logCritical(type, message, throwable)
  - logDebug(type, message, throwable)
  - logError(type, message, throwable)
  - logSuccess(type, message, throwable)
  - logTrace(type, message, throwable)
  - logWarning(type, message, throwable)
- **All EASPI exceptions are automatically logged**

Detecting Intrusions



IntrusionDetector

- **Key Methods**
  - **addException(exception)**
  - **addEvent(event)**
- **Model**
  - **EnterpriseSecurityExceptions automatically added**
  - **Specify a threshold for each event type**

    org.owasp.esapi.ValidationException.count=3
    org.owasp.esapi.ValidationException.interval=3 (seconds)
    org.owasp.esapi.ValidationException.action=logout

    (alternatives are log message, disable account)

SecurityConfiguration

- **Customizable…**
  - **Crypto algorithms**
  - **Encoding algorithms**
  - **Character sets**
  - **Global validation rules**
  - **Logging preferences**
  - **Intrusion detection thresholds and actions**
  - **Etc…**

OWASP Top Ten Coverage

OWASP Top Ten	OWASP ESAPI
A1. Cross Site Scripting (XSS)	Validator, Encoder
A2. Injection Flaws	Encoder
A3. Malicious File Execution	HTTPUtilities (upload)
A4. Insecure Direct Object Reference	AccessReferenceMap
A5. Cross Site Request Forgery (CSRF)	User (csrftoken)
A6. Leakage and Improper Error Handling	EnterpriseSecurityException, HTTPUtils
A7. Broken Authentication and Sessions	Authenticator, User, HTTPUtils
A8. Insecure Cryptographic Storage	Encryptor
A9. Insecure Communications	HTTPUtilities (secure cookie)
A10. Failure to Restrict URL Access	AccessController

Closing Thoughts

- I have learned an amazing amount (I thought I knew)
- An ESAPI is a key part of a balanced breakfast
  - Build coding guidelines, training, tools around your ESAPI
- Secondary benefits
  - May help static analysis do better
  - Enables security upgrades across applications
  - Simplifies developer training
- Next year – experiences moving to ESAPI