Real Time Application Defenses
The Reality of AppSensor & ESAPI

Michael Coates
Mozilla - Web Security Lead
mcoates@mozilla.com

http://michael-coates.blogspot.com
@_mwc

March 23, 2011

The OWASP Foundation
http://www.owasp.org
Agenda

• Power of Application Intrusion Detection
• ESAPI & AppSensor
• Release of AppSensor-Tutorial
• AppSensor @ Mozilla
AppSensor Team

AppSensor Core Team

Michael Coates
John Melton
Colin Watson

Contributors
Ryan Barnett
Simon Bennetts
August Detlefsen
Randy Janida
Jim Manico
Giri Nambari
Eric Sheridan
John Stevens
Kevin Wall
Power of Application Intrusion Detection
Status Quo
Defense Capabilities

• Build secure & hope for the best

• Would you know if your application was currently under attack?

• How confident are you against a skilled attacker?

• Is your attack alert system based on watching the NYT for a front page article?
Attack Points: Requests, Auth, Session
Attack Points: Access Control
Attack Points:
Input Validation
Attack Points:
Business Logic
Numerous Attack Points
Defend with:
Detection Points

Monday, March 21, 2011
Defend with: AppSensor Integration

- Detection Points Report to AppSensor
- AppSensor Integrates w/User Store
- Enables Response Actions against User Object
Detect & Eliminate Threat

- Strong control of authenticated portion
- Lockout user
- Disable account
- Effective attack reporting for unauthenticated portion
AppSensor Eliminates Threats

Requests Needed for Attacker vs. AppSensor

Attacker: find vulnerability

AppSensor: determine user is malicious
AppSensor Eliminates Threats

Requests Needed for Attacker vs. AppSensor

Attacker: find vulnerability
AppSensor: determine user is malicious

Block attacker & minimize threat
Current Approach

Build secure & hope for the best
AppSensor Approach

Build as secure as possible

Secure Design

AppSensor Defense

Detect & eliminate threats
Defend against the unknown

Application Compromise

Add layer of attack detection & prevention

Monday, March 21, 2011
Enhancing App Security

Build Secure
- Integrate Security into SDLC
- Security Code Review & Penetration Testing

Actively Defend
- Attack Detection Points
- Application Trend Anomaly Detection
- Automated Response to Quarantine Attackers
Why This Approach?

- AppSensor - in the app, full user object interaction, full app knowledge
- WAF - generic attack detection
- Log Analysis - slow, reactive, ineffective
ESAPI & AppSensor
Integration Status

- appsensor.jar ready to use w/ESAPI
- AppSensor + ESAPI bundle planned for ESAPI 2.0 rc8
ESAPI / AppSensor Adoption

• AppSensor
  • ModSecurity
  • Major Insurance Company - AppSensor standard for all new web apps
  • Mozilla - AppSensor detection integrated into web apps


AppSensor.jar

- Drop-in support for ESAPI
- 3 Line configuration in ESAPI.properties
- Define policies in appsensor.properties
- Add detection points in code (2-3 lines each)
- Done!
How Easy To Setup?

ESAPI.IntrusionDetector=org.owasp.appsensor.intrusiondetection.AppSensorIntrusionDetector

ESAPI.properties

IntrusionDetector.X1.count=2
IntrusionDetector.X1.interval=35
IntrusionDetector.X1.actions=log logout disableComponentForUser
IntrusionDetector.X1.disableComponentForUser.duration=30
IntrusionDetector.X1.disableComponentForUser.timeScale=m

appsensor.properties

if (AttackDetected){
    new AppSensorException( "X1","User Error Message",
        "Logged Error Message" + "("+ request.getRequestURI()+ ")"
        + " user (" + ESAPI.authenticator().getCurrentUser().getAccountName() + ")"");
}

Monday, March 21, 2011
Detecting Attacks

- 50+ attack detection points and growing
- Grouped into logical areas
  - Request, Auth, Input, Access etc
- Most have nearly zero false positive rate
- POST When Expecting GET
- Evading Presentation Access Control Through Custom POST
- Attempt to Invoke Unsupported HTTP Method

Release of AppSensor-Tutorial
AppSensor-Tutorial

AppSensor-Tutorial

- Lesson Based Application

- Concise & Simple Demo of ESAPI & AppSensor Code

- Purely JSP w/Java libs

- http://defendtheapp.com/
Lesson Format

• Simple form with text input or drop down

• Malicious data checked by ESAPI or AppSensor

• Detection point listed with response actions / intrusion count
Lesson 1:
Validate w/ ESAPI

**Lesson:** lesson 1 - attack detection via ESAPI's input validation

**Objective:** Observe how input is compared against the ESAPI property file. Invalid input is rejected and a generic event is created. Note that specific appsensor codes are not available when only using ESAPI validation.

```java
String dataResult = "";
try {
    dataResult = ESAPI.httpUtilities().getParameter(request,"attackstring");
} catch (ValidationException e){
    //ESAPI Validation Exception
    //Processed by AppSensor
    //Automatically
}
```

**ESAPI.properties**

```
Validator.HTTPParameterName=^[a-zA-Z0-9_]{0,32}$
Validator.HTTPParameterValue=^[a-zA-Z0-9.\-\/+=_ ]*$
```

**IntrusionDetector.properties**

```
IntrusionDetector.Total.count=3
IntrusionDetector.Total.interval=30
IntrusionDetector.Total.actions=logout
```

**appsensor.properties**

```
```
Lesson 2

Validate w/ AppSensor

• Use AppSensor AttackDetectorUtils.verifyXSSAttack

• Customizable black list approach (regex)

• Catch obvious XSS probes
  • alert(document.cookie)
  • <img src=.*script
  • <iframe>.*/iframe>
Lesson 2 - The Code

```java
dataResult = request.getParameter("attackstring");

boolean attackDetected =
    org.owasp.appsensor.AttackDetectorUtils.verifyXSSAttack(dataResult);

if (attackDetected) {
    dataResult = "Exception Caught By ESAPI Validation";
    new AppSensorException(
        appsensorID,"Invalid Input per AppSensor Detection",
        "Attacker is sending input that violates defined whitelists"
        + "("+ request.getRequestURI()+ ")"
        + " user ("
        + ESAPI.authenticator().getCurrentUser().getAccountName() + ")"));
    dataResult = "removed";
}
```

lesson2.jsp
Lesson 2

appsensor.properties

- Define regex black list of XSS patterns
- Black list OK for attack detection
- Define response thresholds as normal

```properties
xss.attack.patterns="\"">\<script>,
<script.*document\.cookie,
<script>,
<IMG.*SRC.*=.*script,
<iframe>.*</iframe>
...
IntrusionDetector.IE1.count=3
IntrusionDetector.IE1.interval=30
IntrusionDetector.IE1.actions=log,logout

appsensor.properties
```
lesson 3
Per User Page Blocking

- Disable user’s access to the page
- Good solution for sensitive operations - transfer funds, update address
- Just affects malicious user
- Simple with AppSensor

Detection Point: IE13
Count: 2
Interval: 35
Response: [disableComponentForUser]
Lesson 3 - The Code

**Lesson:** lesson 3 - Feature Locking Per User

**Objective:** Observe how input is checked by AppSensor for common XSS patterns. After the threshold is reached, the page functionality is locked for this user. Log out to see that the anonymous user can still see the page.

```java
ASUser user = APPSENSOR.asUtilities().getCurrentUser();

boolean isActive =
AppSensorServiceController.isServiceActiveForSpecificUser
(request.getRequestURI(),user);

if (!isActive){
  <%- This page has been disabled %>
} else{
  //display normal page
}
```

lesson3.jsp
Lesson 3

appsensor.properties

- Define normal thresholds
- Define how long page is disabled for user (30 minutes)

IntrusionDetector.IE12.count=2
IntrusionDetector.IE12.interval=35
IntrusionDetector.IE12.actions=disableComponentForUser
disableComponentForUser.duration=30
disableComponentForUser.timeScale=m
Lesson 4
Full Feature Blocking

• Block access to all users
• Possible for critical pages
• Better to shutoff page and investigate than risk compromise
Lesson 4 - The Code

boolean isActiveForEveryone = AppSensorServiceController.isServiceActive(request.getRequestURI());

if (!(isActiveForEveryone)){
  %>Page has been disabled for everyone<%
}

lesson4.jsp

IntrusionDetector.IE12.actions=disableComponent
IntrusionDetector.IE12.disableComponent.duration=10

appsensor.properties
Additional Response Capabilities

http://www.owasp.org/index.php/
File:Owasp-appsensor-responses.pdf
## Additional Response Capabilities

### Table 1: AppSensor Responses

<table>
<thead>
<tr>
<th>Category</th>
<th>Type</th>
<th>Description</th>
<th>Response (Added since v1.1)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Silent</td>
<td>User unaware of application’s response</td>
<td>ASR-A Logging Change</td>
</tr>
<tr>
<td></td>
<td>Passive</td>
<td>Changes to user experience but nothing denied</td>
<td>ASR-B Administrator Notification</td>
</tr>
<tr>
<td></td>
<td>Active</td>
<td>Application functionality reduced for user(s)</td>
<td>ASR-C Other Notification</td>
</tr>
<tr>
<td></td>
<td>Intrusive</td>
<td>User’s environment altered</td>
<td>ASR-D User Status Change</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ASR-E User Notification</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ASR-F Timing Change</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ASR-G Process Terminated</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ASR-H Function Amended</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ASR-I Function Disabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ASR-J Account Logout</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ASR-K Account Lockout</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ASR-L Application Disabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ASR-M Collect Data from User</td>
</tr>
</tbody>
</table>
AppSensor @ Mozilla
Mozilla Threat Profile

• Lots of users
• Many web apps
• Apps constantly growing & changing
• All code open source
Mozilla Services

- Firefox Sync
  - Millions of users
  - Service based app
  - Stores encrypted user data
- Example detection points
  - Credential mismatch within URL request
  - Tampering with reset code
  - Account delete attempt without password

Monday, March 21, 2011
What to Capture

- Threat model attack scenarios
  - Access Control Failures
  - Account lockouts
  - Failed CAPTCHA
- Monitor trends of interesting events
  - New privileged account created
  - Password reset requested
  - Account creations
  - Sensitive bug access
  - New attachment
SIM Deployment
Common Event Format (CEF)

- Emerging standard on logging format
- Easily parsed by security integration manager (sim)
- Enables AppSensor Logging

CEF:0|Mozilla|MozFooApp|1.0 |ACE0|Access Control Violation|8|rt=01 31 2010 18:30:01 suser=janedoe suid=55 act=Action Denied src=1.2.3.4 dst=2.3.4.5 requestMethod=POST request=http://foo.mozilla.org/foo/abc.php?a=b cs1Label=requestClientApplication cs1=Mozilla/5.0 (Macintosh; U; Intel Mac OS X 10.6; en-US; rv:1.9.2.2) Gecko/ 20100316 Firefox/3.6.2 msg=Additional Data here
if (!$authdb->authenticate_user(fix_utf8_encoding($auth_pw)))
{
    if ($cef)
    {
        $message = new CommonEventFormatMessage(
            WEAVE_CEF_AUTH_FAILURE,
            'User Authentication Failed', 3,
            array('username' => $url_user, 'requestip' =>
                get_source_ip()));
        $cef->logMessage($message);
    }
    report_problem('Authentication failed', '401');
}
Python CEF @ PyPI

http://pypi.python.org/pypi/cef/
Trend Analysis

Sync - Weekly Report
7 Day Window 08-29-2010 to 09-04-2010
AppSensor - More Info


http://code.google.com/p/appsensor/

owasp-appsensor-project@lists.owasp.org

mcoates@mozilla.com
michael.coates@owasp.org
http://michael-coates.blogspot.com
@_mwc

Monday, March 21, 2011