3. AppSensor project

Category: Protection

Type: Documentation (& Tool)

Status: Beta

A framework for detecting and responding to attacks from within the application – application layer intrusion detection and prevention
Background

- Established Summer 2008
- AppSensor book, developer guide and planning workbook
- Presented at multiple conferences
- Team:
  - Michael Coates
  - John Melton
  - Colin Watson
- OWASP Live CD & OWASP Broken Web Apps
Resources

- **Source in Google Code, demo WAR**
  

- **Recent video presentations by Michael Coates**
  
  - Real Time Application Defenses - The Reality of AppSensor & ESAPI
    [http://vimeo.com/15726323](http://vimeo.com/15726323)
  
  - Automated Application Defenses to Thwart Advanced Attackers

- **Live demo implementation**
  
The threat: advanced attackers

- Skilled
- Financially Motivated
- Organized
- Patient and Persistent
- In Possession of Your Source Code
- Outside & Inside Your Company
Application defence failures

- “We use SSL”
- “We use firewalls”
- “We use deep packet inspection”
- “We installed a web application firewall”
“We use SSL”

- SSL Protects Transmitted Traffic
- No Guarantee or Inspection of Data
- Zero Impact to Attackers
- Provides Zero Protection to Site Against Attackers
“We use firewalls”

- Purpose of Firewall: Allow or Deny Access via Port
- Necessity of Working Web App: Allowed Access via 80 or 443
- Result: Firewall is an Open Door
“We use deep packet inspection”

- Performed by Generic Network Appliance
- No Knowledge of Application Attacks

- Example Attack: Access Control Attack via Direct Object References
- Not Detected by DPI

GET /updateProfile?id=52473&pass=newpass
Host: yourSite.com
“We installed a web application firewall”

- Custom application + Generic Solution != success
- Application context not available
- No concept of access violations

No attacks here, please proceed
Detecting attacks the right way

Integration
  ▶ Detect INSIDE the application
  ▶ Understand business logic

Effectiveness
  ▶ Minimal false positives
  ▶ Immediate response

Effort
  ▶ Automatic detection
  ▶ No manual work required
Inside the application is best

- Understand application & business context
- Integration with authentication & user store
Establishing detection points

Signature based events:
- Request
- Authentication
- Session
- Access control
- Input
- Exception
- Command injection
- File input/output
- Honey trap

Behaviour based events:
- User trend
- System trend
- Reputation
Detecting malicious users

- Many malicious attacks are obvious and not “user error”
  - POST when expecting GET
  - Tampering with headers
  - Submission of XSS attack
Examples of malicious actions

- Bypassing client side input validation
- Transaction using functionality not visible to user role
- Multiple access control violations
- Change of user agent midsession
- Double encoded data
How does AppSensor protect the app?

Requests Needed for Attacker vs. AppSensor

Attacker: find vulnerability

AppSensor: determine user is malicious

# of malicious requests
**AppSensor is faster than an attacker**

- User identified as malicious & blocked before vulnerability is found

![Diagram showing AppSensor detecting and blocking attacks](image-url)
From theory to reality

- Demo Social Networking Application
- Leverages AppSensor Principles
Detection points

- home.jsp
  - RE4

- updateProfile.jsp
  - RE4, IE1, IE4

- friends.jsp
  - RE4

- friendRequests.jsp
  - RE4, 3 x ACE1

- addFriend.jsp
  - RE4

- search.jsp
  - RE4, IE1, RE3
AppSensor vs scanners

- Tools attempt 10,000s of generic attacks
- AppSensor stops automated scans nearly instantly
AppSensor vs advanced attackers

- Very difficult for attacker
- Requires advanced obfuscation for each attack
- Multiple probes == detection
Detecting/preventing an application worm

- Can you find / fix all XSS?
- Pattern matching easily foiled
- Block the common factor!
  - Worms use XSS and CSRF for propagation
  - 1000% usage increase \(\rightarrow\) problem
  - Our example:
    (updateProfile, updateStatus, updateName)
Case study: Samy

- MySpace Application Worm
- XSS worm embedded in User Profile
  - Added Samy as friend
  - Infected viewer’s profile with XSS
- Exponential Growth of Samy’s friends
  - 10 hours – 560 friends,
  - 13 hours – 6400 friends,
  - 18 hours – 1,000,000 friends,
  - 19 hours – site down for repair
Samy vs AppSensor

- AppSensor detects uptick in addFriend usage
- Compares against trended info
- Automatic response initiated
  - Alerts Admin +%200 Add Friend Usage
  - Alerts Admin 2\textsuperscript{nd} time +%500 Add Friend Usage
  - Automatically shuts down Add Friend Feature

Result:
- Worm Contained,
- Add Friend Temporarily Disabled,
- Site Stays Up
Trend monitoring benefits

- General
  - Insight to scripted traffic / attack probing

- Application worms
  - Auto detection of attacks
  - Automatic worm containment
  - Maintain overall site availability

- Fraud detection
  - Real time detection
  - Context specific
  - System-wide knowledge
AppSensor Specification & Design 1
AppSensor Specification & Design 2

Diagram showing the flow of data and user interactions in a web application, including interactions with users, web servers, web pages, databases, and backup devices.
# AppSensor Specification & Design 3

## Detection Points

<table>
<thead>
<tr>
<th>Summary</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Request Exceptions (RE)</td>
<td>Assert Control Exceptions (ACE)</td>
</tr>
<tr>
<td>3.1 Unexpected HTTP Command</td>
<td>ACE: Identifying Mix-Login within a GET for Direct Object Access Attempt</td>
</tr>
<tr>
<td>3.2 Attempt to Invoke Unsupported HTTP Method</td>
<td>ACE: Modifying Parameter Within POST for Direct Object Access Attempt</td>
</tr>
<tr>
<td>3.3 GET When Expecting POST</td>
<td>ACE: Forced Browsing Attempt</td>
</tr>
<tr>
<td>3.4 POST When Expecting GET</td>
<td>ACE: Coating Presentation Layer Control Through Custom POST</td>
</tr>
<tr>
<td>3.5 Application Exception (ACE)</td>
<td></td>
</tr>
<tr>
<td>3.6 High Value of Input Data Integrity</td>
<td></td>
</tr>
<tr>
<td>3.7 Unexpected Quantity of Characters in Parameter</td>
<td></td>
</tr>
<tr>
<td>3.8 Unexpected Types of Characters in Parameter</td>
<td></td>
</tr>
<tr>
<td>3.9 replay of data attempts</td>
<td></td>
</tr>
<tr>
<td>Authorization Exceptions (ACE)</td>
<td></td>
</tr>
<tr>
<td>4.1 Login Failed</td>
<td></td>
</tr>
<tr>
<td>4.2 Multiple Failed Passwords</td>
<td></td>
</tr>
<tr>
<td>4.3 High Value of Usage Attempts</td>
<td></td>
</tr>
<tr>
<td>4.4 Unexpected IpAddress of Character in Username</td>
<td></td>
</tr>
<tr>
<td>4.5 Unexpected Quantity of Characters in Password</td>
<td></td>
</tr>
<tr>
<td>4.6 Unexpected Types of Character in Username</td>
<td></td>
</tr>
<tr>
<td>4.7 Unexpected Types of Character in Password</td>
<td></td>
</tr>
<tr>
<td>4.8 Providing One IP Address Incorrect</td>
<td></td>
</tr>
<tr>
<td>4.9 Unexpected Only the Password</td>
<td></td>
</tr>
<tr>
<td>5.1 Addressing FTP Transfer</td>
<td></td>
</tr>
<tr>
<td>5.2 Missing FTP Variables</td>
<td></td>
</tr>
<tr>
<td>5.3 Unmatched FTP Variables</td>
<td></td>
</tr>
<tr>
<td>5.4 Delete the Last Name of Common Customers</td>
<td></td>
</tr>
<tr>
<td>Session Exceptions (SE)</td>
<td></td>
</tr>
<tr>
<td>6.1 Unexpected Session Cookie</td>
<td></td>
</tr>
<tr>
<td>6.2 Adding New Cookies</td>
<td></td>
</tr>
<tr>
<td>6.3 Deleting Existing Cookies</td>
<td></td>
</tr>
<tr>
<td>6.4 Deleting Unmatched User's Third Session ID or Cookies</td>
<td></td>
</tr>
<tr>
<td>6.5 Session Information Change During Session</td>
<td></td>
</tr>
<tr>
<td>6.6 Change of User Agent and Session</td>
<td></td>
</tr>
<tr>
<td>6.7 Increase in Session Count</td>
<td></td>
</tr>
<tr>
<td>OWASP</td>
<td>28</td>
</tr>
</tbody>
</table>
## AppSensor Specification & Design 4

### Locally Scoped Detection Points

<table>
<thead>
<tr>
<th>Type</th>
<th>Identity</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AE6</td>
<td>21</td>
<td>Username form value contains invalid characters</td>
</tr>
</tbody>
</table>
AE6: Unexpected type of Char in Username

21 Dublin Site www.example.com:80

Application log in /login.aspx

Validation of username form element value

Form.frmUsername None

The username field is compared to a whitelist of allowable characters. This property is defined in the site’s database, and allocated to the form identity and specific element name.

NB This detection point ONLY matches on the specific form identity, entry point and field name. Invalid entry points, form identities and other field names need to be examined separately.
## AppSensor Specification & Design 6

### Collection Points

<table>
<thead>
<tr>
<th>Identity</th>
<th>Target</th>
<th>Module</th>
<th>Function</th>
<th>Entry Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>21</td>
<td>Username</td>
<td>dite.dbo/auth</td>
<td>checkUser</td>
<td>/login.aspx</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>/loginTask.aspx</td>
</tr>
</tbody>
</table>
AppSensor Specification & Design 7

- Model development
- Optimisation
- Code location
- Attack analysis
## AppSensor Specification & Design

### Response Actions

<table>
<thead>
<tr>
<th>Summary</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ASRA</td>
<td>ASRC</td>
</tr>
<tr>
<td>Classifications</td>
<td>Logging</td>
</tr>
<tr>
<td>Category</td>
<td>Client</td>
</tr>
<tr>
<td>Description</td>
<td>The granularity of logging is changed (typically more logging)</td>
</tr>
<tr>
<td>Considerations</td>
<td>-</td>
</tr>
<tr>
<td>Examples</td>
<td>Example 8: Capture sanitized request headers and response bodies</td>
</tr>
<tr>
<td></td>
<td>Example 2: Full stack trace of error messages logged</td>
</tr>
<tr>
<td></td>
<td>Example 3: Record DM data on user ID</td>
</tr>
<tr>
<td></td>
<td>Example 4: Record DM data on user IP address</td>
</tr>
<tr>
<td></td>
<td>Example 4: Security logging level changed to include informational messages</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Summary</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ASRA</td>
<td>ASRC</td>
</tr>
<tr>
<td>Classifications</td>
<td>Administrator Notification</td>
</tr>
<tr>
<td>Category</td>
<td>Client</td>
</tr>
<tr>
<td>Description</td>
<td>A notification message is sent to the application administrator(s)</td>
</tr>
<tr>
<td>Considerations</td>
<td>-</td>
</tr>
<tr>
<td>Examples</td>
<td>Example 8: Email sent to everyone in the administration team</td>
</tr>
<tr>
<td></td>
<td>Example 2: SMS sent to the user administrator</td>
</tr>
<tr>
<td></td>
<td>Example 3: Visual indicator displayed on an application monitoring dashboard</td>
</tr>
<tr>
<td></td>
<td>Example 4: Audible alarm in the control room</td>
</tr>
</tbody>
</table>
AppSensor Specification & Design 9

- Strategic requirements
- Thresholds
- Model tuning

- Implementation

- Monitoring and tuning
Bring AppSensor into your application

A. Build it into requirements
B. Develop your own
  - Detection points:
  - AppSensor methodology:
C. ESAPI
  - AppSensor Integration into Java ESAPI
D. Security Information/Event Management?
  - Add detection points into application
  - Integrate logging into real time monitor
"Other elements of the Pentagon's strategy include developing active defenses - technologies that detect attacks and probes as they occur, as opposed to defenses that employ only after-the-fact detection and notification..."


"[develop a] framework for capturing and analyzing application and session data in order to isolate criminal behaviors"

CISO, US bank, February 2011
Full day AppSensor training (provisional)

- **AppSec EU**
  6\(^{th}\)-10\(^{th}\) June, Dublin, Ireland
  [http://www.appseceu.org](http://www.appseceu.org)

- **AppSec USA**
  20\(^{th}\)-23\(^{rd}\) September, Minneapolis, USA
  [http://www.appsecusa.org](http://www.appsecusa.org)
End

colin.watson(at)owasp.org