OWASP APPSENSOR

DETECT AND RESPONSE
TO ATTACKS FROM WITHIN
THE APPLICATION

SUMMER OF CODE 2008

Michael Coates
Senior Application Security Engineer
Aspect Security, Inc.
michael.coates@aspectsecurity.com

TABLE OF CONTENTS

Contents

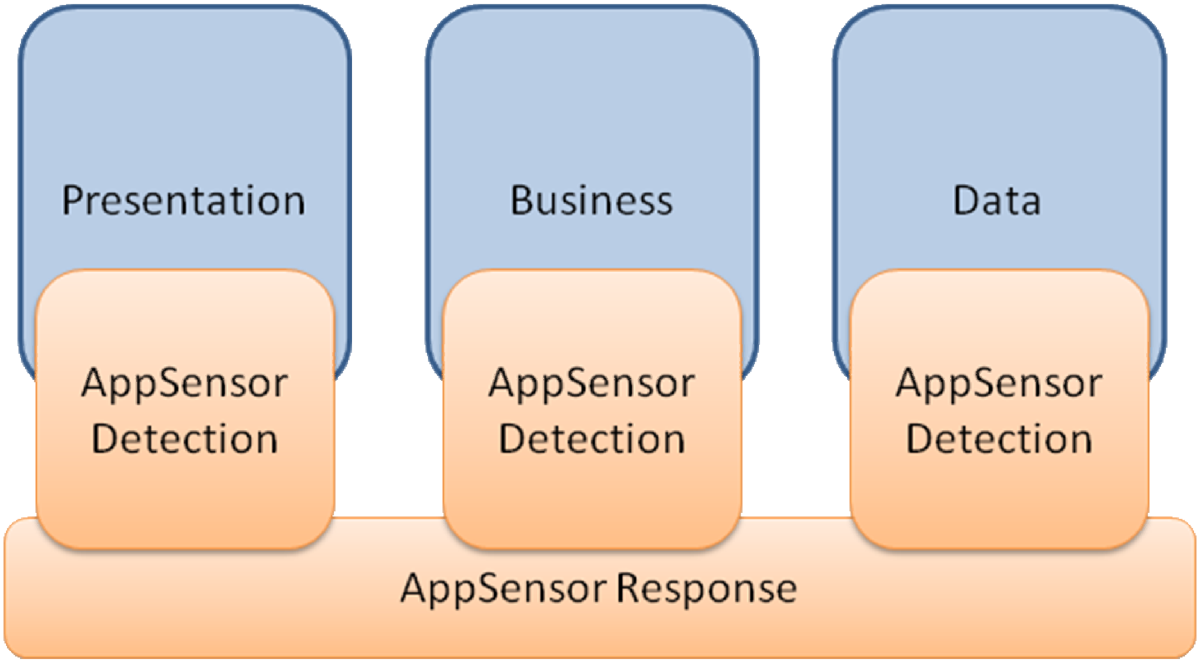
INTRODUCTION

The concept of AppSensor is to detect malicious activity within an application before a user is able to identify and exploit a vulnerability. This objective is possible because many vulnerabilities will only be discovered as a result of trial and error by the attacker. If AppSensor can identify an attacker probing for potential vulnerabilities and take responsive action quickly, it may be possible to prevent the attacker from identifying an exploitable vulnerability.

The AppSesnsor document is a conceptual framework and not a tool or library. This document offers prescriptive guidance to implement intrusion detection capabilities into existing application utilizing standard security controls and recommendations for automated response policies based upon detected behaviour.

ARCHITECTURE

AppSensor contains two modules, a detection unit and a response unit. The detection unit is responsible for identifying malicious behavior based upon defined polices. Detection points can be integrated into presentation, business and data layers of the application. The detection unit reports activity to the response unit. The response unit will take an action against the user. The action taken will depend upon the whether the event is an evident attack or a suspicious event, the user's history of malicious actions and the defined policy of response actions.



AppSensor will be integrated into the application such that a specific exception will be thrown whenever the application detects a suspicious or attack event. AppSensor will be aware of the thrown exception and catalog this event and applicable details. Per the response policy, AppSensor will take action against the responsible user which can include security warnings, account lockout, admin notification etc. AppSensor must have appropriate rights and hooks within the application to perform such response actions.

DETECTION

When detecting malicious activity the system must distinguish between two possible scenarios. First, the detected activity may have been caused by an unintentional user error or by a crafty attacker seeking to hide their attack attempts. However, since the detected activity could result in an undesirable system response, it is important to not disregard this activity entirely. This type of activity will be referred to as “Suspect”. This classification is used since it is not clear from the single event if the user is intentionally performing malicious actions against the system.

Second, the action could be clearly an intentional malicious activity. These types of actions are highly unlikely to have been generated by a user mistake and are highly likely to be an attack attempt against the application. This type of activity will be referred to as “Attack” since it is evident that the user is malicious and attempting to perform an illegal operation on the system.

Detected Activity Possibilities	
Suspect ***Suspicious Activity***	**Example:** The user submits a username which contains the characters ‘; at the end. **Analysis:** This could be the result of the user accidently hitting these two keys when attempting to press enter. Also, this could be a user attempting to discover a SQL injection vulnerability in the login page.
Attack ***Clear Malicious Activity***	**Example:** The user submits a URL with a parameter containing the value ‘1=1—‘. **Analysis:** This is a clear attack using SQL injection and would not be caused by any sort of user error.

In order to determine which category the malicious activity belongs to it is important to consider the following questions:

- Could this activity result from a typo or inadvertent key press by the user?
- Does the user have to leave the normal flow of the application to perform this activity?
- Are additional tools or software needed to perform the identified activity?

It is important to accurately classify detected activities as Suspect or Attack so that responsive action is not unjustly performed against a non-malicious user.

DETECTION: ATTACK TYPE

The following Exception Types are used within AppSensor. The left column "Exception" is the type of exception which is thrown. The right column "# of Detection Points" is the total number of detection points defined for the exception type. The number and types of detection points defined in this document represent detection for common attacks which could be performed on most applications. When implementing AppSensor, it is recommended to enhance or modify the detection points to aptly suit the application.

Exception	# Detection Points
Request	4
Authentication	11
Access Control	6
Session	4
Input	2
Encoding	2
Command Injection	4
File IO	2
User Trend	4
System Trend	3

Each detection point will contain the following information:

- **ID**: A unique identifier for the detection point
- **Event**: The title of the event detected
- **Exception Type**: The exception category of the detected event
- **Description**: Text to describe the malicious activity
- **Considerations**: Any items that should be considered when implementing the detection point
- **Example**: An example of a user action which would trigger this event

DETECTION: INFORMATION CAPTURED

It is vital that when an event is detected that sufficient information is recorded. The following information is recommended to record whenever an event is detected.

- Time of attack
- URI / URL
- Logged in user
- Malicious Activity Detected
- Entire HTTP Request

DETECTION POINTS

The following table provides an overview of the recommended detection points within the application. See Appendix A: Detailed Event Description for a description of each event.

SIGNATURE BASED EVENTS

	ID	Event	Exception
Request	RE1	Unexpected HTTP Commands	RequestException
	RE2	Attempts To Invoke Unsupported HTTP Methods	RequestException
	RE3	GET When Expecting POST	RequestException
	RE4	POST When Expecting GET	RequestException
Authentication	AE1	Use Of Multiple Usernames	AuthenticationException
	AE2	Multiple Failed Passwords	AuthenticationException
	AE3	High Rate Of Login Attempts	AuthenticationException
	AE4	Unexpected Quantity Of Characters In Username	AuthenticationException
	AE5	Unexpected Quantity Of Characters In Password	AuthenticationException
	AE6	Unexpected Types Of Characters In Username	AuthenticationException
	AE7	Unexpected Types Of Characters In Password	AuthenticationException
	AE8	Providing Only The Username	AuthenticationException
	AE9	Providing Only The Password	AuthenticationException
	AE10	Adding Additional POST Variables	AuthenticationException
	AE11	Removing POST Variables	AuthenticationException

Session	SE1	Modifying Existing Cookies	SessionException
	SE2	Adding New Cookies	SessionException
	SE3	Deleting Existing Cookies	SessionException
	SE4	Substituting Another User's Valid Session ID Or Cookie	SessionException
	SE5	Source IP Address Changes During Session	SessionException
	SE6	Change Of User Agent Mid Session	SessionException
Access Control	**ACE1**	Modifying URL Arguments Within A GET For Direct Object Access Attempts	AccessControlException
	ACE2	Modifying Parameters Within A POST For Direct Object Access Attempts	AccessControlException
	ACE3	Force Browsing Attempts	AccessControlException
	ACE4	Evading Presentation Access Control Through Custom Posts	AccessControlException
Input	**IE1**	Cross Site Scripting Attempt	InputException
	IE2	Violations Of Implemented White Lists	InputException
Encoding	**EE1**	Double Encoded Characters	EncodingException
	EE2	Unexpected Encoding Used	EncodingException

Command Injection	**CIE1**	Blacklist Inspection For Common SQL Injection Values	CommandInjectionException
	CIE2	Detect Abnormal Quantity Of Returned Records.	CommandInjectionException
	CIE3	Null Byte Character In File Request	CommandInjectionException
	CIE4	Carriage Return Or Line Feed Character In File Request	CommandInjectionException
File IO	**FIO1**	Detect Large Individual Files	FileIOException
	FIO2	Detect Large Number Of File Uploads	FileIOException

BEHAVIOR BASED EVENTS

User Trend	**UT1**	Irregular Use Of Application	UserTrendException
	UT2	Speed Of Application Use	UserTrendException
	UT3	Frequency Of Site Use	UserTrendException
	UT4	Frequency Of Feature Use	UserTrendException
System Trend	**STE1**	High Number Of Logouts Across The Site	SystemTrendException
	STE2	High Number Of Logins Across The Site	SystemTrendException
	STE3	High Number Of Same Transaction Across The Site	SystemTrendException

RESPONSE

The power of AppSensor is its placement within the application for detection and its ability to respond to malicious activity in real time. The most common response activities will be user warning messages, logout, account lockout and admin notification. However, since AppSensor is connected into the application, the possibilities of response actions are limited only by the capabilities of the application.

When developing the response policy it is vital to determine the appropriate thresholds for response actions. The objective is to appropriately deter malicious activity and prevent determined attackers from successfully identifying vulnerabilities, while minimizing the impact when false positives are recorded from non-malicious user activity.

DETERMINING MALICIOUS INTENT

When responding to detected malicious activity the system must distinguish between three possible scenarios. First, the activity may have been unintentional and caused by user error. Second, the activity could be suspicious activity that is difficult to conclusively determine if malicious and intentional. Third, the action could be clearly an intentional malicious activity.

Malicious Intent

- Possible User Error
- Possible Attack
- Clear Malicious Activity

In order to determine which category the malicious activity belongs to it is important to consider the following questions:

- Could this activity result from a typo or inadvertent key press by the user?
- Does the user have to leave the normal flow of the application to perform this activity?
- Are additional tools or software needed to perform the identified activity?
- Could a common application error be responsible for this activity?

Suspicious Event:

- Could occur during user experience with site or browser
- Could occur as result of non-malicious user error

Attack Event:

- Outside of the normal application flow
- Requires Specials Tools
- Requires Special Knowledge

CATEGORIZING MALICOUS INTENT

Some events detected by AppSensor could be the result of user error and not a malicious attack. AppSensor must work to achieve two goals. First, ensure that non-malicious users that have made inadvertent mistakes are not unjustly punished. Second, detect and respond to malicious attack actions. To achieve these goals the detection events have been categorized into two classes, Suspect and Attack. Suspicious events are those actions which could be the result of a user error or an intentional, but non-malicious, user action. For example, a non-malicious user may inadvertently press the less than character "<" when attempting to press the letter "m" and submit this field. This inadvertent action should not be interpreted as a potential XSS or injection attack.

Attack events are those activities which are highly unlikely to be the actions of a non-malicious user. These events include modifying posts and injecting well formed SQL attack strings. Due to the necessity of specialized tools, security specific knowledge or customized attacks, these events are treated as intentional malicious actions. More generally, any action performed by a user which is not presented in the user interface may be classified as a malicious and intentional attack.

Establishing two categories of events allows AppSensor to immediately react to the clearly malicious actions while still monitoring the suspicious events. It is quite possible that a large number of suspicious events are actually a determined attacker attempting to keep a low profile while performing tests.

See Appendix B: Response – Suspect vs. Attack Events for a breakdown of events by the categories of Suspect or Attack.

RESPONSE ACTIONS

Detection of events is not useful without an automated response to deter and prevent a successful compromise. A response policy should be established which sets specific thresholds and response actions based on the detection actions of a user. In each case, the event and response action taken should be logged.

Example Response Actions	
Security Violation Message	Provide a visual warning message to the user to deter further attack activity. Examples: "A Security Event Has Been Detected And Logged" "A Security Error Has Occurred And Has Been Logged" **Pros**: This may deter the casual attacker by alerting them that their activities are being logged. **Cons**: This will not deter a determined attacker and will provide the attacker with some knowledge of what events are

	being detected as malicious
Account Logout	Log the account out. **Pros**: This action will cause difficulty with most automated tools since the attack or scanning sequence will be interrupted after a small number of attacks. Logging out the user account will also provide a clear indication to a user that the performed actions are being logged and the application is responding to the attacks. **Cons:** Automated tools can be modified to automatically re-authenticate to bypass this response action.
Account Lockout	Lock the user account. The user account could be permanently locked, unlocked after a pre-set amount of time (such as 30 minutes), or unlocked after the user has contacted the help desk. **Pros:** Locking the account will cease the attack activity. **Cons:** If the site does not control the creation of accounts, then an attacker could generate numerous accounts and use each one until it is locked.
Administrator Notification	Notify the administrator via email or other methods of the malicious activity. **Pros:** An administrator could take additional actions or enable additional logging capabilities in real-time. Notification is especially effective for the system trend events which require human analysis. **Cons:** If used too often, this notification could become another type of log which is mostly ignored.

RECOMMENDED THRESHOLDS

All security events generated by a user should be stored in a centralized location. Centralization of event data allows the system to detect a user performing multiple attacks with a single area of the application and also detects a user that is performing attacks across multiple areas of the application.

It is recommended that the response threshold consider the total number of security events generated from all categories. The following table describes a recommended set of response thresholds.

- 3 Suspicious Events = 1 Security Event
- 1 Attack Event = 1 Security Event
- User events totals cleared on rolling 24 hrs basis

The following table illustrates a sample threshold for AppSensor. These values should be customized to meet the specific needs of the application. For example, a highly sensitive application operating within a restricted environment may consider even the most subtle suspicious activity to be a security event where account lockout and administration notification is appropriate.

Security Events	Response Action
2	Security Violation Message
3	Security Violation Message + Account Logout
5	Security Violation Message + Account Lockout 5 minutes
7	Security Violation Message + Account Lockout 30 minutes
10	Administration Notification + Account Lockout Indefinite

MONITORING SYSTEM TREND EVENTS

Normal activity such as logging in or out of the application or performing a particular action such as updating an address or password can be monitored to detect attacks. If these events are monitored on a regular basis to determine expected activity levels, then a dramatic increase in a particular type of activity can be detected and brought to the attention of an administrator.

It is difficult to implement an automated response to trend events since a sudden influx in activity could be the result of a variety of non-attack items. However, there is a point where a spike in activity becomes so dramatic that the spike should be investigated to determine the cause.

For example, advanced XSS worms and CSRF attacks on popular websites can cause significant damage the application and users. XSS worms often spread by planting the payload of the worm within some portion of the user’s profile. The ability to detect a dramatic spike in the use an update user profile method would enable the application to detect the presence of the XSS worm.

Similarly, CSRF attacks may attempt to perform an operation on behalf of the user and then logout the account. A sharp increase in the use of a particular method or the logout feature may enable the application to detect an attack in progress.

RECOMMENDED SYSTEM TREND THRESHOLDS

Thresholds should be established for automated response due to a sudden shift in system trends. System trend monitoring will not be useful without an automated response, since the value of this monitoring is proactively identifying and stopping an attack.

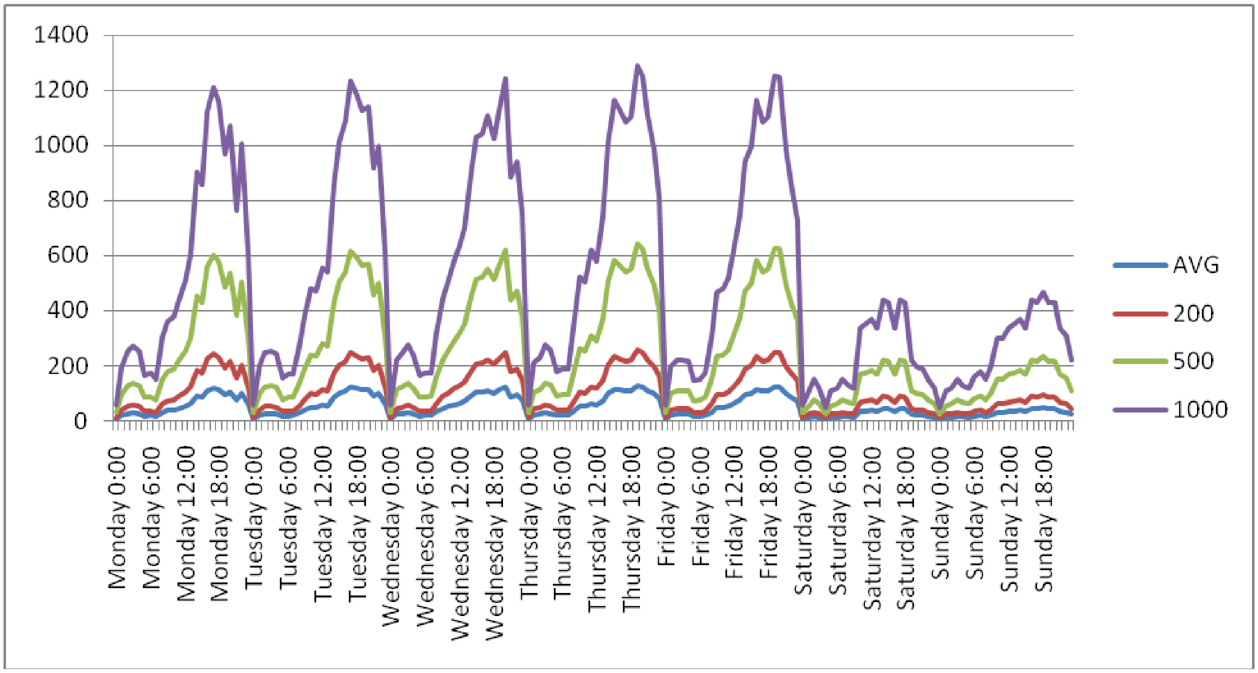
For the first few weeks it will be necessary to simply capture system trend statistics. It is recommended to monitor hourly usage rates and perform comparisons considering the day of the week and time of day. After sufficient statistics’ are available, the automated response can be enabled.

TREND EXAMPLE:

The following policy has been implemented to notify administrators of suspicious activity related to a spike in added friends for a social networking site.

System Trend % Delta	Response Action
+200%	Administration Notification
+500%	Administration Notification
+1000%	Temporarily Disable Add Friend Feature

In order to implement this policy, AppSensor recorded trend data for 1 month and determined the average number of friends added per 1000 users. The results are displayed in the blue line in the chart below. The graph also displays response thresholds for 200%, 500% and 1000% increases in observed activity. Using this data, AppSensor can detect an abnormal level of activity and immediately alert administrators of the suspicious situation. In an extreme cause where use of the friend feature spikes to 1000% of normal activity levels, the policy instructs AppSensor to temporarily disable~~s~~ the add friend feature until the situation can be investigated. By disabling the site feature, a potential worm can be contained and the site may remain operational. Without such automated controls, a worm could bring down the entire social network site before administrators or able to even identify the suspicious activity.



IMPLEMENTATION

AppSensor detection points can be implemented into an application using one of two methods depending upon the type of detection. They can be implemented by using a cross connecting aspect oriented method (ie Java Filters) or by custom code within the business layer of the application. It is recommended to integrate AppSensor using a secure programming approach, such as that provided by ESAPI, for maximum benefit and easy integration in to the program.

See Appendix C: Implementation – Aspect Oriented vs. Business Layer for a recommended implementation of the detection points using Aspect Oriented and Business Layer methods.

ASPECT ORIENTED IMPLEMENTATION

Java filters or similar aspect oriented programming methods can be used to easily integrate the following detection points into the application. The benefit of using an aspect oriented approach is that the addition of AppSensor functionality does not require any modifications to the existing program.

An example of a detection point which can be implemented with aspect oriented programming is CIE1:Blacklist Inspection for Common SQL Injection Values. All GET requests can be inspected by the cross cutting filter for SQL injection keywords such as ‘UNION’, ‘1=1—‘ etc. If a match were found, then CIE1 exception would be thrown.

BUSINESS LAYER IMPLEMENTATION

Some detection points should be implemented within the business layer of the application. It is recommended to use a secure programming methodology, such as that provided by ESAPI, and integrate the detection points with the ESAPI exceptions.

An example of a detection points that should be implemented in the business layer is direct object reference manipulation attempts. In this case the business layer would be designed to create a list of acceptable object references for the user and session. When the user response is received by the application the application will compare the object reference sent by the user to the authorized list. If the item is not present in the authorized list than an exception is thrown and ACE1: Modifying URL Arguments Within A GET For Direct Object Access Attempts is thrown. Note: This particular detection point is easily accomplished using ESAPI’s object referencing code.

CONCLUSION

The threat of attacks against critical business applications continue to rise. AppSensor can provide tremendous value by identifying malicious users and restricting or eliminating application access before a compromise is possible.

The benefits of AppSensor do not come without some effort. For maximum benefits, AppSensor needs to be integrated into the program and, at times, tightly coupled with the application itself. However, after the proper integration is achieved, AppSensor will provide the intended benefits with minimal interaction from developers or system administrators.

For those looking to implement AppSensor into a critical application, it is important to achieve appropriate support from all involved parties (management, architecture, developers, etc). It is recommended to utilize this guide as a blueprint for the design of AppSensor and to make modifications or enhancements as required for the specific application.

REFERENCES

OWASP Development Guide Project: http://www.owasp.org/index.php/Category:OWASP_Guide_Project

OWASP Risk Rating Methodology: http://www.owasp.org/index.php/How_to_value_the_real_risk

OWASP Top Ten Project: http://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project

OWASP Enterprise Security API (ESAPI) Project:
http://www.owasp.org/index.php/Category:OWASP_Enterprise_Security_API

APPENDIX A: DETAILED EVENT DESCRIPTION

SIGNATURE BASED EVENTS DETAILED DESCRIPTION

REQUEST EXCEPTION DETAILED DESCRIPTION

RE1	Unexpected HTTP Commands
RE2	Attempts To Invoke Unsupported HTTP Methods
RE3	GET When Expecting POST
RE4	POST When Expecting GET

RE1	Unexpected HTTP Commands
Exception Type	RequestException
Description	An HTTP request is received which contains unexpected commands. A list of accepted commands should be generated (i.e. GET and POST) and all other HTTP commands should generate an event.
Considerations	
Example(s)	Instead of a GET or POST request, the user sends a TRACE request to the application.

RE2	Attempts To Invoke Unsupported HTTP Methods
Exception Type	RequestException
Description	An http request is received which contains a non-existent HTTP command
Considerations	
Example(s)	Instead of a GET or POST request, the user sends a TEST request to the application (TEST is not a valid http request)

RE3	GET When Expecting POST
Exception Type	RequestException

Description	A page which is expecting only GET requests, receives a POST.
Considerations	
Example(s)	The user sends a GET request to a page which has only been used for POSTs

RE4	**POST When Expecting GET**
Exception Type	RequestException
Description	A page which is expecting only POST requests, receives a GET
Considerations	
Example(s)	The user uses a proxy tool to build a custom POST request and sends it to a page which has been accessed by GET requests.

AUTHENTICATION EXCEPTION DETAILED DESCRIPTION

AE1	Use Of Multiple Usernames
AE2	Multiple Failed Passwords
AE3	High Rate Of Login Attempts
AE4	Unexpected Quantity Of Characters In Username
AE5	Unexpected Quantity Of Characters In Password
AE6	Unexpected Types Of Characters In Username
AE7	Unexpected Types Of Characters In Password
AE8	Providing Only The Username
AE9	Providing Only The Password
AE10	Adding Additional POST Variables
AE11	Removing POST Variables

AE1	**Use Of Multiple Usernames**

Exception Type	AuthenticationException
Description	Multiple usernames are attempted when logging into the application. The assignment of login attempts to a user can be based off of a sessionID given to the user when they visit the website. Correlating based on IP address is difficult since multiple users could be using the site from the same IP address (e.g. corporate NAT)
Considerations	An attacker could bypass this detection by intercepting the post requests and removing the sessionID.
Example(s)	User first tries username bob, then username sue, then steve etc

AE2	**Multiple Failed Passwords**
Exception Type	AuthenticationException
Description	For a single username, multiple bad passwords are entered
Considerations	
Example(s)	User tries username:password combination of user:pass1, user:pass2, user:pass3, etc

AE3	**High Rate Of Login Attempts**
Exception Type	AuthenticationException
Description	The number of logins sent per minute becomes too high indicating an automated login attack
Considerations	An attacker could bypass this detection by intercepting the post requests and removing the sessionID.
Example(s)	User sends the following login attempts within 1 second. user1:pass1, user1:pass2, user2:pass3, user2:pass4

AE4	**Unexpected Quantity Of Characters In Username**
Exception Type	AuthenticationException
Description	The user provides a username with a large number of characters
Considerations	
Example(s)	The user sends a username that is 200 characters long

AE5	Unexpected Quantity Of Characters In Password
Exception Type	AuthenticationException
Description	The user provides a password with a large number of characters
Considerations	
Example(s)	The user sends a password that is 200 characters long

AE6	Unexpected Types Of Characters In Username
Exception Type	AuthenticationException
Description	The user provides non-printable characters such as the null byte. Any characters below hex value 20 or above 7E are considered illegal (decimal values of below 32 or above 126)
Considerations	The range will need to be adjusted for international characters.
Example(s)	The user sends a username that contains ascii characters below 20 or above 7E

AE7	Unexpected Types Of Characters In Password
Exception Type	AuthenticationException
Description	The user provides characters such as the null byte, alt-characters, (WHAT IS THE NAME FOR THOSE)
Considerations	The range will need to be adjusted for international characters.
Example(s)	The user sends a password that contains ascii characters below 20 or above 7E

AE8	Providing Only The Username
Exception Type	AuthenticationException
Description	The user submits a post request which only contains the username variable. The password variable has been removed. This is different from only providing the username in the login form since in that case the password variable would be present and empty.
Considerations	
Example(s)	The user uses a proxy tool to remove the password variable from the submitted post

	request.

AE9	Providing Only The Password
Exception Type	AuthenticationException
Description	The user submits a post request which only contains the password variable. The username variable has been removed. This is different from only providing the password in the login form since in that case the username variable would be present and empty.
Considerations	
Example(s)	The user uses a proxy tool to remove the username variable from the submitted post request.

AE10	Adding Additional POST Variables
Exception Type	AuthenticationException
Description	Additional, unexpected post variables are received during an authentication request.
Considerations	
Example(s)	The user uses a proxy tool to add the additional post variable of admin=true to the post request

AE11	Removing POST Variables
Exception Type	AuthenticationException
Description	Expected post variables are not present within the submitted authentication requests
Considerations	
Example(s)	The user uses a proxy tool to remove an additional post variable, such as guest=true, from the post request

SESSION EXCEPTION DETAILED DESCRIPTION

SE1	Modifying Existing Cookies
SE2	Adding New Cookies

SE3	Deleting Existing Cookies
SE4	Substituting Another User's Valid Session ID Or Cookie
SE5	Source IP Address Changes During Session
SE6	Change Of User Agent Mid Session

SE1	**Modifying Existing Cookies**
Exception Type	SessionException
Description	A request is received containing a cookie with a modified value. This could be determined if the cookie is modified to an illegal value.
Considerations	
Example(s)	The user uses a proxy tool to change the encrypted cookie to an alternative value which does not properly decode within the application. Or, the user modifies an unencrypted cookie and sets an illegal value for a particular variable.

SE2	**Adding New Cookies**
Exception Type	SessionException
Description	A request is received which contains additional cookies that are not expected by the application.
Considerations	
Example(s)	The user uses a proxy tool to add additional cookies to the request.

SE3	**Deleting Existing Cookies**
Exception Type	SessionException
Description	A request is received which does not contain the expected cookies.
Considerations	
Example(s)	The user uses a proxy tool to remove cookies or portions of cookies from a request.

SE4	**Substituting Another User's Valid Session ID Or Cookie**

Exception Type	SessionException
Description	A request is received which contains cookie data that is clearly from another user or another session.
Considerations	This may only be possible to detect in unique situations where the cookie value is clearly not valid for this user.
Example(s)	The user uses a proxy tool to substitute valid data from another user or session into the cookie. An example would be changing some sort of identification number within the cookie.

SE5	**Source IP Address Changes During Session**
Exception Type	SessionException
Description	Valid requests, containing valid session credentials, are received from multiple source IP addresses.
Considerations	Detection of a different IP address may be difficult since some network providers change source IP address between requests. However, it may be safe to flag events if the IP address changes to one which is located in a different country than the previous request.
Example(s)	User A's session is compromised and User B begins using the account. The requests originating from User B will possibly contain a different source IP address the User A. The source IP addresses could be the same if both users where behind the same NAT.

SE6	**Change Of User Agent Mid Session**
Exception Type	SessionException
Description	The User-Agent value of the header changes during an authenticated session. This indicates a different browser is now being used. Although this value is under the control of the sender, a change in this may indicates that the session has been compromised and is being used another individual. This will likely not be the case that the user has simply copied and pasted the URL from one browser to another on the same system because this action would not copy over the appropriate session identifiers.
Considerations	This detection point may inhibit the ability for an authorized penetration test. However, that is a good thing in all other situations.
Example(s)	Midsession, the User-Agent changes from Firefox to Internet Explorer Mozilla/5.0 (Windows; U; Windows NT 6.0; en-US; rv:1.8.1.14) Gecko/20080404 Firefox/2.0.0.14

	to Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.0; SLCC1; .NET CLR 2.0.50727; Media Center PC 5.0; .NET CLR 3.0.04506; InfoPath.2)

ACCESS CONTROL DETAILED DESCRIPTION

ACE1	Modifying URL Arguments Within A GET For Direct Object Access Attempts
ACE2	Modifying Parameters Within A POST For Direct Object Access Attempts
ACE3	Force Browsing Attempts
ACE4	Evading Presentation Access Control Through Custom Posts

ACE1	Modifying URL Arguments Within a GET For Direct Object Access Attempts
Exception Type	AccessControlException
Description	The application is designed to use an identifier for a particular object, such as using categoryID=4 or user=guest within the URL. A user modifies this value in an attempt to access unauthorized information. This exception should be thrown anytime the identifier received from the user is not authorized due to the identifier being nonexistent or the identifier not authorized for that user.
Considerations	
Example(s)	The user modifies the following URL from site.com/viewpage?page=1&user=guest to site.com/viewpage?page=22&user=admin

ACE2	Modifying Parameters Within A POST For Direct Object Access Attempts
Exception Type	AccessControlException
Description	The value of a non-free text html form element (i.e. drop down box, radio button) is modified to an illegal value. The value either does not exist or is not authorized for the user.
Considerations	
Example(s)	The user uses a proxy tool to intercept a post request and changes the posted value to a value that was not available through the normal display. For example, the user encounters

	a dropdown box containing the numbers 1 through 10. The user selects 5 and then intercepts the post to change the submitted value to 100.

ACE3	Force Browsing Attempts
Exception Type	AccessControlException
Description	An authenticated user sends a request for a non-existent page or a page that is not authorized for the user.
Considerations	
Example(s)	The user is authenticated and requests site.com/PageThatDoesNotExist

ACE4	Evading Presentation Access Control Through Custom Posts
Exception Type	AccessControlException
Description	A post request is received which is not authorized for the current user and the user could not have performed this action without crafting a custom POST request. This situation is most likely to occur when presentation layer access controls are in place and have removed the user's ability to initiate the action through the presentation of the application. An attacker may be aware of the functionality and attempt to bypass this presentation layer access control by crafting their own custom message and sending this in an attempt to execute the functionality.
Considerations	Detecting this event requires the application to be aware of which controls/functionality or hidden at the presentation layer due to access controls. If an access control violation occurs for any of these items, then this event should be fired.
Example(s)	The application contains the ability for an administrator to delete a user. This method is normally invoked by entering the username and posting to https://oursite/deleteuser Presentation layer access controls ensure the delete user form is not displayed to non-administrator users. A malicious user has access to a non-administrator account and is aware of the delete user functionality. The malicious user sends a custom crafted post message to https:/ oursite/deleteuser in an attempt to execute the delete user method.

INPUT EXCPTION DETAILED DESCRIPTION

IE1	Cross Site Scripting Attempt
IE2	Violations Of Implemented White Lists

IE1	Cross Site Scripting Attempt
Exception Type	InputException
Description	The HTTP request contains common XSS attacks which are often used by attackers probing for XSS vulnerabilities. Detection should be configured to test all GET and POST values as well as all header names and values for the following values.
Considerations	
Example(s)	The user uses a proxy tool to add an XSS attack to the header value and the ""displayname"" post variable. The header value could be displayed to an admin viewing log files and the ""displayname"" post variable may be stored in the application and displayed to other users. <script>alert(document.cookie);</script> <script>alert(xss);</script> <script>alert(test);</script> <script>alert(hi);</script> alert(String.fromCharCode(88,83,83)) <IMG SRC="javascript:alert('XSS');"> <IMG SRC=javascript:alert('XSS')> <IMG SRC=javascript:alert(&#38;quot;XSS&#38;quot;)> <BODY ONLOAD=alert('XSS')>

IE2	Violations Of Implemented White Lists

Exception Type	InputException
Description	The application receives user-supplied data that violates an established white list validation.
Considerations	
Example(s)	The user submits data that is not correct for the particular field. This may not be attack data necessarily, but repeated violations could be an attempt by the attacker to determine how an application works or to discover a flaw.

ENCODING EXCEPTION DETAILED DESCRIPTION

EE1	Double Encoded Characters
EE2	Unexpected Encoding Used

EE1	**Double Encoded Characters**
Exception Type	EncodingException
Description	An HTTP request is received which contains values that have been double encoded.
Considerations	
Example(s)	The user sends encodes the % symbol to %25 and appends 3C. The user is sending %253C which may be interpreted by the application as %3C which is actually <.

EE2	**Unexpected Encoding Used**
Exception Type	EncodingException
Description	An HTTP request is received which contains values that have encoded in an unexpected format.
Considerations	
Example(s)	The user encodes an attack such as alert(document.cookie) into the UTF-7 format and sends this data the application. This could bypass validation filters and be rendered to a user in certain situations.

COMMAND INJECTION DETAILED DESCRIPTION

CIE1	Blacklist Inspection For Common SQL Injection Values
CIE2	Detect Abnormal Quantity Of Returned Records.
CIE3	Null Byte Character In File Request
CIE4	Carriage Return Or Line Feed Character In File Request

CIE1	Blacklist Inspection For Common SQL Injection Values
Exception Type	CommandInjectionException
Description	A request is received which contains common SQL injection attack attempts. The point of this detection is not to detect all variations of a SQL injection attack, but to detect the common probes which an attacker or tool might use to determine if a SQL injection vulnerability is present. Add regular expressions here too?
Considerations	Unless the site contains some sort of message board for discussing SQL injection, there is little reason that the SQL injection examples should ever be received from a user request. Use caution when adding SQL statements such as UNION or JOIN. These may create false positives depending on valid information passed through parameters which may contain these words.
Example(s)	The user sends a request and modifies a URL parameter from category=5 to category =5' OR '1'='1 in an attempt to perform an SQL injection attack. The user could perform similar attacks by modifying post variables or even the request headers to contain SQL injection attacks. • ' OR '1'='1 • ' OR 'a'='a • ' OR 1=1— • xp_cmdshell • UNION • JOIN

CIE2	Detect Abnormal Quantity Of Returned Records

Exception Type	CommandInjectionException
Description	A database query is executed which returns more records than expected. For example, if the query should only return 1 record and 100 records are returned, then something has likely gone wrong.
Considerations	Detection of this event will not likely indicate what malicious action the attacker has taken, but it will indicate that the attacker has successfully bypasses the intended actions of the application and accessed more data than intended.
Example(s)	The application is designed to allow a user to maintain 5 profiles. A user makes a request to view all of their profiles. The database query, which is expected to always return 5 or less results, returns 10000 records. Something in the application, or user's actions, has caused unauthorized data to be returned.

CIE3	**Null Byte Character In File Request**
Exception Type	CommandInjectionException
Description	A request is received to download a file from the server. The filename requested contains the null byte the file name. This is an attempted OS injection attack.
Considerations	
Example(s)	The user modifies the filename of the requested file to download to contain the null byte. The null byte can be added by inserting the hex value %00.

CIE4	**Carriage Return Or Line Feed Character In File Request**
Exception Type	CommandInjectionException
Description	A request is received which contains the carriage return or line feed characters within the posted data or the URL parameters. This is an attempted HTTP split response attack.
Considerations	
Example(s)	The user includes the hex value %0D or %0A in the http request post data or URL parameters.

FILE IO DETAILED DESCRIPTION

FIO1	Detect Large Individual Files
FIO2	Detect Large Number Of File Uploads

FIO1	Detect Large Individual Files
Exception Type	FileIOException
Description	A file upload feature detects that a large file has been submitted for upload which exceeds the maximum upload size
Considerations	
Example(s)	The user attempts to upload a large file to occupy resources or fill up disk space

FIO2	Detect Large Number Of File Uploads
Exception Type	FileIOException
Description	A user uploads an excessively large number of files.
Considerations	
Example(s)	A single user attempts to upload multiple small files to occupy resources or fill up disk space

BEHAVIOR BASED EVENTS DETAILED DESCRIPTION

USER TREND DETAILED DESCRIPTION

UT1	Irregular Use Of Application
UT2	Speed Of Application Use
UT3	Frequency Of Site Use
UT4	Frequency Of Feature Use

UT1	Irregular Use Of Application
Exception Type	UserTrendException
Description	The application receives numerous requests for the same page or feature from a user. The user may be sending different data combinations or trying to detect errors in the page.
Considerations	This detection point is appropriate to add to pages which perform operations or respond to user supplied data versus pages which display static information.
Example(s)	The user requests a particular page, such as the address update page, numerous times.

UT2	Speed Of Application Use
Exception Type	UserTrendException
Description	The speed of requests from a user indicates that an automated tool is being used to access the site. The use of a tool may indicate reconnaissance for an attack or attempts to identify vulnerabilities in the site.
Considerations	Search spiders may request large quantities of pages from the unauthenticated portion of the website. However, any scripted tool requesting large quantities of pages within the authenticated portion of the site would be suspicious.
Example(s)	The user utilizes an automated tool to request hundreds of pages per minute.

UT3	Frequency Of Site Use

Exception Type	UserTrendException
Description	Does the user normally access the site 1 per week, and this is now many times per day
Considerations	
Example(s)	

UT4	**Frequency Of Feature Use**
Exception Type	UserTrendException
Description	The rate of a user utilizing a particular application feature changes dramatically.
Considerations	
Example(s)	

SYSTEM TREND DETAILED DESCRIPTION

STE1	High Number Of Logouts Across The Site
STE2	High Number Of Logins Across The Site
STE3	High Number Of Same Transaction Across The Site

STE1	**High Number Of Logouts Across The Site**
Exception Type	SystemTrendException
Description	A sudden spike in logouts across the application could indicate a XSS and CSRF attack placed within the application which is automatically logging off users.
Considerations	This requires the application to maintain normal usage levels of the application feature usage.
Example(s)	The hourly usage of the logoff feature of the application suddenly spikes by 500%.

STE2	**High Number Of Logins Across The Site**

Exception Type	SystemTrendException
Description	A sudden spike in logins across the application could indicate users being redirected to the site from a phishing email looking to exploit a XSS vulnerability in the site.
Considerations	This requires the application to maintain normal usage levels of the application feature usage.
Example(s)	The hourly usage of the logon feature of the application suddenly spikes by 500%.

STE3	**High Number Of Same Transaction Across The Site**
Exception Type	SystemTrendException
Description	A sudden spike in similar activity across numerous users of the application may indicate a phishing attack or CSRF attack against the users.
Considerations	This requires the application to maintain normal usage levels of the application feature usage.
Example(s)	The hourly usage of the update email address feature of the application suddenly spikes by 500%.

APPENDIX B: RESPONSE – SUSPECT VS. ATTACK EVENTS

	ID	Title	Exception	Suspect	Attack
Request	**RE1**	Unexpected HTTP Commands	RequestException		x
	RE2	Attempts To Invoke Unsupported HTTP Methods	RequestException		x
	RE3	GET When Expecting POST	RequestException	x	
	RE4	POST When Expecting GET	RequestException		x
Authentication	**AE1**	Use Of Multiple Usernames	AuthenticationException	x	
	AE2	Multiple Failed Passwords	AuthenticationException		x
	AE3	High Rate Of Login Attempts	AuthenticationException		x
	AE4	Unexpected Quantity Of Characters In Username	AuthenticationException		x
	AE5	Unexpected Quantity Of Characters In Password	AuthenticationException		x
	AE6	Unexpected Types Of Characters In Username	AuthenticationException		x
	AE7	Unexpected Types Of Characters In Password	AuthenticationException	x	
	AE8	Providing Only The Username	AuthenticationException		
	AE9	Providing Only The Password	AuthenticationException		x
	AE10	Adding Additional POST Variables	AuthenticationException		x
	AE11	Removing POST Variables	AuthenticationException		x
Session	SE1	Modifying Existing Cookies	SessionException		x
	SE2	Adding New Cookies	SessionException		x
	SE3	Deleting Existing Cookies	SessionException	x	
	SE4	Substituting Another User's Valid Session ID Or Cookie	SessionException		x

	SE5	Source IP Address Changes During Session	SessionException	**x**	
	SE6	Change Of User Agent Mid Session	SessionException		**x**
Access Control	**ACE1**	Modifying URL Arguments Within A GET For Direct Object Access Attempts	AccessControlException	**x**	
	ACE2	Modifying Parameters Within A POST For Direct Object Access Attempts	AccessControlException		**x**
	ACE3	Force Browsing Attempts	AccessControlException	**x**	
	ACE4	Evading Presentation Access Control Through Custom Posts	AccessControlException		**x**
Input	**IE1**	Cross Site Scripting Attempt	InputException	**x**	
	IE2	Violations Of Implemented White Lists	InputException	**x**	
Encoding	**EE1**	Double Encoded Characters	EncodingException	**x**	
	EE2	Unexpected Encoding Used	EncodingException		**x**
Command Injection	**CIE1**	Blacklist Inspection For Common SQL Injection Values	CommandInjectionException		**x**
	CIE2	Detect Abnormal Quantity Of Returned Records.	CommandInjectionException		**x**
	CIE3	Null Byte Character In File Request	CommandInjectionException		**x**
	CIE4	Carriage Return Or Line Feed Character In File Request	CommandInjectionException		**x**
File IO	**FIO1**	Detect Large Individual Files	FileIOException	**x**	
	FIO2	Detect Large Number Of File Uploads	FileIOException		**x**
User Trend	**UT1**	Irregular Use Of Application	UserTrendException	**x**	
	UT2	Speed Of Application Use	UserTrendException	**x**	
	UT3	Frequency Of Site Use	UserTrendException	**x**	

	UT4	Frequency Of Feature Use	UserTrendException	**x**	
System Trend	**STE1**	High Number Of Logouts Across The Site	SystemTrendException	**x**	
	STE2	High Number Of Logins Across The Site	SystemTrendException	**x**	
	STE3	High Number Of Same Transaction Across The Site	SystemTrendException	**x**	

APPENDIX C: IMPLEMENTATION – ASPECT ORIENTED VS. BUSINESS LAYER

	ID	Title	Exception	Aspect Oriented	Business Layer
Request	**RE1**	Unexpected HTTP Commands	RequestException	x	
	RE2	Attempts To Invoke Unsupported HTTP Methods	RequestException	x	
	RE3	GET When Expecting POST	RequestException		x
	RE4	POST When Expecting GET	RequestException		x
Authentication	**AE1**	Use Of Multiple Usernames	AuthenticationException		x
	AE2	Multiple Failed Passwords	AuthenticationException		x
	AE3	High Rate Of Login Attempts	AuthenticationException		x
	AE4	Unexpected Quantity Of Characters In Username	AuthenticationException	x	
	AE5	Unexpected Quantity Of Characters In Password	AuthenticationException	x	
	AE6	Unexpected Types Of Characters In Username	AuthenticationException	x	
	AE7	Unexpected Types Of Characters In Password	AuthenticationException	x	
	AE8	Providing Only The Username	AuthenticationException	x	
	AE9	Providing Only The Password	AuthenticationException	x	
	AE10	Adding Additional POST Variables	AuthenticationException	x	
	AE11	Removing POST Variables	AuthenticationException	x	
Session	SE1	Modifying Existing Cookies	SessionException	x	
	SE2	Adding New Cookies	SessionException	x	
	SE3	Deleting Existing Cookies	SessionException	x	

	SE4	Substituting Another User's Valid Session ID Or Cookie	SessionException	**x**	
	SE5	Source IP Address Changes During Session	SessionException		**x**
	SE6	Change Of User Agent Mid Session	SessionException		**x**
Access Control	**ACE1**	Modifying URL Arguments Within A GET For Direct Object Access Attempts	AccessControlException		**x**
	ACE2	Modifying Parameters Within A POST For Direct Object Access Attempts	AccessControlException		**x**
	ACE3	Force Browsing Attempts	AccessControlException		**x**
	ACE4	Evading Presentation Access Control Through Custom Posts	AccessControlException		**x**
Input	**IE1**	Cross Site Scripting Attempt	InputException		**x**
	IE2	Violations Of Implemented White Lists	InputException		**x**
Encoding	**EE1**	Double Encoded Characters	EncodingException		**x**
	EE2	Unexpected Encoding Used	EncodingException		**x**
Command Injection	**CIE1**	Blacklist Inspection For Common SQL Injection Values	CommandInjectionException	**x**	
	CIE2	Detect Abnormal Quantity Of Returned Records.	CommandInjectionException		**x**
	CIE3	Null Byte Character In File Request	CommandInjectionException		**x**
	CIE4	Carriage Return Or Line Feed Character In File Request	CommandInjectionException		**x**
File IO	**FIO1**	Detect Large Individual Files	FileIOException		**x**
	FIO2	Detect Large Number Of File Uploads	FileIOException		**x**
User Tren d	**UT1**	Irregular Use Of Application	UserTrendException		**x**

	UT2	Speed Of Application Use	UserTrendException		**x**
	UT3	Frequency Of Site Use	UserTrendException		**x**
	UT4	Frequency Of Feature Use	UserTrendException		**x**
System Trend	**STE1**	High Number Of Logouts Across The Site	SystemTrendException		**x**
	STE2	High Number Of Logins Across The Site	SystemTrendException		**x**
	STE3	High Number Of Same Transaction Across The Site	SystemTrendException		**x**