

Checksum
Src IP Address
Dest IP Address

Addressing Software Risks Throughout the Supply Chain
SwA Forum Workshops Track 1 - SwA at the Code Level

Planning the Implementation of Attack-Aware Software with Active Defenses

Using Materials Based on the OWASP AppSensor Project*

September 2011 SwA Forum, SEI, Arlington, VA, 14th September 2011

Colin Watson, Watson Hall Ltd

* The OWASP Foundation does not support, advocate, or recommend any particular vendor, product or technology

SwA at the Code Level

As the entire computer security industry is fully and painfully aware, applications are the #1 target for malicious attack. The root of this problem is vulnerable software -- trillions of lines worth of code and counting. On an internet-wide scale, how do we go about writing more secure code? How do we deal with the massive backlog of vulnerable code already in wide circulation? What are the best strategies for ensuring code remains secure as threats evolve?

One Issue

- Skilled and motivated attackers

Two Questions

1) Is the application being attacked now?

Yes No Don't know

2) Have any unknown vulnerabilities been exploited today?

Yes No Don't know

Three Test Cases

1) Stepping through a process in the incorrect order

Step five,
then step two

/step5/
/step2/

2) Requesting an unauthorized resource identifier

Show my account,
then show me someone else's

/updateProfile?id=1005
/updateProfile?id=1006

3) Payment transfer exceeding limit

Send 27 pounds,
then send rather more

/transfer?amount=27.00
/transfer?amount=270000

How Do We Protect Applications?

- Preliminary requirements
 - Risk assessment
 - Secure coding
 - Application logging
 - Hardened environment
- And then?

Conventional Defenses

- Transport layer security
- Firewall (stateful or deep packet inspection)
- Web application firewall
- Application aware firewall (“next generation”)
- Intrusion detection and prevention
 - Network
 - Host

Transport Layer Security

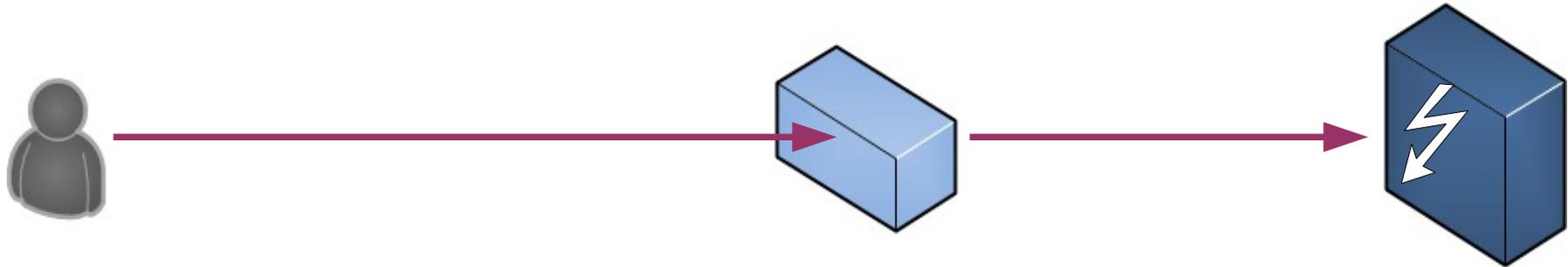


3) Payment transfer exceeding limit
Send 27 pounds,
then send rather more

`/transfer?amount=27.00`
`/transfer?amount=270000`

Protected Unprotected

Firewall



3) Payment transfer exceeding limit
Send 27 pounds,
then send rather more

`/transfer?amount=27.00`
`/transfer?amount=270000`

Protected

Unprotected

Web Application Firewall



2) Requesting an unauthorized resource identifier

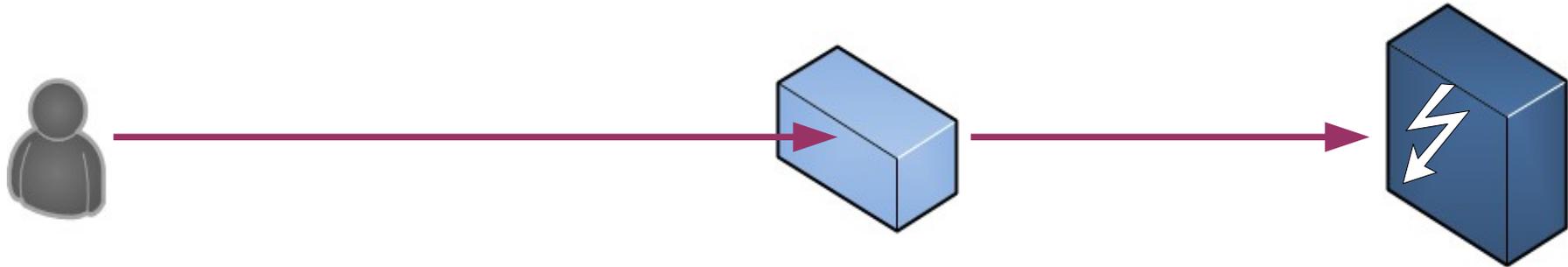
Show my account,
then show me someone else's

`/updateProfile?id=1005`
`/updateProfile?id=1006`

Protected

Unprotected

Application Aware Firewall



- 1) Stepping through a process in the incorrect order
Step five, `/step5/`
then step two `/step2/`

Protected Unprotected

Detecting Attacks the Right Way

- Integrated
 - Understands the application
 - Understands normal vs. malicious use
 - Updated when the business process changes
- Effective
 - Minimal false positives
 - Immediate response
- Scalable
 - Automatic detection
 - Real time

Inside The Application

- Applications have:
 - Full knowledge of the business logic
 - An understanding of the roles & permissions of users
 - Knowledge of malicious vs. normal use
 - Access to user and system history and trends
 - Information to instantly detect attackers
 - The ability to respond automatically in real-time such as taking a more defensive posture

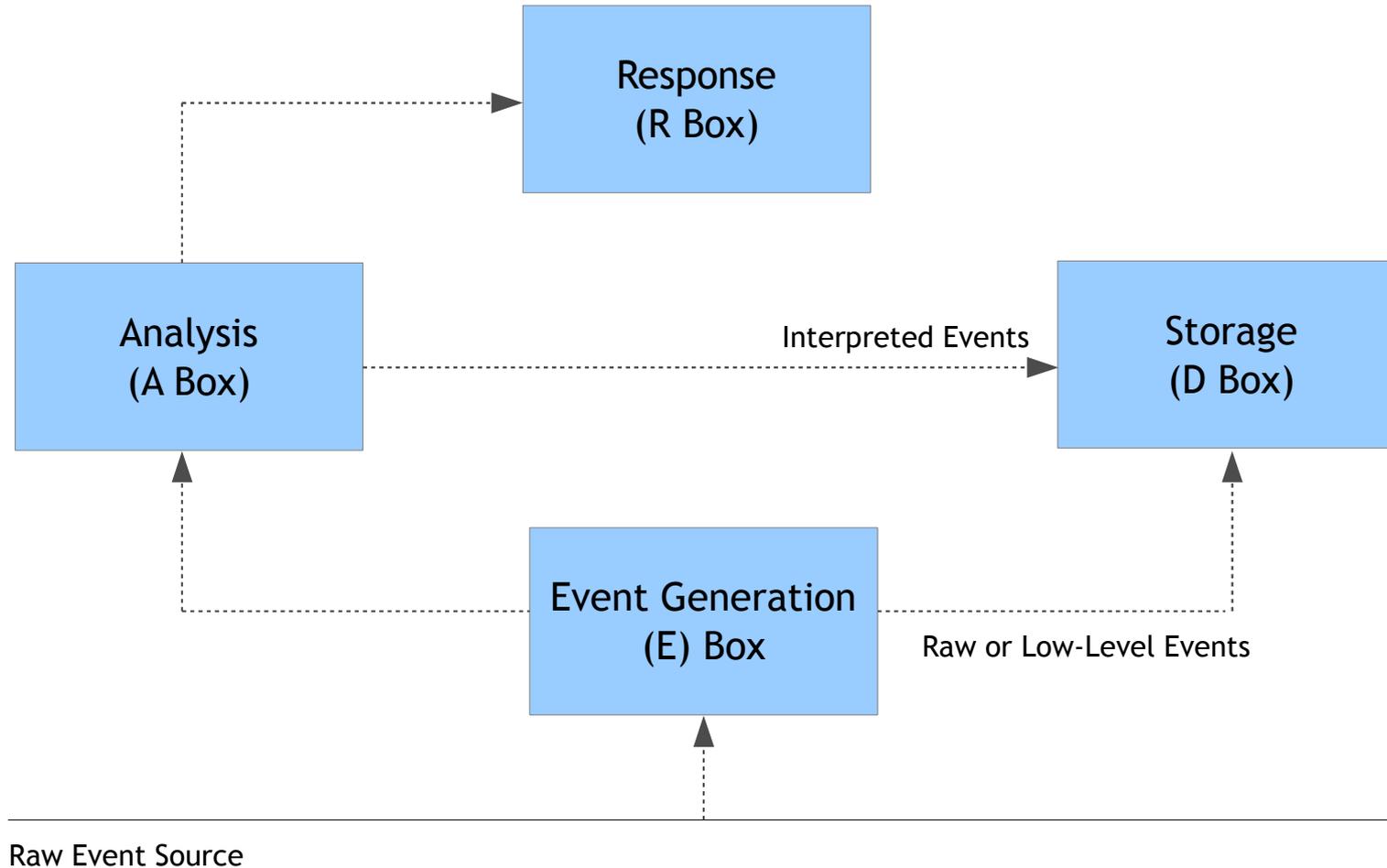
Existing Application Countermeasure Examples

- Non-critical functions disabled by a car engine management system when intrusion detected
- Blocking access to an airplane's avionics control system when the source is identified as coming from the passenger network
- Tamper detection erases encryption keys
- Raising an alert when the external time is detected to be different to an internal time reference
- Logging of system power outages
- Application disabled by an operator due to unusual conditions
- Access blocked when single sign on message fails integrity check
- Application logging
- Disable non-core function

Further Examples

- Terminating a request when blacklisted inputs are received
- Fraud detection
- Adding time delays to each successive failed authentication attempt
- Locking a user account after a number of failed authentication attempts
- Application honey pot functionality
- Logging a user out if the browser's “back” button is used
- Terminating a session if a user's geo-location changes
- Blocking access by certain IP addresses when malicious behavior is detected
- Recording unexpected actions
- Blocking certain HTTP verbs

Common Intrusion Detection Framework Model



Common Intrusion Detection Framework Architecture, Porras, Schnackenberg, Staniford-Chen, Davis, Stillman and Wu, 1999
Insertion, Evasion and Denial Of Service: Eluding Network Intrusion Detection, Ptacek and Newsham, 1998

IDS Raw Event Sources

- Network
- Host
 - Kernel
 - Network
 - File system
 - Log files
- Application

Attack-Aware with Active Defenses

1. Event detection
2. Analysis
3. Attack determination
4. Response selection
5. Response execution

Primary Objective

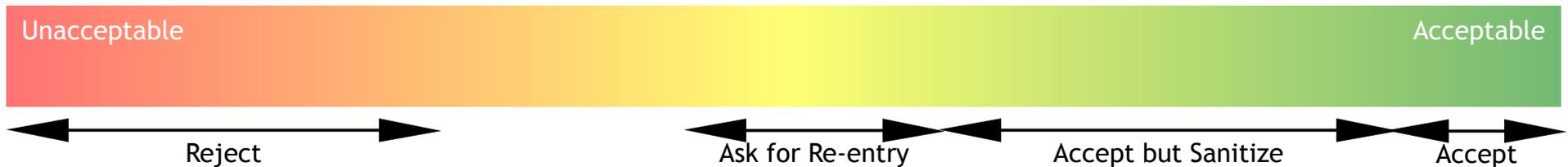
- Maintain an extremely low false-positive attack detection rate
- Must not identify normal behavior as
 - an attack
 - suspicious

Detecting Malicious Users

- “Users” are not perfect



- Application-specific actions



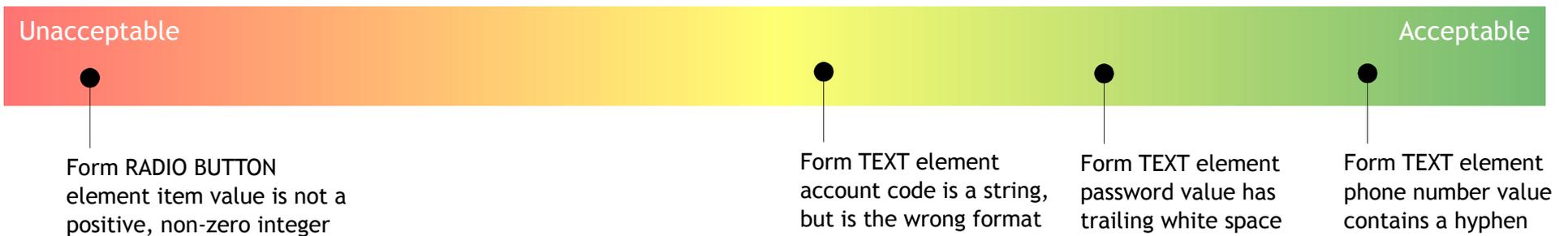
Good Detection Points

- Clearly malicious behavior
- Activities that would require use of a tool
- Bypass of presentation controls

- Some examples
 - POST instead of GET (HTTP method)
 - Altered HIDDEN form parameter values
 - Change of User Agent string mid-session

Importance of Context

- Server-side validation only



- Server-side with duplicate client-side validation



Poor Detection Points

- Anything which is normal behavior
- Anything that could occur by accident

- Examples
 - GET instead of POST (HTTP method)
 - Invalid characters in a TEXTAREA form parameter value
 - An apostrophe/single tick in a TEXT form parameter value
 - Loss of external connectivity
 - Search engine activity

Effect of Poor Detection Points

- Increased number of events identified
- Harder to spot real attacks
- More false-positives
- Possibility of disrupting normal behavior
- Loss of confidence
- Business impact

Indeterminate Detection Points

- Anything which could be normal behavior but is suspicious
- Unusual events and something to take note of
- Examples
 - Duplicate POST request with same parameter values
 - Faster use of the application

Context Dependent

- Good, indeterminate and poor are completely application-specific
- Good detection points for one application may be poor for another
- That's why it is “application-specific”

Step 1

- Detection point selection and specification

AppSensor Detection Points

- A detection point is a highly-tuned “instrumentation” sensor (within the application’s code) and used to identify a particular type of suspicious or malicious activity
- Over 50 detection points
- 12 exception types
 - 9 signature based (request, authentication, session, etc)
 - 3 behavior based (user, system, reputation)
- Latest list of detection points with descriptions, considerations and examples is maintained at:

https://www.owasp.org/index.php/AppSensor_DetectionPoints

Detection Point Types

- Signature based
 - Request
 - Authentication
 - Session
 - Access control
 - Input
 - Encoding
 - Command injection
 - File input/output
 - Honey trap
- Behavior based
 - User trend
 - System trend
 - Reputation

Signature

Request Exceptions (RE)

- RE1: Unexpected HTTP Command
- RE2: Attempt to Invoke Unsupported HTTP Method
- RE3: GET When Expecting POST
- RE4: POST When Expecting GET
- RE5: Additional/Duplicated Data in Request
- RE6: Data Missing from Request
- RE7: Unexpected Quantity of Characters in Parameter
- RE8: Unexpected Type of Characters in Parameter

Signature

Authentication Exceptions (AE)

- 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 Type of Character in Username
- AE7: Unexpected Type of Character in Password
- AE8: Providing Only the Username
- AE9: Providing Only the Password
- AE10: Additional POST Variable
- AE11: Missing POST Variable
- AE12: Utilization of Common Usernames

Signature

Session Exceptions (SE)

- SE1: Modifying Existing Cookie
- SE2: Adding New Cookie
- SE3: Deleting Existing Cookie
- SE4: Substituting Another User's Valid Session ID or Cookie
- SE5: Source Location Changes During Session
- SE6: Change of User Agent Mid Session

Signature

Access Control Exception (ACE)

- ACE1: Modifying URL Argument Within a GET for Direct Object Access Attempt
- ACE2: Modifying Parameter Within A POST for Direct Object Access Attempt
- ACE3: Force Browsing Attempt
- ACE4: Evading Presentation Access Control Through Custom POST

Signature

Input Exception (IE)

- IE1: Cross Site Scripting Attempt
- IE2: Violation Of Implemented White Lists
- IE3: Violation Of Implemented Black Lists
- IE4: Violation of Input Data Integrity
- IE5: Violation of Stored Business Data Integrity
- IE6: Violation of Security Log Integrity

Signature

Encoding Exception

- EE1: Double Encoded Character
- EE2: Unexpected Encoding Used

Signature

Command Injection Exception (CIE)

- 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

Signature

File IO Exception (FIO)

- FIO1: Detect Large Individual File
- FIO2: Detect Large Number of File Uploads

Signature

Honey Trap (HT)

- HT1: Alteration to Honey Trap Data
- HT2: Honey Trap Resource Requested
- HT3: Honey Trap Data Used

Behavior

User Trend Exception (UTE)

- UT1: Irregular Use of Application
- UT2: Speed of Application Use
- UT3: Frequency of Site Use
- UT4: Frequency of Feature Use

Behavior

System Trend Exception (STE)

- STE1: High Number of Logouts Across The Site
- STE2: High Number of Logins Across The Site
- STE3: Significant Change in Usage of Same Transaction Across The Site

Behavior

Reputation (RP)

- RP1: Suspicious or Disallowed User Source Location
- RP2: Suspicious External User Behavior
- RP3: Suspicious Client-Side Behavior
- RP4: Change to Environment Threat Level

Related Detection Points

- Some named points are just more specific versions of others
- RE6: Data Missing From Request
 - AE11: Missing POST Variable
 - AE8: Providing Only the Username
 - AE9: Providing Only the Password
- RE8: Unexpected Type of Characters in Parameter
 - IE3: Violation of Implemented Black Lists
 - IE1: Cross Site Scripting Attempt

Chart of Related Detection Points



Custom

- Many of the best detection points will not be ones listed previously, but are highly customized to the individual application.
- *Example 1:* Request received by a industrial control device from a public, or incorrect private network, address range.
- *Example 2:* Multi-step business process undertaken in the wrong sequence.
- *Example 3:* A mismatch between a unique token allocated to an individual authenticated user at logon, and the one received back as part of a URL path.
- *Example 4: Your application*

Detection Point Categorizations

- Signature vs. Behavior based
- Users monitored
 - All Users
 - Single user
- Event type
 - Suspicious
 - Attack
- Class
 - Discrete
 - Application-scope
 - Local-scope
 - Aggregating
 - Modifying
- Stage
 - Request properties
 - Outcome/result

Users Monitored

- All Users
 - All three System Trend Exception detection points
 - Reputation RP4 (Change to Environment Threat Level)
- Single User
 - Every other detection point

Event Type

- Suspect
 - Suspicious behavior
 - Could possibly result from a typo or inadvertent key press
 - *Examples:* RE3 (GET when expecting POST), RE5 (Additional/Duplicated Data in Request) and RE6 (Data Missing From Request)
- Attack
 - Clear malicious activity
 - Not possible in normal flow of application use
 - Additional tools or software needed
 - *Examples:* RE1, RE2 and RE4 (POST when expecting GET)

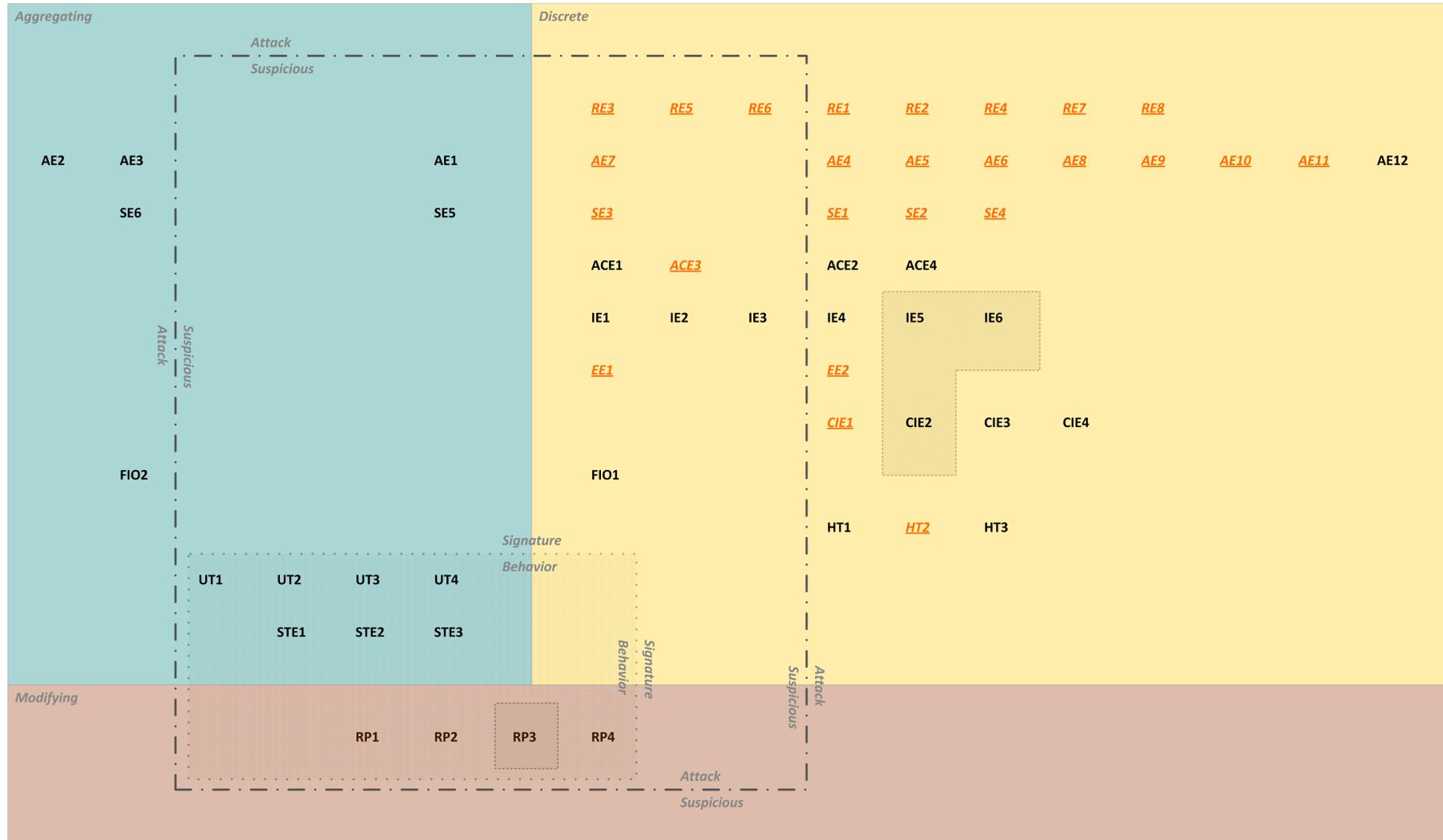
Class

- Discrete
 - Events that do not need any prior knowledge of the user's behavior
 - Activity of a single request/response
- Aggregating
 - Requiring a number of prior identical events to occur before being activated
 - Activities over the duration of a single or multiple sessions
- Modifying
 - Typically normally only used to alter attack detection thresholds or response actions

Scope

- Application-scoped (also called generic filters, generic pre-processing, or aspect-orientated)
- Locally-scoped (also called business layer)

Chart of Detection Point Categorizations



Approaches to Detection Point Selection

- Class categorization
 - Discrete
 - Application-scoped
 - Locally-scoped
 - Aggregating
 - Modifying
- Application risk classification
- Threat assessment
- Standards?

Application Risk Classification Approach

CATEGORIZATION		APPLICATION RISK CLASSIFICATION			
SOURCE	TYPE/CLASS	LOW	MEDIUM	HIGH	CRITICAL
See 2a) on page 14	Inputs	Applicable	Applicable	Applicable	Applicable
	Outcomes	Not Applicable	Possible	Possible	Applicable
Table A-1	Signature	Applicable	Applicable	Applicable	Applicable
	Behavior	Not Applicable	Possible	Applicable	Applicable
Table A-2	Suspicious	Not Applicable	Possible	Applicable	Applicable
	Attack	Possible	Applicable	Applicable	Applicable
Table A-3	Discrete	Applicable	Applicable	Applicable	Applicable
	Aggregating	Not Applicable	Possible	Applicable	Applicable
	Modifying	Not Applicable	Not Applicable	Possible	Possible

KEY

Applicable	Applicable
Possible	Possible
Not Applicable	Not Applicable

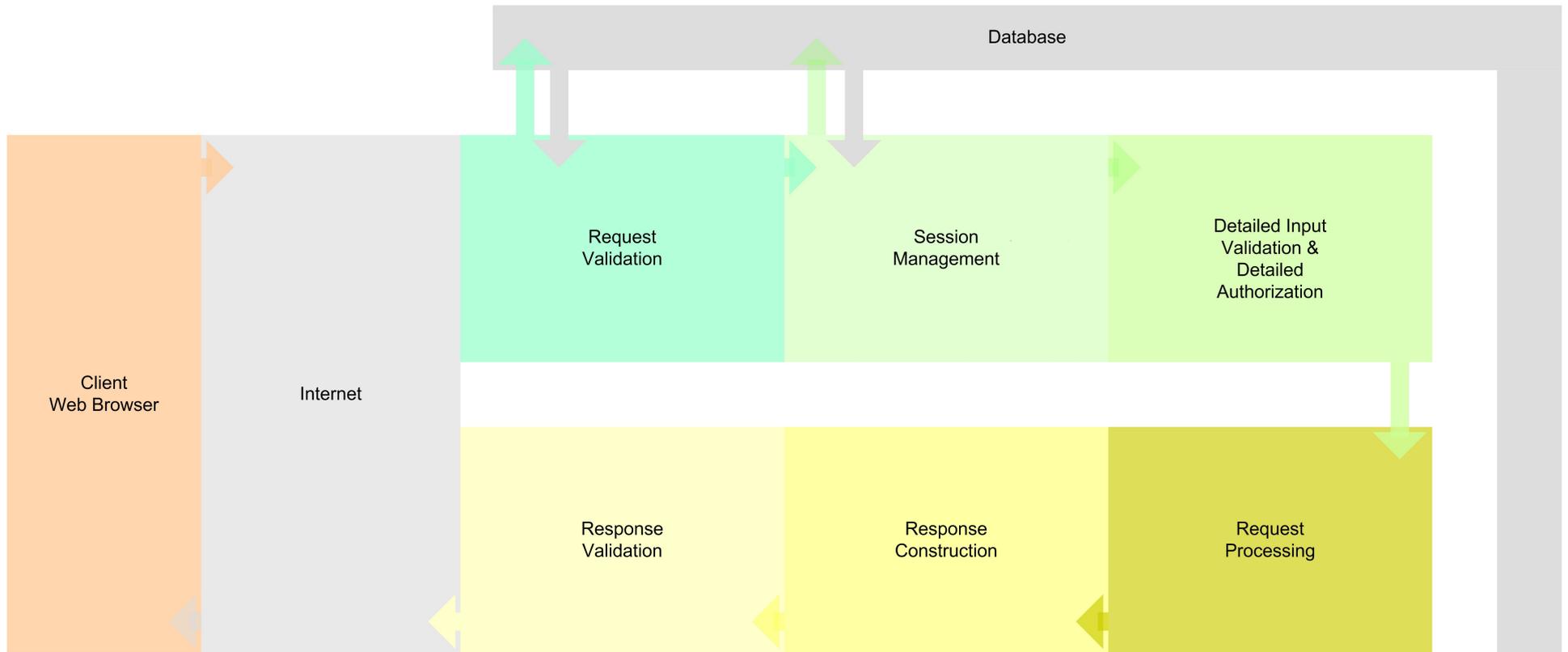
Threat Assessment Approach

ID	Name	AccessControlException				InputException						EncodingEx*				CommandInjectionException				FileIOExcept*			Honey Trap			Behavior Based Events			
		ACE1	ACE2	ACE3	ACE4	IE1	IE2	IE3	IE4	IE5	IE6	EE1	EE2	CIE1	CIE2	CIE3	CIE4	FIO1	FIO2	HT1	HT2	HT3	UT1	UT2	UT3	UT4	STE1	STE2	STE3
WASC Threat Classification – Attacks																													
WASC-03	Integer Overflows																												
WASC-05	Remote File Inclusion																												
WASC-06	Format String																												
WASC-07	Buffer Overflow																												
WASC-08	Cross-site Scripting																												
WASC-09	Cross-site Request Forgery																												
WASC-10	Denial of Service																												
WASC-11	Brute Force																												
WASC-12	Content Spoofing																												
WASC-18	Credential/Session Prediction																												
WASC-19	SQL Injection																												
WASC-23	XML Injection																												
WASC-24	HTTP Request Splitting																												
WASC-25	HTTP Response Splitting																												
WASC-26	HTTP Request Smuggling																												
WASC-27	HTTP Response Smuggling																												
WASC-28	Null Byte Injection																												
WASC-29	LDAP Injection																												
WASC-30	Mail Command Injection																												
WASC-31	OS Commanding																												
WASC-32	Routing Detour																												
WASC-33	Path Traversal																												
WASC-34	Predictable Resource Location																												
WASC-35	SOAP Array Abuse																												
WASC-36	SSI Injection																												
WASC-37	Session Fixation																												
WASC-38	URL Redirector Abuse																												
WASC-39	XPath Injection																												
WASC-41	XML Attribute Blowup																												
WASC-42	Abuse of Functionality																												
WASC-43	XML External Entities																												
WASC-44	XML Entity Expansion																												
WASC-45	Fingerprinting																												
WASC-46	XQuery Injection																												
WASC Threat Classification – Weaknesses																													
WASC-15	Application Misconfiguration																												
WASC-16	Directory Indexing																												
WASC-17	Improper Filesystem Permissions																												
WASC-20	Insufficient Logging & Monitoring																												

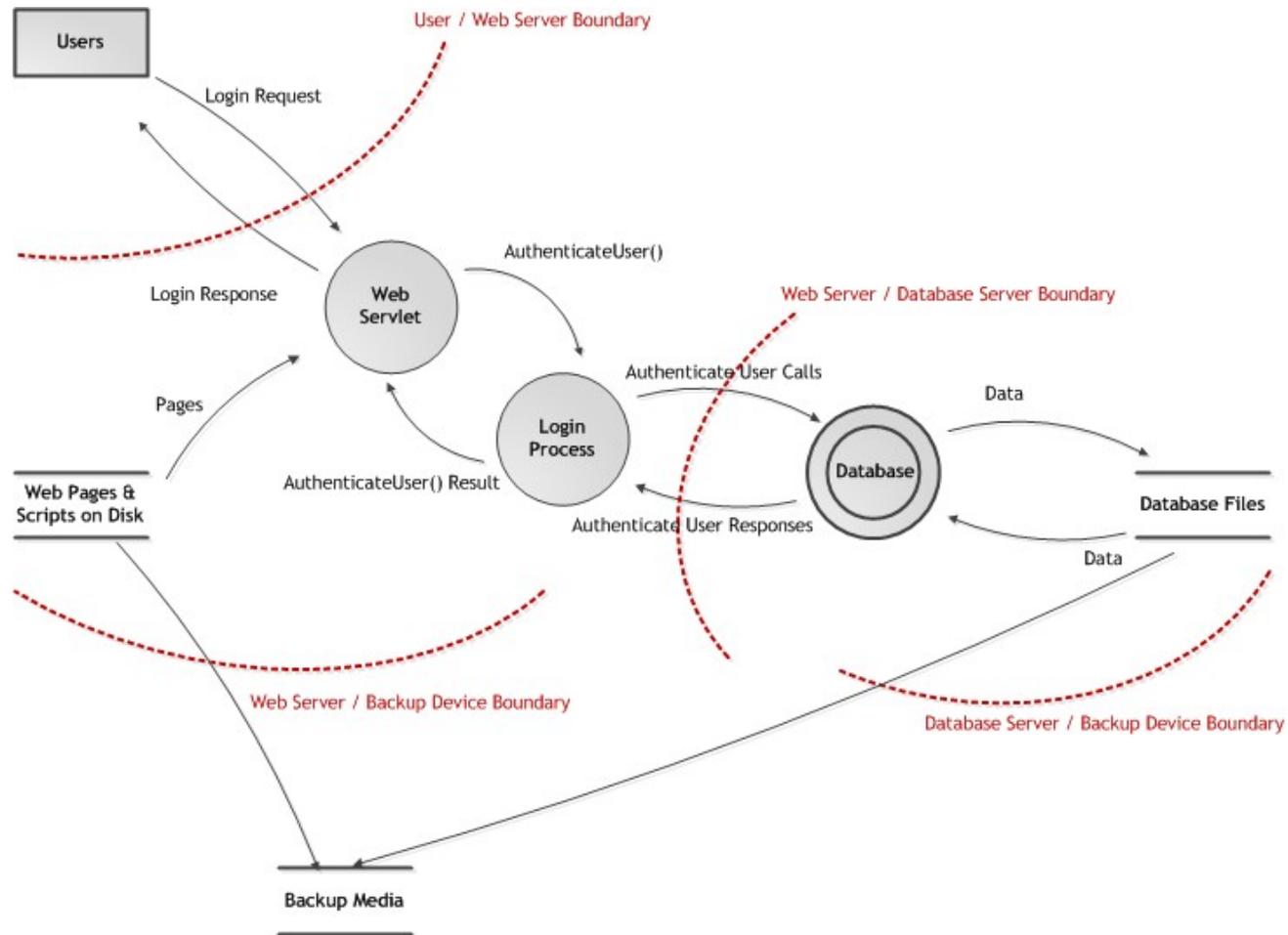
Detection Points Definition

- One application, but what scope?
 - Some or all the time?
 - Some or all users?
 - Some or all entry points
 - Some or all parameters
 - Some or all functions?
- What settings?
 - What is in the whitelist or blacklist?
 - What determines detection?
 - What else?
- Exclude normal behavior?

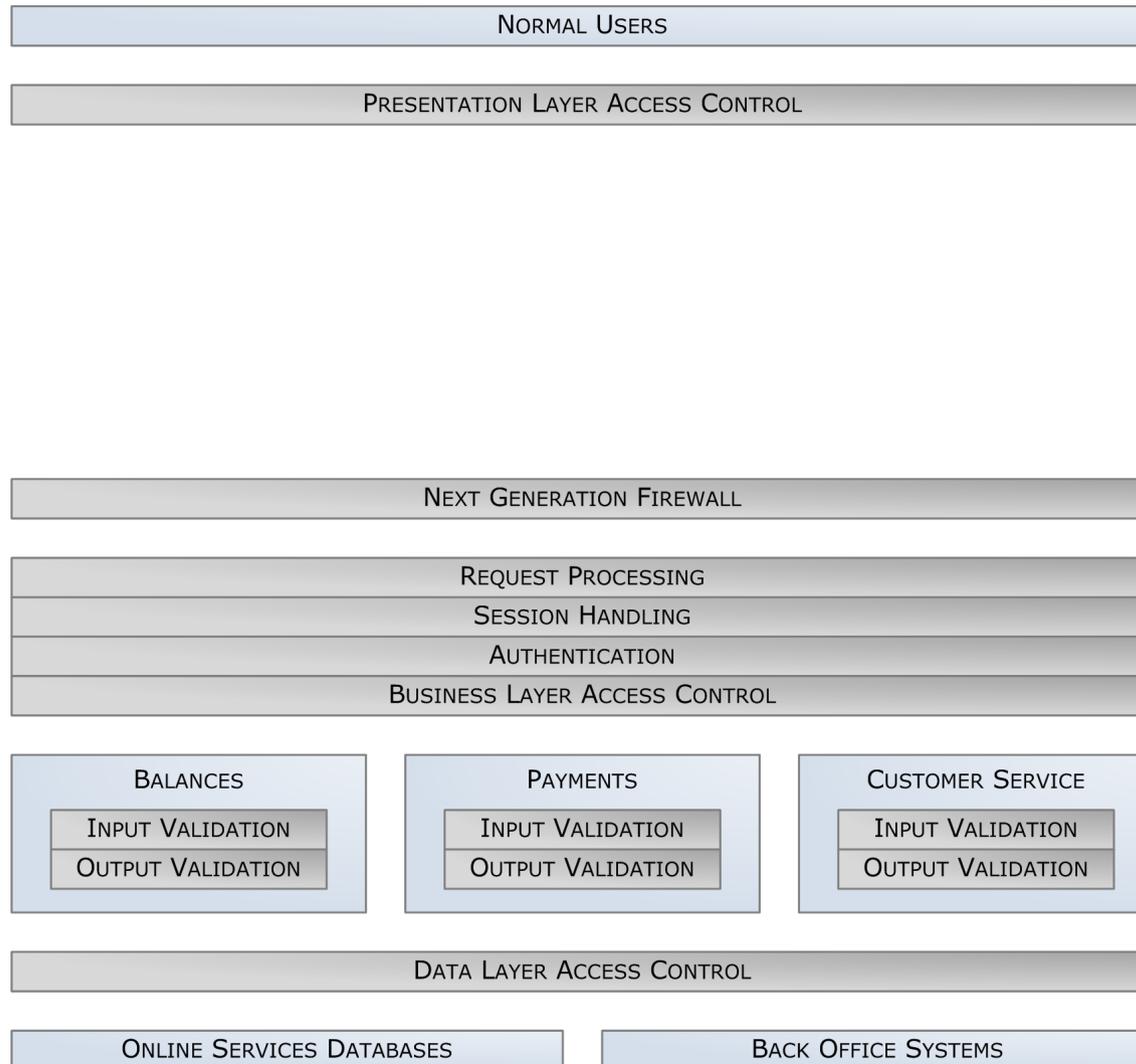
Diagrams 1



Diagrams 2



Diagrams 3



Workshop Example I

Detection Point Selection and Definition

- Objectives
 - Choose and define the ten most appropriate detection points that can be used to help defend your application.
- Method: Choose detection points
 1. Review your application's architecture and business purpose.
 2. Consider what data is important and what attackers' objectives might be in targeting your application (or your company or your users via the application).
 3. Examine the information on detection points and select up to 10 which you will use to detect suspicious behavior and attacks. Give each one a unique ID (e.g. 101, 102) and note what general detection point type it is.
 4. Write the detection points in the schedule. Two tables - application-scoped detected points, and locally-scoped ones.

Workshop Example I

Detection Point Selection and Definition

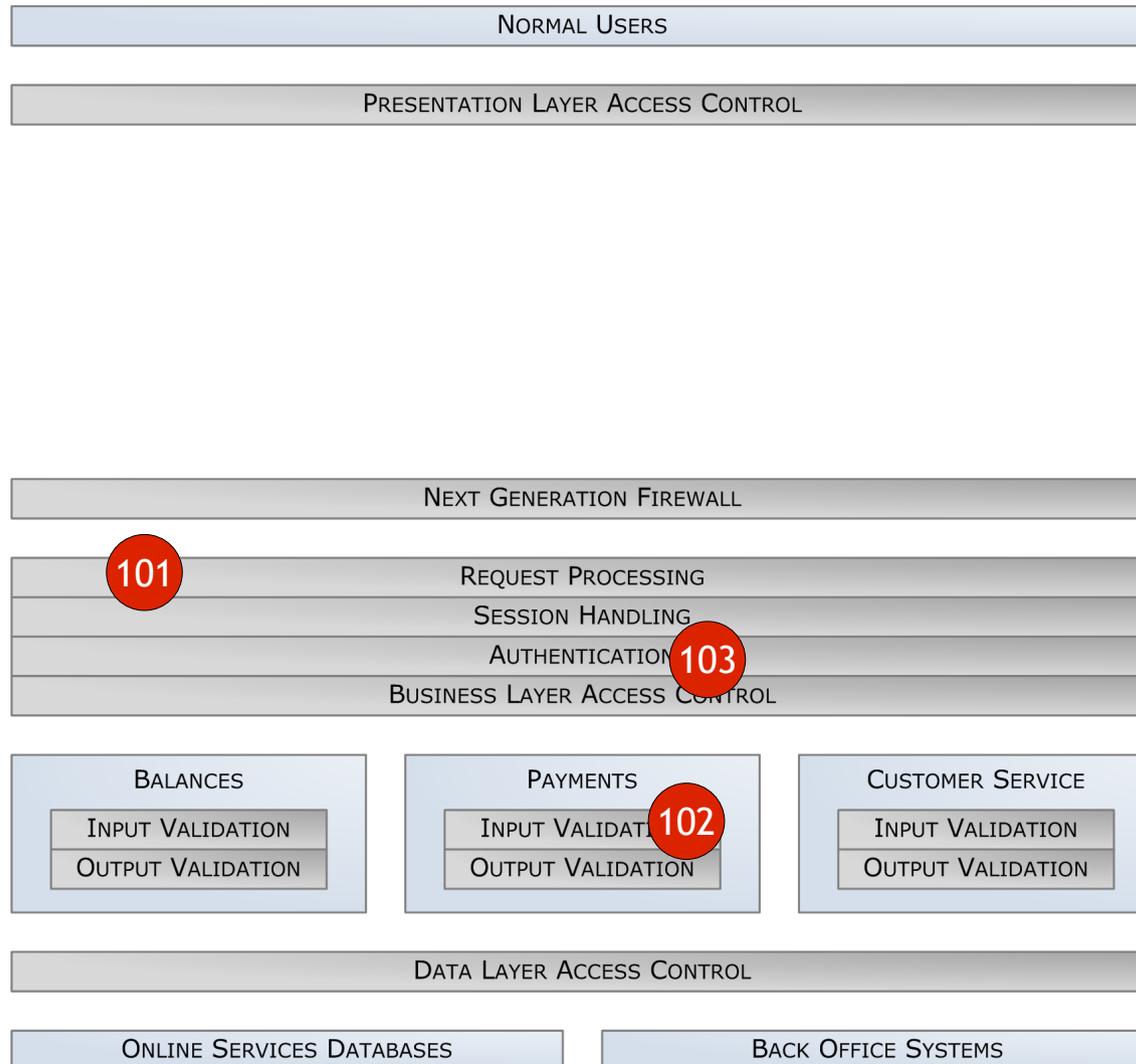
- Three examples

Application Scoped Detection Points		
ID	Type	Title
101	RE2	<i>HTTP Request Using an Unsupported Method</i>

Locally Scoped Detection Points		
ID	Type	Title
102	IE1	<i>Payment Destination Violates Whitelist Input Data Validation</i>
103	HT3	<i>Hidden Field Honey Trap Data Received in Password Field</i>

Workshop Example I

Detection Point Selection and Definition



Application Logging Inspiration

- How to Do Application Logging Right, Anton Chuvakin and Gunnar Peterson, IEEE Security & Privacy Journal
<http://arctecgroup.net/pdf/howtoapplogging.pdf>
- OWASP ESAPI Logger (Java), OWASP
http://owasp-esapi-java.googlecode.com/svn/trunk_doc/latest/org/owasp/esapi/Logger.html
- See also:
 - SP 800-92 Guide to Computer Security Log Management, NIST
<http://csrc.nist.gov/publications/nistpubs/800-92/SP800-92.pdf>
 - OWASP Logging Project, OWASP
https://www.owasp.org/index.php/Category:OWASP_Logging_Project#tab=Main

Application Event Logging Aspiration

When
Event date/time
Log date/time

Security Event
Type
Severity
Confidence
Custom classifications
Owner

Location
Host
Service/application name
Port
Protocol
HTTP method
Entry point
Request number

Request
Purpose
Target

AppSensor Detection
Sensor ID
Sensor location
AppSensor ID(s)
Description

Result
Status
Reason for status
HTTP status code
AppSensor Result ID(s)
Description
Message

Who/what
Source
User identity
HTTP User Agent
Client fingerprint

Extra?
Request headers
Request body
Response headers
Response body
Error stack trace
Error message

Record integrity
Identity
Hash

Definition Properties

- Application
 - Name or identity
- ID
 - 101, 102, etc
- Type
 - AppSensor detector type
- Domain(s) and ports(s)
- Purpose
 - The intent in a sentence
- Processes
 - Business processes and areas
- Entry points
 - Definition of URLs or interfaces
- Target users
 - Which users?
- Description
 - Detailed definition describing how the detection point works and any considerations
- Related detection points
 - Reference to other detection points that are similar or might have an effect on the operation of the one being defined

Workshop Example II

Detection Point Selection and Definition

- Objectives
 - Choose and define the ten most appropriate detection points that can be used to help defend your application.
- Method: Define detection points
 1. For each detection point, complete a specification sheet. The information in the sheet should be sufficiently detailed that a supplier could cost the project, and a development team could implement your intentions quickly and accurately, but not so specific that it prevents developers from thinking creatively about how to implement it.

Workshop Example II

Detection Point Selection and Definition

Detection Point Definition			
ID	102	Type	IE1
Title	Payment Destination Violates Whitelist Input Data Validation		
Purpose	To detect & record when an invalid value has been detected for the payment destination field at any stage of the process		
Process(es)	International payment transfers		
Entry Point(s) / Method(s)	/customers/procs/do?proc=346		
Description	<p>This detection point triggers when the server-side business-layer input validation checks identify that the payment destination does not match the whitelist of acceptable values.</p> <p>The payment destination options are rendered as radio buttons with the values set by the application when the page is generated.</p> <p>The payment destination values are small positive, non-zero, integer, indicating which pre-assigned and pre-validated linked international bank account number is the destination for the new payment instruction. Typical values might be one integer from the set {1,2,3}, or for a customer with only one linked account, {1}.</p>		
Related Detection Points	101 has previously checked that a valid HTTP method is being used		

Step 2

- Defining a Response Policy

Strategic Requirements

- Organizational risk tolerance
- User experience
- Application's purpose

Example “Policy”

Similar idea:

“Among the most effective measures which can be implemented to provide a level of real-time protection is reactive session termination ...in a nutshell consists of terminating the user’s session every time an anomalous request is submitted. These include requests with unexpected or invalid parameters, requests submitted out of order, requests prohibited according to the application framework’s access controls, and so on.”

Source: Development and Implementation of Secure Web Applications, August 2011, UK Centre for the Protection of National Infrastructure (CPNI)

Other Examples

- “Application functionality will not be changed unless the user's source location is in a higher-risk country.”
- “Authenticated administrators who have access to the most functionality and the greatest data access permissions should have the strictest thresholds before a response action is undertaken.”
- “Do not prevent users doing anything, but log, monitor and alert fervently.”

Workshop Example III

Defense Policy

- Objectives
 - Create a high-level policy that defines how the defensive mechanisms will respond to different types of attack detection.
- Method
 1. Members of your team should take one of the following roles each, behave with their role's motivations and discuss with the other team members, what the responses should be.
 2. Document 5-10 statements that define your policy.

Workshop Example III

Defense Policy

- Mandatory roles
 - Sales Managers
 - Information Security Officer
- Other roles
 - Product Manager
 - Development Manager
 - Head of Risk
 - Local Union Representative
 - Head of Human Resources

Workshop Example III

Defense Policy

- Two examples

Defense Policy	
No	Policy Statement
1	No administrative users will ever be logged out or locked out of the application, regardless of the detection points activated.
2	Active responses will only be used against users from the internal company network.

Step 3

- Response Action Selection and Specification

Conventional Attack Responses

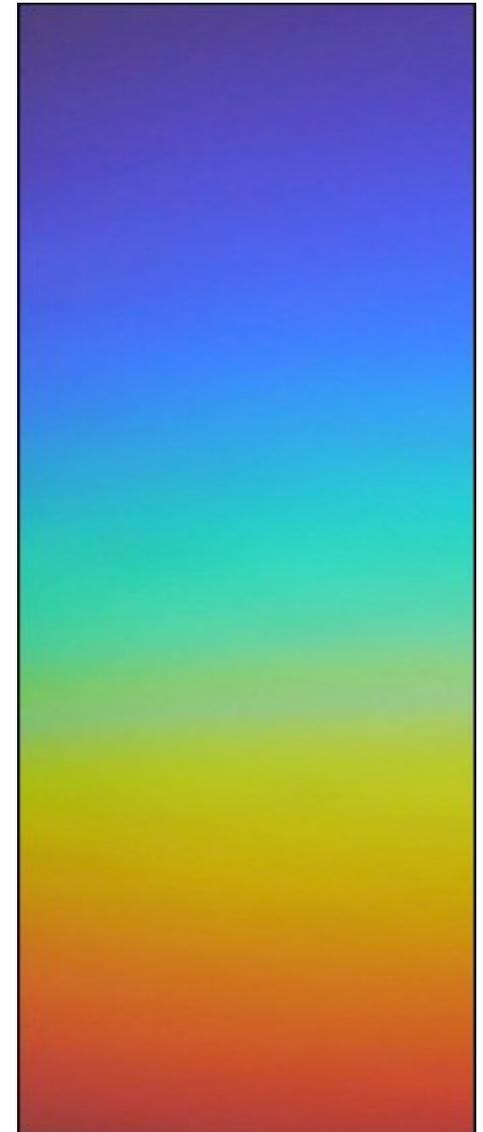
- No change (e.g. just continue logging)

- Process terminated (e.g. reset connection)



Full Spectrum Responses

- No change
- Logging increased
- Administrator notification
- Other notification (e.g. other system)
- Proxy
- User status change
- User notification
- Timing change
- **Process terminated**
- Function amended
- Function disabled
- Account log out
- Account lock out
- Application disabled
- Collect data from user



Application Response Capabilities

- Often already exist
 - Logging level
 - Alerting (email?)
 - User messages
 - Logout
 - Account lockout
 - Redirects
- Much less likely to exist
 - Proxy
 - Disabling individual functions
 - Disabling the application

AppSensor Response Actions

- Response actions describe a change to the application's behavior
- 14 response actions
- Many ways to categorize them, including:
 - Effect on user / application
 - Target of response
 - Duration of response
- Latest list of response actions with descriptions, considerations and examples is maintained at:
https://www.owasp.org/index.php/AppSensor_ResponseActions

AppSensor Response Types and Examples

Response		Examples
ID	Description	
ASR-A	Logging Change	<ul style="list-style-type: none"> Capture sanitised request headers and response bodies Full stack trace of error messages logged Record DNS data on user's IP address Security logging level changed to include 'informational' messages
ASR-B	Administrator Notification	<ul style="list-style-type: none"> Email alert sent to everyone in the administration team SMS alert sent to the on-call administrator Visual indicator on application monitoring dashboard Audible alarm in control room
ASR-C	Other Notification	<ul style="list-style-type: none"> Broadcast event to SIEM Signal sent to upstream network firewall, application firewall (e.g. XML, web) or load balancer Alert sent to fraud protection department Record added to server event log Event highlighted in daily management report Email alert sent to staff member's manager Proactive entry added to customer support system
ASR-D	User Status Change	<ul style="list-style-type: none"> Internal trustworthiness scoring about the user changed Reduce payment transfer limit before additional out-of-band verification is required Reduce maximum file size limit for each file upload by the forum user Increase data validation strictness for all form submissions by this citizen Reduce number of failed authentication attempts allowed before the user's account is locked (ASR-K)

Response Types and Examples (continued)

Response		Examples
ID	Description	
ASR-E	User Notification	<p>On-screen message about mandatory form fields</p> <p>On-screen message about data validation issues</p> <p>Message sent by email to the registered email address to inform them their password has been changed</p>
ASR-F	Timing Change	<p>Extend response time for each failed authentication attempt</p> <p>File upload process duration extended artificially</p> <p>Add fixed time delay into every response</p> <p>Order flagged for manual checking</p> <p>Goods despatch put on hold (e.g. despatch status changed)</p>
ASR-G	Process Terminated	<p>Discard data, display message and force user to begin business process from start</p> <p>Redirection of an unauthenticated user to the log-in page</p> <p>Redirection to home page</p> <p>Display other content (i.e. terminate process but display the output of some other page without redirect)</p> <p>Redirection to a page on another website</p>
ASR-H	Function Amended	<p>Limit on feature usage rate imposed</p> <p>Reduce number of times/day the user can submit a review</p> <p>Additional registration validation steps</p> <p>Additional anti-automation measures (e.g. out-of-band verification activated, CAPTCHA introduced)</p> <p>Static rather than dynamic content returned</p> <p>Additional validation requirements for delivery address</p> <p>Watermarks added to pages, images and other content</p>

Response Types and Examples (continued)

Response		Examples
ID	Description	
ASR-I	Function Disabled	'Add friend' feature inactivated 'Recommend to a colleague' feature links removed and disabled Document library search disabled Prevent new site registrations Web service inactivated Content syndication stopped Automated Direct Debit system turned off and manual form offered instead
ASR-J	Account Logout	Session terminated and user redirected to logged-out message page Session terminated only (no redirect)
ASR-K	Account Lockout	User account locked for 10 minutes User account locked permanently until an Administrator resets it One user's IP address range blocked Unauthenticated user's session terminated
ASR-L	Application Disabled	Website shut down and replaced with temporary static page Application taken offline
ASR-M	Collect Data from user	Deploy additional browser fingerprinting using JavaScript in responses Deploy a Java applet to collect remote IP address Deploy JavaScript to collect information about the user's network
ASR-N	Proxy	Requests from the user invisibly passed through to a hardened system Request are proxied to a special honeypot system which closely mimics or has identical user functionality

Response Actions from the User's Viewpoint

Category		Response	
Type	Description	ID	Description
Silent	User unaware of application's response	ASR-A	Logging Change
		ASR-B	Administrator Notification
		ASR-C	Other Notification
		ASR-N	Proxy
Passive	Changes to user experience but nothing denied	ASR-D	User Status Change
		ASR-E	User Notification
		ASR-F	Timing Change
Active	Application functionality reduced for user(s)	ASR-G	Process Terminated
		ASR-H	Function Amended
		ASR-I	Function Disabled
		ASR-J	Account Logout
		ASR-K	Account Lockout
		ASR-L	Application Disabled
Intrusive	User's environment altered	ASR-M	Collect Data from User

Response Action Classifications

Response		Classification ● indicates 'always' and ○ 'sometimes'								
		Purpose				Target User		Response Duration		
ID	Description	Logging	Notifying	Disrupting	Blocking	One	All	Instantaneous	Period	+
ASR-A	Logging Change	●				●	○	○	○	
ASR-B	Administrator Notification	●	●			●	●	●		
ASR-C	Other Notification	●	●			●		●		
ASR-D	User Status Change	●				●			●	
ASR-E	User Notification	●	●	●		●		●		
ASR-F	Timing Change	●		●		●	○	○	○	
ASR-G	Process Terminated	●	○	●		●		●		
ASR-H	Function Amended	●	○	●	●	●	○		●	○
ASR-I	Function Disabled	●	○	●	●	●	○		●	○
ASR-J	Account Logout	●	○	●	●	●		●		
ASR-K	Account Lockout	●	○	●	●	●			●	○
ASR-L	Application Disabled	●	○	●	●		●			●
ASR-M	Collect Data from User	●				●			●	
ASR-N	Proxy	●				●	○		●	○

Workshop Example IV

Response Action Specification

- Objectives
 - Derive the necessary response actions from the policy. These will be all the capabilities the application needs in the application's response arsenal.
- Method
 1. Work through each policy item and map them to matching ASR codes.
 2. Record the response capabilities required by defining them on the worksheet, giving a unique identifier (e.g. R01, R02, etc), the matching AppSensor Response Type and a brief description.

Workshop Example IV

Response Action Specification

- Three examples

Response Actions		
ID	Type	Description
R01	ASR-A	Additionally log request and response bodies for a user for the remainder of their session
R02	ASR-K	A single user account is locked out for a period of 20 minutes and automatically resets; it may also be manually reset by an application administrator
R03	ASR-K	A single user account is locked indefinitely and can only be reset by an application administrator

Response Threshold Considerations

- Approaches
 - Role dependent
 - Per detection point / per application
- Weightings of suspicious and attack events
- User events and user trends
 - Individual detection points
 - Overall number of security events
- System trend detection points
- Modifying detection points

Example “Policy” (revisited)

“Among the most effective measures which can be implemented to provide a level of real-time protection is reactive session termination ...in a nutshell consists of **terminating the user’s session every time an anomalous request** is submitted. These include requests with unexpected or invalid parameters, requests submitted out of order, requests prohibited according to the application framework’s access controls, and so on.”

Source: Development and Implementation of Secure Web Applications, August 2011, UK Centre for the Protection of National Infrastructure (CPNI)

- Response action
 - ASR-J Account Logout
- Threshold
 - 1

Workshop Example V

Response Threshold Definition

- Objectives
 - For each detection point, and groups of points, define the thresholds when the responses are initiated.
- Method
 1. Take the list of detection points and copy their identifiers and titles into the worksheets. Indicate whether they measure an individual user by indicating “Users=One”, or whether they are measuring all users i.e. “Users=All”.
 2. Then consider what number of security events (the threshold) warrants a response - this will be a small integer often between 2 and 10.
 3. Specify the period over which this threshold is applied. A longer period is more strict e.g. 3 detection events over the course of 20 minutes is stricter than 3 detection events in a minute. You can also write “session” in here to indicate the count is reset at each new user session, but accumulates all the time the session is current.
 4. Define which of your responses occur when the threshold is reached.

Workshop Example V

Response Threshold Definition

- Two examples

A1: Threshold Definition		Users	One
Title	Payment Destination Violates Whitelist Input Data Validation		
Security event threshold	3	Period	1 hour
Response ID(s)	R02 (account lockout)		
Notes	See also C1 where logging is increased for any user detection point activated, whenever that occurs (i.e. threshold=1, period not applicable) until the end of the user's session		

C1: Overall Threshold Definition		Users	One
Title	Increased Logging		
Security event threshold	1 (i.e. as soon as any detection point is activated)	Period	Session
Response ID(s)	R01 (increase logging)		
Notes	As soon as any user detection point fires, logging is increased for the particular user		

Step 4

- Model Tuning

Simplification

- Requirements for each detection point
 - Purpose
 - General statement of its functionality
 - Details of any prerequisites
 - Related detection points
- Duplicate codes
- Aggregating detection points

Tuning

- Typical user activities
- Speed of use
- Static content
- Missing content
- Actions that disable features or the application
- Other systems
- Business metrics
- Defense analysis

Defense Analysis

- Attack scenarios
 - Determine what type(s) of attack methods are being used
 - Consider what vulnerabilities the attacker might be trying to find or exploit
 - What detection point types are relevant?
 - Do these target an individual user, groups or everyone?
 - How many times is the attack performed?
 - What rate is it performed at?
 - How do these compare to the specified detection points, thresholds and response actions in the model being evaluated?
 - Examine usage of words “all” and “every” carefully

Additional Considerations

- Only log
- Overrides
 - Source location
 - Time
- Operational tuning of thresholds
- Export
 - Vulnerability management programmes
 - Security integration manager (SIM) systems
 - Dashboards
- Secure development practices

Optimization

- Three aims
 - Ensure we maintain a low false positive rate through adjusting the sensitivity
 - Consider relationships with other systems and the effects these may have on detection points
 - Identify if any detection points can be removed to eliminate overlaps and duplicates
- Test cases

Implementation

- New project requirements
- Retrofitting existing applications
 - Build into application code
 - Own code
 - Libraries (ESAPI)
 - Modify code at runtime?
 - Externalize?
 - SIEM?
 - Add detection points into application
 - Integrate logging into real time monitor
 - WAF?

Roundup and Conclusions

- Implementation
- Benefits
- Reference materials

Lifecycle

- Secure development practices:
 - Risk analysis
 - Design and code review
 - Testing
 - Operational enablement
- Change management
- Monitoring and tuning
- Ongoing testing

Review

- Very low false positive detection rate
- No need to build business logic elsewhere
- Defenses altered as code is changed
- Cross integration
- Unknown threats
- Information insight
- Application security metrics
- Application worms
- Fraud detection
- Operational risk reduction

Unknown Attacks

- [This list is intentionally left blank]

Operational Risk

1. Actions of people

- Inadvertent
- **Deliberate**
- Inaction

2. Systems and technology failures

- Hardware
- Software
- **Systems**

3. Failed internal processes

- Process design or execution
- **Process controls**
- Supporting processes

4. External events

- Hazards
- Legal issues
- Business issues
- **Service dependencies**

CrossTalk

- September/October 2011
Protecting Against Predatory Practices
- Creating Attack-Aware Software Applications with Real-Time Defenses
Colin Watson, Michael Coates, John Melton, Dennis Groves

OWASP AppSensor Video Demonstrations

- Video presentations by Michael Coates, AppSensor Project Leader:
 - Automated Application Defenses to Thwart Advanced Attackers, June 2010
<http://michael-coates.blogspot.com/2010/06/online-presentation-thursday-automated.html>
 - Attack Aware Applications, April 2011
https://www.owasp.org/index.php/Minneapolis_St_Paul#tab=Video.2FAudio.2FSlides.2FHandouts
- Videos of AppSensor attack detection demonstrations:
 - AppSensor Project media
https://www.owasp.org/index.php/Minneapolis_St_Paul#tab=Video.2FAudio.2FSlides.2FHandouts

OWASP AppSensor Documentation

- Written guidance:
 - OWASP AppSensor, v1.1, Michael Coates, 2008
https://www.owasp.org/images/2/2f/OWASP_AppSensor_Beta_1.1.pdf
 - Implementation Planning Methodology, Colin Watson, 2010
<https://www.owasp.org/index.php/File:Appsensor-planning.zip>

OWASP AppSensor Code

- OWASP ESAPI for Java
 - Developer Guide
https://www.owasp.org/index.php/AppSensor_Developer_Guide
 - <http://meri-stuff.blogspot.com/2011/05/appsensor-intrusion-detection.html>
- Apache Shiro (Java security framework) aka JSecurity
<http://meri-stuff.blogspot.com/2011/05/appsensor-integration-with-shiro.html>

Two Questions (Revisited)

1) Is the application being attacked now?

2) Have any unknown vulnerabilities been exploited today?

Yes

No

~~Don't know~~

Presenter

Colin Watson MSc CITP CISSP CSSLP CISA

- colin@watsonhall.com
- colin.watson@owasp.org

Watson Hall Ltd

- <https://www.watsonhall.com>
- +44 (0) 20 7183 3710

