

The Web Application Strikes Back

JavaZone 2016



Dominik Schadow | bridgingIT

Application Intrusion Detection with OWASP AppSensor

Basics

Architecture

Implementation

Basics

Attacks are often executed as trial & error

Most web
applications don't
inform about
ongoing attacks



You learn about
successful attack
after the data is
leaked

Automatically react on identified attacks in realtime

Stop attackers before they are successful

It's impossible to detect every attack

It's enough to know about a user's intentions

*What makes application intrusion
detection more effective than a
web application firewall?*

?accountNo=66666&amount=10000¤cy=Euro

mybank.com

Account

11111
22222
33333
44444

Amount

10000

Currency

Euro
Dollar

Transfer Money

?accountNo=66666&amount=10000¤cy=Euro

mybank.com

Account

11111
22222
33333
44444

Amount

10000

Currency

Euro
Dollar

Transfer Money

?accountNo=66666&amount=10000¤cy=Euro

Context matters

Application is aware of the business context

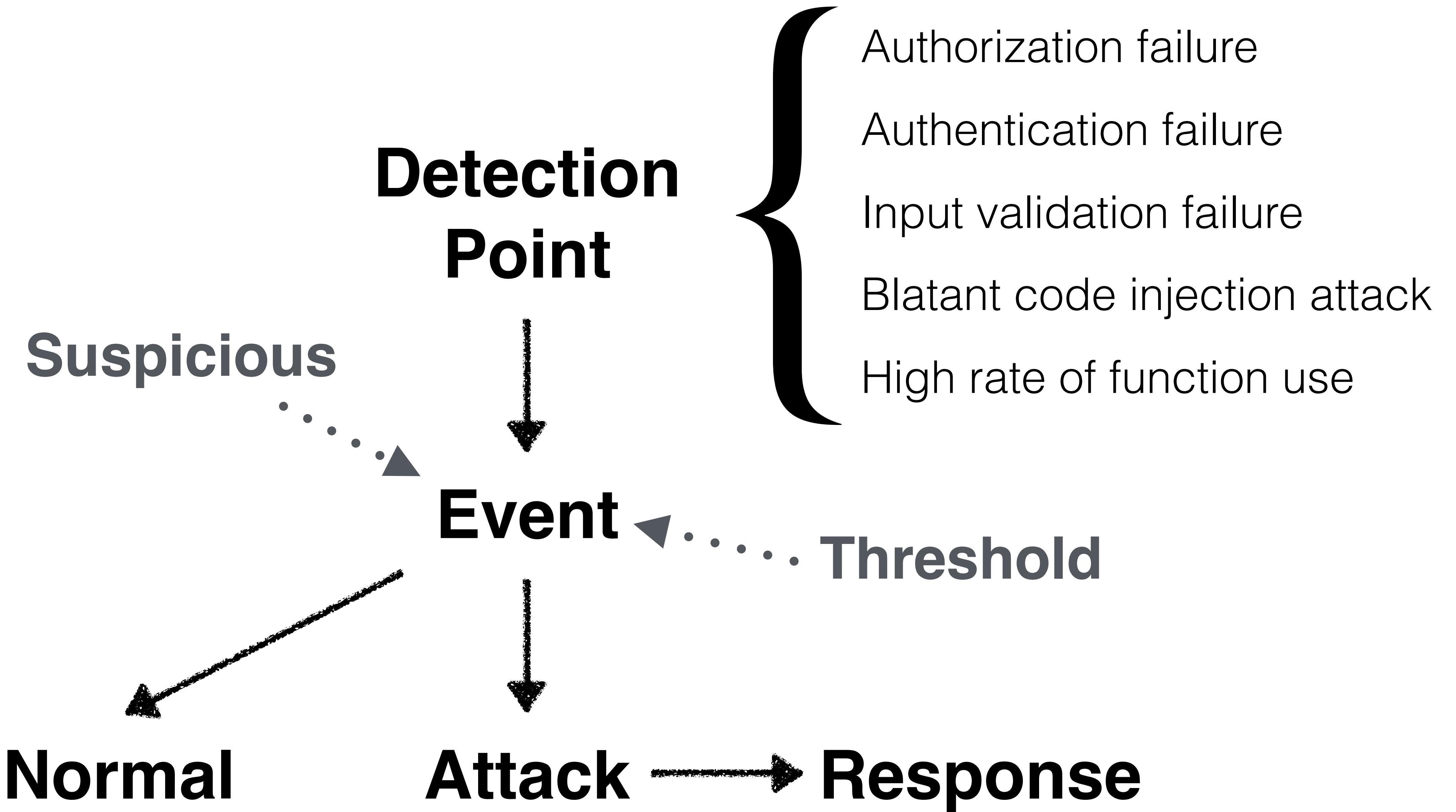
Application is aware of valid and invalid data

Application aware defense

Part of a defense in depth strategy

Application has to be secure

Detect attacks, not vulnerabilities



Detection Points

Detect events in the application (like a sensor)

Tightly integrated in the application code

Useable in any backend application layer

```
@Named  
public class EncounterValidator implements Validator {  
    @Autowired  
    private SpringValidatorAdapter validator;  
  
    public void validate(Object target, Errors errors) {  
        validator.validate(target, errors);  
        // ...  
    }  
}
```

```
@Named  
public class EncounterValidator implements Validator {  
    @Autowired  
    private SpringValidatorAdapter validator;  
  
    public void validate(Object target, Errors errors) {  
        validator.validate(target, errors);  
        // ...  
    }  
}
```

Often already available as isolated
implementations

```
@Autowired  
private DetectionSystem detectionSystem;  
@Autowired  
private EventManager ids;  
  
public void validate(Object target, Errors errors) {  
    validator.validate(target, errors);  
    Encounter encounter = (Encounter) target;  
  
    if (hasXssPayload(encounter.getEvent())) {  
        fireXsseEvent();  
        errors.rejectValue("event", "xss.attempt", "Stop!");  
    }  
}
```

```
@Autowired  
private DetectionSystem detectionSystem;  
@Autowired  
private EventManager ids;  
  
public void validate(Object target, Errors errors) {  
    validator.validate(target, errors);  
    Encounter encounter = (Encounter) target;  
  
    if (hasXssPayload(encounter.getEvent())) {  
        fireXsseEvent();  
        errors.rejectValue("event", "xss.attempt", "Stop!");  
    }  
}
```

May use **blacklists** to identify attacks

```
public boolean hasXssPayload(@NotNull String payload) {  
    return containsAnyIgnoreCase(payload, "<", "script",  
        "onload", "eval", "document.cookie");  
}
```

```
public boolean hasXssPayload(@NotNull String payload) {  
    return containsAnyIgnoreCase(payload, "<", "script",  
        "onload", "eval", "document.cookie");  
}
```

```
@Autowired  
private DetectionSystem detectionSystem;  
@Autowired  
private EventManager ids;  
  
public void validate(Object target, Errors errors) {  
    validator.validate(target, errors);  
    Encounter encounter = (Encounter) target;  
  
    if (hasXssPayload(encounter.getEvent())) {  
        fireXssEvent();  
        errors.rejectValue("event", "xss.attempt", "Stop!");  
    }  
}
```

Extend with central analysis and response unit

Detection Point categories

Request Exception (RE)

Authentication Exception (AE)

Session Exception (SE)

Access Control Exception (ACE)

Input Exception (IE)

Encoding Exception (EE)

Command Injection Exception (CIE)

FileIO Exception (FIO)

User Trend Exception (UT)

System Trend Exception (STE)

Input Exception

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

IE7 - detect abnormal content output structure

```
private void fireXSSEvent() {  
    DetectionPoint dp = new DetectionPoint(  
        DetectionPoint.Category.INPUT_VALIDATION, "IE1");  
    ids.addEvent(new Event(user, dp, detectionSystem));  
}
```

```
private void fireXSSEvent() {  
    DetectionPoint dp = new DetectionPoint(  
        DetectionPoint.Category.INPUT_VALIDATION, "IE1");  
    ids.addEvent(new Event(user, dp, detectionSystem));  
}
```

Category



```
private void fireXSSEvent() {  
    DetectionPoint dp = new DetectionPoint(  
        DetectionPoint.Category.INPUT_VALIDATION, "IE1");  
    ids.addEvent(new Event(user, dp, detectionSystem));  
}
```

Label

```
private void fireXSSEvent() {  
    DetectionPoint dp = new DetectionPoint(  
        DetectionPoint.Category.INPUT_VALIDATION, "IE1");  
    ids.addEvent(new Event(user, dp, detectionSystem));  
}
```

IE1-001, IE1-002, ...

Events

Observable occurrences in an application

Monitored and analyzed to determine an attack

```
private void fireXSSEvent() {  
    DetectionPoint dp = new DetectionPoint(  
        DetectionPoint.Category.INPUT_VALIDATION, "IE1");  
    ids.addEvent(new Event(user, dp, detectionSystem));  
}
```

```
@Autowired  
private DetectionSystem detectionSystem;  
@Autowired  
private EventManager ids;  
  
public void validate(Object target, Errors errors) {  
    validator.validate(target, errors);  
    Encounter encounter = (Encounter) target;  
  
    if (hasXssPayload(encounter.getEvent())) {  
        fireXsseEvent();  
        errors.rejectValue("event", "xss.attempt", "Stop!");  
    }  
}
```

Thresholds

React immediately to malicious events (**threshold = 1**)

Requires **count**

React delayed to suspicious events (**threshold > 1**)

Requires **count** and **interval**

Threshold examples

2 events over the last 5 minutes

10 events over the last 24 hours

200% increase over one hour

User and System Trends

More difficult to configure

Require a large user base/ frequent usage

Consider special situations (sale, holidays, ...)

Without UI validation

wrong
format in
date field



potentially
dangerous
character



unknown
drop down
value



With UI validation



potentially
dangerous
character



unknown
drop down
value



wrong
format in
date field

Responses

Defend the application, its data and its users

Respond to an attack in any way the application supports

Must happen automatically in realtime

No retaliation



Logging change

No response

User status change

Timing change

Function disabled

Account lockout

Account logout

Collect data from user

User notification

Proxy

Admin notification

Function amended

Process terminated

Application disabled

Respond to suspicious events and
identified attacks differently

Responses and Thresholds

Threshold

Response

2. event

Log request

4. event

Logout user

6. event

Disable user

```
<detection-point>
  <category>Input Validation</category>
  <id>IE1</id>
  <threshold>
    <count>2</count>
    <interval unit="minutes">5</interval>
  </threshold>
  <responses>
    <response>
      <action>log</action>
    </response>
    <response>
      <action>logout</action>
    </response>
  </responses>
</detection-point>
```

```
<detection-point>
  <category>Input Validation</category>
  <id>IE1</id>
  <threshold>
    <count>2</count>
    <interval unit="minutes">5</interval>
  </threshold>
  <responses>
    <response>
      <action>log</action>
    </response>
    <response>
      <action>logout</action>
    </response>
  </responses>
</detection-point>
```

```
<detection-point>
  <category>Input Validation</category>
  <id>IE1</id>
  <threshold>
    <count>2</count>
    <interval unit="minutes">5</interval>
  </threshold>
  <responses>
    <response>
      <action>log</action>
    </response>
    <response>
      <action>logout</action>
    </response>
  </responses>
</detection-point>
```

```
<detection-point>
  <category>Input Validation</category>
  <id>IE1</id>
  <threshold>
    <count>2</count>
    <interval unit="minutes">5</interval>
  </threshold>
  <responses>
    <response>
      <action>log</action>
    </response>
    <response>
      <action>logout</action>
    </response>
  </responses>
</detection-point>
```

```
<detection-point>
  <category>Input Validation</category>
  <id>IE1</id>
  <threshold>
    <count>2</count>
    <interval unit="minutes">5</interval>
  </threshold>
  <responses>
    <response>
      <action>log</action>
    </response>
    <response>
      <action>logout</action>
    </response>
  </responses>
</detection-point>
```

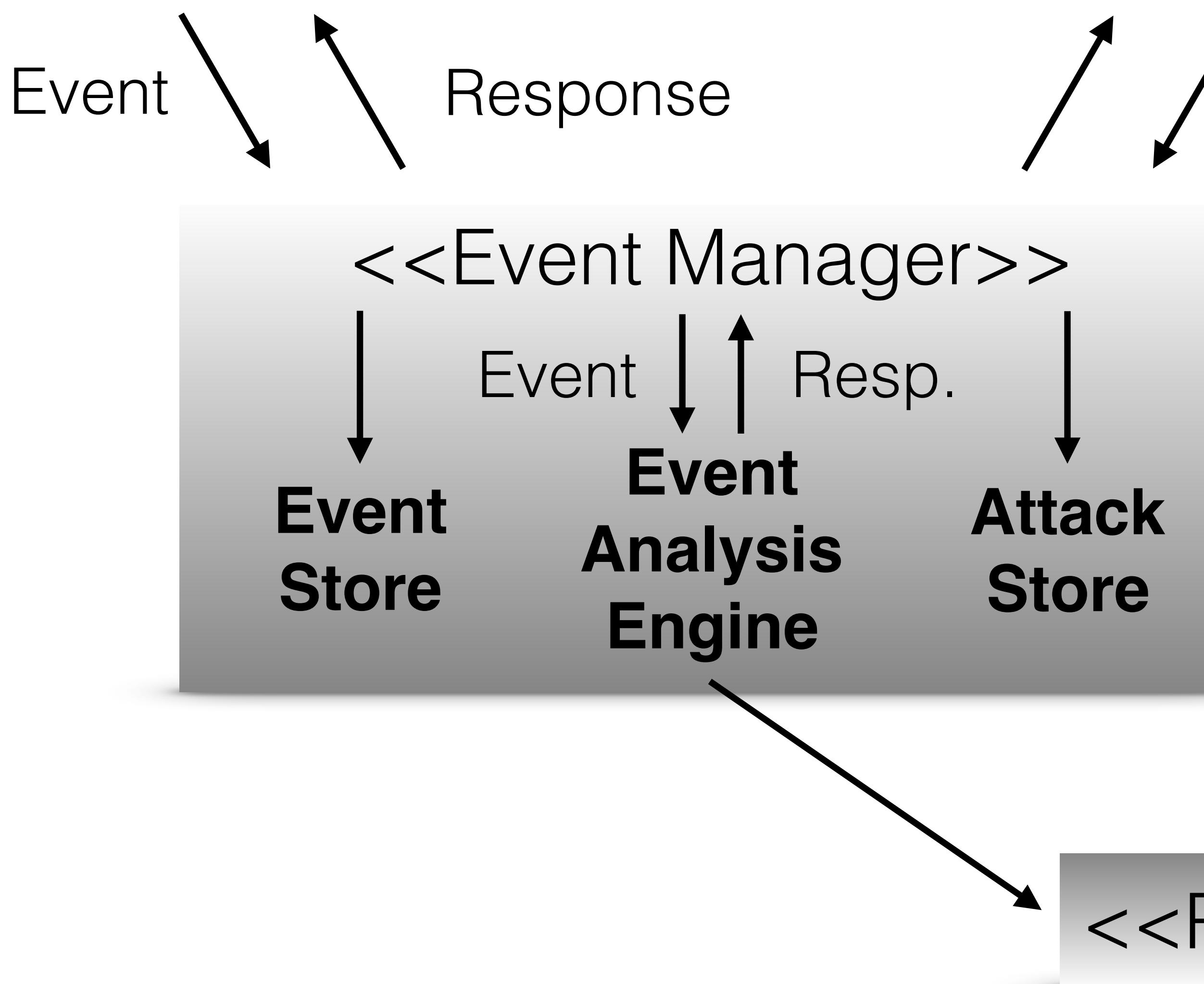
Architecture

Execution modes

	REST	SOAP	Thrift	RabbitMQ	Kafka	Local
Application language			Java & non Java			Java
# of applications			multiple			1
Cluster support			yes			no

<<Client Application 1>>
Detection Points

<<Client Application n>>
Detection Points



Handling detections and
responses in one application

One application and it is Java

Implementation

My Profile arthur@dent.com (Rookie)

[Edit Userdata](#)[Change Password](#)

My Encounters

JavaOne 2014 (09/30/2014)

San Francisco (USA)

5

JavaOne 2012 (10/01/2012)

San Francisco (USA)

0

My Confirmations

JavaOne 2008 (10/10/2012)

San Francisco (USA)

JavaOne 2005 (10/10/2005)

San Francisco (USA)

[Add Encounter](#)[Add Confirmation](#)

Duke Encounters

The leading online platform for Java Duke spotting.

About

This demo web application is developed by [Dominik Schadow](#), source code is available on [GitHub](#).

Navigation

[Home](#)
[Encounters](#)
[Search](#)
[Account](#)

Follow me

[Blog](#)
[Twitter](#)
[GitHub](#)

Demo

React on anonymous and authenticated users

```
private void fireEvent() {
    if (user.isAnonymous()) {
        DetectionPoint dp = new DetectionPoint(
            INPUT_VALIDATION, "IE1-001");
        ids.addEvent(new Event(user, dp, detectionSystem));
    } else {
        DetectionPoint dp = new DetectionPoint(
            INPUT_VALIDATION, "IE1-002");
        ids.addEvent(new Event(user, dp, detectionSystem));
    }
}
```

```
private void fireEvent() {
if (user.isAnonymous()) {
    DetectionPoint dp = new DetectionPoint(
        INPUT_VALIDATION, "IE1-001");
    ids.addEvent(new Event(user, dp, detectionSystem));
} else {
    DetectionPoint dp = new DetectionPoint(
        INPUT_VALIDATION, "IE1-002");
    ids.addEvent(new Event(user, dp, detectionSystem));
}
}
```

```
private void fireEvent() {
    if (user.isAnonymous()) {
        DetectionPoint dp = new DetectionPoint(
            INPUT_VALIDATION, "IE1-001");
        ids.addEvent(new Event(user, dp, detectionSystem));
    } else {
        DetectionPoint dp = new DetectionPoint(
            INPUT_VALIDATION, "IE1-002");
        ids.addEvent(new Event(user, dp, detectionSystem));
    }
}
```

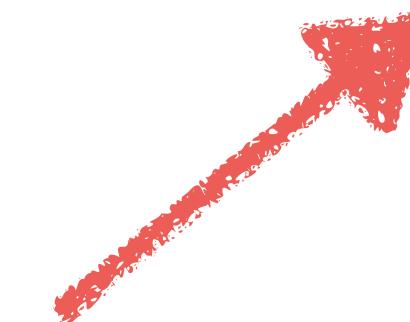
Separate detection unit and
response unit

How to start

Identify critical business functionality

Set detection points and thresholds

Determine proportional responses



Increased logging of
'attacks' is a good start

Wrap up

Add another layer of defense to web applications

Respond to ongoing attacks in realtime

Reduce impact of successful attacks

Develop securely and strike back



Marienstr. 17
70178 Stuttgart
Germany

dominik.schadow@bridging-it.de
www.bridging-it.de

Blog blog.dominiksshadow.de
Twitter @dschadow

Demo Projects

github.com/dschadow/ApplicationIntrusionDetection

OWASP AppSensor

appsensor.org

Detection Points

www.owasp.org/index.php/AppSensor_DetectionPoints

Pictures

www.dreamstime.com

