



Web Application Defense with Bayesian Attack Analysis

Presented by:

Ryan Barnett

Senior Security Researcher

OWASP ModSecurity CRS Leader

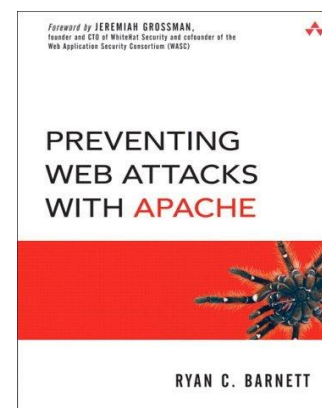
Ryan Barnett - Background

- Trustwave

- Senior Security Researcher
- Member of SpiderLabs Research
- Surveillance Team Lead
 - IDS/IPS
 - MailMax
 - WAF
- Web Application Defense
- ModSecurity Project Leader

- Author

- “Preventing Web Attacks with Apache”
 - Pearson Publishing - 2006
- “The Web Application Defenders’ Cookbook”
 - Wiley Publishing – (Due end of 2012)

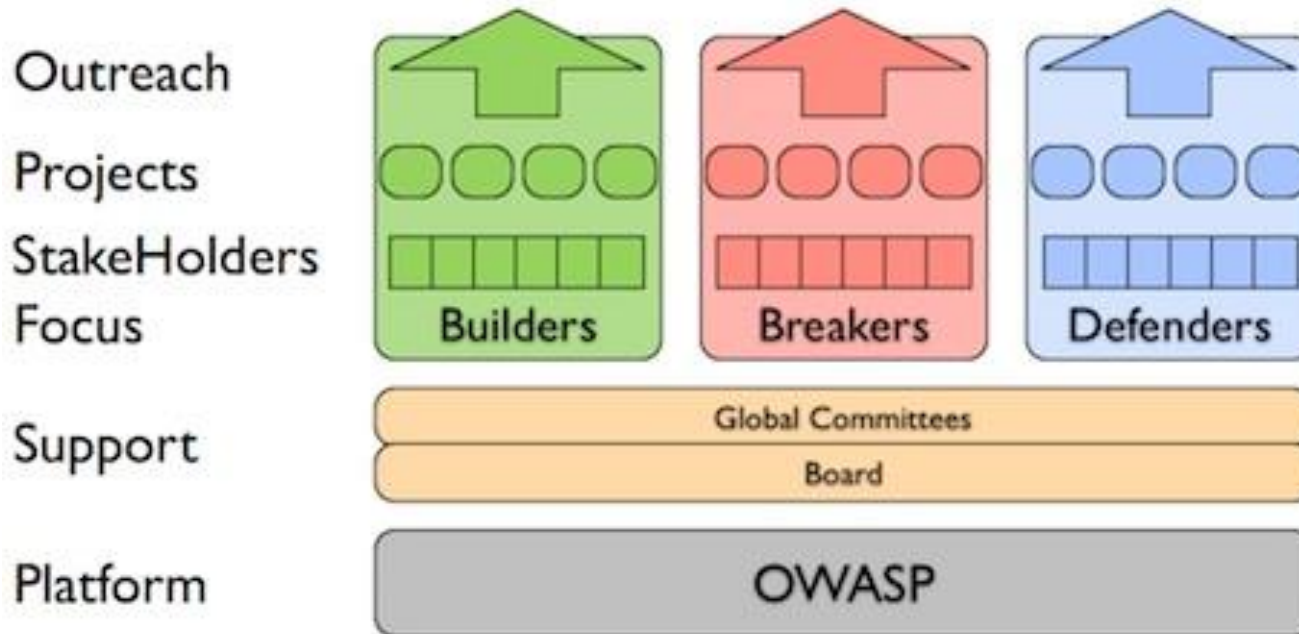


Agenda

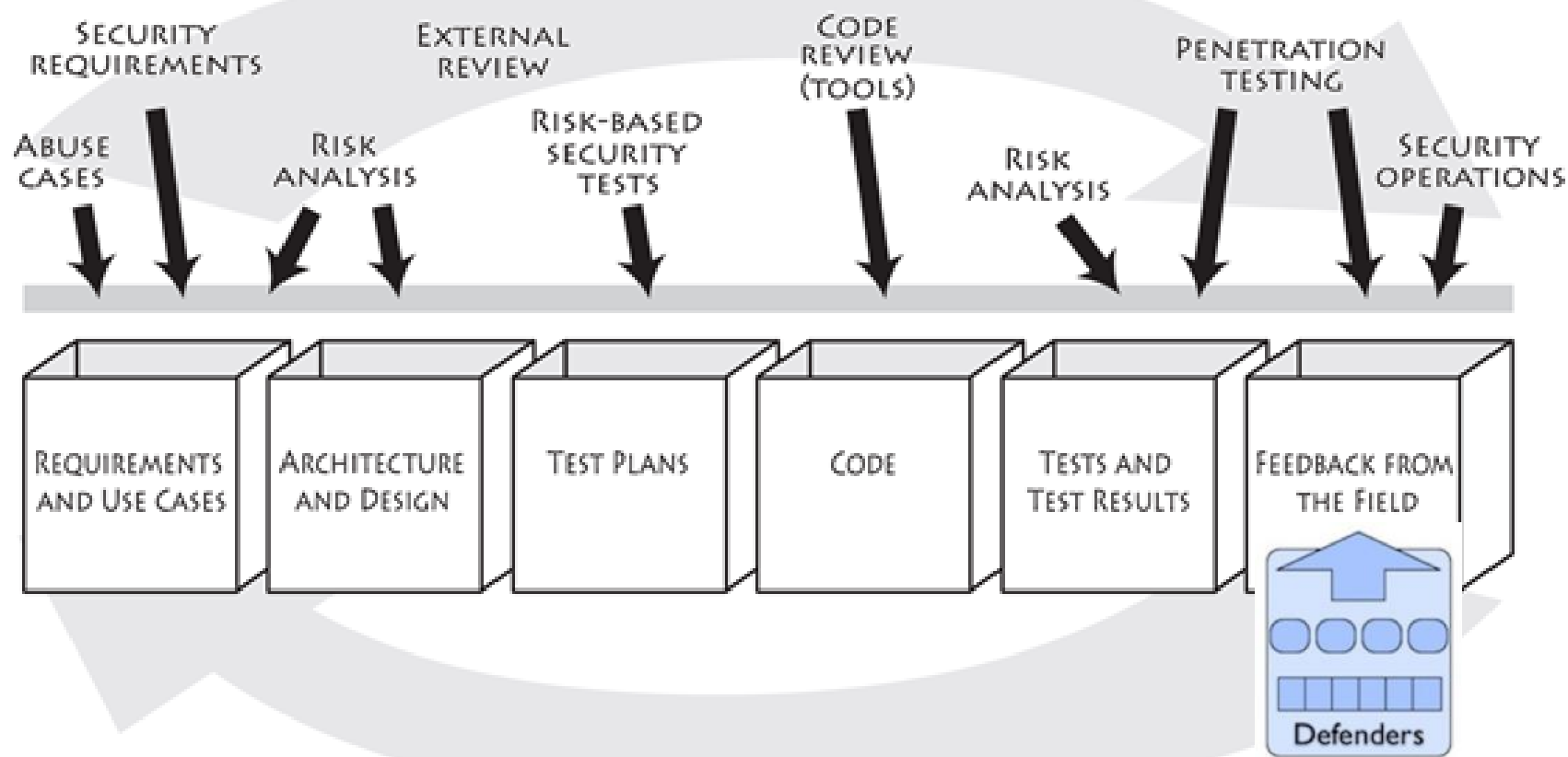
- Attack Resistance Testing
 - Blacklist Filter Evasions
 - ModSecurity SQL Injection Challenge Result Example
- Evasion Analysis
 - Time-to-Hack Metrics
 - Common Evasion Methodology
- Using Bayesian Analysis for Attack Detection
 - OSBF-Lua within ModSecurity
 - Ham/SPAM Training
 - Attack Detection Examples
- Conclusion
 - Development Plans
 - Call for participation

Target Audience: Defender Community

A Vision for OWASP



Defending Live Web Applications



Tuesday, April 3, 2012

OWASP Security Blitz - April : Injection Attacks

OWASP is starting a monthly security blitz where we will rally the security community around a particular topic. The topic may be a vulnerability, defensive design approach, technology or even a methodology. All members of the security community are encouraged to write blog posts, articles, patches to tools, videos etc in the spirit of the current monthly topic. Our goal is to show a variety of perspectives on the topic from the different perspectives of builders, breakers and defenders.

Today I'm happy to kick off our first month of the OWASP Security Blitz with the topic of:
Injection Attacks - SQL Injection

Please tweet your contributions with **hashtag #OWASP** and also add a **comment to this post** with a link to the material.

At the end of the month we will gather the new articles and include a summary in an upcoming OWASP newsletter. We may even hold a small vote to determine the best contribution of the month.

Let's start the rally!

Attack Resistance Testing: *Blacklist Filter Evasions*

OWASP ModSecurity Core Rule Set Project



This project is part of the OWASP [Defenders](#) community.
Feel free to browse other projects within the [Defenders](#), [Builders](#), and [Breakers](#) communities.

[Home](#) [Download](#) [Bug Tracker](#) [Demo](#) [Installation](#) [Documentation](#) [Presentations and Whitepapers](#) [Related Projects](#)

[Latest News and Mail List](#) [Contributors, Users and Adopters](#) [Project About](#)

Overview

ModSecurity™ is a web application firewall engine that provides very little protection on its own. In order to become useful, ModSecurity™ must be configured with rules. In order to enable users to take full advantage of ModSecurity™ out of the box, Trustwave's SpiderLabs is providing a free certified rule set for ModSecurity™ 2.x. Unlike intrusion detection and prevention systems, which rely on signatures specific to known vulnerabilities, the Core Rules provide generic protection from unknown vulnerabilities often found in web applications, which are in most cases custom coded. The Core Rules are heavily commented to allow it to be used as a step-by-step deployment guide for ModSecurity™.

Core Rules Content

In order to provide generic web applications protection, the Core Rules use the following techniques:

- **HTTP Protection** - detecting violations of the HTTP protocol and a locally defined usage policy.
- **Real-time Blacklist Lookups** - utilizes 3rd Party IP Reputation
- **Web-based Malware Detection** - identifies malicious web content by check against the Google Safe Browsing API.
- **HTTP Denial of Service Protections** - defense against HTTP Flooding and Slow HTTP DoS Attacks.
- **Common Web Attacks Protection** - detecting common web application security attack.
- **Automation Detection** - Detecting bots, crawlers, scanners and other surface malicious activity.
- **Integration with AV Scanning for File Uploads** - detects malicious files uploaded through the web application.
- **Tracking Sensitive Data** - Tracks Credit Card usage and blocks leakages.
- **Trojan Protection** - Detecting access to Trojans horses.
- **Identification of Application Defects** - alerts on application misconfigurations.
- **Error Detection and Hiding** - Disguising error messages sent by the server.



Search

Search



SpiderLabs Services

- [360 Application Security](#)
- [App Code Review](#)
- [App PenTesting](#)
- [Incident Response](#)
- [Net PenTesting](#)
- [ModSecurity Rules](#)
- [Physical Security](#)

Resources

- [Advisories](#)
- [Papers](#)
- [Projects](#)
- [Security Tools](#)

Categories

[\[HoneyPot Alert\]](#) [Advisories](#)
[Application Security](#)
[Conferences](#) [Global Security](#)

« [ModSecurity Advanced Topic of the Week: Application Logout Response Actions](#) | [Main](#) | [Announcing Release of ModSecurity v2.6.1-RC1](#) »

22 June 2011

Announcing the ModSecurity SQL Injection Challenge

The ModSecurity Project Team is happy to announce our [first community hacking challenge](#)!

This is a **SQL Injection and Filter Evasion Challenge**. We have setup ModSecurity to proxy to the following 4 commercial vuln scanner demo sites:

- [IBM \(AppScan\) - demo.testfire.net site](#)
- [Cenzic \(HailStorm\) - CrackMe Bank site](#)
- [HP \(WebInspect\) - Free Bank site](#)
- [Acunetix \(Acunetix\) - Acuart site](#)

Challenge Details

To successfully complete the challenge, participants must do the following:

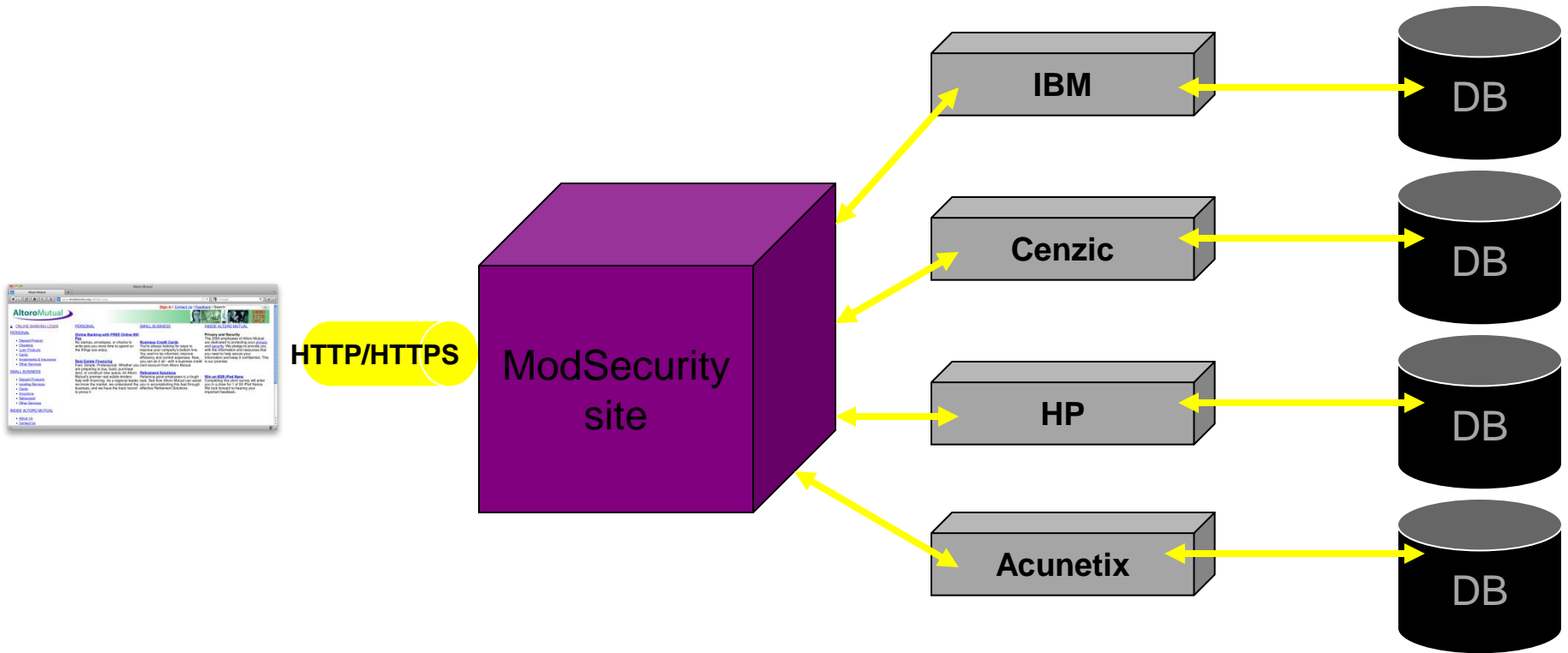
1. Identify a SQL Injection vector within one of the demo websites listed above.
2. Successfully enumerate the following information about the database:
 - DB User(s) - provide request data.
 - DB Name(s) - provide request data.
 - Table Name(s) - provide request data.
 - Column Name(s) - provide request data.

Challenge Submission

Please send challenge submissions to security@modsecurity.org with the details from above.

<http://blog.spiderlabs.com/2011/06/announcing-the-modsecurity-sql-injection-challenge.html>

SQL Injection Challenge Architecture



Two Challenge Levels

- Level I – Speed Hacking
 - Find an SQLi attack vector
 - Exploit the SQLi vulnerability
 - Enumerate the required DB data
 - Submit the data to us for review
- Level II – Blacklist Filter Evasion
 - Same as Level I, however you must evade the OWASP ModSecurity CRS Blacklist Filters

Level II – Filter Evasions

Altoro Mutual: Server Error

www.modsecurity.org/bank/login.aspx

ModSecurity Alert Message:
Inbound Alert: 981243-Detects classic SQL injection probings 2/2
Outbound Alert: The application is not available
TX ID: hCVXc8Co8AoAABq52dMAAAAA

[Sign In](#) | [Contact Us](#) | [Feedback](#) | Search

Altoro Mutual

DEMO SITE ONLY

An Error Has Occurred

Summary:

Syntax error in date in query expression 'username = " or '1='1#' AND password = 'assda'".

Error Message:

System.Data.OleDb.OleDbException: Syntax error in date in query expression 'username = " or '1='1#' AND password = 'assda'. at System.Data.OleDb.OleDbCommand.ExecuteNonQueryErrorHandling(OleDbHResult hr) at System.Data.OleDb.OleDbCommand.ExecuteNonQueryForSingleResult(tagDBPARAMS dbParams, Object& executeResult) at System.Data.OleDb.OleDbCommand.ExecuteNonQuery(Object& executeResult) at System.Data.OleDb.OleDbCommand.ExecuteNonQuery(CommandBehavior behavior, Object& executeResult) at

Challenge Participation

- More than 650 participants (in 18 Countries)



Challenge Winners

- Winners received the following:
 - Recognition - Name(s) listed on the Challenge website
 - Shwag - ModSecurity t-shirt
- Everyone is Happy 😊



Level 1 Winners

IBM Testfire:

- [PT Research](#)

Cenzic CrackMe Bank:

- [Ahmad Maulana](#)

HP Free Bank:

- [Alexander Zaitsev](#)

Acunetix Acuart:

- [Travis Lee](#)

Level 2 Winners

- [Johannes Dahse](#)
- [Vladimir Vorontsov](#)
- [PT Research](#)
- [Ahmad Maulana](#)
- [Travis Lee](#)
- [Roberto Salgado](#)
- [SQLMap Developers](#)
- [HackPlayers](#)
- [Georgi Geshev](#)
- TBD
- TBD
- TBD

Level II Filter Evasion: *Example*

Attacking the RegEx Logic

```
SecRule REQUEST_FILENAME|ARGS_NAMES|ARGS|XML:/* \
"\bunion\b.{1,100}?\bselect\b" \
"phase:2,rev:'2',capture,t:none,t:urlDecodeUni,t:html
EntityDecode t:1,ctl:auditLogParts=
t:replaceCom,ctl:auditLogParts=
+E,block,msg:'SQL Injection Attack',id:'WASC-
ASCTC/WASC-19',tag:'OWASP_TOP_10/A1',tag:'OWASP_AppSensor/CIE1',tag:
'PCI/6.5.2',logdata:'%{TX.0}',severity:'2',setvar:'tx.msg
=%{rule.msg}',setvar:tx.sql_injection_score=+{%tx.critical_
l_anomaly_score},setvar:tx.anomaly_score=+{%tx.critical_a
nomaly_score},setvar:tx.%{rule.id}-
WEB_ATTACK/SQL_INJECTION-%{matched_var_name}=%{tx.0}"
```

Regex allows up to 100 characters between "union" and "select"

8.6. Comment Syntax

MySQL Server supports three comment styles:

- From a “#” character to the end of the line.
- From a “-- ” sequence to the end of the line. In MySQL, the “-- ” (double-dash) comment style requires the second dash to be followed by at least one whitespace or control character (such as a space, tab, newline, and so on). This syntax differs slightly from standard SQL comment syntax, as discussed in [Section 1.8.5.5. “'--' as the Start of a Comment”](#).
- From a /* sequence to the following */ sequence, as in the C programming language. This syntax enables a comment to extend over multiple lines because the beginning and closing sequences need not be on the same line.

The following example demonstrates all three comment styles:

```
mysql> SELECT 1+1;      # This comment continues to the end of line
mysql> SELECT 1+1;      -- This comment continues to the end of line
mysql> SELECT 1 /* this is an in-line comment */ + 1;
mysql> SELECT 1+
/*
this is a
multiple-line comment
*/
1;
```

Nested comments are not supported. (Under some conditions, nested comments might be permitted, but usually are not, and users should avoid them.)

Excessive Comment Text

- %40%40new%20union%23sqlmaps%0Aselect%201,2,database%23sqlmap%0A%28%29

- URL Decoded

- @@new

union#sqlmaps%0Aselect%201,2,database%23sqlmap%0A%28%29

103 chars of random text
bypasses the regex rule logic

Evasion Analysis

Common Methodology

- Automation to identify injection points
 - NetSparker
 - Arachni
 - Sqlmap
 - Havij
- Manual testing to develop working SQLi payloads
 - ***An iterative process of trial and error***
 1. Send initial payloads and observe DB responses
 2. Use obfuscation tactics (comments, encodings, etc...)
 3. Send payload and observe DB response
 4. Repeat steps 2 - 3

Time-to-Hack Metrics

Time-to-Hack Metric	Speed Hacking	Filter Evasion
Avg. # of Requests	170	433
Avg. Duration (Time)	5 hrs 23 mins	72 hrs
Shortest # of Requests	36	118
Shortest Duration (Time)	46 mins	10 hrs

Filter Evasion Conclusions

- Blacklist filtering will only slow down determined attackers
- Attackers need to try **many permutations** to identify a working filter evasion
- The OWASP ModSecurity Core Rules Set's blacklists SQLi signatures **caught several hundred** attempts before an evasion was found

Questions

- How can we use this methodology to our advantage?
- What detection technique can we use other than regular expressions?

Application Intrusion Detection

- Positive/Whitelist Security Model Input Validation
 - Allowed characters
 - Length
 - WAF Traffic Profiling
- Response Time Latency Tracking
 - Deviations of response data due to blind SQLi queries (waitfor delay, benchmark() or pg_sleep)
- Response Page Fingerprint Deviations
 - Changes to the page construction (title, size, etc...)
 - Deviation in the amount of sensitive records returned

Using Bayesian Analysis for Attack Detection

Bayesian Analysis for HTTP

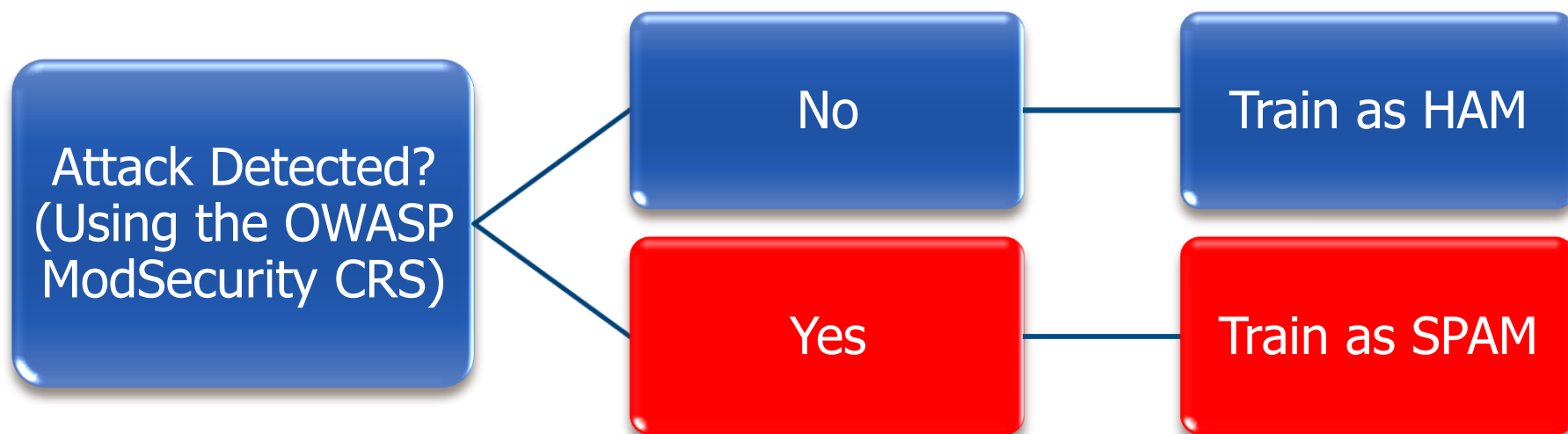
- RegEx detection is binary
 - The operator either matched or it didn't
 - Need a method of detecting ***attack probability***
- Bayesian analysis has achieved great results in Anti-SPAM efforts for email
- Can't we use the same detection logic for HTTP data?
 - Data Source
 - Email – OS level text files
 - HTTP – text taken directly from HTTP transaction
 - Data Format
 - Email – Mime headers + Email body
 - HTTP – URI + Request Headers + Parameters
 - Data Classification
 - Non-malicious HTTP request = HAM
 - HTTP Attack payloads = SPAM

OSBF-Lua

- OSBF-Lua by Fidelis Assis
 - Orthogonal Sparse Bigrams with Confidence Factor (OSBF)
 - Uses space characters for tokenization (which means that it factors in meta-characters)
 - Very fast
 - Accurate classifiers
 - <http://osbf-lua.luaforge.net/>
- Moonfilter by Christian Siefkes
 - Wrapper script for OSBF
 - <http://www.siefkes.net/software/moonfilter/>
- Integrate with ModSecurity's Lua API



Training the OSBF Classifiers



Theory of Operation - HAM

1. Non-malicious user data does not trigger any blacklist rules
2. Lua script trains OSBF classifier that payloads are HAM

Lua: Executing script: /etc/httpd/modsecurity.d/bayes_train_ham.lua

Arg Name: ARGS:txtFirstName and Arg Value: Bob.

Arg Name: ARGS:txtLastName and Arg Value: Smith.

Arg Name: ARGS:txtSocialSecurityNo and Arg Value: 123-12-9045.

Arg Name: ARGS:txtDOB and Arg Value: 1958-12-12.

Arg Name: ARGS:txtAddress and Arg Value: 123 Someplace Dr..

Arg Name: ARGS:txtCity and Arg Value: Fairfax.

Arg Name: ARGS:drpState and Arg Value: VA.

Arg Name: ARGS:txtTelephoneNo and Arg Value: 703-794-2222.

Arg Name: ARGS:txtEmail and Arg Value: bob.smith@mail.com.

Arg Name: ARGS:txtAnnualIncome and Arg Value: \$90,000.

Arg Name: ARGS:drpLoanType and Arg Value: Car.

Arg Name: ARGS:sendbutton1 and Arg Value: Submit.

Low Bayesian Score: . Training payloads as non-malicious.

Theory of Operation - SPAM

1. Attacker sends malicious payloads during initial testing phase
2. Payloads are caught by our blacklist rules
3. Lua script trains OSBF classifier that payloads are SPAM

```
[Thu Nov 03 15:21:08 2011] [error] [client  
72.192.214.223] ModSecurity: Warning. Pattern match  
".*" at TX:981231-WEB_ATTACK/SQL_INJECTION-  
ARGS:artist. [file  
"/etc/httpd/modsecurity.d/crs/base_rules/modsecurity_c  
rs_48_bayes_analysis.conf"] [line "1"] [data  
"Completed Bayesian Training on SQLi Payload: @@new  
union#sqlmapsqlmapsqlmapsqlmapsqlmapsqlmapsqlmap  
sqlmapsqlmapsqlmapsqlmapsqlmapsqlmapsqlmapsql  
\x0aselect 1,2,database#sqlmap\x0a() ."] [hostname  
"www.modsecurity.org"] [uri  
"/testphp.vulnweb.com/artists.php"] [unique_id  
"VCq1xsCo8AoAADYJV3kAAAAH"]
```

Theory of Operation - Unknown

- Previous evasion payload is now caught

```
[Thu Nov 03 15:28:18 2011] [error] [client 72.192.214.223]
ModSecurity: Warning. Bayesian Analysis Alert for
ARGS:artist with payload: "@@new
union#sqlmapsqlmapsqlmapsqlmapsqlmapsqlmapsqlmapsqlmapsqlmap
sqlmapsqlmapsqlmapsqlmapsqlmapsqlmapsqlmapsqlmapsqlmap
1,2,database#sqlmap\n()" [file
"/etc/httpd/modsecurity.d/crs/base_rules/modsecurity_crs_48_
bayes_analysis.conf"] [line "3"] [msg "Bayesian Analysis
Detects Probable SQLi Attack."] [data "Score:
{prob=0.99999999965698,probs={0.99999999965698,3.43018986145
48e-
10},class=\\x22/var/log/httpd/spam\\x22,pR=5.5841622861233,r
einforce=true}"] [severity "CRITICAL"] [tag
"WEB_ATTACK/SQL_INJECTION"] [tag "WASCTC/WASC-19"] [tag
"OWASP TOP 10/A1"] [tag "OWASP AppSensor/CIE1"] [tag
"PCI/6.5.2"] [hostname "www.modsecurity.org"] [uri
"/testphp.vulnweb.com/artists.php"] [unique_id
"bcjElMCo8AoAADYlSXMAAAAI"]
```

Bayesian Alert for Evasion Payload

The screenshot shows a web browser window with the title "artists". The address bar contains the URL "www.modsecurity.org/testphp.vulnweb.com/artists.php?artist=%40%40new". The main content area displays a red alert message:

ModSecurity Alert Message:
Inbound Alert: Bayesian Analysis Detects Probable SQLi Attack.
Outbound Alert: Bayesian Analysis Detects Probable SQLi Attack.
TX ID: Z6sARsCo8AoAADYdGfgAAAAA

Below the alert, there is a logo for "acunetix acuart" and a text box stating "TEST and Demonstration site for Acunetix Web Vulnerability Scanner". A navigation menu includes links for "home", "categories", "artists", "disclaimer", "your cart", "guestbook", and "AJAX Demo". A search bar is present with the text "search art" and a "go" button. The main content area shows "artist: 2" with a list of items: "acuart", "view pictures of the artist", and "comment on this artist". A sidebar on the left contains "Browse categories", "Browse artists", "Your cart", "Signup", "Your profile", "Our guestbook", "AJAX Demo", "Links", "Security art", and "Fractal Explorer".

Conclusion

Development Plans/Call for Assistance

- This proof of concept will eventually be put into the OWASP ModSecurity CRS
 - Other projects may consider using it too (AppSensor, ESAPI, etc...)
- Need to include HTTP Header data in training
 - For accurate Bayesian classification, more data is better.
 - Including HTTP Header data may also help to identify non-browser/tool attacks
- Need more testing
 - If you would like to help with testing, please contact me and I will provide you access to the Lua scripts.

ModSecurity T-Shirt Giveaway

- What was the shortest “Time-to-Evasion” from Level II?
- 10 hrs.



Contact/Resources

- Email
 - OWASP: ryan.barnett@owasp.org
 - Trustwave: rbarnett@trustwave.com
- Twitter
 - @ryancbarnett
 - @ModSecurity
 - @SpiderLabs
- Blog
 - <http://tacticalwebappsec.blogspot.com>
 - <http://blog.spiderlabs.com>