Web Application Incident Response & Forensics: A Whole New Ball Game!

Chuck Willis  chuck.willis@mandiant.com
Rohyt Belani  rohyt.belani@mandiant.com

Black Hat Briefings USA 2006
August 2, 2006
Founded in February 2004

Full spectrum information security company:
- Professional Services
- Education
- Software

Services include:
- Application Security
- Network Security
- Incident Response
- Computer Forensics

Located in Alexandria, VA & NYC

Former US military investigators and industry security professionals
Why Are We Here?

- “They” say that attacks against web applications are on the rise
- “We” see it – 70% of the attacks we have responded to in the last year have been against web applications
- Responding to such attacks is different
  - Need to understand application security
  - Need to look elsewhere for evidence
Agenda

- Background
- How web application incident response and forensics is different
- Case Studies
- Log discovery, review, and analysis
  - Web Server
  - Application Server
  - Database
- Remediation
Background
Three Tier Web Application

These servers may be independent or may run on the same machine.
Standard Incident Response & Forensics

- Capture volatile data
  - Processes
  - Ports and network connections
  - Memory dumps
  - Logged in users
- Perhaps capture some non-volatile data
  - Event logs
  - File listing and timestamps
- Shutdown system
- Make forensic image
Analyze image with forensic tools
- Examine file timestamps
- Check for known malicious software
- Examine deleted files
- Conduct string searches
- Carve files based on headers
How Does Web App Forensics Differ?

Let’s find out...
Why Standard Process Doesn’t Work

- Web applications are often distributed across multiple servers
- Web applications are often business critical and downtime for imaging may not be allowed
- Database servers usually have large disk arrays
- Web application attacks usually do not leave evidence in the same places as other attacks
- Web application forensics and incident response requires a solid understanding of web application security issues – not a conventional “forensicator” skill
Web Application Forensics Overview

- Understand the “normal” flow of the application
- Review log files:
  - Web Server
  - Application Server
  - Database Server
  - Application
- Capture application and server configuration files
- Identify potential anomalies:
  - Malicious input from client
  - Breaks in normal web access trends
  - Unusual referrers
  - Mid-session changes to cookie values
- Determine a remediation plan
A Report from the Trenches - Case #1
Symptoms

- “I see a trade executed from my account …10000 shares of a company I haven’t even heard about, were purchased on January 17 (2006) @ 2 pm from my account!” – a client of a well-established brokerage firm in NYC

- 7 other clients of the same brokerage firm report the same issue – in January 2006
Investigation

- Computer security breaches were the prime suspect

- Was the brokerage firm hacked? Was it the end user who was hacked?

- We had dates and times of the trade executions as a clue
Investigation

- Our team began reviewing the brokerage firm’s online trading application for clues
  - Network logs
  - Web server logs
  - Security mechanisms of the application

- We asked to duplicate the victim’s hard drive and review it for indicators of compromise
Web Server Logs

- Requested IIS logs for January 17, 2006 from all the (load balanced) servers

- Combined the log files into one common repository = 1 GB

- Microsoft’s Log Parser to the rescue
Microsoft LogParser

- LogParser is an excellent and free tool for analyzing log files
- Available from www.microsoft.com
- More information on unofficial LogParser support site: http://www.logparser.com/
- Supports a variety of log formats
- Uses SQL syntax to process log files
Microsoft LogParser

- Parsed out all requests to execute.asp using Microsoft Log Parser:

```
LogParser -o:csv "select * INTO execute.csv from *.log where cs-uri-stem like '/execute.asp%'"
```
Can You Find The Smoking Gun?

Software: Microsoft Internet Information Services 5.0
Version: 1.0
Date: 2006-01-01 01:03:15

<table>
<thead>
<tr>
<th>Fields: time</th>
<th>c-ip</th>
<th>cs-method</th>
<th>cs-uri-stem</th>
<th>cs-uri-query</th>
<th>Status</th>
<th>version</th>
</tr>
</thead>
<tbody>
<tr>
<td>1:03:15</td>
<td>172.16.22.33</td>
<td>POST</td>
<td>/execute.asp</td>
<td>sessionid=90198e1525e4b03797f833ff4320af39</td>
<td>200</td>
<td>HTTP/1.0</td>
</tr>
<tr>
<td>1:04:35</td>
<td>172.16.54.33</td>
<td>POST</td>
<td>/execute.asp</td>
<td>sessionid=3840943093874b3484c3839de9340494</td>
<td>200</td>
<td>HTTP/1.0</td>
</tr>
<tr>
<td>1:08:15</td>
<td>172.16.22.33</td>
<td>POST</td>
<td>/execute.asp</td>
<td>sessionid=90198e1525e4b03797f833ff4320af39</td>
<td>200</td>
<td>HTTP/1.0</td>
</tr>
<tr>
<td>1:10:19</td>
<td>172.16.87.231</td>
<td>POST</td>
<td>/execute.asp</td>
<td>sessionid=298230e0393bc09849d839209883993</td>
<td>200</td>
<td>HTTP/1.0</td>
</tr>
<tr>
<td>1:13:15</td>
<td>172.16.22.33</td>
<td>POST</td>
<td>/execute.asp</td>
<td>sessionid=90198e1525e4b03797f833ff4320af39</td>
<td>200</td>
<td>HTTP/1.0</td>
</tr>
<tr>
<td>1:18:15</td>
<td>172.16.22.33</td>
<td>POST</td>
<td>/execute.asp</td>
<td>sessionid=90198e1525e4b03797f833ff4320af39</td>
<td>200</td>
<td>HTTP/1.0</td>
</tr>
<tr>
<td>1:19:20</td>
<td>172.16.121.3</td>
<td>POST</td>
<td>/execute.asp</td>
<td>sessionid=676db8787ab0393898de0398348c89</td>
<td>200</td>
<td>HTTP/1.0</td>
</tr>
<tr>
<td>1:21:43</td>
<td>172.16.41.53</td>
<td>POST</td>
<td>/execute.asp</td>
<td>sessionid=3840943093874b3484c3839de9340494</td>
<td>200</td>
<td>HTTP/1.0</td>
</tr>
<tr>
<td>1:23:16</td>
<td>172.16.22.33</td>
<td>POST</td>
<td>/execute.asp</td>
<td>sessionid=90198e1525e4b03797f833ff4320af39</td>
<td>200</td>
<td>HTTP/1.0</td>
</tr>
<tr>
<td>1:28:15</td>
<td>172.16.22.33</td>
<td>POST</td>
<td>/execute.asp</td>
<td>sessionid=90198e1525e4b03797f833ff4320af39</td>
<td>200</td>
<td>HTTP/1.0</td>
</tr>
</tbody>
</table>
Next Step

- Noticed repeated use of same sessionid at regular intervals from the same IP

- Parsed out all requests with the suspicious sessionid

LogParser -o:csv "select * INTO sessionid.csv from *.log where cs-uri-query like '%90198e1525e4b03797f833ff4320af39'"
Can You Find The Smoking Gun?

#Software: Microsoft Internet Information Services 5.0
#Version: 1.0
#Date: 2006-01-017 01:03:15

<table>
<thead>
<tr>
<th>Fields: time</th>
<th>c-ip</th>
<th>cs-method</th>
<th>cs-uri-stem</th>
<th>cs-uri-query</th>
<th>Status</th>
<th>version</th>
</tr>
</thead>
<tbody>
<tr>
<td>1:03:15</td>
<td>172.16.22.33</td>
<td>POST</td>
<td>/execute.asp</td>
<td>sessionid=90198e1525e4b03797f833ff4320af39</td>
<td>200</td>
<td>HTTP/1.0</td>
</tr>
<tr>
<td>1:08:15</td>
<td>172.16.22.33</td>
<td>POST</td>
<td>/execute.asp</td>
<td>sessionid=90198e1525e4b03797f833ff4320af39</td>
<td>200</td>
<td>HTTP/1.0</td>
</tr>
<tr>
<td>1:13:15</td>
<td>172.16.22.33</td>
<td>POST</td>
<td>/execute.asp</td>
<td>sessionid=90198e1525e4b03797f833ff4320af39</td>
<td>200</td>
<td>HTTP/1.0</td>
</tr>
<tr>
<td>1:18:15</td>
<td>172.16.22.33</td>
<td>POST</td>
<td>/execute.asp</td>
<td>sessionid=90198e1525e4b03797f833ff4320af39</td>
<td>200</td>
<td>HTTP/1.0</td>
</tr>
<tr>
<td>1:23:16</td>
<td>172.16.22.33</td>
<td>POST</td>
<td>/execute.asp</td>
<td>sessionid=90198e1525e4b03797f833ff4320af39</td>
<td>200</td>
<td>HTTP/1.0</td>
</tr>
<tr>
<td>1:28:15</td>
<td>172.16.22.33</td>
<td>POST</td>
<td>/execute.asp</td>
<td>sessionid=90198e1525e4b03797f833ff4320af39</td>
<td>200</td>
<td>HTTP/1.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13:53:15</td>
<td>172.16.22.33</td>
<td>POST</td>
<td>/execute.asp</td>
<td>sessionid=90198e1525e4b03797f833ff4320af39</td>
<td>200</td>
<td>HTTP/1.0</td>
</tr>
<tr>
<td>13:58:15</td>
<td>172.16.22.33</td>
<td>POST</td>
<td>/execute.asp</td>
<td>sessionid=90198e1525e4b03797f833ff4320af39</td>
<td>200</td>
<td>HTTP/1.0</td>
</tr>
<tr>
<td>14:03:15</td>
<td>172.16.22.33</td>
<td>POST</td>
<td>/execute.asp</td>
<td>sessionid=90198e1525e4b03797f833ff4320af39</td>
<td>200</td>
<td>HTTP/1.0</td>
</tr>
<tr>
<td>14:07:23</td>
<td>172.16.14.166</td>
<td>POST</td>
<td>/login.asp</td>
<td>sessionid=90198e1525e4b03797f833ff4320af39</td>
<td>200</td>
<td>HTTP/1.0</td>
</tr>
<tr>
<td>14:07:54</td>
<td>172.16.14.166</td>
<td>POST</td>
<td>/account.asp</td>
<td>sessionid=90198e1525e4b03797f833ff4320af39</td>
<td>200</td>
<td>HTTP/1.0</td>
</tr>
<tr>
<td>14:08:15</td>
<td>172.16.22.33</td>
<td>POST</td>
<td>/execute.asp</td>
<td>sessionid=90198e1525e4b03797f833ff4320af39</td>
<td>200</td>
<td>HTTP/1.0</td>
</tr>
<tr>
<td>14:10:09</td>
<td>172.16.22.33</td>
<td>POST</td>
<td>/confirm.asp</td>
<td>sessionid=90198e1525e4b03797f833ff4320af39</td>
<td>200</td>
<td>HTTP/1.0</td>
</tr>
</tbody>
</table>
Phishing?

- No indications of key logging trojans, malware, viruses, etc. were found on the victim’s computer.
- Look what we found in the archived .pst file:

```
From: A.customer-service@xyz.com
To: 
Cc: 
Subject: Valued Customer Feedback

At we are always striving to improve the customer’s online experience. We are currently experimenting with a new user interface and have selected a few of our valued customers to provide feedback on it.

We would appreciate it if you, Mr., would review the changes by logging into your account by clicking here and sending us e-mail telling us what you like and what you don’t about what you see.

We appreciate your participation in this process.
Thank You once again,

Customer Service
```

URL: https://www.xyzbrokerage.com/login.asp?sessionid=90198e1525e4b03797f833ff4320af39
The application was confirmed to be vulnerable to session fixation:

• A session id was issued before login
• The same session id was used by the application after login for the purposes of user authorization
• This allowed an attacker to hijack legitimate user sessions using a bit of social engineering
Web Server Logs
IIS 6.0

- Default logs are plain text in W3C Extended log file format
- Logs stored in LogFiles\W3SVCx
- Easily parsed with text parsing tools or with LogParser
- Log files can capture cookies and referrer headers
- Cannot log HTTP POST data
IIS 6.0 – Logged by Default

- Date / Time
- Client IP
- Server Info
- HTTP Method
- URL and Parameters
- HTTP Status Code
- User Agent
IIS 6.0 – Not Logged by Default

Can be enabled:
- Transfer Sizes
- Host Header
- Cookies
- Referrer

Not even an option…
- POST Data
Why Do We Care About POST Data?

- Much of the user input to a web application is passed to the server as POST parameters.
- Manipulating these parameters is the prime mechanism for attacking an application.
- POST data logging provides insight into such attacks.
- POST data is necessary to perform an accurate damage assessment.
What is the Referrer header?

Referrer headers are an indicator of browsing flow.

Can be used to identify abnormal browsing trends that may be indicative of an attack.

Not a reliable measure.

Referrer spoofing is easy and results in false positives.
May 2006

Multi-national food and beverages company requested bids for a machinery maintenance contract

The bids were to be provided over the Web

One of the bidders appeared to have inside knowledge

Chief counsel ordered an investigation
Cookie Crunching

- Application authorized requests based on the “uid” cookie
- Reviewed IIS 6.0 server logs
- Server was configured to log cookies
- Parsed all requests to bid.aspx
- Multiple requests from the same IP address with different uid cookies
- Whois on the IP address revealed the culprit
- Cookie logging saved the day!
URLScan

- URLScan is a free IIS filter from Microsoft that can prevent some types of HTTP requests from making it to the web server.
- If URLScan is in use, the logs will include details on blocked requests.
- Logs are stored by default in the same directory as URLScan.
- Automated attacks can often be detected by reviewing URLScan logs.
Apache Web Server Logs

- Log format and locations are highly customizable
- Log configuration set in httpd.conf
- Access log – records all requests
  - access.log on Windows, access_log on Unix
- Error log – holds diagnostic and error messages
  - error.log / error_log
- Some modules have their own logs:
  - rewrite.log
Apache Logs – Default Access Log

- LogFormat "%h %l %u %t "%r" %>s %b"
  - Remote Host
  - Remote logname (from identd)
  - Remote user (from HTTP authentication)
  - Time
  - First line of request
  - Status
  - Bytes sent

- mod_log_config can be used to enhance Apache logging to capture additional fields
Application Server Logs
Application Server Logs

- Application servers will log data
- Logged events will include:
  - Unhandled application exceptions
  - Application errors
  - Loader problems (references to classes that are not available)
  - Other implementation dependent items
  - Some messages from applications
ASP.NET Application Server

- ASP.NET does not maintain its own log files
- Errors and unhandled exceptions are logged to the Windows event logs
- In .NET 2.0, an unhandled exception will halt the application by default
BEA WebLogic

- BEA WebLogic is a common Java application server and HTTP server
- Maintains a variety of logs:
  - Server Log
    - Messages and errors from the server, applications and subsystems
    - ServerName/ServerName.log
  - Domain Log
    - Messages forwarded from the servers in the domain
    - Not all messages are forwarded or logged at the domain level
    - DomainName.log
Other logs that may be present:

- HTTP Log – similar to Apache access log
- Node Manager Logs – startup and status messages
- Standard Output – Messages from the server and also from the applications
- Standard Error
- Java Transaction API (JTA) Logs
- Java Database Connectivity (JDBC) Logs
IBM’s WebSphere Application Server is another common Java App Server

Logs created by WebSphere:
- Apache Web Server Logs
  - Access Log
  - Error Log
- IBM Service Log
  - Logs events for servers under a node
  - File name is activity.log
  - Log is binary data – use showlog script to convert
WebSphere Application Server

- Stream logs on WebSphere:
  - JVM logs – streams from Java code
    - SystemOut.log
    - SystemErr.log
  - Process logs – streams from native code
    - native_stdout.log
    - native_stderr.log
A Report from the Trenches - Case #2
Symptoms

- The CEO of a retail organization received an extortion threat of $250,000 via snail mail
- The threat – 125,000 customer credit card numbers would be sold to the mafia
- The response was demanded in the form of a footer on the main page of the retailer’s website
Response

- In-house counsel used several ploys to buy time – a mere 72 hours were granted by the extorter
- 3 members of our team were brought in to investigate round the clock for the next 3 days
- Our job was to determine how the credit card database may have been compromised and more importantly who was the culprit
What Followed?

- Frenzied web server log analysis to detect anomalous activity – Nothing!
- Reviewed all employee email inboxes to detect internal fraud – Nothing!
- Database login/logout activity reviewed – nothing suspicious
- Web application scanned for SQL injection flaws – No luck!
- Last resort – application code review
Racing Against Time

- Over 100,000 lines of code
- A comprehensive code review was ruled out
- Resorted to scripted searches through code
Scripted Searches

- Did the code contain raw SQL statements?
- Searched for occurrences of the “SELECT” in the code

Regex = .*SELECT.*

- The search resulted in an overwhelming number of hits
Scripted Searches

- The results from the previous search were searched for occurrences of the “SELECT *” string to identify SQL statements where the scope was not properly limited

  Regex =  \*.*FROM.*

- The search resulted in 5 hits
- One of the hits was:

  SELECT * FROM CardTable
The Code That Made The Call

NameValueCollection coll = Request.QueryString;
String[] arr1 = coll.AllKeys;
...
String[] arr5 = coll.getValues(arr1[4]);
string extra = Server.HtmlEncode(arr5[0]).ToString();

if (extra.Equals("letmein"))
{
    Cmd = "SELECT * FROM CardTable";
}

...
Eureka!

- This was a backdoor – an insider job?
- Reviewed code archives to detect addition of code
- The first check-in with this code was made by a developer contracted from a third-party in Asia
- Found the URL with the additional parameter in the web server logs
- The client IP traced back to Asia!
Another One Bites The Dust…

- The development company was notified of this rogue activity
- Local law enforcement was cooperative
Post Mortem

- What could have been done better:
  - Encryption of sensitive info in the DB
  - More advanced DB logging
  - Security reviews of code
Database Server Logging
Database Server Logging

- Common databases have little or no logging enabled by default
- Logging of additional database events can be enabled
- Table or data specific logging can be accomplished with database triggers
Captures login/logout and other activity in the Windows Application Log

ErrorLog file – server errors and other messages
  • New log created on DB startup
  • By default, 6 previous logs are stored

Server-Side Traces can be used for fine-grained auditing
MS SQL C2 Auditing – Advantages

- Records detailed information
  - Execution of stored procedures
  - Creation or deletion of objects like tables
  - Querying of tables
  - Permission changes

- Logs stored in .trc files that can be viewed using SQL Server Profiler
MS SQL C2 Auditing – Disadvantages

- Databases and audit logs share the same directory
- C2 auditing affects SQL server performance
- If the disk is full and C2 log cannot be written SQL server execution is halted
- C2 auditing is not practical as a long-term solution
Oracle Database Auditing

- Events logged to the OS log by default:
  - Instance startup and shutdown
  - Connections to DB with administrator privileges
- Additional auditing of database events can be enabled
- Additional audit entries can be stored in a database table or in the OS Log
Oracle Database Auditing

- **Audit.log**
  - Flat text file
  - Records important information about the database operation
  - Records errors
  - References to trace files and dump files

- **Trace files can result from:**
  - An error in a background process
  - Administrator action
Application Logging
Application Level Logging

- Application logs can provide key information
  - Detailed knowledge of business logic
  - Good signal to noise ratio
- Ask developers or administrators:
  - Where are application logs?
  - What is format?
  - What messages would result from likely malicious activity?
  - How long are logs stored?
Application should log these events:

- Invalid Input
  - SQL Injection Attempts
  - Cross Site Scripting Attempts
- Failed Authentication
- Authorization Failures
- Session Tracking Problems
- Critical portions of business logic
Application Level Logging

- Application should log this information:
  - Server Identity
  - Client IP Address
  - Username
  - Date/Time
  - URL
  - POST data
  - Cookies
Logging Frameworks

- Logging frameworks provide an easy way for developers to implement and configure logging
- Common logging frameworks:
  - Log4j / Log4net / Log4PLSQL
  - Java’s java.util.logging
  - The Object Guy’s dotnetlog / javalog
Remediation
Remediation

- When web application analysis is exhausted, need to determine if a standard forensic analysis is warranted
- Need to determine a remediation plan:
  - Recover from current state
  - Restore from backup
  - Rebuild from scratch
- Ensure that causes of the incident are addressed
Conclusion

- Application forensics requires a concerted effort between system administrators, network administrators, security staff and developers
- Responders need to be intimately familiar with application security issues
- Enhance your forensics and incident response checklists
- There is no one right way!
Questions?

Chuck Willis    chuck.willis@mandiant.com
Rohyt Belani    rohyt.belani@mandiant.com