Spotlights on the Digital Battlefield

WEB: UNDER (THE) COVER

"Better to light a candle than to curse the darkness."

Chinese Proverb

Case Study: Reconnaissance Leaks & Corporate Assets

- Hacking is not always malicious
  - Stealing credit card numbers
  - Defacing a Web site
  - Denial of service attacks
- Sometimes, hackers want information
  - Sales projections
  - Personnel records
  - Corporate "proprietary" information

Case Study: Reconnaissance Leaks & Corporate Assets

- `Example.com` is an on-line video store
- They have a new Web site that has been described as “unbreakable”!
- This is a challenge to most hackers!
- Remember, the hackers only tools (at least at this point) are the browser and the URL
- What can the attacker gather from a simple URL?
- Let’s load the page and find out…

Slide 4

Case Study: Reconnaissance Leaks & Corporate Assets

```
http://www.example.com/
  load.cgi?file=main.dhtml
```

- Loading the home page gives the hacker some information:
  - Site is using some form of CGI (`load.cgi`)
  - Dynamic HTML is being used (`main.dhtml`)
  - Because the parameters are shown in the URL, the programmer used GET requests

Slide 5

Case Study: Reconnaissance Leaks & Corporate Assets

```
http://www.example.com/load.cgi?file=main.dhtml
```

- Check to see if the programmer did a good job at input validation by putting a known file (`load.cgi`) in the `file=` parameter

```
http://www.example.com/load.cgi?file=load.cgi
```

- Now, you can load any file on the system!
  - Try `robots.txt` (used by crawlers to exclude files and directories from a crawl)
  - From that, the hacker sees a `Forecasts` directory

Slide 6

Case Study: Reconnaissance Leaks & Corporate Assets

- The hacker tries a few obvious attempts; none work

```
http://www.example.com/Forcasts/index.dhtml
http://www.example.com/Forcasts/index.html
http://www.example.com/Forcasts/load.cgi
```

Slide 7

Case Study: Reconnaissance Leaks & Corporate Assets

- The hacker uses Teleport Pro (a Web crawler) to mirror the Web site
- Discovers pdf copies of company quarterly report in the Web server *document root* tree!
- Hacker downloads them…

```
http://www.example.com/Forcasts/q1-09.pdf
http://www.example.com/Forcasts/q2-09.pdf
http://www.example.com/Forcasts/q3-09.pdf
```

- Reads Q3 results and sells his stock!

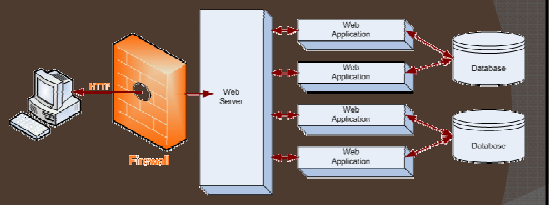
Slide 8

Components of a Web Application

- In order to understand how to secure your Web site, you need to know how Web site components fit together
- A typical Web application has 3 main components:
  1. Front-end Web server
  2. Web application execution environment
  3. Database server
- We'll consider each component separately…

Slide 9

Components of a Web Application



Slide 10

The Front-end Web Server

- The 4 most popular Web servers:
  - Apache (73%)
  - Microsoft IIS (20%)
  - Zeus Web server (0.5%)
  - Netscape / iPlanet server (0.28%)
- The Front-end Web server is:
  - An HTTP receiver and HTTP replier
  - High throughput and efficient
  - Able to handle many concurrent connections

Chapter 6 –Web: Under (the) Cover - V1.0.0

Slide 11

The Front-end Web Server

- Desirable characteristics of a Front-end Web server:
  - Scalable and robust
  - Tried and tested against known attacks
  - Able to handle high loads and concurrent connections
  - Versatile configuration facilities
  - API's or plug-in support for integrating external components

Slide 12

The WAEE = Application Server

- WAEE = Web Application Execution Environment
- The WAEE is a platform for writing customized apps. that receive input from Web forms or URL's and generate HTML output dynamically
- Sometimes referred to as an *Application Server*
- Can be very simple or very complex
- Contains a "language" used to program the environment
- Ex: Perl/CGI, PHP, ASP/VB, ASP.NET, JSP

Slide 13

Which WAEE?

- When deciding on a WAEE, keep in mind:
  1. Suitability of task – Do you really need .NET? Or will Perl/CGI due?
  2. Front-end Web Server Interface – Must interface easily with the Web Server
  3. Database Interfaces – Must interface with the back-end database: Oracle, DB2, SQL Server, MySQL, etc.
  4. Are there "corporate commitments"?

Slide 14

The Database Server

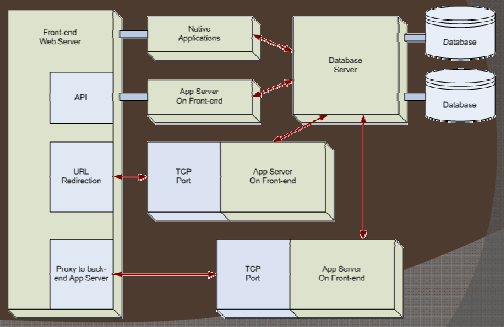
- The Database Server is the Web App's back-end data store
- The back-end DB contains all of the business logic and all of the business data
- The interface is usually an API that allows the WAEE access to the DB
- All interaction with the DB Server is via SQL

Wiring the Components

- Now that we know what the components are, how can we "wire" them together?
- Many different ways to do it…
- There are 4 common schemes for interfacing components:
  1. The native application processing environment
  2. Web server API's and plug-ins
  3. URL mapping & internal proxying
  4. Proxying with a back-end app server

Slide 16

Wiring the Components



Slide 17

The Native App Processing Env

- Using this scheme, developers write apps using the env's scripting language:
  - IIS → ASP/VB
  - IIS → ASP.NET (C#, VB.NET, J#)
  - Apache → PHP
  - Apache → Perl/CGI
  - Apache/Tomcat → JSP

Slide 18

Web Server API's & Plug-ins

- With this scheme, developers write apps by using API libraries
- API's provided by Web server and become an extension of the server
- Examples:
  - Apache Dynamic Shared Objects (DSO)
  - Microsoft Internet Server API (ISAPI)
  - Netscape NSAPI
  - NewAtlanta Inc.'s ServletExec (Java, EJBean, JSP) extension for Apache or IIS

Slide 19

URL Mapping & Internal Proxying

- With this method, the App server operates independently of the front-end Web server
- The App server is its own HTTP server listening on a different port from the Web server
- The Web server is configured to map certain URL's over to the App server
- Allows for a more generic interface
- Tradeoff: performance

Slide 20

Proxying with a Back-end App Server

- With this method, the App server runs on a separate system with an internal IP address
- The App server only accessible via the front-end Web server
- The Web server acts as a proxy server to the back-end App server
- This method is an extension of URL mapping allowing the App server to be hosted by more than 1 system

Slide 21

Connecting with the Database

- In the 70's & 80's
  - DBMS were colossal beasts
  - Apps were written in proprietary languages
- Today
  - SQL standardized the way data was retrieved from a DBMS
  - Client-Server segregated db from apps
  - Apps developed in some language, db API provides a connection and query capability
- 3 most popular ways of connecting to DB: Native DB API's, ODBC & JDBC

Slide 22

Native DB API's

- Web App languages have programming API's that provide
  - Db connection
  - Querying ability
  - Result set parsing
- The API's are provided by the DB Co.

Slide 23

Native DB API's

- Example: Apache/PHP/MySQL

```
$link_id = mysql_connect ( $dbhost, $dbusername,
                           $dbuserpassword );

$query = "select * from tblStudent order by
  $sortMethod";

$result = mysql_query ( $query );

while ( $row = mysql_fetch_array ( $result ) )
{ … }
```

Slide 24

Using ODBC / JDBC

- Open Database Connectivity (ODBC)
- A standard of universal methods for connecting to any DB
- The OBDC driver is the “go between” from program to DB
- To change DB’s simply change the driver; everything else stays the same
- JDBC = ODBC for Java
- JDBC standardized along with J2EE

Slide 25

Specialized Web App Servers

- We’ve been discussing general purpose app servers
- Specialized app servers are used for very specific purposes:
  - Vignette’s StoryServer: geared toward delivering up-to-the-minute new stories on demand
  - IBM’s n-commerce and Net.Data provide interactive transaction processing systems

Slide 26

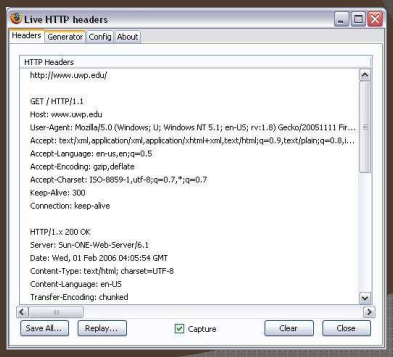
Identifying Web App Components from URL

- The URL, as we’ve seen, tells us much about the Web app systems
- As a hacker, the URL is the key to Web site infiltration
- Being able to id the different technologies in a Web app tells the hacker how and what to take advantage of

Slide 27

Basics of Technology Identification

- Step 1: Look at the HTTP header returned from the server
  - Many different programs that can help you view HTTP headers:
    - Netcat
    - LiveHttpHeaders
  - Some servers send back too much info in the HTTP response
  - Servers can (and should) be configured to return only the bare minimum



Basics of Technology Identification

- Step 2: Look at the type of resource or file being requested (most popular below)

Extension	Technology	Server Platform
.pl	Perl CGI script	Generic; usually running on UNIX
.asp	Active Server Page	Microsoft IIS
.aspx	ASP+	Microsoft .NET
.php	PHP script	Generic; usually Apache
.cfm	ColdFusion	Generic; usually Microsoft IIS
.nsf	Lotus Domino	Lotus Domino

Technology Identification: Examples

1. `http://www1.example.com/homepage.nsf?Open`

- The `.nsf` indicates a Lotus Domino server
- HTTP header confirms this

```
HTTP/1.1 302 Found
Server: Lotus-Domino/5.0.5
Date: Mon, 04 May 2009 17:52:59 GMT
Location: homepage.nsf?Open
Connection: Close
Content-Type: text/html
Content-Length: 305
```

Slide 31

Technology Identification: Examples

2. `http://www2.example.com/software/buy.jhtml;jsessionid=ZY ↵ QFN5WHKORD5QFIAE0SFF GAVAAUIIV0`

- The HTTP header identifies this site as *Microsoft-IIS/4.0*
- IIS does not support `.jhtml` natively
- This tells us IIS is working with an app server
- The "`;jsessionid=xxxx`" is indicative of a ATG Dynamo App server
- This server is a typical Java-based App server that serves up Java HTML as servlets

Slide 32

Technology Identification: Examples

3. `http://www3.example.com/cgi-bin/ncommerce3/ExecMacro/webstore/home.d2w/report`

- This URL is a typical of IBM's Net.Data
- The `ncommerce3` and the `ExecMacro` reveal the technology
- `home.d2w` is a macro written in Net.Data's scripting language
- `report` is a method of the macro that provides a reporting mechanism

Slide 33

Technology Identification: Examples

4. `http://www4.example.com/category.jsp?id=21&StoreSession=PC1…V7b|388…820/167838525/6/7001/7001/7002/7002/7001/-1`
   - The file being requested is a Java Server Page (`.jsp`)
   - The HTTP header reveals the underlying server is Netscape Enterprise Server 4.1
   - However, the URL "signature" is not typical of Netscape Enterprise Server

Slide 34

Technology Identification: Examples

4. `http://www4.example.com/category.jsp?id=21&StoreSession=PC1…V7b|388…820/167838525/6/7001/7001/7002/7002/7001/-1`
   - The URL "signature" is typical of BEA's WebLogic
     - First part is a string followed by a "|"
     - The 7001 & 7002 refer to the HTTP and SSL TCP ports

Slide 35

Technology Identification: Examples

5. `http://www5.example.com/site/index/0,10017,2578,00.html`
   - This URL has numbers separated by commas and a `.html` extension
   - URL typical of Vignette's Story Server
     - Content server used in conjunction with either Netscape Enterprise server of MS IIS.
     - Comments in the HTML confirm our detective work

Slide 36

Technology Identification: Examples

- Sometimes, the URL's themselves are deceptive and lead us to believe a different system is running
- Viewing the HTTP header can reveal a different underlying technology
- Also, servers that return cookies can expose its identity!

Slide 37

Technology Identification: Examples

- The following list shows examples of cookies returned by certain Web/App servers:

Apache	`Apache=202.86.136.115.308…729`
IIS	`ASPSESSIONIDGGQGGCVC=KEL…DME`
ATG Dynamo	`JSESSIONID=H4T…IVO`
IBMNet.Data	`SESSION_ID=30783,wFX…w8e/TUD…EZq`
ColdFusion	`CFID=573208, CFTOKEN=86241965`

Slide 3

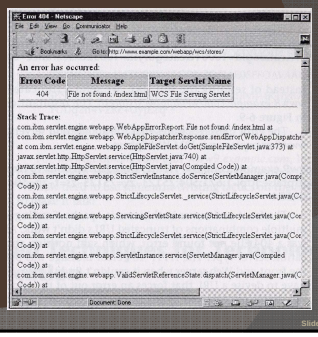
Adv. Techniques for Technology ID

- If previous stuff works don't work, what next?
- Force an error! Error messages from the server can give away:
  - The type of Web server & App server
  - Physical path mappings
  - References to files and libraries
  - Entire SQL queries!
- You can force errors in different ways:
  - Truncated URL's
  - Requests for non-existent files
  - Parameter tampering

Slide 39

Adv. Techniques for Technology ID

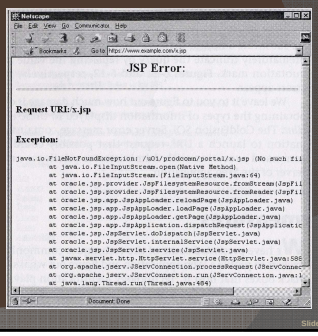
- This shows the result of a truncated URL…
- What can you tell me about this Web site?



Slide 40

Adv. Techniques for Technology ID

- This shows the result when requesting a non-existent file
- What can you tell me about this Web site?



Slide 41

Identifying DB Servers

- Quite a bit more tricky than identifying Web or App servers
- Usually, the only way to id a DB server is to get it to generate an error
- Consider:
  - `http://www.example.com/public/index.php?ID=27`
  - `http://www.example.org/Profile.cfm?id=3&page=1`
- How can we make these URL's generate errors?

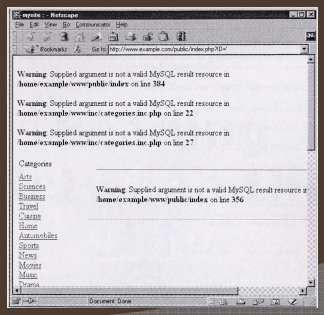
Slide 42

Identifying DB Servers

- Both URL's have these "ID=" in them
- We simply substitute "bad" values for these parameters:
  - First: `ID=qq` (some non-numeric value)
  - Second: `id=“` (truncate URL prematurely)
- The next slides show the results. Which DB is being used for each?

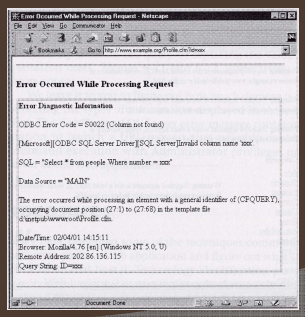
Slide 43

Identifying DB Servers



Slide 44

Identifying DB Servers



Slide 45

Countermeasures

- System identification is important to hackers because it lets them “choose their weapons”
- 2 General Rules of thumb:
  1. Minimize information leaked from the HTTP header
  2. Prevent error information from being returned to the client browser

Slide 46

Summary

- Understanding how each part of a Web site fits together is vital
- Hackers can piece together information about your systems from clues in the HTTP header and error information
- This lets them “choose their weapons”
- Keeping the amount of information leaked from HTTP and errors is important

Slide 47