Web Application Security

Lecture:03

Mapping the Application

Miss Maryam Malik
Lecturer
Department of cyber security
Air university Islamabad

Mapping

- **Enumerate application's content and functionality**
- **Some is hidden, requiring guesswork and luck to discover**
- **Examine every aspect of behavior, security mechanisms, and technologies**
- **Determine *attack surface* and *vulnerabilities***

Enumerating Content and Functionality

Web Spiders

- **Load web page, find all links on it**
  - **(into the targeted domain)**
- **Load those pages, find more links**
- **Continue until no new content is discovered**

Web Application Spiders

- **Also parse HTML forms**
  - **Fill in the forms with preset or random values and submit them**
  - **Trying to walk through multistage functionality**
- **Can also parse client-side JavaScript to extract URLs**
- **Tools: WebScarab, Zed Attack Proxy, and CAT**

Robots.txt

- **Intended to stop search engines**
- **May guide spiders to interesting content**

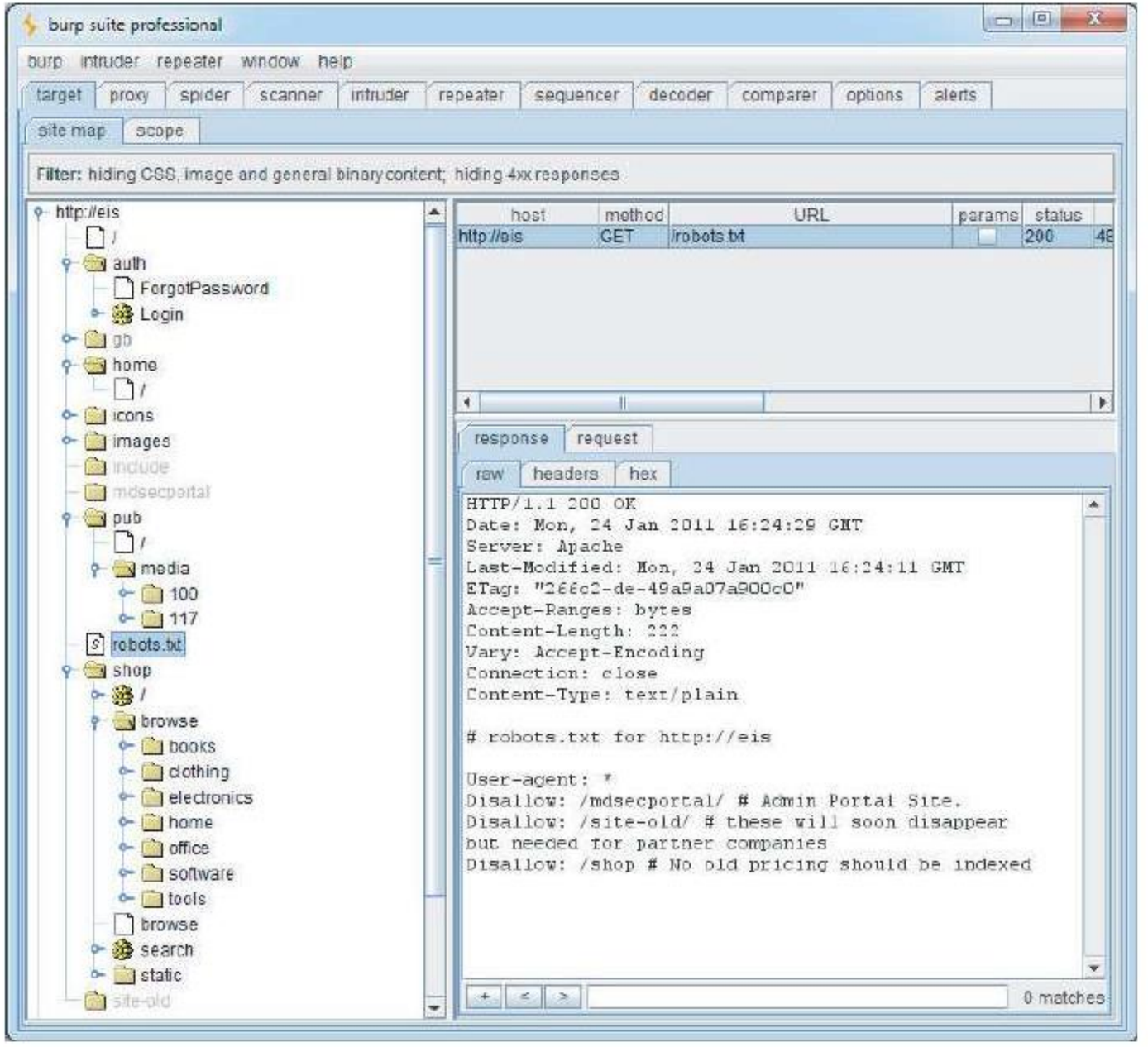


Figure 4-1: Mapping part of an application using Burp Spider

Limitations of Automatic Spidering

May fail to handle unusual navigation mechanisms, such as dynamically created JavaScript menus

- **So it may miss whole areas of an application**

Links buried in compiled client-side objects like Flash or Java may be missed

Limitations of Automatic Spidering

- **Forms may have validation checks, such as user registration forms**
  - **Email address, telephone number, address, zip code**
  - **Too complex for most spiders, which use a single text string for all form fields**
  - **Spider cannot understand the "Invalid" error messages**

Limitations of Automatic Spidering

- **Spiders only fetch each URL once**
  - **But applications use forms-based navigation, in which the same URL may return different content and functions**
  - **For example, a bank may implement every user action with POST to /account.jsp with parameters determining the action**
  - **Spiders aren't smart enough to handle that**

Example

View Account Balance Request:
Get/account.jsp?action=view_balance
The server returns the account balance.
Make a Transfer Request:
POST/account.jsp?action=make_transfer&amount=100&to_account=12345
The server processes the transfer request and deducts $100 from the user's account.
Pay a Bill Request:
POST /account.jsp?action=pay_bill&bill_id=789
The server processes the bill payment.

Limitations of Automatic Spidering

- **Some applications place volatile data within URLs**
  - **Parameters containing timers or random number seeds**
- **Spider will fetch the same page over and over, thinking it's new**
  - **May freeze up**

Example

/account.jsp?user_id=12345&timestamp=1634567891&random_seed=xyz987

/account.jsp?user_id=12345&timestamp=1634567891&random_seed=xyz988

/account.jsp?user_id=12345&timestamp=1634567892&random_seed=xyz989

All leads to same page

Limitations of Automatic Spidering

- **Authentication: spider must be able to submit valid credentials**
  - **Perhaps using a valid cookie**
- **However, spiders often break the authenticated session, by**
  - **Requesting a logout function**
  - **Submitting invalid input to a sensitive function**
  - **Requesting pages out-of-sequence**

Warning

- **Spiders may find an administrative page and click every link**
  - **Delete User, Shut Down Database, Restart Server...**

User-Directed Spidering

- **More sophisticated and controlled technique than automated spidering, usually preferable**
- **User walks through application using a browser connected to Burp (or another proxy)**
- **The proxy collects all requests and responses**

Advantages of User-Directed Spidering

- **User can follow unusual or complex navigation mechanisms**
- **User can enter valid data where needed**
- **User can log in as needed**
- **User can avoid dangerous functionality, such as deleteUser.jsp**

Browser Tools

- **Chrome's Developer Tools can show details of requests and responses within the browser**
- **No proxy needed**
- **Often useful; shows timing as well as content**

Discovering Hidden Content

- **Finding it requires automates testing, manual testing, and luck**
- **Testing or debugging features left in application**
- **Different functionality for different categories of users**
  - **Anonymous, authenticated, administrators**
- **Backup copies of live files**
  - **May be non-executable and reveal source code**

Discovering Hidden Content

- **Backup archives that contain snapshot of entire application**
- **New functionality implemented for testing but not yet linked from main application**
- **Default functionality in an off-the-shelf application that has been superficially hidden from the user but not removed**
- **Old versions of files--may still be exploitable**

Discovering Hidden Content

- **Configuration and include files containing sensitive data such as database credentials**
- **Source files from which application functions were compiled**
- **Comments in source code; may contain usernames and passwords, "test this" marks, and other useful data**
- **Log files--may contain valid usernames, session tokens, etc.**

Brute-Force Techniques

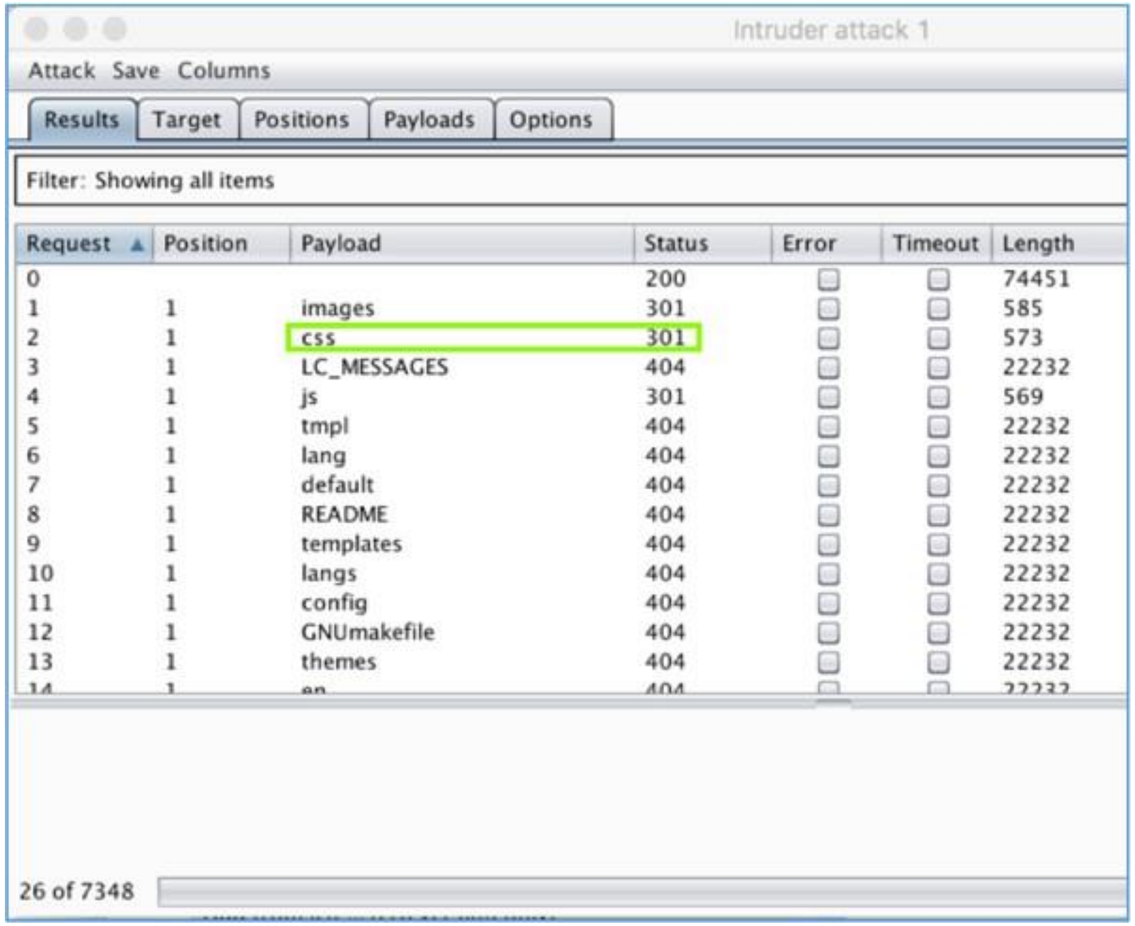
- **Suppose user-directed spidering finds the URLs on the left**
- **A brute-forcer will try names as shown on the right**

```
http://eis/auth/Login
http://eis/auth/ForgotPassword
http://eis/home/
http://eis/pub/media/100/view
http://eis/images/eis.gif
http://eis/include/eis.css
```

```
http://eis/About/
http://eis/abstract/
http://eis/academics/
http://eis/accessibility/
http://eis/accounts/
http://eis/action/
...
```

Burp's Brute-Forcer

- **Burp's brute-forcer is crippled in the free version**



Inference from Published Content

- **Look for patterns**
  - **All subdirectories of "auth" start with a capital letter**
  - **One is "ForgotPassword", so try these**

```
http://eis/auth/AddPassword
http://eis/auth/ForgotPassword
http://eis/auth/GetPassword
http://eis/auth/ResetPassword
http://eis/auth/RetrievePassword
http://eis/auth/UpdatePassword
...
```

Discovering Hidden Parameters

- **Try adding "debug=true" to requests**
  - **Or test, hide, source, etc.**
- **Burp Intruder can do this (see Ch 14)**

Analyzing the Application

- **Key areas**
  - **Core functionality**
  - **Peripheral behavior: off-site links, error messages, administrative and logging functions, and use of redirects**
  - **Core security mechanisms: session state, access control, authentication**
    - **User registration, password change, account recovery**

Key Areas (continued)

- **Everywhere the application processes user-supplied input**

  - **URL, query string, POST data, cookies**

- **Client-side technologies**

  - **Forms, scripts, thick-client components (Java applets, ActiveX controls, and Flash), and cookies**

Key Areas (continued)

- **Server-side technologies**
  - **Static and dynamic pages, request parameters, SSL, Web server software, interaction with databases, email systems, and other back-end components**

Entry Points for User Input

- Every URL string up to the query string marker
- Every parameter submitted within the URL query string
- Every parameter submitted within the body of a `POST` request
- Every cookie
- Every other HTTP header that the application might process — in particular, the `User-Agent`, `Referer`, `Accept`, `Accept-Language`, and `Host` headers

URL File Paths

- **RESTful URLs put parameters where folder names would go**

```
http://eis/shop/browse/electronics/iPhone3G/
```

In this example, the strings `electronics` and `iPhone3G` should be treated as parameters to store a search function.

Request Parameters

- **Here are some nonstandard parameter formats**

- `/dir/file;foo=bar&foo2=bar2`
- `/dir/file?foo=bar$foo2=bar2`
- `/dir/file/foo%3dbar%26foo2%3dbar2`
- `/dir/foo.bar/file`
- `/dir/foo=bar/file`
- `/dir/file?param=foo:bar`
- `/dir/file?data=%3cfoo%3ebar%3c%2ffoo%3e%3cfoo2%3ebar2%3c%2ffoo2%3e`

HTTP Headers

- **User-Agent is used to detect small screens**
  - **Sometimes to modify content to boost search engine rankings**
  - **May allow XSS and other injection attacks**
- **Changing User-Agent may reveal a different user interface**

HTTP Headers

- **Applications behind a load balancer or proxy may use X-Forwarded-For header to identify source**
- **Can be manipulated by attacker to inject content**

Out-of-Band Channels

- **User data may come in via**
  - **Email**
  - **Publishing content via HTTP from another server**
  - **IDS that sniffs traffic and puts it into a Web application**
  - **API interface for non-browser user agents, such as cell phone apps, and then shares data with the primary web application**

Identifying Server-Side Technologies

Banner Grabbing

BanneMany web servers disclose fine-grained version information, both about the web server software itself and about other components that have been installed.

For example, the HTTP Server header discloses a huge amount of detail about some installations:

```
Server: Apache/1.3.31 (Unix) mod_gzip/1.3.26.1a mod_auth_passthrough/
1.8 mod_log_bytes/1.2 mod_bwlimited/1.4 PHP/4.3.9 FrontPage/
5.0.2.2634a mod_ssl/2.8.20 OpenSSL/0.9.7a
```

HTTP Fingerprinting

Httprecon:

- This is a tool specifically designed for HTTP fingerprinting.
- It performs a variety of tests against a web server and analyzes its responses.
- The tool can provide: Possible Server Identifications: Based on the responses received, it will suggest what type of server might be running, along with the confidence level of each guess.

Example

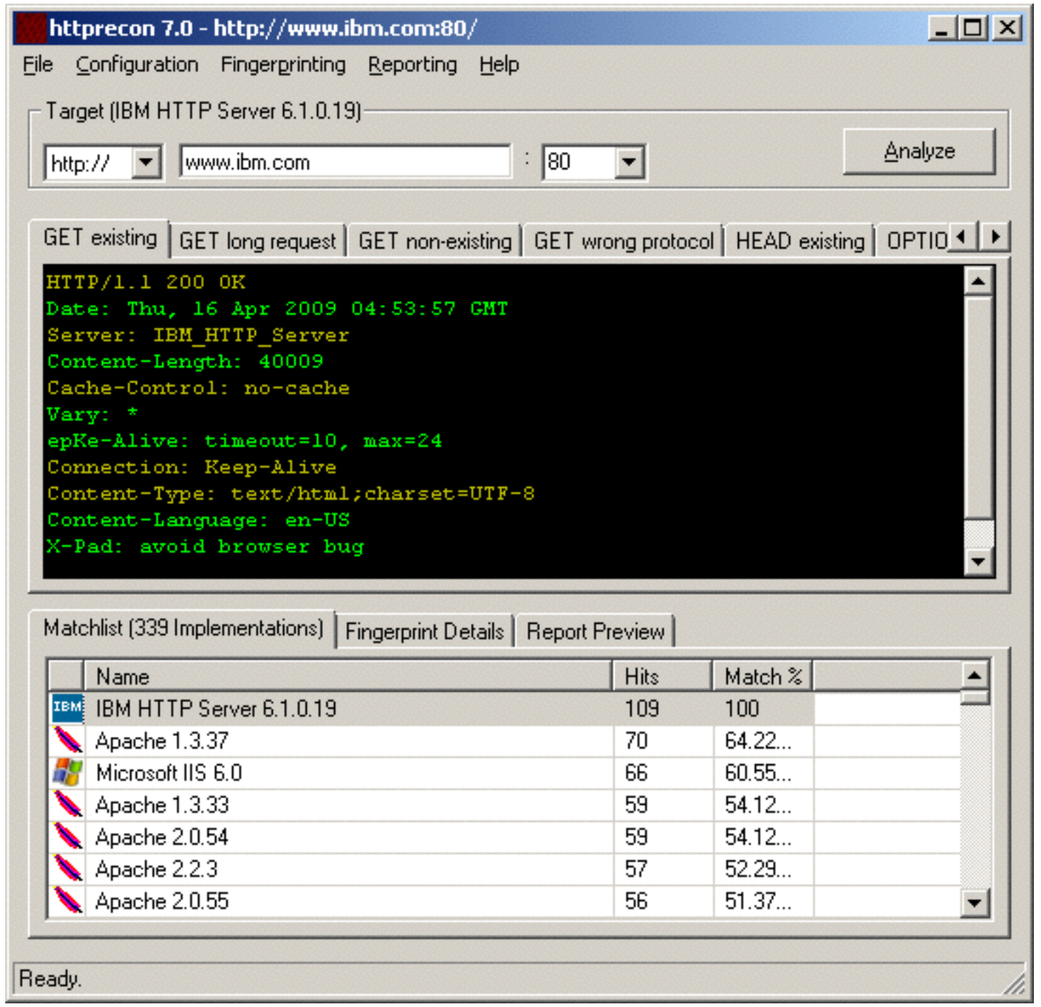
see output like this:Detected

Servers:

Apache 2.4: Confidence level 90%

Nginx 1.18: Confidence level 70%

IIS 10: Confidence level 50%This means that



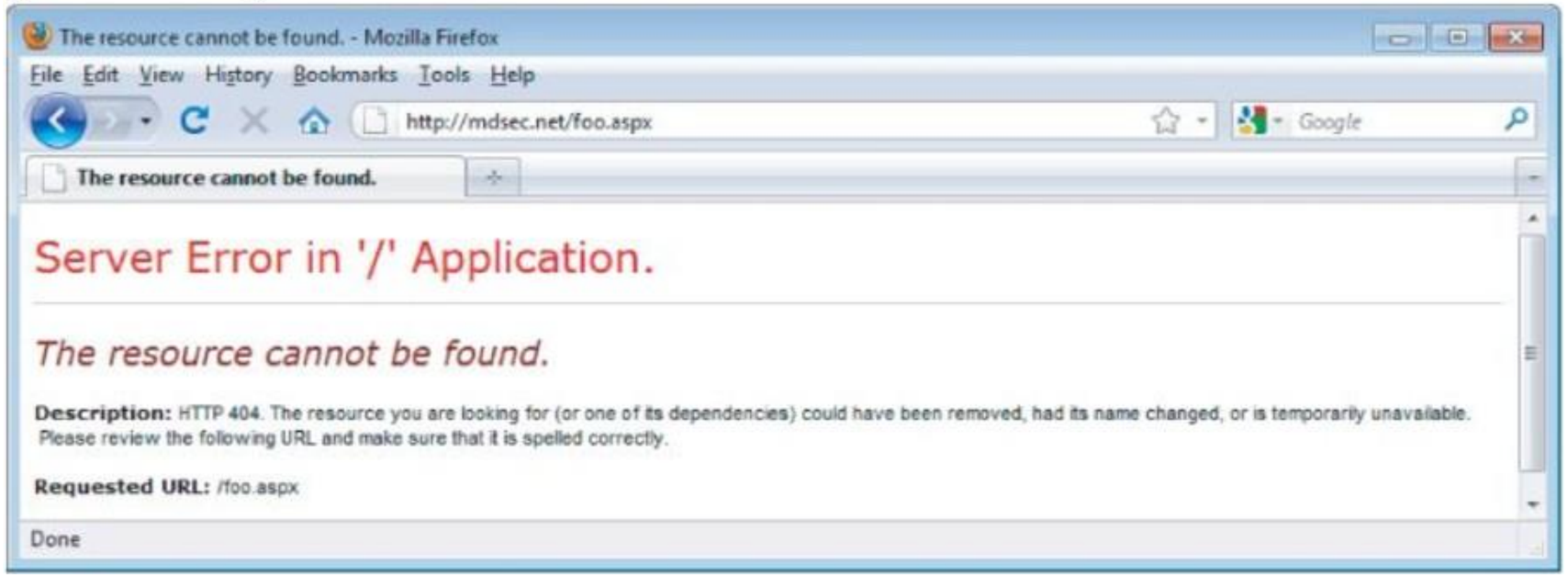
File Extensions

- **Disclose platform or language**

  - `asp` — Microsoft Active Server Pages
  - `aspx` — Microsoft ASP.NET
  - `jsp` — Java Server Pages
  - `cfm` — Cold Fusion
  - `php` — The PHP language
  - `d2w` — WebSphere
  - `pl` — The Perl language
  - `py` — The Python language
  - `dll` — Usually compiled native code (C or C++)
  - `nsf` or `ntf` — Lotus Domino

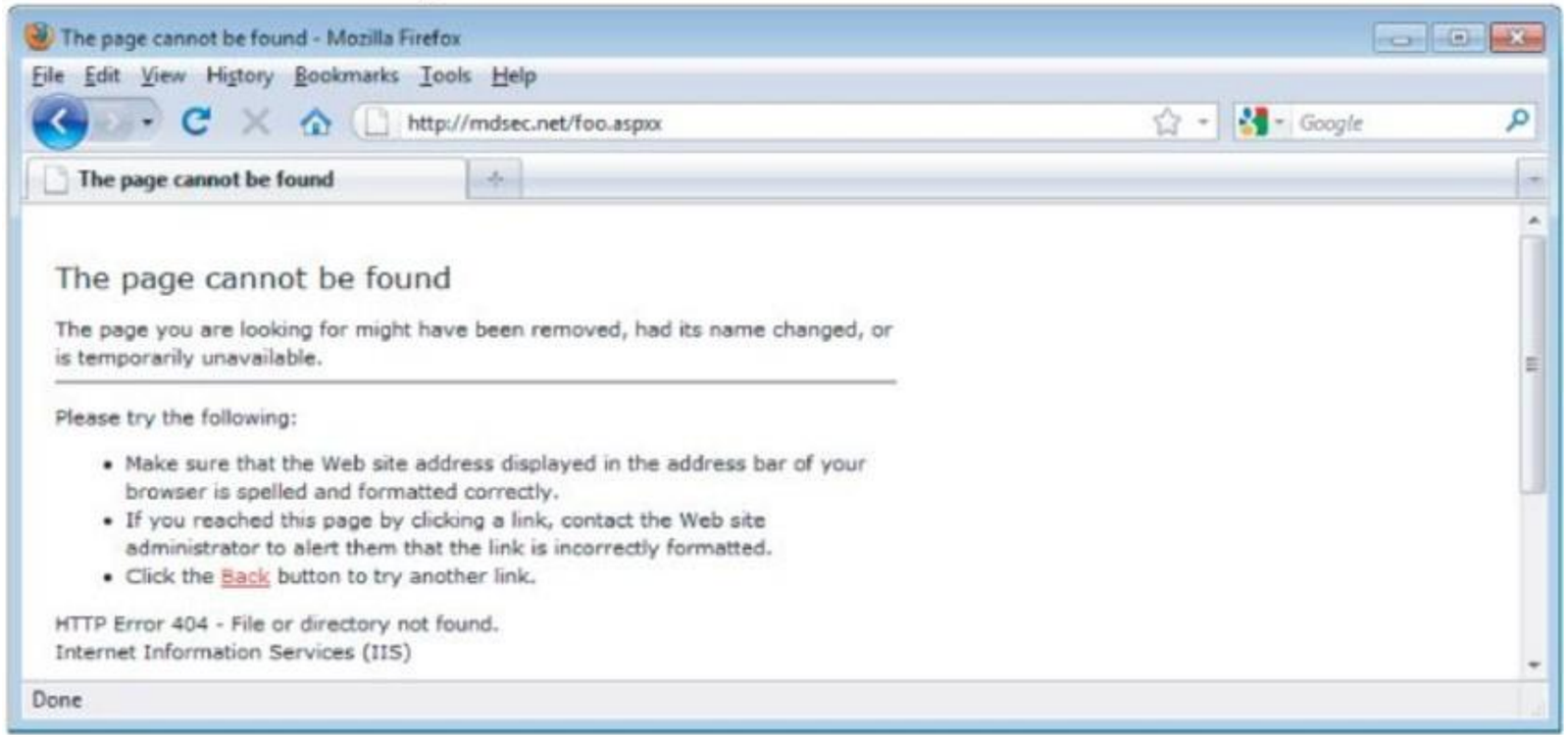
Error Messages

Figure 4.12 A customized error page indicating that the ASP.NET platform is present on the server



Error Message

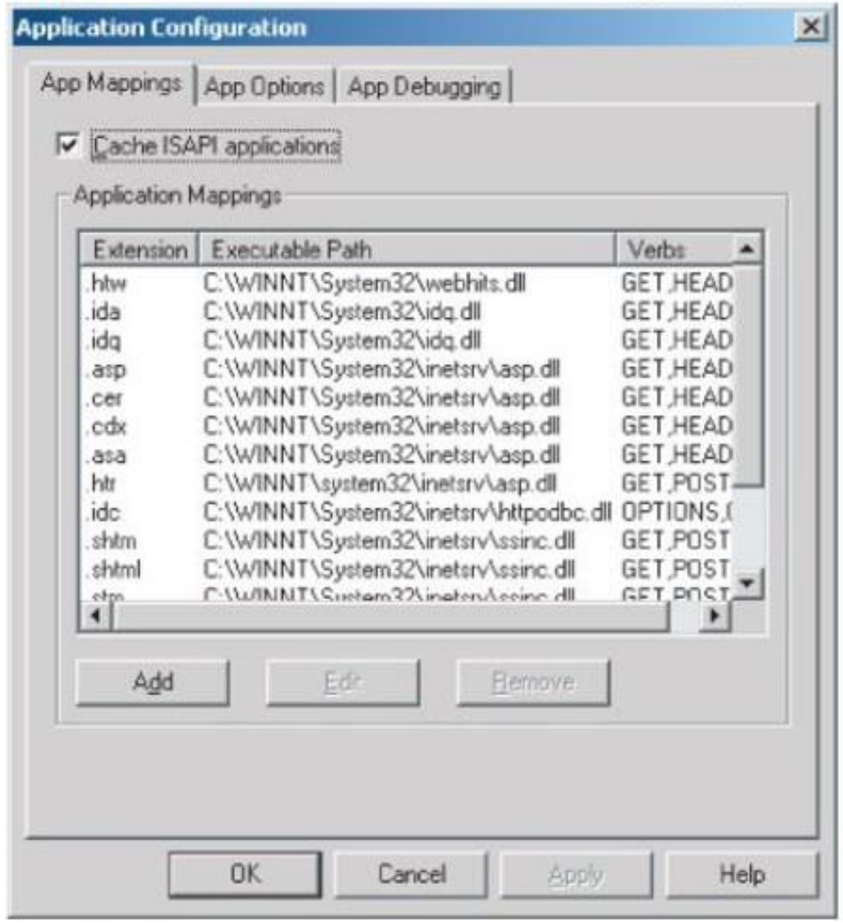
Figure 4.13 A generic error message created when an unrecognized file extension is requested



File Extension Mappings

- **Different DLLs may lead to different error messages**

Figure 4.14 File extension mappings in IIS 5.0



OpenText

`https://wahh-app/news/0,,2-421206,00.html`

The comma-separated numbers toward the end of the URL are usually generated by the Vignette content management platform.

- **Vignette is now rebranded as OpenText**
  - **Link Ch 4i**

Directory Names

- **Indicate technology in use**

  - `servlet` — Java servlets
  - `pls` — Oracle Application Server PL/SQL gateway
  - `cfdocs` or `cfide` — Cold Fusion
  - `SilverStream` — The SilverStream web server
  - `WebObjects` or `{function}.woa` — Apple WebObjects
  - `rails` — Ruby on Rails

Session Tokens

- `JSESSIONID` — The Java Platform
- `ASPSESSIONID` — Microsoft IIS server
- `ASP.NET_SessionId` — Microsoft ASP.NET
- `CFID/CFTOKEN` — Cold Fusion
- `PHPSESSID` — PHP

Third-Party Code Components

- **Add common functionality like**
  - **Shopping carts**
  - **Login mechanisms**
  - **Message boards**
- **Open-Source or commercial**
- **May contain known vulnerabilities**

Hack Steps

1. Identify all entry points for user input

- **URL, query string parameters, POST data, cookies, HTTP headers**

2.Examine query string format; should be some variation on name/value pair

3.Identify any other channels that allow user-controllable or third-party data into the app

Hack Steps

4.View HTTP server banner returned by the app; it may use several different servers

5.Check for other software identifiers in custom HTTP headers or HTML source code

6. Run httprint to fingerprint the web server

7. Research software versions for vulnerabilities

8.Review map of URLs to find interesting file extensions, directories, etc. with clues about the technologies in use

httprint

- **Not updated since 2005 (link Ch 4j)**
- **Alternatives include nmap, Netcraft, and SHODAN (Link Ch 4k)**
- **Also the Wappalyzer Chrome extension**

Hack Steps

9. **Review names of session tokens to identify technologies being used**

10. **Use lists of common technologies, or Google, to identify technologies in use, or discover other websites that use the same technologies**

11. **Google unusual cookie names, scripts, HTTP headers, etc. If possible, download and install the software to analyze it and find vulnerabilities**

Identifying Server-Side Functionality

```
https://wahh-
app.com/calendar.jsp?name=new%20applicants&isExpired=
0&startDate=22%2F09%2F2010&endDate=22%2F03%2F2011&OrderBy=name
```

- **.jsp - Java Server Pages**
- **OrderBy parameter looks like SQL**
- **isExpired suggests that we could get expired content by changing this value**

Identifying Server-Side Functionality

```
https://wahh-app.com/workbench.aspx?template=NewBranch.tpl&loc=
/default&ver=2.31&edit=false
```

- **.aspx - Active Server Pages (Microsoft)**
- **template - seems to be a filename and loc - looks like a directory; may be vulnerable to path traversal**
- **edit - maybe we can change files if this is true**
- **ver - perhaps changing this will reveal other functions to attack**

Identifying Server-Side Functionality

```
POST /feedback.php HTTP/1.1
Host: wahh-app.com
Content-Length: 389

from=user@wahh-mail.com&to=helpdesk@wahh-app.com&subject=
Problem+logging+in&message=Please+help...
```

- **.php - PHP**
- **Connecting to an email server, with user-controllable content in all fields**
- **May be usable to send emails**
- **Any fields may be vulnerable to email header injection**

Identifying Server-Side Functionality

```
http://eis/pub/media/117/view
```

The handling of this URL is probably functionally equivalent to the following:

```
http://eis/manager?schema=pub&type=media&id=117&action=view
```

- **Change action to "edit" or "add"**
- **Try viewing other collections by changing the ip number**

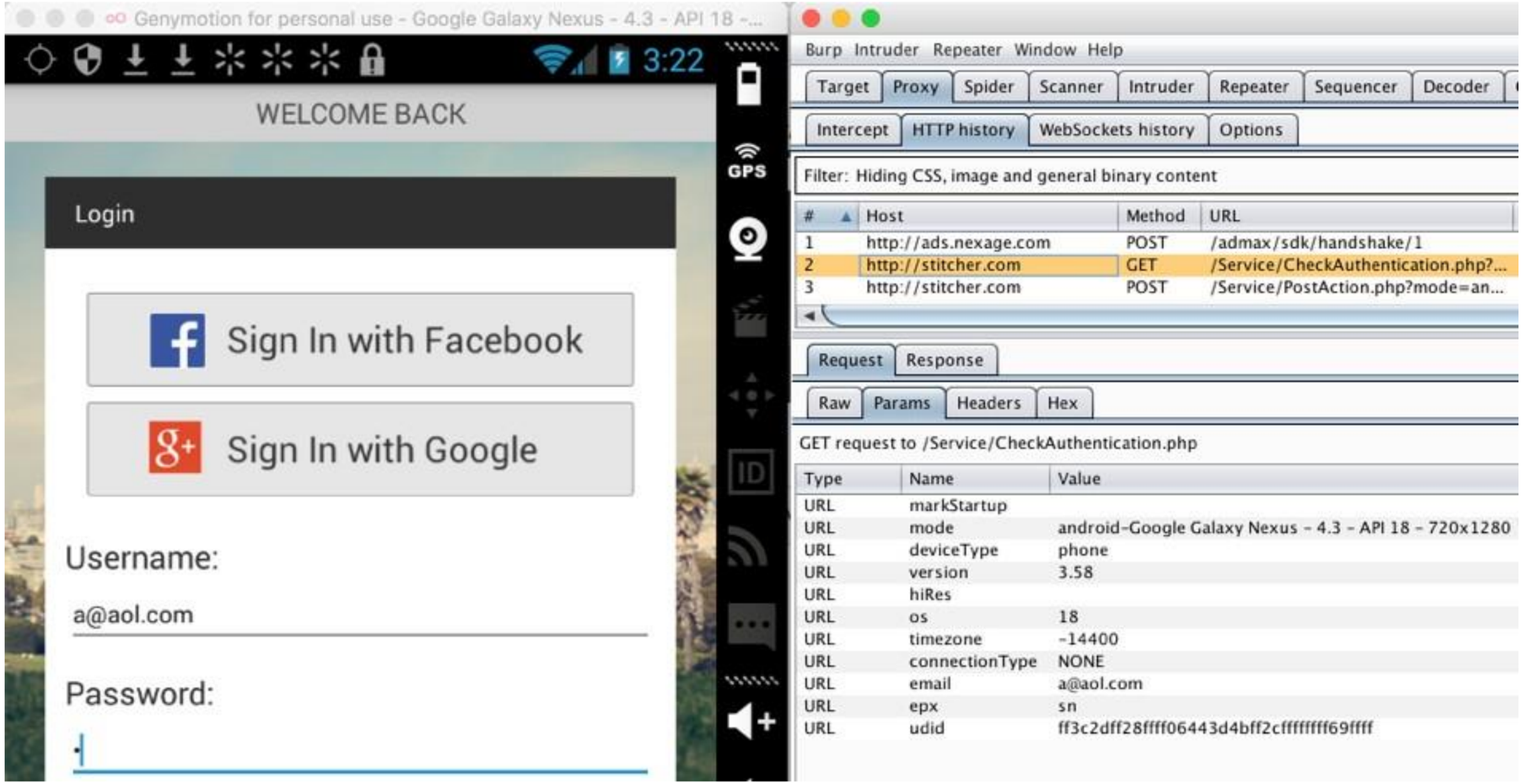
Extrapolating Application Behavior

- **An application often behaves consistently across the range of its functionality**
  - **Because code is re-used or written by the same developer, or to the same specifications**
- **So if your SQL injections are being filtered out, try injecting elsewhere to see what filtering is in effect**

Extrapolating Application Behavior

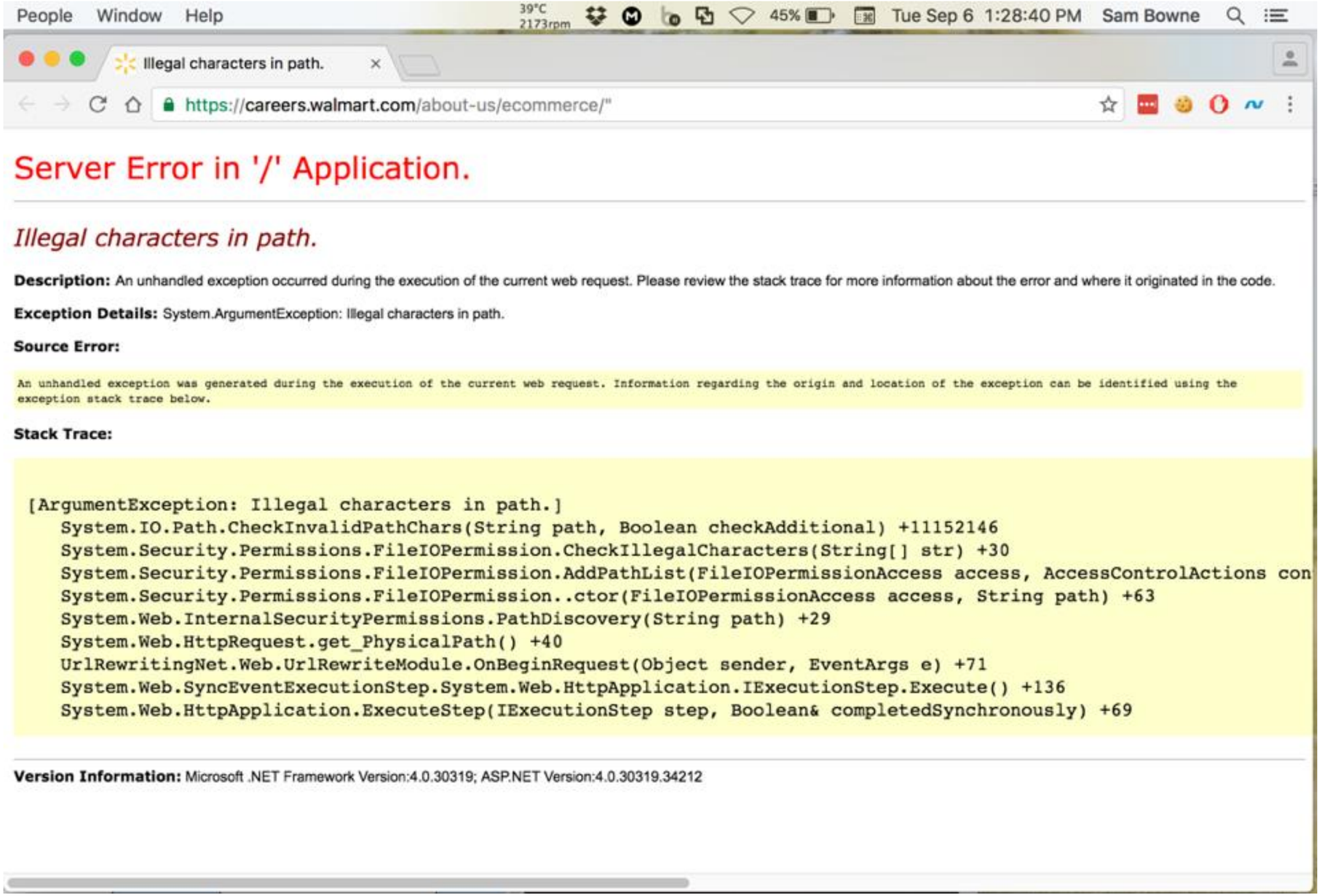
- **If app obfuscates data, try finding a place where a user can enter an obfuscated sting and retrieve the original**
  - **Such as an error message**
- **Or test systematically-varying values and deduce the obfuscation scheme**

Demo: Stitcher



Error Handling

- **Some errors may be properly handled and give little information**
  Others may crash and return verbose error information


People Window Help
39°C 2173rpm
45%
Tue Sep 6 1:28:40 PM
Sam Bowne
Illegal characters in path.
https://careers.walmart.com/about-us/ecommerce/"
Server Error in '/' Application.
Illegal characters in path.
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.ArgumentException: Illegal characters in path.
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:
[ArgumentException: Illegal characters in path.]
System.IO.Path.CheckInvalidPathChars(String path, Boolean checkAdditional) +11152146
System.Security.Permissions.FileIOPermission.CheckIllegalCharacters(String[] str) +30
System.Security.Permissions.FileIOPermission.AddPathList(FileIOPermissionAccess access, AccessControlActions con
System.Security.Permissions.FileIOPermission..ctor(FileIOPermissionAccess access, String path) +63
System.Web.InternalSecurityPermissions.PathDiscovery(String path) +29
System.Web.HttpRequest.get_PhysicalPath() +40
UrlRewritingNet.Web.UrlRewriteModule.OnBeginRequest(Object sender, EventArgs e) +71
System.Web.SyncEventExecutionStep.System.Web.HttpApplication.IExecutionStep.Execute() +136
System.Web.HttpApplication.ExecuteStep(IExecutionStep step, Boolean& completedSynchronously) +69
Version Information: Microsoft .NET Framework Version:4.0.30319; ASP.NET Version:4.0.30319.34212

Isolate Unique Application Behavio

- **App may use a consistent framework that prevents attacks**
- **Look for extra parts "bolted on" later, which may not be integrated into the framework**
  - **Debug functions, CAPTCHAs, usage tracking, third-party cod**
  - **Different GUI appearance, parameter naming conventions, comments in source code**

Mapping the Attack Surface

- **Client-side validation**
- **Database interaction -- SQL injection**
- **File uploading and downloading -- Path traversal, stored XSS**
- **Display of user-supplied data - XSS**
- **Dynamic redirects -- Redirection and header attacks**

Mapping the Attack Surface

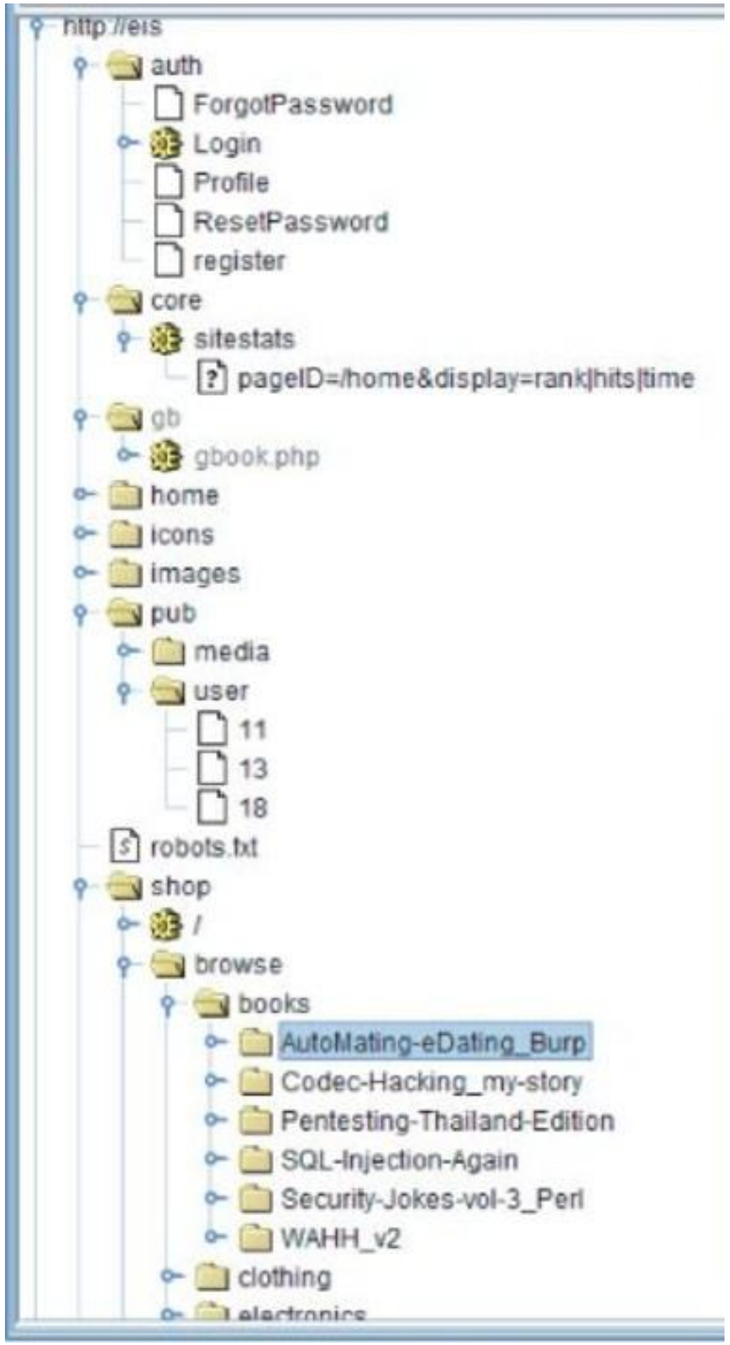
- **Social networking features -- username enumeration, stored XSS**
- **Login -- Username enumeration, weak passwords, brute-force attacks**
- **Multistage login -- Logic flaws**
- **Session state -- Predictable tokens, insecure token handling**

Mapping the Attack Surface

- **Access controls -- Horizontal and vertical privilege escalation**
- **User impersonation functions -- Privilege escalation**
- **Cleartext communications -- Session hijacking, credential theft**
- **Off-site links -- Leakage of query string parameters in the Referer header**
- **Interfaces to external systems -- Shortcuts handling sessions or access controls**

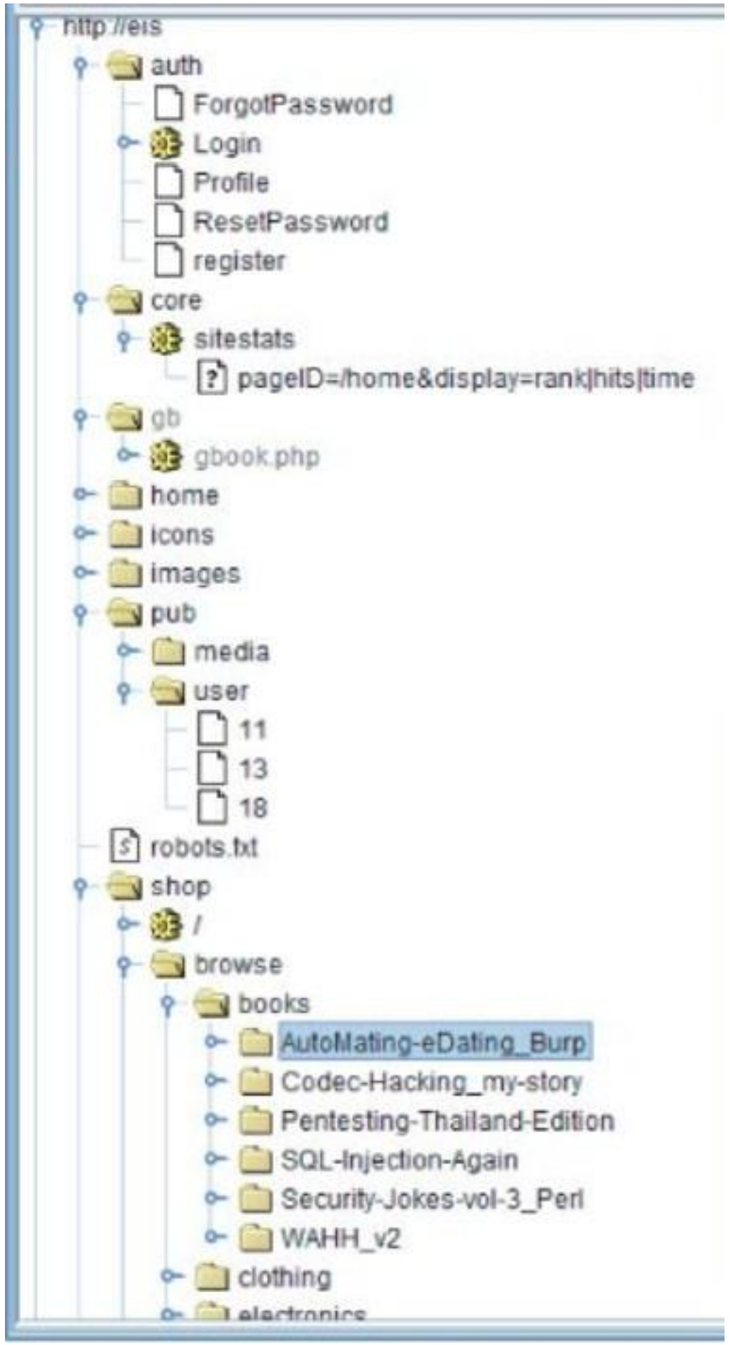
Mapping the Attack Surface

- **Error messages -- Information leakage**
- **Email interaction -- Email or command injection**
- **Native code components or interaction -- Buffer overflows**
- **Third-party components -- Known vulnerabilities**
- **Identifiable Web server -- Common configuration errors, known bugs**



Example

- **/auth contains authentication functions -- test session handling and access control**
- **/core/sitestats -- parameters; try varying them; try wildcards like *all* and * ; PageID contains a path, try traversal**
- **/home -- authenticated user content; try horizontal privilege escalation to see other user's info**



Example

- **/icons and /images -- static content, might find icons indicating third-party content, but probably nothing interesting here**
- **/pub -- RESTful resources under /pub/media and /pub/user; try changing the numerical value at the end**
- **/shop -- online shopping, all items handled similarly; check logic for possible exploits**