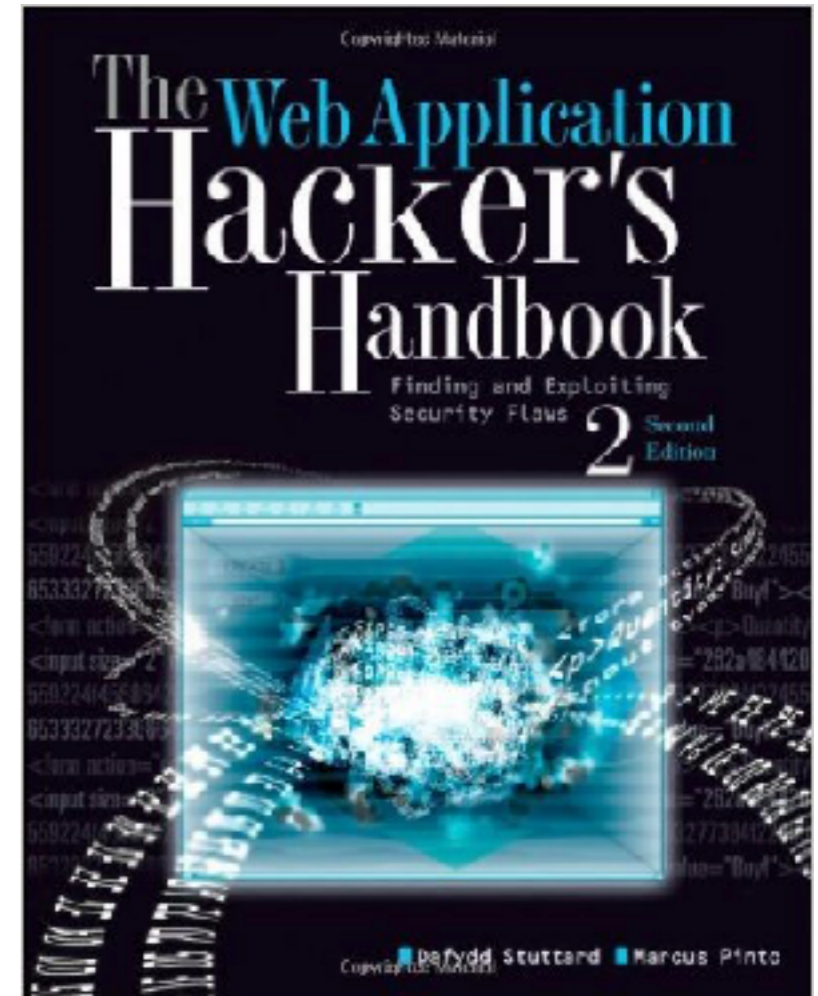


CNIT 129S: Securing Web Applications

Ch 4: Mapping the Application



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**

```
← → ↻ 🏠 ⓘ www.ccsf.edu/robots.txt

# /robots.txt for http://www.ccsf.cc.ca.us/
# comments to jjah@cloud.ccsf.cc.ca.us

User-agent: *
Disallow: /autodiscover.xml
Disallow: /crossdomain.xml
Disallow: /browserconfig.xml
Disallow: /api
Disallow: /admin
Disallow: /Schedule/Archive/
Disallow: /Schedule/Archives/
Disallow: /translate.google.com
Disallow: /Shared_Files
Disallow: /shared_images
Disallow: /applications
Disallow: /sitemap.xml
Disallow: /Campuses/
Disallow: /Chat/
Disallow: /Departments/
```

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**

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**

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**

Example (Not Logged In)

The screenshot shows a web browser window with the URL `hackazon.samsclass.info/product/view?id=16`. The browser's address bar includes a search field and navigation icons. A notification bar at the top states: "Firefox has prevented the outdated plugin 'Adobe Flash' from running on http://hackazon.samsclass.info." with buttons for "Continue Blocking" and "Allow...".

The website header features the "HACKAZON" logo in red, followed by navigation links: "FAQ", "Contact Us", "Wish List", "Sign In / Sign Up", and a shopping cart icon. A search bar is located below the logo, with a dropdown menu set to "All".

A shopping cart overlay is displayed in the center, listing two items:

- 1x Cricut Explore Electronic Cutting Machine with Cricut Design Space Free Online **\$250**
- 1x Molton Brown Indian Cress Purifying Shampoo, 10 fl. oz. **\$ 30,-**

Below the list, there is a button labeled "Show all items in shopping cart >".

On the right side of the page, there is a product detail section with the following elements:

- An "Add To Wish List" button.
- A "Back" button.
- A "Count" dropdown menu set to "1".
- A blue "Add to cart" button with a shopping cart icon.

At the bottom left, there are two buttons: "Special Offers" and "Best selling products".

- Note items in cart
- "user" contains only 5 URLs

The screenshot displays the Burp Suite interface. The top menu bar includes 'Burp Intruder', 'Repeater', 'Window', and 'Help'. Below this is a toolbar with buttons for 'Sequencer', 'Decoder', 'Comparer', 'Extender', 'Project options', 'User options', and 'Alerts'. A secondary toolbar contains 'Target', 'Proxy', 'Spider', 'Scanner', 'Intruder', and 'Repeater'. The 'Site map' and 'Scope' tabs are visible, with a filter bar below them stating: 'Filter: Hiding not found items; hiding CSS, image and general binary content; hiding 4xx responses; hiding empty folders'. The site map on the left shows a tree structure for 'http://hackazon.samsclass.info', with the 'product' folder selected. The right pane shows a list of HTTP requests with columns for Host, Method, URL, Params, and S. The selected request is a GET request to '/product/view?id=16'. Below the list, the 'Request' and 'Response' tabs are visible, with the 'Raw' tab selected, showing the raw HTTP request details.

Host	Method	URL	Params	S
http://hackazon.sams...	GET	/product/view?id=16	<input checked="" type="checkbox"/>	2
http://hackazon.sams...	GET	/product/view?id=64	<input checked="" type="checkbox"/>	2
http://hackazon.sams...	GET	/product/view	<input type="checkbox"/>	
http://hackazon.sams...	GET	/product/view?id=1	<input checked="" type="checkbox"/>	
http://hackazon.sams...	GET	/product/view?id=101	<input checked="" type="checkbox"/>	
http://hackazon.sams...	GET	/product/view?id=104	<input checked="" type="checkbox"/>	
http://hackazon.sams...	GET	/product/view?id=108	<input checked="" type="checkbox"/>	
http://hackazon.sams...	GET	/product/view?id=109	<input checked="" type="checkbox"/>	
http://hackazon.sams...	GET	/product/view?id=11	<input checked="" type="checkbox"/>	
http://hackazon.sams...	GET	/product/view?id=111	<input checked="" type="checkbox"/>	
http://hackazon.sams...	GET	/product/view?id=113	<input checked="" type="checkbox"/>	

```
GET /product/view?id=16 HTTP/1.1
Host: hackazon.samsclass.info
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.11; rv:46.0) Gecko/20100101 Firefox/46.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://hackazon.samsclass.info/
Cookie: visited_products=%2C1%2C72%2C16%2C64%2C;
__cfduid=d9c56d090ba7a8cde02df2adcce0fb34f1453389371;
PHPSESSID=isulfocru7tcth4ppe85vj4tp0
Connection: close
```

- Login event seen in Burp

Filter: Hiding not found items; hiding CSS, image and general binary content; hiding 4xx responses; hiding empty folders

Host Method URL Param

http://hackazon.sams...	POST	/user/login	<input checked="" type="checkbox"/>
-------------------------	------	-------------	-------------------------------------

Request Response

Raw Params Headers Hex

```
POST /user/login HTTP/1.1
Host: hackazon.samsclass.info
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.11; rv:46.0) Gecko/20100101 Firefox/46.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://hackazon.samsclass.info/product/view?id=16

Cookie: visited_products=%2C1%2C72%2C16%2C64%2C;
__cfduid=d9c56d090ba7a8cde02df2adcce0fb34f1453389371; PHPSESSID=isulfocru7tcth4ppe85vj4tp0
Connection: close
Content-Type: application/x-www-form-urlencoded
Content-Length: 33

username=samtest&password=samtest
```

? < + > Type a search term 0 matches

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**

Browser window: Hackazon | URL: hackazon.samsclass.info

HACKAZON

All Search products... Search!

Register on the site

Get the Best Price

Special selection

Martha Stewart Crafts Garland, Pink Pom Pom Small
Fun, festive party decorative pop poms. Perfect for any...

\$9

Elements Console Sources Network Timeline Profiles Application

View: [Icons] Preserve log Disable cache Offline No throttling

Filter Regex Hide data URLs

All XHR JS CSS Img Media Font Doc WS Manifest Other

Name Path	Status Text	Type	Initiator	Size Conter	Time Latenc	Timeline - Start Time	
hackazon.samsclass.info	200 OK	doc...	Other	9.4 KB 67.3 ...	327 ... 326 ...		
bootstrap.css /css	200 OK	style...	[index]:12 Parser	19.4 ... 129 ...	825 ... 294 ...		
font-awesome.min.css /font-awesome/css	200 OK	style...	[index]:17 Parser	4.9 KB 20.3...	1.11 s 1.11 s		
ekko-lightbox.css /css	200 OK	style...	[index]:20 Parser	813 B 1.1 KB	1.11 s 1.11 s		
star-rating.min.css /css	200 OK	style...	[index]:21 Parser	1.2 KB 2.6 KB	1.11 s 1.11 s		
nivo-slider.css /css	200 OK	style...	[index]:22 Parser	1.1 KB 1.8 KB	1.11 s 1.11 s		
bar.css /css/nivo-themes/bar	200 OK	style...	[index]:23 Parser	1.4 KB 3.4 KB	1.11 s 1.11 s		
light.css /css/nivo-themes/light	200 OK	style...	[index]:24 Parser	1.1 KB 1.9 KB	1.11 s 1.11 s		
bootstrapValidator.css /css	200 OK	style...	[index]:25 Parser	642 B 472 B	1.52 s 1.51 s		
modern-business.css /css	200 OK	style...	[index]:26 Parser	1.5 KB 3.2 KB	1.51 s 1.51 s		
ladda-themeless.min.css /css	200 OK	style...	[index]:27 Parser	1.5 KB 7.5 KB	1.52 s 1.52 s		
subcategory.css /css	200 OK	style...	[index]:30 Parser	617 B 543 B	1.52 s 1.52 s		
site.css /css	200 OK	style...	[index]:31 Parser	5.7 KB 25.9...	1.52 s 1.52 s		
sidebar.css	200	...	[index]:32	833 B	1.52 s		

64 requests | 460 KB transferred | Finish: 6.90 s | DOMContentLoaded: 4.74 s | Load: 6.90 s


HACKAZON

All ▾ Search products... Search!


[Register on the site](#)


[Get the Best Price](#)

Special selection



[Martha Stewart Crafts Garland, Pink Pom Pom Small](#)
Fun, festive party decorative pop poms. Perfect for any...



\$9



Elements Console Sources Network Timeline Profiles Application

Capture: Network JS Profile Screenshots Memory Paint

1000 ms 2000 ms 3000 ms 4000 ms 5000 ms 6000 ms 7000 ms 8000 ms 9000 ms 10000 ms

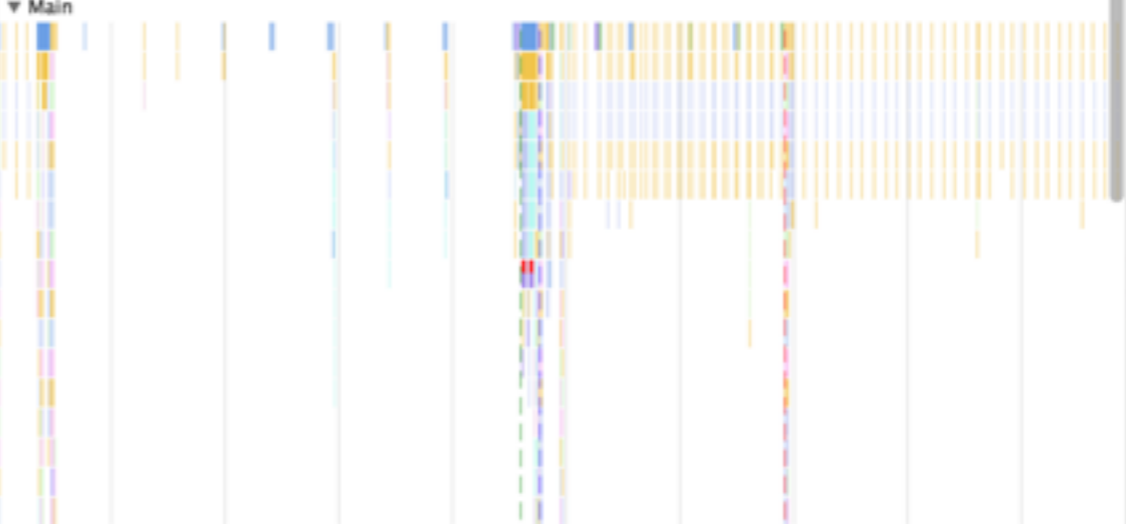


FPS
CPU
NET

1000 ms 2000 ms 3000 ms 4000 ms 5000 ms 6000 ms 7000 ms 8000 ms 9000 ms 10000 ms


Interactions

▼ Main



Summary Bottom-Up Call Tree Event Log

Range: 0 – 9.95 s



- 32.1 ms ■ Loading
- 372.7 ms ■ Scripting
- 186.2 ms ■ Rendering
- 11.6 ms ■ Painting
- 96.5 ms ■ Other
- 9247.3 ms ■ Idle

Total: 9.95 s

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**
- **A custom Python script is much better**

Intruder attack 1

Attack Save Columns

Results Target Positions Payloads Options

Filter: Showing all items

Request ▲	Position	Payload	Status	Error	Timeout	Length
0			200	<input type="checkbox"/>	<input type="checkbox"/>	74451
1	1	images	301	<input type="checkbox"/>	<input type="checkbox"/>	585
2	1	css	301	<input type="checkbox"/>	<input type="checkbox"/>	573
3	1	LC_MESSAGES	404	<input type="checkbox"/>	<input type="checkbox"/>	22232
4	1	js	301	<input type="checkbox"/>	<input type="checkbox"/>	569
5	1	tmpl	404	<input type="checkbox"/>	<input type="checkbox"/>	22232
6	1	lang	404	<input type="checkbox"/>	<input type="checkbox"/>	22232
7	1	default	404	<input type="checkbox"/>	<input type="checkbox"/>	22232
8	1	README	404	<input type="checkbox"/>	<input type="checkbox"/>	22232
9	1	templates	404	<input type="checkbox"/>	<input type="checkbox"/>	22232
10	1	langs	404	<input type="checkbox"/>	<input type="checkbox"/>	22232
11	1	config	404	<input type="checkbox"/>	<input type="checkbox"/>	22232
12	1	GNUmakefile	404	<input type="checkbox"/>	<input type="checkbox"/>	22232
13	1	themes	404	<input type="checkbox"/>	<input type="checkbox"/>	22232
14	1	en	404	<input type="checkbox"/>	<input type="checkbox"/>	22232

26 of 7348

Python Brute-Forcer

```
Sams-MBP-3:proj sambowne$ cat brute.py
#!/usr/bin/python
import socket, time

a = []
t0 = time.time()
with open("Directories_Common.wordlist", "r") as f:
    for line in f:
        a.append(line.rstrip('\n'))
f.close()

n = len(a)

for i in range(0, n, 100):
    s = socket.socket()
    s.connect(("hackazon.samsclass.info", 80))
    for j in range(0,100):
        k = i + j
        if k < n:
            req = "HEAD /" + a[k] + " HTTP/1.1\nHost: hackazon.samsclass.info\n\n"
            s.send(req)
            r = s.recv(8192)[9:12]
            if r != "404":
                print a[k], r
    s.shutdown(socket.SHUT_WR)
    s.close()

print "Guesses: ", n, " Elapsed time: ", time.time() - t0, " sec."
```

```
[Sams-MBP-3:proj sambowne$ ./brute.py
images 301
css 301
js 301
admin 302
faq 200
search 200
upload 301
fonts 301
app 301
install 302
log 200
blog 503
home 200
contact 200
Search 200
swf 301
.htpasswd 403
cart 302
account 302
Guesses: 1837 Elapsed time: 50.6177461147 sec.
Sams-MBP-3:proj sambowne$ █
```

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  
...
```

Other Patterns

- **Names may use numbers or dates**
- **Check include files from HTML and JavaScript**
 - **They may be publicly readable**
- **Comments may include database names, SQL query strings**
- **Java applets and ActiveX controls may contain sensitive data**

More Clues

- **Search for temporary files created by tools and file editors**
- **.DS_Store file (a directory index created by Mac OS X)**
- **file.php-1 created when file.php is edited**
- **.tmp files created by many tools**

Burp Pro's Content Discovery

- Brute force using built-in lists of common file and directory names
- Dynamic generation of wordlists based on resource names observed within the target application
- Extrapolation of resource names containing numbers and dates
- Testing for alternative file extensions on identified resources
- Spidering from discovered content
- Automatic fingerprinting of valid and invalid responses to reduce false positives

Google's Skipfish

- **Vulnerability scanner but main strength is finding files and folders**
- **Links Ch 4d, 4e**



DirBuster

- **Seems abandoned; perhaps now included in Zed Attack Proxy**
- **Link Ch 4f**

→ ↻ 🏠 https://www.owasp.org/index.php/Category:OWASP_DirBuster_Project 🔍 ☆ 📄 🌟 🌟

👤 Log in Request account

Category **Discussion** Read View source View history Search

Category:OWASP DirBuster Project

This historical page is now part of the OWASP archive.
This page contains content that is outdated and is no longer being maintained. It is provided as a courtesy for individuals who are still using these technologies. This page may contain URLs that were once valid but may now link to sites or pages that no longer exist.
Please use the newer Edition(s) like **OWASP Zed Attack Proxy Project**

Home
About OWASP
Acknowledgements
Advertising
AppSec Events
Books
Brand Resources
Chapters
Donate to OWASP
Downloads
Funding
Governance
Initiatives

OWASP PROJECTS
INACTIVE OWASP PROJECT

 **OWASP**
Open Web Application
Security Project

Public Information

- **Search engines (and cached content)**
- **Web archives such as the Wayback Machine**
- **Posts to forums like Stack Exchange**

Advanced Search

- `site:www.wahh-target.com` returns every resource within the target site that Google has a reference to.
- `site:www.wahh-target.com login` returns all the pages containing the expression `login`. In a large and complex application, this technique can be used to quickly home in on interesting resources, such as site maps, password reset functions, and administrative menus.
- `link:www.wahh-target.com` returns all the pages on other websites and applications that contain a link to the target. This may include links to old content, or functionality that is intended for use only by third parties, such as partner links.
- `related:www.wahh-target.com` returns pages that are “similar” to the target and therefore includes a lot of irrelevant material. However, it may also discuss the target on other sites, which may be of interest.

Web Server Vulnerabilities

- **Some Web servers let you list directory contents or see raw source code**
- **Sample and diagnostic scripts may contain vulnerabilities**

Nikto and Wikto

- **Scans servers for known vulnerable files and versions**
- **Wikto is the Windows version**
- **Nikto is the Linux version**
 - **Included in Kali**
- **Fast and easy to use**
- **Has false positives like all vulnerability scanners**
- **Must verify results with manual testing**

Example

```
root@kali:~# nikto -host hackazon.samsclass.info
- Nikto v2.1.6
-----
+ Target IP:          45.55.27.75
+ Target Hostname:    hackazon.samsclass.info
+ Target Port:        80
+ Start Time:         2016-08-29 18:22:38 (GMT-4)
-----
+ Server: Apache/2.4.7 (Ubuntu)
+ Cookie PHPSESSID created without the httponly flag
+ Retrieved x-powered-by header: PHP/5.5.9-lubuntu4.14
+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-XSS-Protection header is not defined. This header can hint to the user agent to protect against some forms of XSS
+ The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ Server leaks inodes via ETags, header found with file /robots.txt, fields: 0x2 0x525c6d2b9ad55
+ /crossdomain.xml contains a full wildcard entry. See http://jeremiahgrossman.blogspot.com/2008/05/crossdomainxml-invites-cross-site.html
+ Apache/2.4.7 appears to be outdated (current is at least Apache/2.4.12). Apache 2.0.65 (final release) and 2.2.29 are also current.
+ ERROR: Error limit (20) reached for host, giving up. Last error: error reading HTTP response
+ Scan terminated: 20 error(s) and 8 item(s) reported on remote host
+ End Time:           2016-08-29 18:29:36 (GMT-4) (418 seconds)
-----
+ 1 host(s) tested
root@kali:~# █
```

Functional Paths

- **Different from old-fashioned tree-structured file system**
- **Every request goes to the same URL**
- **Parameters specify function**
- **Very different structure to explore**

```
POST /bank.jsp HTTP/1.1
```

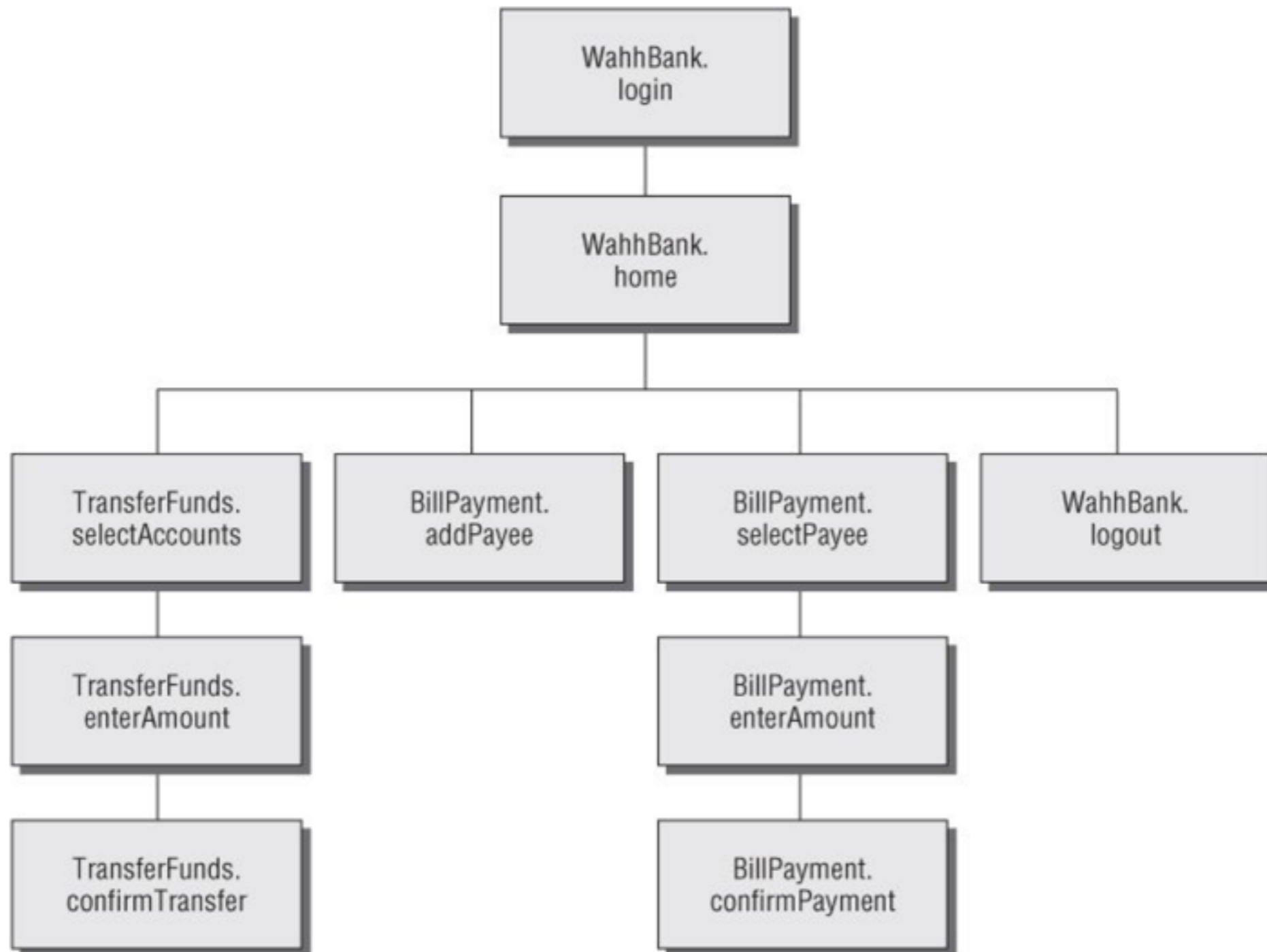
```
Host: wahn-bank.com
```

```
Content-Length: 106
```

```
servlet=TransferFunds&method=confirmTransfer&fromAccount=10372918&to  
Account=
```

```
3910852&amount=291.23&Submit=Ok
```

Map of Functional Paths



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

- **Normally, google.com?q=duck**
- **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 (e.g. WebDAV)**
 - **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

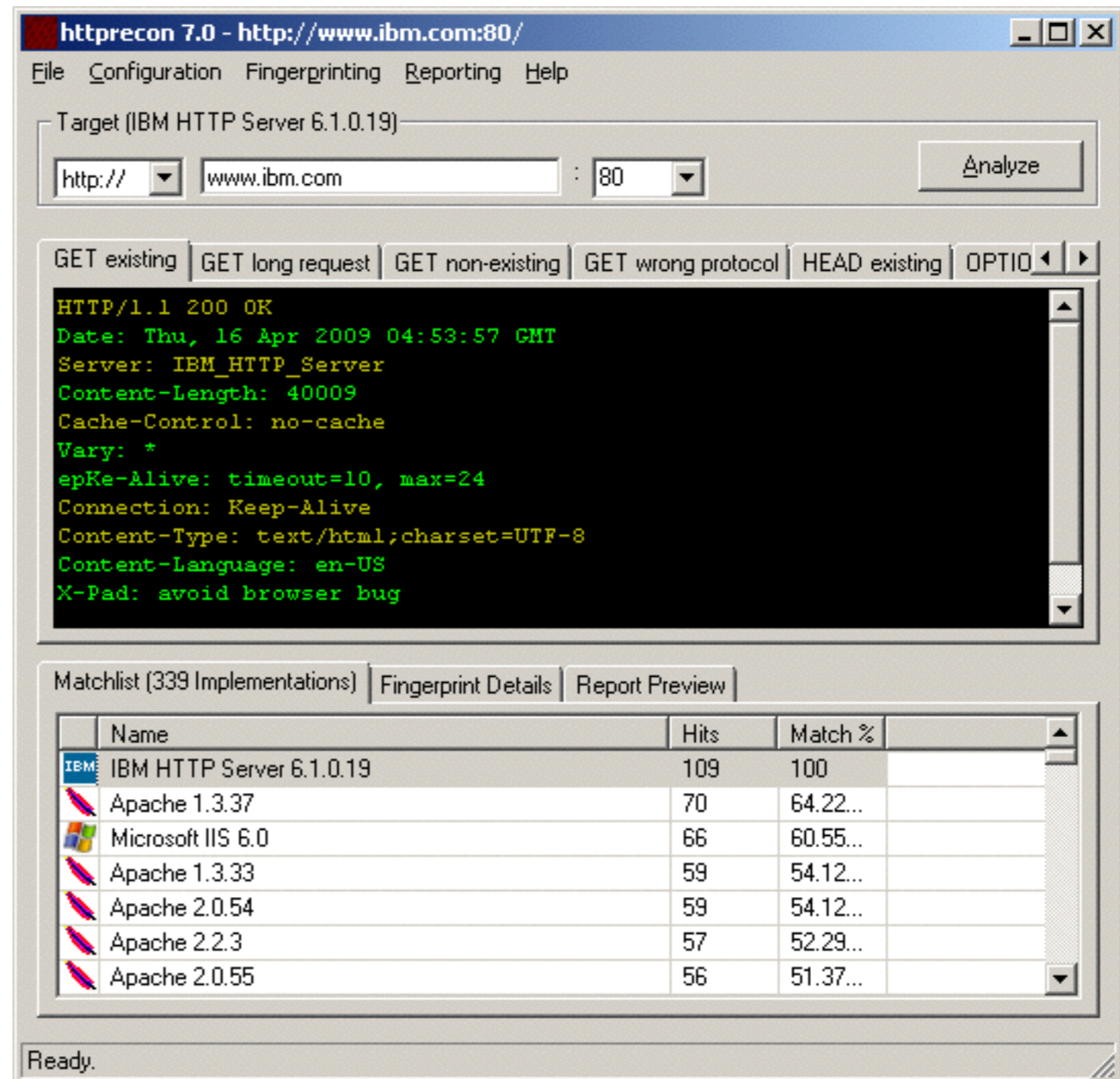
- **Banners often leak version information**
- **Also Web page templates**
- **Custom HTTP headers**
- **URL query string parameters**

```
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 uses subtle clues to identify versions, not just banners**

- **Link Ch 4h**



The screenshot shows the httprecon 7.0 application window. The title bar reads "httprecon 7.0 - http://www.ibm.com:80/". The menu bar includes "File", "Configuration", "Fingerprinting", "Reporting", and "Help". The "Target" field is set to "http://www.ibm.com:80" with an "Analyze" button. Below this, there are tabs for different request types: "GET existing", "GET long request", "GET non-existing", "GET wrong protocol", "HEAD existing", and "OPTIO". The main display area shows the response from the IBM HTTP Server 6.1.0.19:

```
HTTP/1.1 200 OK
Date: Thu, 16 Apr 2009 04:53:57 GMT
Server: IBM_HTTP_Server
Content-Length: 40009
Cache-Control: no-cache
Vary: *
epKe-Alive: timeout=10, max=24
Connection: Keep-Alive
Content-Type: text/html;charset=UTF-8
Content-Language: en-US
X-Pad: avoid browser bug
```

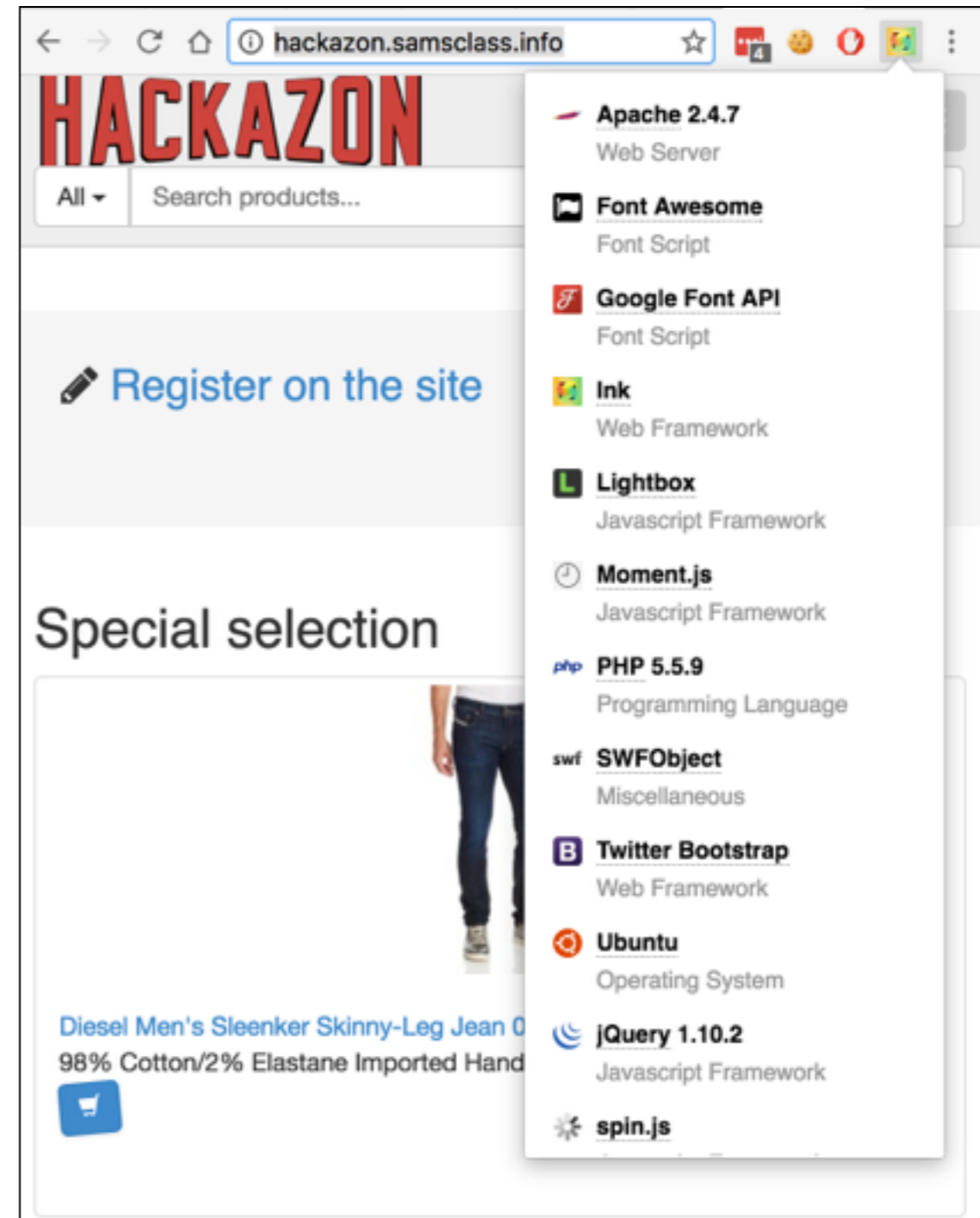
At the bottom, there is a "Matchlist (339 Implementations)" section with tabs for "Matchlist (339 Implementations)", "Fingerprint Details", and "Report Preview". A table displays the results:

Name	Hits	Match %
IBM HTTP Server 6.1.0.19	109	100
Apache 1.3.37	70	64.22...
Microsoft IIS 6.0	66	60.55...
Apache 1.3.33	59	54.12...
Apache 2.0.54	59	54.12...
Apache 2.2.3	57	52.29...
Apache 2.0.55	56	51.37...

The status bar at the bottom left shows "Ready."

Wappalyzer

- **Browser extension**

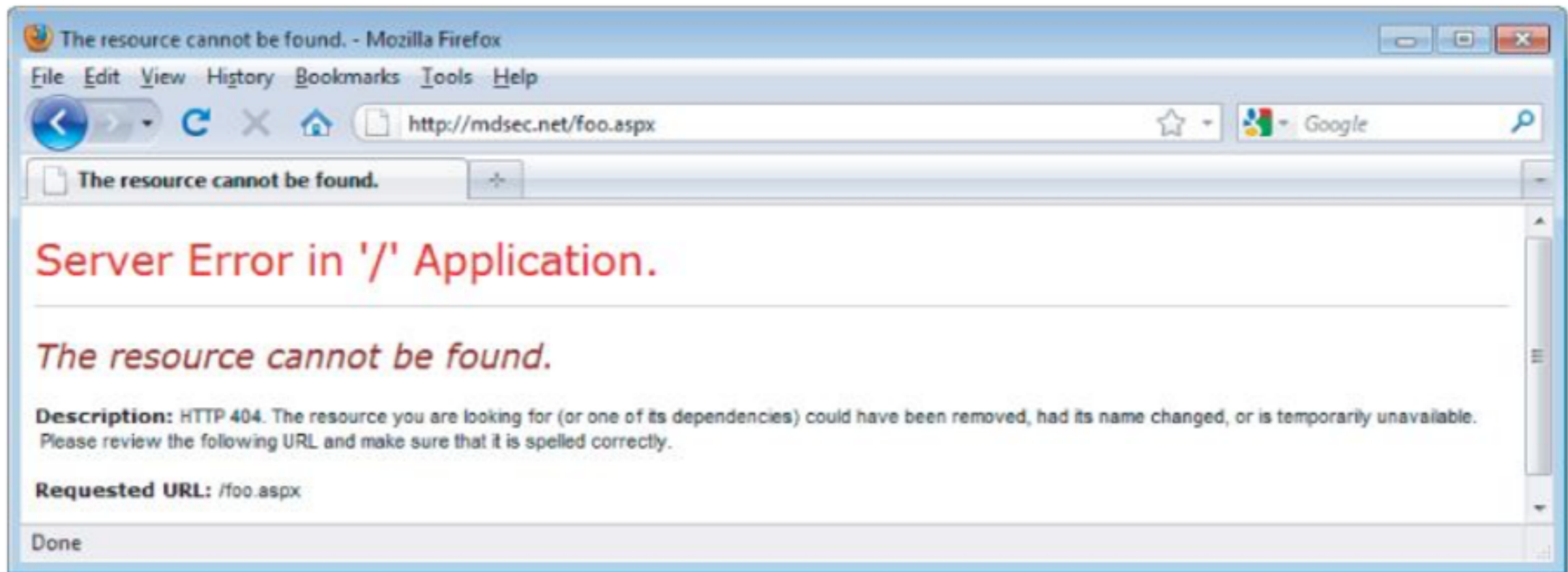


File Extensions

- **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
 - `p1` — The Perl language
 - `py` — The Python language
 - `dll` — Usually compiled native code (C or C++)
 - `nsf` or `ntf` — Lotus Domino

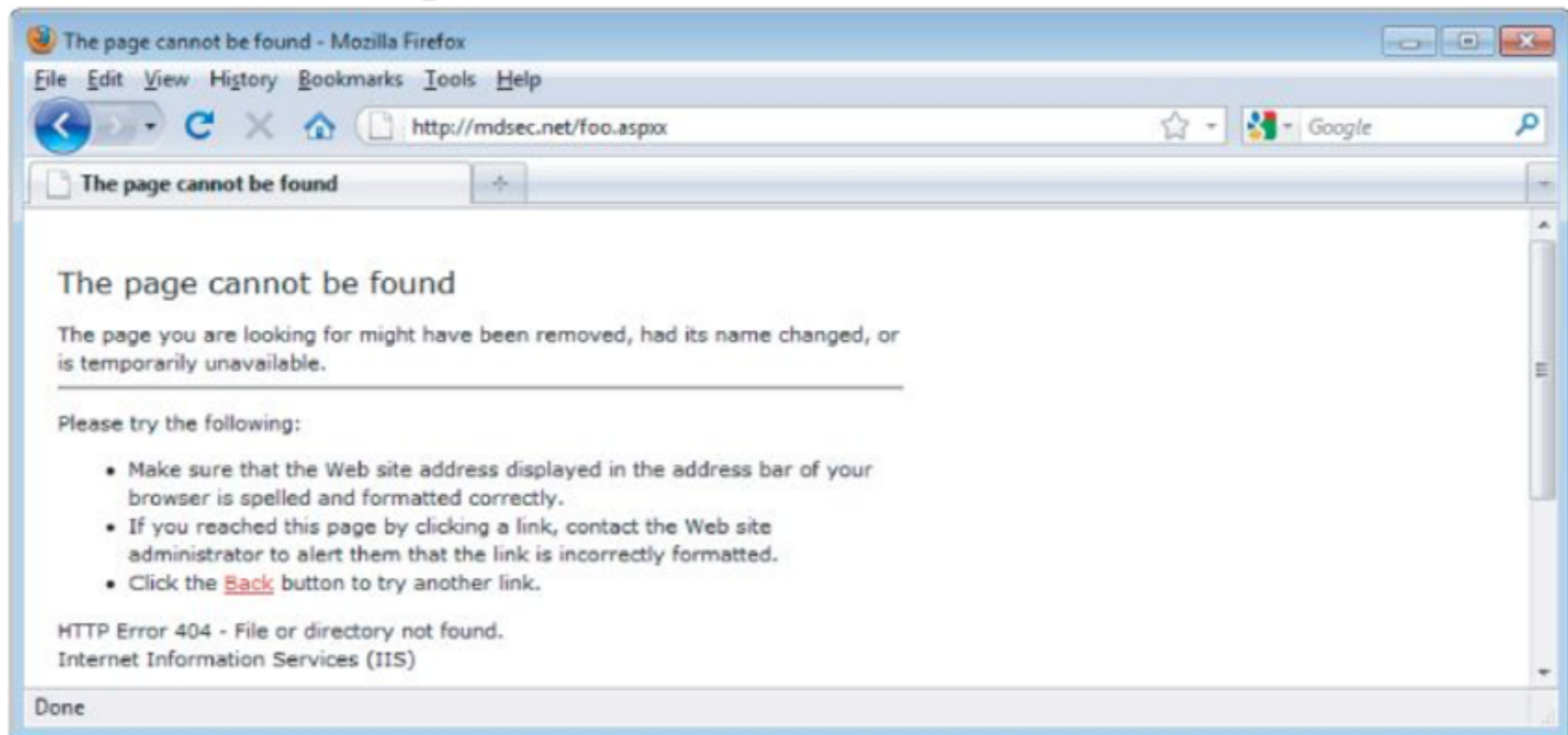
Error Messages

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



Error Message

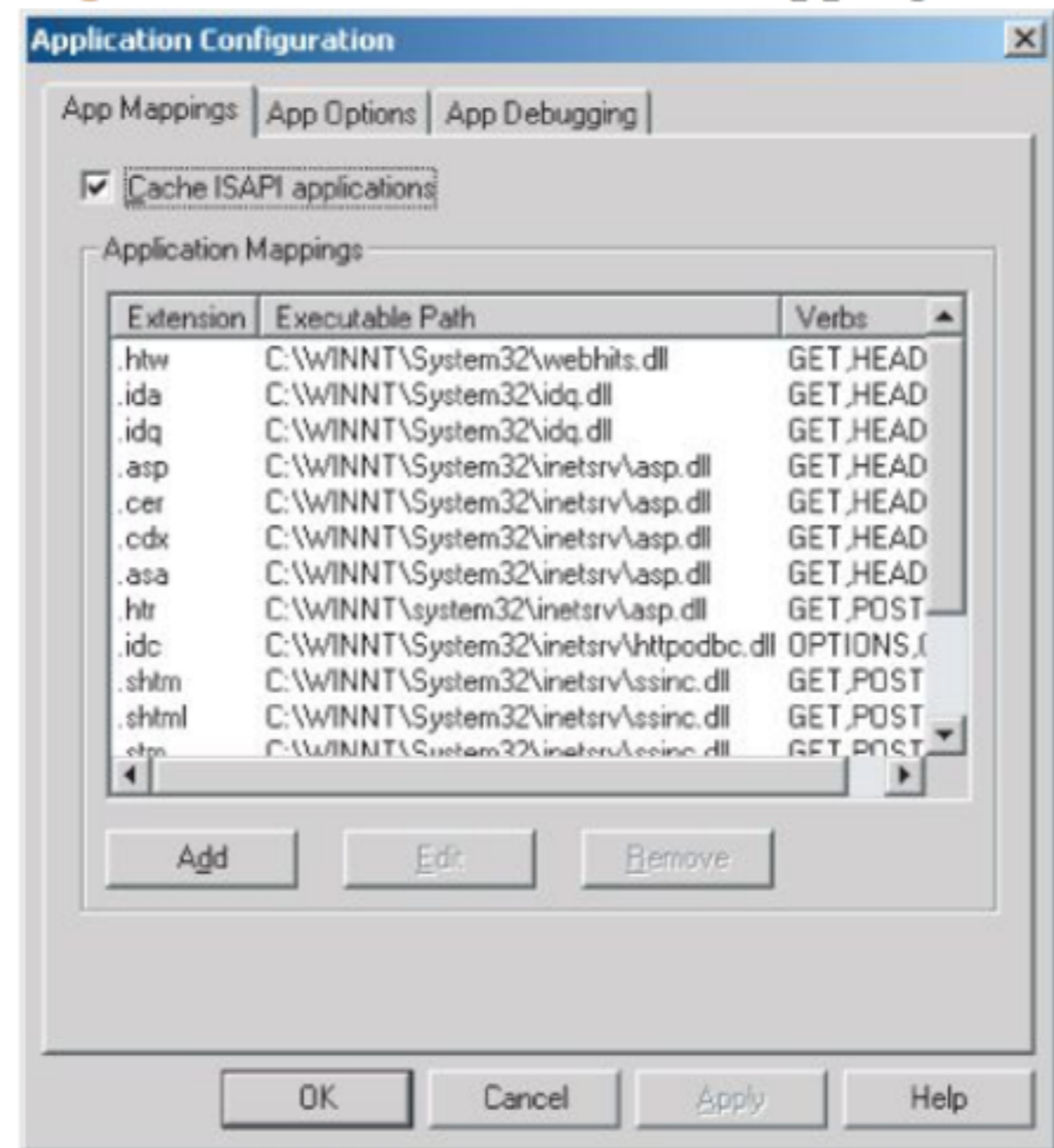
Figure 4.13 A generic error message created when an unrecognized file extension is requested



File Extension Mappings

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

Figure 4.14 File extension mappings in IIS 5.0



OpenText

```
https://wahn-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**

httpprint



- **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://wahn-  
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://wahn-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: wahn-app.com  
Content-Length: 389
```

```
from=user@wahn-mail.com&to=helpdesk@wahn-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 string and retrieve the original**
 - **Such as an error message**
- **Or test systematically-varying values and deduce the obfuscation scheme**

Demo: Stitcher

The image shows a mobile application interface on the left and a Burp Suite HTTP history window on the right. The mobile app is titled "WELCOME BACK" and has a "Login" section. It features two social login buttons: "Sign In with Facebook" and "Sign In with Google". Below these are input fields for "Username:" (containing "a@aol.com") and "Password:". The Burp Suite window shows a list of HTTP requests. The second request is highlighted, showing a GET request to "http://stitcher.com/Service/CheckAuthentication.php?". Below this, the request parameters are listed in a table.

#	Host	Method	URL
1	http://ads.nexage.com	POST	/admax/sdk/handshake/1
2	http://stitcher.com	GET	/Service/CheckAuthentication.php?...
3	http://stitcher.com	POST	/Service/PostAction.php?mode=an...

Type	Name	Value
URL	markStartup	
URL	mode	android-Google Galaxy Nexus - 4.3 - API 18 - 720x1280
URL	deviceType	phone
URL	version	3.58
URL	hiRes	
URL	os	18
URL	timezone	-14400
URL	connectionType	NONE
URL	email	a@aol.com
URL	epx	sn
URL	udid	ff3c2dff28ffff06443d4bff2cffffff69ffff

Error Handling

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



Illegal characters in path. x

🔍 <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 Behavior

- **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 code**
- **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

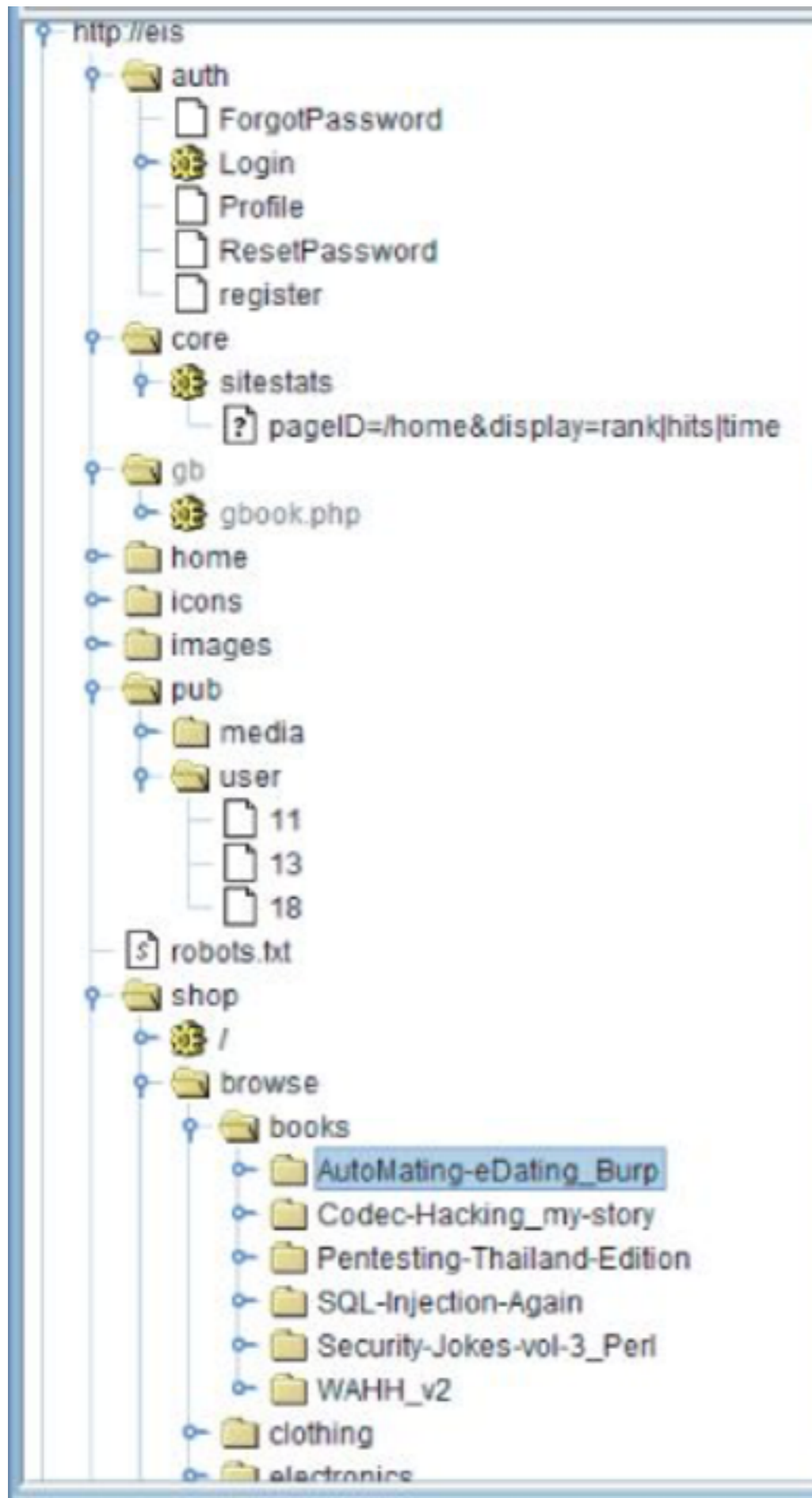
- **Access controls -- Horizontal and vertical privilege escalation**
- **User impersonation functions -- Privilege escalation**
- **Cleartext communications -- Session hijacking, credential theft**
- **Off-site links -- Leakage of query string parameters in the Referer header**
- **Interfaces to external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

