

Information Disclosure

Kameswari Chebrolu

Department of CSE, IIT Bombay

Information Disclosure

- What is the job of a web server?
 - Information dissemination
- Why information disclosure a problem?
 - Not all information should be public! Violates confidentiality
 - E.g. database passwords, usernames, debug info
 - Attackers often gather this information by maliciously interacting with server

- Say server tells its web server **type/version**, what is the big deal?
- Information may not be directly exploitable , but attacker can use it in an attack lifecycle
 - Attackers frequently use publicized security vulnerabilities
 - E.g. **Zero-day vulnerabilities** made public in the last 24 hours
 - zero-day vulnerability published → hackers scan web servers running the vulnerable software → Launch Attack
 - Web server should not leak software version and make itself a target!

Point to Note

- Many web security attacks force web server to disclose some information and violate confidentiality
 - Access control, directory traversal attacks, SQL injection, XSS, CSRF etc (will be covered in detail)
- Here, we will cover some miscellaneous ways

Outline

- What information is useful to attackers?
- How can such information leak?
- Impact of such leaks
- Common sources of such leaks
- Testing for such leaks
- Best Practices to defend against leaks

Types of Information

- OS/Application Name and Version
 - Attacker can refer to publicly available database (CVE) to check for vulnerabilities
 - Results in a highly targeted and specific attack with a high chance of success
- Sensitive data leaks (also called data breaches)
 - E.g. personal information like names, addresses, pan card details, financial data
 - Attackers can sell such data in black market!
 - Explicitly protected under various legislative and regulatory measures (e.g. GDPR in Europe)
 - Can trigger significant financial penalties for an organisation

- Username Enumeration:
 - Helps attackers gain access to a user account → can act within authorised context of that user
 - This however needs password as well
 - But if able to enumerate valid usernames, job half-done!
- Server Configuration:
 - An inherent property of the server (e.g. Apache) or leak of a **configuration file**
 - Such leaks can reveal internal-only endpoints (e.g. IP), folder paths, software versions, database access credentials, API keys etc

- Application code:
 - Servers execute code and provide results (HTTP responses), details of code normally not exposed
 - But in some situations, code can leak
 - Server uses an interpreted language (e.g. PHP) but **interpreter not enabled** in configuration file
 - When a version control accidentally exposes source code
 - Attacker can get sensitive info like passwords/api keys from code or statistically analyse code for weaknesses

- Internal organisational data (not customer data)
 - E.g. employee data
 - Permits social engineering attacks (spear-phishing) to gain access to high-value accounts (e.g. admins)

Why information leaks?

3 Main Reasons

- Sensitive content not removed from public content
 - Comments in Markup
 - Exposed metadata in git store
 - Git stores metadata such as usernames, filenames, file paths, host IP addresses, detailed “diff” (source code snippets)
 - Git store can be accidentally uploaded as part of the CI/CD pipeline

- Insecure configuration

- Not disabling debugging and diagnostic features
 - E.g. `phpinfo()` function is used on many web servers to test if PHP installation was successful
 - Exposes detailed information about PHP config as well as underlying system
 - `phpinfo()` as such should not be used in production environments!
- Configuring incorrect MIME types
 - E.g. A web server determines the correct handling for a file based on MIME type (e.g. `.php`)
 - Developer may create a backup (e.g. “`file.php.bak`”)
 - If MIME type not specified properly, the handler would not execute the file but return the code to the requester

PHP Version 5.5.9-1ubuntu4.13



System	Linux ns1 3.19.1-x86_64-linode53 #1 SMP Tue Mar 10 15:30:28 EDT 2015 x86_64
Build Date	Sep 29 2015 15:27:05
Server API	FPM/FastCGI
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php5/fpm
Loaded Configuration File	/etc/php5/fpm/php.ini
Scan this dir for additional .ini files	/etc/php5/fpm/conf.d
Additional .ini files parsed	/etc/php5/fpm/conf.d/05-opcache.ini, /etc/php5/fpm/conf.d/10-pdo.ini, /etc/php5/fpm/conf.d/20-apcu.ini, /etc/php5/fpm/conf.d/20-curl.ini, /etc/php5/fpm/conf.d/20-gd.ini, /etc/php5/fpm/conf.d/20-imagick.ini, /etc/php5/fpm/conf.d/20-json.ini, /etc/php5/fpm/conf.d/20-mysql.ini, /etc/php5/fpm/conf.d/20-mysqli.ini, /etc/php5/fpm/conf.d/20-pdo_mysql.ini, /etc/php5/fpm/conf.d/20-readline.ini
PHP API	20121113
PHP Extension	20121212
Zend Extension	220121212

- Over verbose Logs/error messages
 - When logging, can specify levels of reporting: “critical” or “detailed”
 - Detailed logs: function executed, received input, snippets of relevant code, call stack details etc
 - Logs helpful in development environments to help bugs but problematic when
 - Left in production systems
 - Collected logs inadvertent publishing to web root folder
 - Configuration setting where errors are returned to users as error messages
 - Differential error messages (sending different responses under different circumstances) particularly problematic
 - Can expose internal state

Impact of Leaks

- Direct harm: leaked information is inherently sensitive
 - Can cause financial or reputation damage
 - E.g. credit card data, financial data
 - Needs immediate attention
- Indirect harm: depends on what attacker can do with info
 - Latest patched version of software → no risk
 - Old version with vulnerability → high risk
- Focus on impact and exploitability of the leak, not just presence

Outline

- ~~What information is useful to attackers?~~
- ~~How can such information leak?~~
- ~~Impact of such leaks~~
- Common sources of such leaks
- Testing for such leaks
- Best Practices to defend against leaks

Some Common Sources of Leaks

- Files for web crawlers
- Directory listings
- Developer comments
- Error messages
- Debugging data
- User account pages
- Backup files
- Insecure configuration
- Version control history

Example robots.txt

```
User-agent: Googlebot  
Disallow: /admin/  
Disallow: /includes/  
Disallow: /content/plugins/  
Disallow: /content/themes/  
Crawl-delay: 50  
Visit-time: 0400-0500
```

Google can crawl each page at a delay of 50 ms; specified URLs cannot be crawled, crawling can happen between 4-5 am only

- **Files used by web crawlers** (**robots.txt** and **sitemap.xml**)
 - robots.txt: controls and restrict web crawlers by specifying which parts they can and cannot access
 - sitemap.xml: lists all the pages of a website that should be **indexed** by **search engines**
 - Location: <https://www.example.com/robots.txt> or <https://www.example.com/sitemap.xml>
 - Directories being asked to skip may contain sensitive information → worth exploring!

Sample Apache Configuration

Global configuration

ServerRoot "/etc/apache2"

Listen on port 80

Listen 80

Server-wide defaults

<Directory />

Options FollowSymLinks

AllowOverride None

Require all denied

</Directory>

<Directory /var/www/>

Options Indexes FollowSymLinks

AllowOverride None

Require all granted

</Directory>

Logging

ErrorLog \${APACHE_LOG_DIR}/error.log

LogLevel warn

CustomLog \${APACHE_LOG_DIR}/access.log combined

Include module configurations

IncludeOptional mods-enabled/*.load

IncludeOptional mods-enabled/*.conf

Include additional directory configurations

IncludeOptional conf-enabled/*.conf

Virtual hosts

IncludeOptional sites-enabled/*.conf

<u>Name</u>	<u>Last modified</u>	<u>Size</u>	<u>Description</u>
-------------	----------------------	-------------	--------------------



Parent Directory		-	
----------------------------------	--	---	--



FTP_ls.log	2020-04-27 09:20	63K	
----------------------------	------------------	-----	--



database_connect.php	2020-04-27 09:20	300	
--------------------------------------	------------------	-----	--



db_dump.sql	2020-04-27 09:21	96K	
-----------------------------	------------------	-----	--



old_pass.txt	2020-04-27 09:22	6.3K	
------------------------------	------------------	------	--

Apache/2.4.43 (Win64) OpenSSL/1.1.1g PHP/7.4.5 Server at 127.0.0.1 Port 80

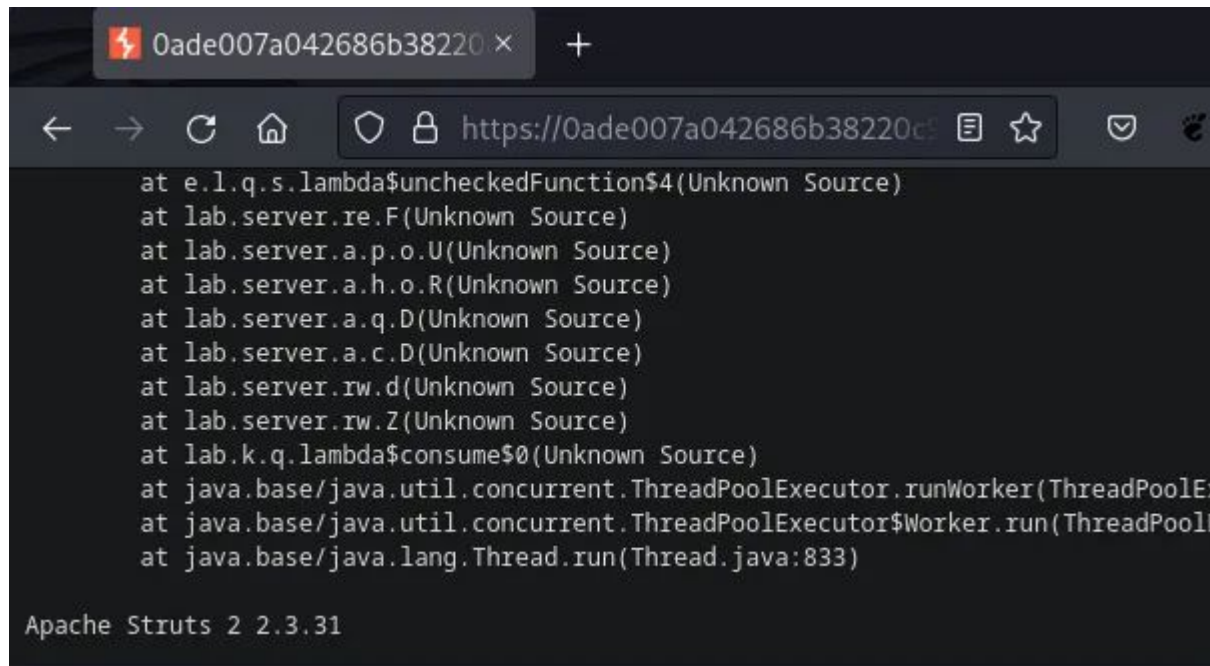
- **Directory listings:** If configured, web servers can auto list contents of directories without index page
 - If index page “**index.php or index.html**” is say absent in root folder, server will show directory listing of root
 - Attackers can see temporary files or crash dumps available!

- **Developer comments:** in-line **HTML comments** added to markup
 - Should be stripped in production environment
 - But if forgotten or missed can help attackers
 - Hidden directories, hints about application logic etc

- **Error messages:** most common cause of information leakage
 - Can reveal information on input/data type → can identify **exploitable parameters** without wasting time
 - Provide information about different technologies being used
 - E.g. template engine, database type, or server version numbers
 - Can search for exploits against given technology
 - Can browse code of open-source frameworks and get even more info

<https://example.com/product?productId=1> (valid)

<https://example.com/product?productId=67> (invalid)

A screenshot of a web browser window. The address bar shows a URL starting with 'https://0ade007a042686b38220c...'. The main content area displays a stack trace from an Apache Struts application. The stack trace lists several method calls, including 'uncheckedFunction\$4', 'F', 'U', 'R', 'D', 'D', 'Z', 'consume\$0', and 'runWorker', all originating from 'Unknown Source'. The bottom of the stack trace shows 'Thread.run' from 'java.lang.Thread' at line 833. At the bottom left of the browser window, the text 'Apache Struts 2 2.3.31' is visible.

```
at e.l.q.s.lambda$uncheckedFunction$4(Unknown Source)
at lab.server.re.F(Unknown Source)
at lab.server.a.p.o.U(Unknown Source)
at lab.server.a.h.o.R(Unknown Source)
at lab.server.a.q.D(Unknown Source)
at lab.server.a.c.D(Unknown Source)
at lab.server.rw.d(Unknown Source)
at lab.server.rw.Z(Unknown Source)
at lab.k.q.lambda$consume$0(Unknown Source)
at java.base/java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolEx
at java.base/java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPool
at java.base/java.lang.Thread.run(Thread.java:833)

Apache Struts 2 2.3.31
```

<https://cyberw1ng.medium.com/6-1-lab-information-disclosure-in-error-messages-2023-40c410fec85e>

- Differences between error messages can reveal application behavior
 - E.g. username enumeration
 - One of the guesses returns a different code from others → guess may be correct
- Should carefully audit all error messages

Some Common Sources of Leaks

- ~~Files for web crawlers~~
- ~~Directory listings~~
- ~~Developer comments~~
- ~~Error messages~~
- Debugging data
- User account pages
- Backup files
- Insecure configuration
- Version control history

- **Debugging Data:** logs can contain large amounts of information about application's behavior
 - Can help understand application's runtime state → can craft input to manipulate the application
 - Dangerous to expose in production environment
 - Examples:
 - Session variables (authentication tokens, personal data etc) expose sensitive information
 - Hostnames and credentials for back-end components
 - File and directory names on the server
 - Keys used to encrypt data

ValueError at /exams/1/save/

The view `Exams.views.save_exam_view` didn't return an `HttpResponse` object. It returned `None` instead.

```
Request Method: GET
Request URL: http://127.0.0.1:8000/exams/1/save/
Django Version: 4.0.5
Exception Type: ValueError
Exception Value: The view Exams.views.save_exam_view didn't return an HttpResponse object. It returned None instead.
Exception Location: /home/deniz/SeniorProject/django_project/vEnv/venv/lib/python3.8/site-packages/django/core/handlers/base.py, line 332, in check_response
Python Executable: /home/deniz/SeniorProject/django_project/vEnv/venv/bin/python
Python Version: 3.8.0
Python Path: ['/home/deniz/SeniorProject/django_project',
              '/home/deniz/.pyenv/versions/3.8.0/lib/python38.zip',
              '/home/deniz/.pyenv/versions/3.8.0/lib/python3.8',
              '/home/deniz/.pyenv/versions/3.8.0/lib/python3.8/lib-dynload',
              '/home/deniz/SeniorProject/django_project/vEnv/venv/lib/python3.8/site-packages']
Server time: Sat, 30 Jul 2022 18:26:38 +0000
```

Traceback [Switch to copy-and-paste view](#)

/home/deniz/SeniorProject/django_project/vEnv/venv/lib/python3.8/site-packages/django/core/handlers/exception.py, line 55, in inner

```
55.         response = get_response(request)
```

► Local vars

/home/deniz/SeniorProject/django_project/vEnv/venv/lib/python3.8/site-packages/django/core/handlers/base.py, line 204, in _get_response

```
204.         self.check_response(response, callback)
```

► Local vars

/home/deniz/SeniorProject/django_project/vEnv/venv/lib/python3.8/site-packages/django/core/handlers/base.py, line 332, in check_response

```
332.         raise ValueError(
```

<https://forum.djangoproject.com/t/failed-to-load-resource-the-server-responded-with-a-status-of-500-internal-server-error/15105/2>

- **User Accounts:** can contain sensitive information
 - Email address, phone number, API key etc
 - Is vulnerable if logic flaws allow attacker to view other user's data
 - E.g. GET /user/info?user=ravi

- **Source code Leakage**: can reveal application logic as well as API keys and credentials for accessing back-end systems
 - Easy with open source systems (i.e. for logic)
 - Tough to normally get source code of website
 - Request source code (e.g. example.php) → server will execute it and send results
 - Requesting code file using a backup file extension (e.g ~ or .bak) may help
 - Robots.txt can help identify backup folders

- **Insecure Configuration:**

- Forgot to disable **debugging** options
 - E.g. **HTTP TRACE** enabled on server
 - Echoes the exact request received → reveal any (authentication) headers appended to request by **reverse proxies**

- **Version Control:** Most websites use some version control for code (e.g. git)
 - Data stored in a folder called .git
 - If folder exposed, can download folder and explore
 - May not give access to full source code but can look at **diffs**
 - small snippets of code which can have sensitive hard-coded data

Testing for Leaks

- Many commercial as well as open-source scanners
 - E.g. Burpe Suite, appcheck, qualys, acunetix, invicti etc
- Can detect known weaknesses published as CVEs

Best Practices

- Use **generic error messages** as much as possible
- Ensure debugging or diagnostic features are disabled in the production environment
- Disable any features and settings that you don't need
- Do not upload any files that don't need to be on the web root
- Ensure proper access controls and authorizations

- Train staff on what type of information is sensitive and how to securely handle it
- Audit code and configuration settings for potential information disclosure
 - Details about backend technology type, version, setup, builds
 - In code: hardcoded credentials, API keys, IP addresses
 - Configure the correct MIME types for all the different files being used
 - Can use automated tools for all this

Web Server Settings

- Via web server configuration, **disable any HTTP response headers** that convey details about server
 - E.g. server technology, language, version etc
 - Browsers don't use such information
- Avoid file suffixes in URLs that reveal details of technology (e.g .php, .asp, and .jsp)
- Name of the cookie to store session can also reveal server-side technology
 - **Java** web servers usually store session ID under a cookie named **JSESSIONID**

- Minify/obfuscate javascript files
 - Minifiers remove spaces, comments, replace code with shorter but semantically identical statements
 - Obfuscators make code less readable; replace function, variable names with tokens without changing code behaviour
 - E.g. UglifyJS does both of above for javascript
 - Developers normally use these for performance reasons, but it can help with security as well
 - Makes it harder for an attacker to casually inspect code and reverse engineers
 - Note: These do not offer foolproof protection against determined attackers, just raise the bar

- Keep upto date with security advisories
 - Attackers can use fingerprinting techniques still to glean information
 - E.g. corrupted HTTP requests or send requests with unusual HTTP verbs to see how server responds
 - Different server technologies may respond differently!
 - Following advisories regularly will keep you on top of the game

Apache Config for MIME types

AddType text/html .html

AddType text/css .css

AddType application/javascript .js

AddType application/pdf .pdf

AddType image/jpeg .jpeg .jpg

AddType image/png .png

Summary

- Attackers interested in a variety of information
 - Some to sell in black market (e.g. customer data), some to launch further attacks as part of kill chain (towards some attack goal)
- Information types: sensitive data, OS/app versions, user names, server configurations, app code etc
- Leaks mostly via insecure configuration, log/error messages, comments/metadata in files

References

- <https://portswigger.net/web-security/information-disclosure>