GALAXY INTERNATIONAL INTERDISCIPLINARY RESEARCH JOURNAL (GIIRJ)
ISSN (E): 2347-6915
Vol. 12, Issue 5 May (2024)

INFORMATION DISCLOUSURE VULNERABILITIES

O`ralov Javlonbek Bahodir o`g`li
Teacher of "Information Security" Department of Urganch
Branch of Tashkent University of Information Technologies
uralovjavlonbek0001@gmail.com, +998999679809

Surayyo Rajabboyeva Baxrom qizi
Head of the "Information Security" Department of Urganch
Branch of Tashkent University of Information Technologies
Surayyobaxromqizi@gmail.com, +998937472702

ABSTRACT

Information disclosure, the act of revealing pertinent data and information to relevant stakeholders, plays a critical role in modern organizational dynamics. This article delves into the multifaceted nature of information disclosure, examining its implications for corporate transparency, stakeholder trust, and regulatory compliance. We explore the theoretical underpinnings of information disclosure, differentiating between voluntary and mandatory disclosures, and assess the impact of technological advancements on the ease and scope of information dissemination. The article also investigates the strategic considerations organizations must balance when deciding what information to disclose, including the potential risks and benefits. By analyzing case studies from various industries, we highlight best practices and common pitfalls in information disclosure strategies. Ultimately, this article provides a comprehensive overview of the current landscape of information disclosure, offering insights for both scholars and practitioners seeking to navigate this complex and evolving field.

Keywords: Burp suit, Git, kali linux, portswigger, feroxbuster, apache, fremwork, disclousure, backup, login

What is information disclosure?

Information disclosure, also known as information leakage, is when a website unintentionally reveals sensitive information to its users. [1] Depending on the context, websites may leak all kinds of information to a potential attacker, including:

- Data about other users, such as usernames or financial information
- Sensitive commercial or business data
- Technical details about the website and its infrastructure

The dangers of leaking sensitive user or business data are fairly obvious, but disclosing technical information can sometimes be just as serious. Although some of this information will be of limited use, it can potentially be a starting point for exposing an additional attack surface, which may contain other interesting vulnerabilities. [2] The knowledge that you are able to gather could even provide the missing piece of the puzzle when trying to construct complex, high-severity attacks.

Occasionally, sensitive information might be carelessly leaked to users who are simply browsing the website in a normal fashion. More commonly, however, an attacker needs to elicit

the information disclosure by interacting with the website in unexpected or malicious ways. [3] They will then carefully study the website's responses to try and identify interesting behavior.

Examples of information disclosure

Some basic examples of information disclosure are as follows:

- Revealing the names of hidden directories, their structure, and their contents via a robots.txt file or directory listing
- Providing access to source code files via temporary backups
- Explicitly mentioning database table or column names in error messages
- Unnecessarily exposing highly sensitive information, such as credit card details
- Hard-coding API keys, IP addresses, database credentials, and so on in the source code
- Hinting at the existence or absence of resources, usernames, and so on via subtle differences in application behavior [4]

Let's consider this type of weakness in practice. The portswigger.net website helps us learn about this type of vulnerability.

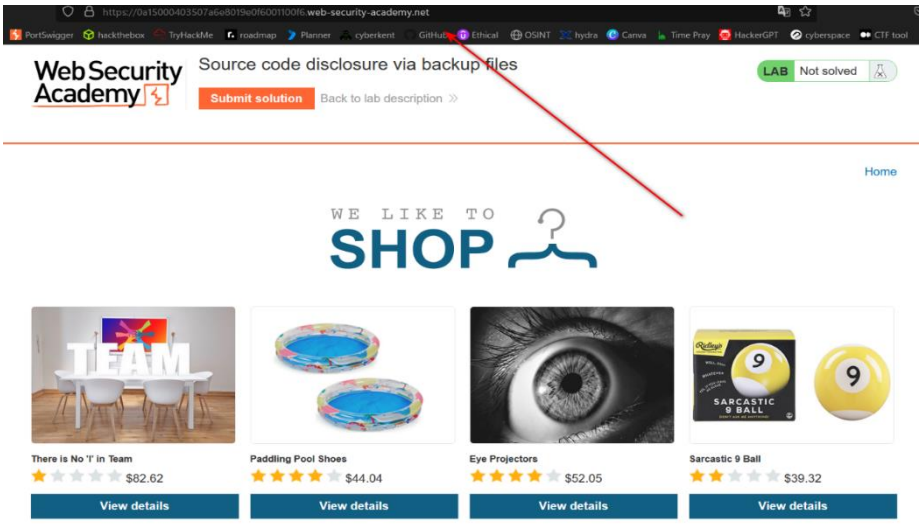

Figure 1. Vulnerable website

To exploit the vulnerability shown in Figure 1, we need to add the /robots.txt directory to this website directory.

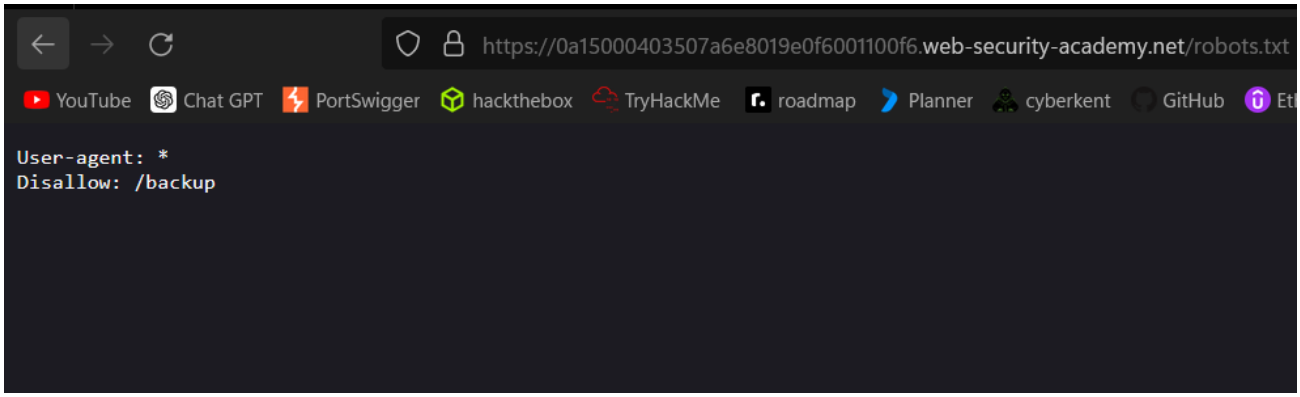

Figure 2. Hidden directory

After adding the /robots.txt extension, Figure 2 which we can see in this picture, it appears that we have a /backup directory. If we enter this directory, we can download the file in Figure 3.

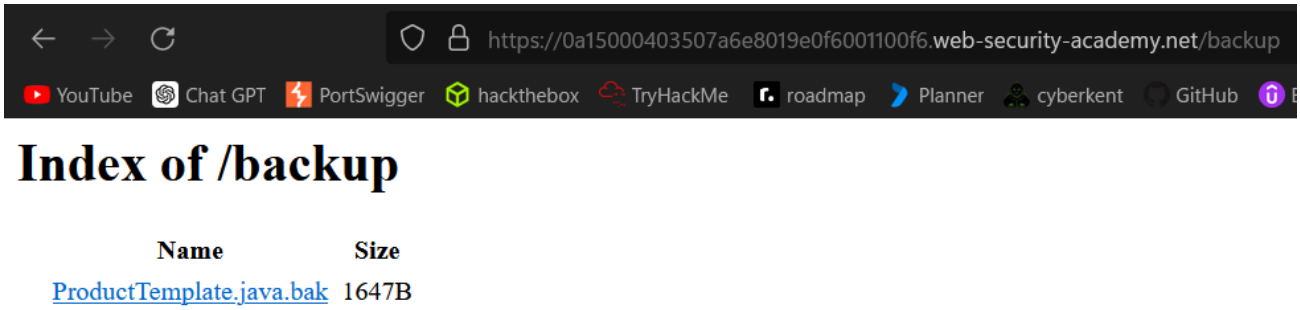

Figure 3.

By extracting the file shown in Figure 3, we can obtain the database login and password of this vulnerable website Figure 4.

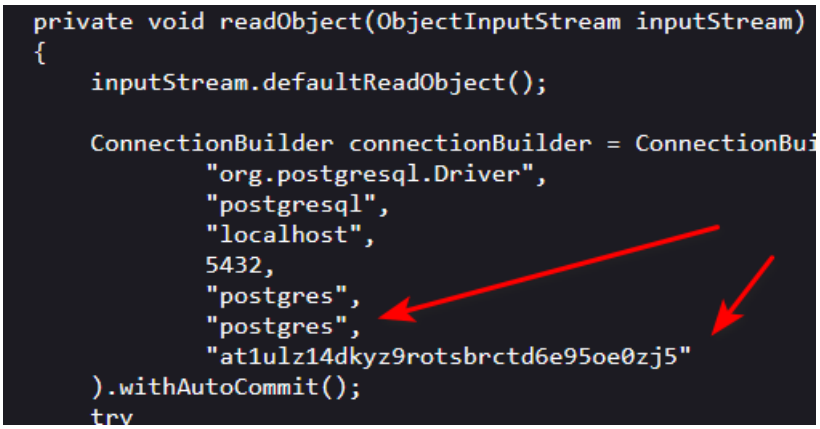

Figure 4.

From Figure 4, we can see that the database login is **"postgres"** and the password is **"at1ulz14dkyz9rotsbrctd6e95oe0zj5".**

How do information disclosure vulnerabilities arise?

Information disclosure vulnerabilities can arise in countless different ways, but these can broadly be categorized as follows:

- **Failure to remove internal content from public content**. For example, developer comments in markup are sometimes visible to users in the production environment.
- **Insecure configuration of the website and related technologies**. For example, failing to disable debugging and diagnostic features can sometimes provide attackers with useful tools to help them obtain sensitive information. Default configurations can also leave websites vulnerable, for example, by displaying overly verbose error messages.
- **Flawed design and behavior of the application**. For example, if a website returns distinct responses when different error states occur, this can also allow attackers to enumerate sensitive data, such as valid user credentials.

What is the impact of information disclosure vulnerabilities?

Information disclosure vulnerabilities can have both a direct and indirect impact depending on the purpose of the website and, therefore, what information an attacker is able to obtain. In some cases, the act of disclosing sensitive information alone can have a high impact on the affected parties. [6] For example, an online shop leaking its customers' credit card details is likely to have severe consequences.

On the other hand, leaking technical information, such as the directory structure or which third-party frameworks are being used, may have little to no direct impact. However, in the wrong hands, this could be the key information required to construct any number of other exploits. The severity in this case depends on what the attacker is able to do with this information. [5]

How to assess the severity of information disclosure vulnerabilities

Although the ultimate impact can potentially be very severe, it is only in specific circumstances that information disclosure is a high-severity issue on its own. During testing, the disclosure of technical information in particular is often only of interest if you are able to demonstrate how an attacker could do something harmful with it.

For example, the knowledge that a website is using a particular framework version is of limited use if that version is fully patched. [7] However, this information becomes significant when the website is using an old version that contains a known vulnerability. In this case, performing a devastating attack could be as simple as applying a publicly documented exploit.

It is important to exercise common sense when you find that potentially sensitive information is being leaked. It is likely that minor technical details can be discovered in numerous ways on many of the websites you test. [8] Therefore, your main focus should be on the impact and exploitability of the leaked information, not just the presence of information disclosure as a standalone issue. The obvious exception to this is when the leaked information is so sensitive that it warrants attention in its own right.

Common sources of information disclosure

Information disclosure can occur in a wide variety of contexts within a website. The following are some common examples of places where you can look to see if sensitive information is exposed.

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

Files for web crawlers. Many websites provide files at /robots.txt and /sitemap.xml to help crawlers navigate their site. Among other things, these files often list specific directories that the crawlers should skip, for example, because they may contain sensitive information. As these files are not usually linked from within the website, they may not immediately appear in Burp's site map. However, it is worth trying to navigate to /robots.txt or /sitemap.xml manually to see if you find anything of use.

Directory listings. Web servers can be configured to automatically list the contents of directories that do not have an index page present. This can aid an attacker by enabling them to quickly identify the resources at a given path, and proceed directly to analyzing and attacking those resources. It particularly increases the exposure of sensitive files within the directory that are not intended to be accessible to users, such as temporary files and crash dumps. Directory listings themselves are not necessarily a security vulnerability. However, if the website also fails to implement proper access control, leaking the existence and location of sensitive resources in this way is clearly an issue. [9]

Developer comments. During development, in-line HTML comments are sometimes added to the markup. These comments are typically stripped before changes are deployed to the production environment. However, comments can sometimes be forgotten, missed, or even left in deliberately because someone wasn't fully aware of the security implications. Although these comments are not visible on the rendered page, they can easily be accessed using Burp, or even the browser's built-in developer tools. Occasionally, these comments contain information that is useful to an attacker. For example, they might hint at the existence of hidden directories or provide clues about the application logic.

Error messages. One of the most common causes of information disclosure is verbose error messages. As a general rule, you should pay close attention to all error messages you encounter during auditing.

The content of error messages can reveal information about what input or data type is expected from a given parameter. This can help you to narrow down your attack by identifying exploitable parameters. It may even just prevent you from wasting time trying to inject payloads that simply won't work. Verbose error messages can also provide information about different technologies being used by the website. For example, they might explicitly name a template engine, database type, or server that the website is using, along with its version number. This information can be useful because you can easily search for any documented exploits that may exist for this version. Similarly, you can check whether there are any common configuration errors or dangerous default settings that you may be able to exploit. Some of these may be highlighted in the official documentation. [10]

How to prevent information disclosure vulnerabilities

Preventing information disclosure completely is tricky due to the huge variety of ways in which it can occur. However, there are some general best practices that you can follow to minimize the risk of these kinds of vulnerability creeping into your own websites.

- Make sure that everyone involved in producing the website is fully aware of what information is considered sensitive. Sometimes seemingly harmless information can be much more useful to an attacker than people realize. Highlighting these dangers can help make sure that sensitive information is handled more securely in general by your organization.
- Audit any code for potential information disclosure as part of your QA or build processes. It should be relatively easy to automate some of the associated tasks, such as stripping developer comments.
- Use generic error messages as much as possible. Don't provide attackers with clues about application behavior unnecessarily.
- Double-check that any debugging or diagnostic features are disabled in the production environment.
- Make sure you fully understand the configuration settings, and security implications, of any third-party technology that you implement. Take the time to investigate and disable any features and settings that you don't actually need.

REFERENCES

1. "The Web Application Hacker's Handbook: Finding and Exploiting Security Flaws" - Dafydd Stuttard va Marcus Pinto (2011)
2. "Hacking: The Art of Exploitation" - Jon Erickson (2008)
3. "Web Security Testing Cookbook: Systematic Techniques to Find Problems Fast" - Paco Hope va Ben Walther (2008)
4. "The Tangled Web: A Guide to Securing Modern Web Applications" - Michal Zalewski (2011)
5. "OWASP Testing Guide v4" - OWASP Foundation (2014)
6. "Information Security: Principles and Practice" - Mark Stamp (2011)
7. "The Art of Deception: Controlling the Human Element of Security" - Kevin D. Mitnick va William L. Simon (2002)
8. "The Shellcoder's Handbook: Discovering and Exploiting Security Holes" - Chris Anley, John Heasman, Felix Lindner, Gerardo Richarte (2007)
9. "Security Engineering: A Guide to Building Dependable Distributed Systems" - Ross J. Anderson (2008)
10. "The Basics of Hacking and Penetration Testing: Ethical Hacking and Penetration Testing Made Easy" - Patrick Engebretson (2013).