Content Discovery

What is Content Discovery?

In the context of web application security, content can be many things: a file, video, picture, backup, or website feature. When we talk about content discovery, we're not talking about the obvious things we can see on a website; it's the things that aren't immediately presented to us and that weren't always intended for public access.

This content could be, for example:

- Pages or portals intended for staff usage
- Older versions of the website
- Backup files
- Configuration files
- Administration panels

There are **three main ways** of discovering content on a website:

1. **Manually**
2. **Automated**
3. **OSINT** (Open-Source Intelligence)

Manual Discovery Methods

There are multiple places we can manually check on a website to start discovering more content.

Robots.txt

The `robots.txt` file is a document that tells search engines which pages they are and aren't allowed to show on their search engine results or ban specific search engines from crawling the

website altogether. It can be common practice to restrict certain website areas so they aren't displayed in search engine results. These pages may be areas such as administration portals or files meant for the website's customers. This file gives us a great list of locations on the website that the owners don't want us to discover as penetration testers.

Practical Exercise:

```
http://MACHINE_IP/robots.txt
```

Favicon

The favicon is a small icon displayed in the browser's address bar or tab used for branding a website. Sometimes when frameworks are used to build a website, a favicon that is part of the installation gets leftover, and if the website developer doesn't replace this with a custom one, this can give us a clue on what framework is in use.

OWASP hosts a database of common framework icons that you can use to check against the target's favicon:

- https://wiki.owasp.org/index.php/OWASP_favicon_database

Practical Exercise:

Visit: https://static-labs.tryhackme.cloud/sites/favicon/

To download and hash the favicon:

Using curl (Linux/Mac):

```
curl https://static-labs.tryhackme.cloud/sites/favicon/images/favicon.ico | md5sum
```

Using PowerShell (Windows):

```
PS C:\> curl https://static-labs.tryhackme.cloud/sites/favicon/images/favicon.ico -
UseBasicParsing -o favicon.ico
PS C:\> Get-FileHash .\favicon.ico -Algorithm MD5
```

> **Note:** If your hash ends with 427e then your curl failed, and you may need to try it again.

Sitemap.xml

Unlike the `robots.txt` file, which restricts what search engine crawlers can look at, the `sitemap.xml` file gives a list of every file the website owner wishes to be listed on a search engine. These can sometimes contain areas of the website that are a bit more difficult to navigate to or even list some old webpages that the current site no longer uses but are still working behind the scenes.

Example:

```
http://10.67.149.33/sitemap.xml
```

HTTP Headers

When we make requests to the web server, the server returns various HTTP headers. These headers can sometimes contain useful information such as the webserver software and possibly the programming/scripting language in use.

Example using curl:

```
curl http://10.67.149.33 -v
```

Sample output:

```
*   Trying 10.67.149.33:80...
* TCP_NODELAY set
* Connected to 10.67.149.33 (10.67.149.33) port 80 (#0)
> GET / HTTP/1.1
> Host: 10.67.149.33
> User-Agent: curl/7.68.0
> Accept: */*
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
```

```
< Server: nginx/1.18.0 (Ubuntu)
< X-Powered-By: PHP/7.4.3
< Date: Mon, 19 Jul 2021 14:39:09 GMT
< Content-Type: text/html; charset=UTF-8
< Transfer-Encoding: chunked
< Connection: keep-alive
```

In this example, we can see the webserver is NGINX version 1.18.0 and runs PHP version 7.4.3. Using this information, we could find vulnerable versions of software being used.

Framework Stack

Once you've established the framework of a website, either from the favicon example above or by looking for clues in the page source such as comments, copyright notices or credits, you can then locate the framework's website. From there, we can learn more about the software and other information, possibly leading to more content we can discover.

Looking at the page source, you might find comments linking to framework documentation, which can reveal paths to administration portals or other sensitive areas.

OSINT (Open-Source Intelligence) Methods

There are external resources available that can help in discovering information about your target website. These resources are freely available tools that collect information.

Google Hacking / Dorking

Google hacking/Dorking utilizes Google's advanced search engine features, which allow you to pick out custom content.

Common Google Dork Filters:

Filter	Example	Description

site	site:tryhackme.com	Returns results only from the specified website address
inurl	inurl:admin	Returns results that have the specified word in the URL
filetype	filetype:pdf	Returns results which are a particular file extension
intitle	intitle:admin	Returns results that contain the specified word in the title

Example usage:

```
site:tryhackme.com admin
```

This would only return results from the tryhackme.com website which contain the word "admin" in its content.

More information: https://en.wikipedia.org/wiki/Google_hacking

Wappalyzer

Wappalyzer (https://www.wappalyzer.com/) is an online tool and browser extension that helps identify what technologies a website uses, such as:

- Frameworks
- Content Management Systems (CMS)
- Payment processors
- Version numbers

Wayback Machine

The Wayback Machine (https://archive.org/web/) is a historical archive of websites that dates back to the late 90s. You can search a domain name, and it will show you all the times the service scraped the web page and saved the contents. This service can help uncover old pages that may still be active on the current website.

GitHub

Git is a **version control system** that tracks changes to files in a project. GitHub is a hosted version of Git on the internet. Repositories can either be set to public or private and have various

access controls.

You can use GitHub's search feature to look for:

- Company names
- Website names
- Source code
- Passwords or credentials
- Configuration files

Once discovered, you may have access to content that you hadn't yet found.

S3 Buckets

S3 Buckets are a storage service provided by Amazon AWS, allowing people to save files and even static website content in the cloud accessible over HTTP and HTTPS. Sometimes access permissions are incorrectly set and inadvertently allow access to files that shouldn't be available to the public.

S3 Bucket Format:

```
http(s)://{name}.s3.amazonaws.com
```

Common naming patterns:

- {name}-assets
- {name}-www
- {name}-public
- {name}-private

S3 buckets can be discovered by:

- Finding URLs in the website's page source
- Searching GitHub repositories
- Automating the process with common naming patterns

Automated Discovery

What is Automated Discovery?

Automated discovery is the process of using tools to discover content rather than doing it manually. This process is automated as it usually contains hundreds, thousands or even millions of requests to a web server. These requests check whether a file or directory exists on a website, giving us access to resources we didn't previously know existed.

What are Wordlists?

Wordlists are text files that contain a long list of commonly used words. For content discovery, we require lists containing the most commonly used directory and file names.

Recommended wordlist resource:

- SecLists by Daniel Miessler: https://github.com/danielmiessler/SecLists

Automation Tools

Three popular content discovery tools:

1. ffuf

```
ffuf -w /usr/share/wordlists/SecLists/Discovery/Web-Content/common.txt -u
http://10.67.149.33/FUZZ
```

2. dirb

```
dirb http://10.67.149.33/ /usr/share/wordlists/SecLists/Discovery/Web-Content/common.txt
```

3. Gobuster

```
gobuster dir --url http://10.67.149.33/ -w /usr/share/wordlists/SecLists/Discovery/Web-
Content/common.txt
```

Summary

Content discovery combines multiple techniques:

- **Manual methods** help understand the structure and technology stack
- **OSINT** provides external information and historical data
- **Automated tools** efficiently scan for hidden directories and files

Each method has its strengths, and using them in combination provides the most comprehensive results for web application security testing.

Revision #1
Created 2026-01-03 17:53:16 UTC by David Rizzo
Updated 2026-01-03 17:53:34 UTC by David Rizzo