Content Discovery

**Task 1 – What is Content Discovery?**

Firstly, we should ask, in the context of web application security, what is content? Content can be many things, a file, video, picture, backup, a 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, etc.

There are three main ways of discovering content on a website which we'll cover. Manually, Automated and OSINT (Open-Source Intelligence).

**Answer the question:**

1.What is the Content Discovery method that begins with M?

Ans – Manually

2. What is the Content Discovery method that begins with A?

Ans- Automated

3. What is the Content Discovery method that begins with O?

Ans - OSINT

**Task 2 – Manual Discovery – Robots.txt**

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

Q - What is the directory in the robots.txt that isn't allowed to be viewed by web crawlers?

Ans - /staf-portal

**Task 3 – Manual Discovery - Favicon**

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 host a database of common framework icons that you can use to check against the targets favicon https://wiki.owasp.org/index.php/OWASP\_favicon\_database. Once we know the framework stack, we can use external resources to discover more about it (see next section).

**Task 4 – Manual Discovery – Sitemap.xml**

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.

Take a look at the sitemap.xml file on the Acme IT Support website to see if there's any new content we haven't yet discovered: http://10.10.249.97/sitemap.xml

**Task 5 – Manual Discovery – HTTP Headers**

**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. In the below 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. Try running the below curl command against the web server, where the -v switch enables verbose mode, which will output the headers (there might be something interesting!).

Curl <http://10.10.249.97/> -v

\* Trying 10.10.249.97:80...

\* Connected to 10.10.249.97 (10.10.249.97) port 80 (#0)

> GET / HTTP/1.1

> Host: 10.10.249.97

< Server: nginx/1.18.0 (Ubuntu)

< Date: Wed, 20 Oct 2021 14:37:03 GMT

< Content-Type: text/html; charset=UTF-8

< Transfer-Encoding: chunked

< X-FLAG: THM{HEADER\_FLAG}

**Task 6 – Manual Discovery – Framework Stack**

Framework Stack

Once you've established the framework of a website, either from the above favicon example 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 of our Acme IT Support website (http://10.10.249.97), you'll see a comment at the end of every page with a page load time and also a link to the framework's website, which is https://static-labs.tryhackme.cloud/sites/thm-web-framework. Let's take a look at that website. Viewing the documentation page gives us the path of the framework's administration portal, which gives us a flag if viewed on the Acme IT Support website.

**Task 7 – OSINT – Google Hacking / Dorking**

There are also external resources available that can help in discovering information about your target website; these resources are often referred to as OSINT or (Open-Source Intelligence) as they're 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. You can, for instance, pick out results from a certain domain name using the site: filter, for example (site:tryhackme.com) you can then match this up with certain search terms, say, for example, the word admin (site:tryhackme.com admin) this then would only return results from the tryhackme.com website which contain the word admin in its content. You can combine multiple filters as well. Here is an example of more filters you can use:

|  |  |  |
| --- | --- | --- |
| **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 |

More information about google hacking can be found here: <https://en.wikipedia.org/wiki/Google_hacking>

**Task 8 – OSINT - Wappalyzer**

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 and much more, and it can even find version numbers as well.

**Task 9 – OSINT – Wayback Machine**

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.

Q. What is the website address for the Wayback machin ?

Ans - https://archive.org/web/)

**Task 10 – OSINT – Github**

To understand GitHub, you first need to understand Git. Git is a version control system that tracks changes to files in a project. Working in a team is easier because you can see what each team member is editing and what changes they made to files. When users have finished making their changes, they commit them with a message and then push them back to a central location (repository) for the other users to then pull those changes to their local machines. 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 or website names to try and locate repositories belonging to your target. Once discovered, you may have access to source code, passwords or other content that you hadn't yet found.

**Task 11 – OSINT – S3 Buckets**

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. The owner of the files can set access permissions to either make files public, private and even writable. Sometimes these access permissions are incorrectly set and inadvertently allow access to files that shouldn't be available to the public. The format of the S3 buckets is http(s)://{name}.s3.amazonaws.com where {name} is decided by the owner, such as tryhackme-assets.s3.amazonaws.com. S3 buckets can be discovered in many ways, such as finding the URLs in the website's page source, GitHub repositories, or even automating the process. One common automation method is by using the company name followed by common terms such as {name}-assets, {name}-www, {name}-public, {name}-private, etc.

Q - What URL format do amazon S3 buckets end in?

Ans - .s3.amazon.com

**Task 12 – 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. This process is made possible by using a resource called wordlists.

What are wordlists?

Wordlists are just text files that contain a long list of commonly used words; they can cover many different use cases. For example, a password wordlist would include the most frequently used passwords, whereas we're looking for content in our case, so we'd require a list containing the most commonly used directory and file names. An excellent resource for wordlists that is preinstalled on the THM AttackBox is https://github.com/danielmiessler/SecLists which Daniel Miessler curates.

Automated Tools

Although there are many different content discovery tools available, all with their features and flaws, we're going to cover three which are preinstalled on our attack box, ffuf, dirb and gobuster.

Using ffuf:

ffuf -w /usr/share/wordlists/SecLists/Discovery/Web-Content/common.txt -u http://10.10.249.97/FUZZ

Using dirb:

user@machine$ dirb http://10.10.249.97/ /usr/share/wordlists/SecLists/Discovery/Web-Content/common.txt

Using Gobuster:

dir --url http://MACHINE\_IP/ -w /usr/share/wordlists/SecLists/Discovery/Web-Content/common.txt