# robots.txt

Imagine you're a guest at a grand house party. While you're free to mingle and explore, there might be certain rooms marked "Private" that you're expected to avoid. This is akin to how `robots.txt` functions in the world of web crawling. It acts as a virtual "`etiquette guide`" for bots, outlining which areas of a website they are allowed to access and which are off-limits.

## What is robots.txt?

Technically, `robots.txt` is a simple text file placed in the root directory of a website (e.g., `www.example.com/robots.txt`). It adheres to the Robots Exclusion Standard, guidelines for how web crawlers should behave when visiting a website. This file contains instructions in the form of "directives" that tell bots which parts of the website they can and cannot crawl.

### How robots.txt Works

The directives in robots.txt typically target specific user-agents, which are identifiers for different types of bots. For example, a directive might look like this:

Code: txt

```
User-agent: *
Disallow: /private/
```

This directive tells all user-agents (`*` is a wildcard) that they are not allowed to access any URLs that start with `/private/`. Other directives can allow access to specific directories or files, set crawl delays to avoid overloading a server or provide links to sitemaps for efficient crawling.

### Understanding robots.txt Structure

The robots.txt file is a plain text document that lives in the root directory of a website. It follows a straightforward structure, with each set of instructions, or "record," separated by a blank line. Each record consists of two main components:

1. `User-agent`: This line specifies which crawler or bot the following rules apply to. A wildcard (`*`) indicates that the rules apply to all bots. Specific user agents can also be targeted, such as "Googlebot" (Google's crawler) or "Bingbot" (Microsoft's crawler).
2. `Directives`: These lines provide specific instructions to the identified user-agent.

Common directives include:

| Directive | Description | Example |
|---|---|---|
| `Disallow` | Specifies paths or patterns that the bot should not crawl. | `Disallow: /admin/` (disallow access to the admin directory) |
| `Allow` | Explicitly permits the bot to crawl specific paths or patterns, even if they fall under a broader `Disallow` rule. | `Allow: /public/` (allow access to the public directory) |
| `Crawl-delay` | Sets a delay (in seconds) between successive requests from the bot to avoid overloading the server. | `Crawl-delay: 10` (10-second delay between requests) |
| `Sitemap` | Provides the URL to an XML sitemap for more efficient crawling. | `Sitemap: https://www.example.com/sitemap.xml` |

### Why Respect robots.txt?

While robots.txt is not strictly enforceable (a rogue bot could still ignore it), most legitimate web crawlers and search engine bots will respect its directives. This is important for several reasons:

- `Avoiding Overburdening Servers`: By limiting crawler access to certain areas, website owners can prevent excessive traffic that could slow down or even crash their servers.
- `Protecting Sensitive Information`: Robots.txt can shield private or confidential information from being indexed by search engines.
- `Legal and Ethical Compliance`: In some cases, ignoring robots.txt directives could be considered a violation of a website's terms of service or even a legal issue, especially if it involves accessing copyrighted or private data.

## robots.txt in Web Reconnaissance

For web reconnaissance, robots.txt serves as a valuable source of intelligence. While respecting the directives outlined in this file, security professionals can glean crucial insights into the structure and potential vulnerabilities of a target website:

- `Uncovering Hidden Directories`: Disallowed paths in robots.txt often point to directories or files the website owner intentionally wants to keep out of reach from search engine crawlers. These hidden areas might house sensitive information, backup files, administrative panels, or other resources that could interest an attacker.
- `Mapping Website Structure`: By analyzing the allowed and disallowed paths, security professionals can create a rudimentary map of the website's structure. This can reveal sections that are not linked from the main navigation, potentially leading to undiscovered pages or functionalities.
- `Detecting Crawler Traps`: Some websites intentionally include "honeypot" directories in robots.txt to lure malicious bots. Identifying such traps can provide insights into the target's security awareness and defensive measures.

### Analyzing robots.txt

Here's an example of a robots.txt file:

Cheat Sheet

Table of Contents

My Workstation

OFFLINE

Start Instance

∞ / 1 spawns left

Code: txt

```
User-agent: *
Disallow: /admin/
Disallow: /private/
Allow: /public/

User-agent: Googlebot
Crawl-delay: 10

Sitemap: https://www.example.com/sitemap.xml
```

This file contains the following directives:

- All user agents are disallowed from accessing the `/admin/` and `/private/` directories.
- All user agents are allowed to access the `/public/` directory.
- The `Googlebot` (Google's web crawler) is specifically instructed to wait 10 seconds between requests.
- The sitemap, located at `https://www.example.com/sitemap.xml`, is provided for easier crawling and indexing.

By analyzing this robots.txt, we can infer that the website likely has an admin panel located at `/admin/` and some private content in the `/private/` directory.