Crawling

`Crawling`, often called `spidering`, is the `automated process of systematically browsing the World Wide Web`. Similar to how a spider navigates its web, a web crawler follows links from one page to another, collecting information. These crawlers are essentially bots that use pre-defined algorithms to discover and index web pages, making them accessible through search engines or for other purposes like data analysis and web reconnaissance.

How Web Crawlers Work

The basic operation of a web crawler is straightforward yet powerful. It starts with a seed URL, which is the initial web page to crawl. The crawler fetches this page, parses its content, and extracts all its links. It then adds these links to a queue and crawls them, repeating the process iteratively. Depending on its scope and configuration, the crawler can explore an entire website or even a vast portion of the web.

1. `Homepage`: You start with the homepage containing `link1`, `link2`, and `link3`.

```txt
Homepage
├── link1
├── link2
└── link3
```

2. `Visiting link1`: Visiting `link1` shows the homepage, `link2`, and also `link4` and `link5`.

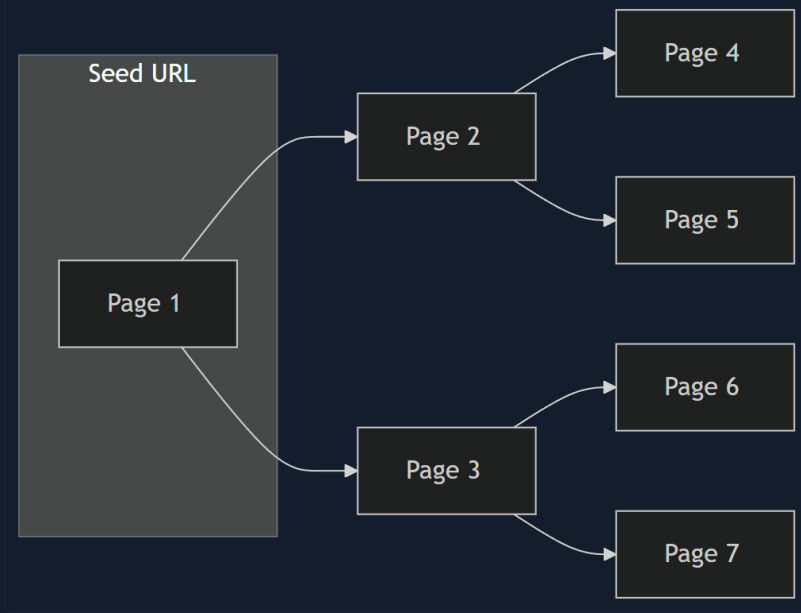
```txt
link1 Page
├── Homepage
├── link2
├── link4
└── link5
```

3. `Continuing the Crawl`: The crawler continues to follow these links systematically, gathering all accessible pages and their links.

This example illustrates how a web crawler discovers and collects information by systematically following links, distinguishing it from fuzzing which involves guessing potential links.

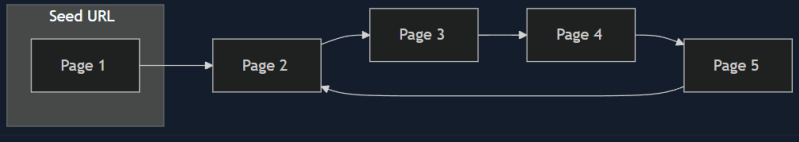
There are two primary types of crawling strategies.

Breadth-First Crawling



`Breadth-first crawling` prioritizes exploring a website's width before going deep. It starts by crawling all the links on the seed page, then moves on to the links on those pages, and so on. This is useful for getting a broad overview of a website's structure and content.

Depth-First Crawling



In contrast, `depth-first crawling` prioritizes depth over breadth. It follows a single path of links as far as possible before backtracking and exploring other paths. This can be useful for finding specific content or reaching deep into a website's structure.

My Workstation

OFFLINE

Start Instance

∞ / 1 spawns left

The choice of strategy depends on the specific goals of the crawling process.

Extracting Valuable Information

Crawlers can extract a diverse array of data, each serving a specific purpose in the reconnaissance process:

- `Links (Internal and External)`: These are the fundamental building blocks of the web, connecting pages within a website (`internal links`) and to other websites (`external links`). Crawlers meticulously collect these links, allowing you to map out a website's structure, discover hidden pages, and identify relationships with external resources.
- `Comments`: Comments sections on blogs, forums, or other interactive pages can be a goldmine of information. Users often inadvertently reveal sensitive details, internal processes, or hints of vulnerabilities in their comments.
- `Metadata`: Metadata refers to `data about data`. In the context of web pages, it includes information like page titles, descriptions, keywords, author names, and dates. This metadata can provide valuable context about a page's content, purpose, and relevance to your reconnaissance goals.
- `Sensitive Files`: Web crawlers can be configured to actively search for sensitive files that might be inadvertently exposed on a website. This includes `backup files` (e.g., `.bak`, `.old`), `configuration files` (e.g., `web.config`, `settings.php`), `log files` (e.g., `error_log`, `access_log`), and other files containing passwords, `API keys`, or other confidential information. Carefully examining the extracted files, especially backup and configuration files, can reveal a trove of sensitive information, such as `database credentials`, `encryption keys`, or even source code snippets.

The Importance of Context

Understanding the context surrounding the extracted data is paramount.

A single piece of information, like a comment mentioning a specific software version, might not seem significant on its own. However, when combined with other findings—such as an outdated version listed in metadata or a potentially vulnerable configuration file discovered through crawling—it can transform into a critical indicator of a potential vulnerability.

The true value of extracted data lies in connecting the dots and constructing a comprehensive picture of the target's digital landscape.

For instance, a list of extracted links might initially appear mundane. But upon closer examination, you notice a pattern: several URLs point to a directory named `/files/`. This triggers your curiosity, and you decide to manually visit the directory. To your surprise, you find that directory browsing is enabled, exposing a host of files, including backup archives, internal documents, and potentially sensitive data. This discovery wouldn't have been possible by merely looking at individual links in isolation; the contextual analysis led you to this critical finding.

Similarly, seemingly innocuous comments can gain significance when correlated with other discoveries. A comment mentioning a "file server" might not raise any red flags initially. However, when combined with the aforementioned discovery of the `/files/` directory, it reinforces the possibility that the file server is publicly accessible, potentially exposing sensitive information or confidential data.

Therefore, it's essential to approach data analysis holistically, considering the relationships between different data points and their potential implications for your reconnaissance goals.