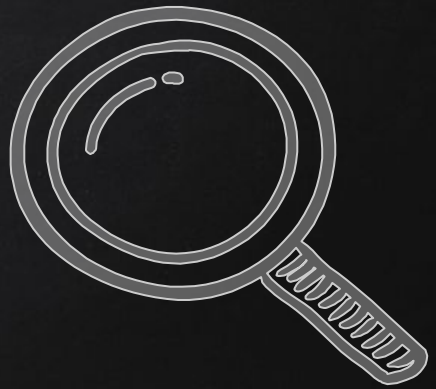
# Information Gathering

- IP address.
- Domain name info.
- Technologies used.
- Other websites on the same server.
- DNS records.
- Files, sub-domains, directories.



# Crawling

## Subdomains

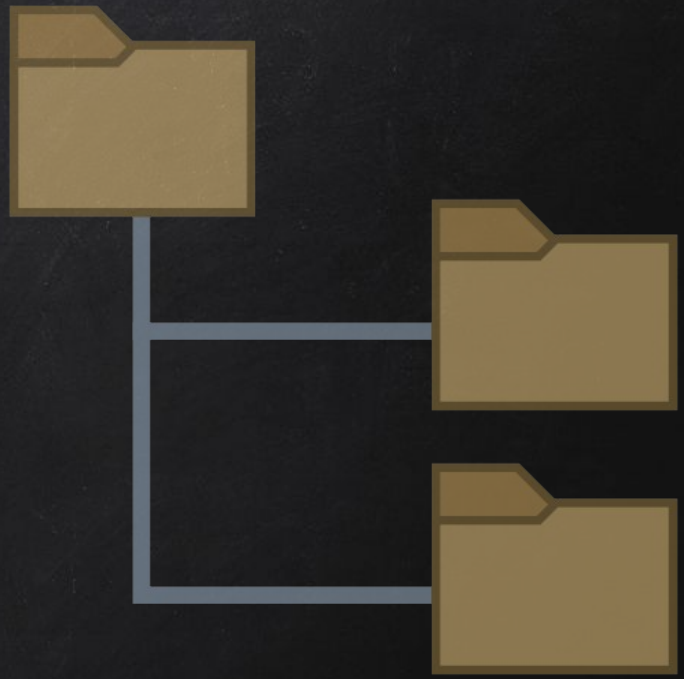
- Domain before the actual domain name.
- Part of the main domain.

Ex:

- subdomain.target.com
- mail.google.com
- plus.google.com

# Crawling

## Directories



- Directories/folders inside the web root.
- Can contain files or other directories.

Ex:

- target.com/directory
- plus.google.com/discover

# Crawling

## Summary

Our crawler so far can **guess**:

- Subdomains.
- Directories.
- Files.

Advantages:

→ Discover "hidden" paths/paths admin does not want us to know.

Disadvantage:

→ Will not discover everything.

# Crawling

## Summary

Advantages:
→ Discover "hidden" paths/paths admin does not want us to know.

Disadvantage:
→ Will not discover everything.

Solution:
→ Analyse discovered paths to discover more paths.

# Crawling

## Spider

Goal → Recursively list all links starting from a base url.

1. Read page html.
2. Extract all links.
3. Repeat for each new link that is not already in the list

# Lists

- List of values/elements, all can be stored in one variable.

Ex:

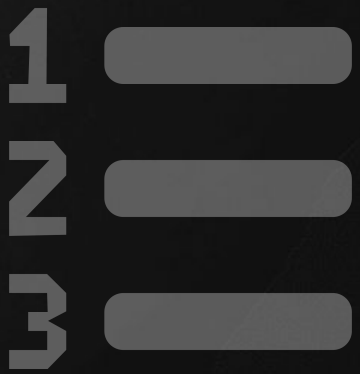
```
lucky_numbers_list = [3, 7, 8, 17, 24]
```

Python will interpret this as

| index | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| value | 3 | 7 | 8 | 17 | 24 |

Elements can be accessed using their index

```
print(lucky_numbers_list[0])  #prints 3
print(lucky_numbers_list[1])  #prints 7
print(lucky_numbers_list[2])  #prints 8
```



# Lists

- List of values/elements, all can be stored in one variable.

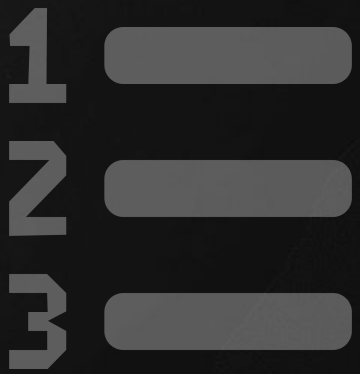
Ex:

lucky_numbers_list = [3, 7, 8, 17, 24]

| index | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| value | 3 | 7 | 8 | 17 | 24 |

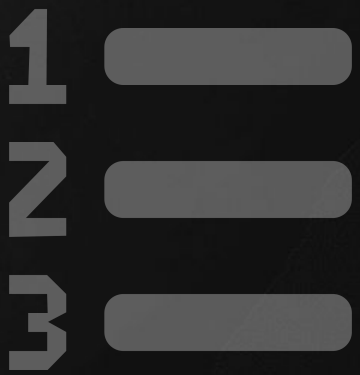
lucky_numbers_list.pop[2])

| index | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| value | 3 | 7 | 17 | 24 |



# Website Hacking

1. Information gathering.
2. Discover vulnerabilities.
3. Exploit discovered vulnerabilities.



# HTTP Requests

## Basic information Flow

- User clicks on a link.
- HTML website generates a request (client side)
- Request is sent to the server.
- Server performs the request (Server Side)
- Sends response back.

# HTTP Requests - GET vs POST

Two main methods used to send data to the web application:

1. Through the URL (Usually using GET).
    a. http://website.com/news.php?id=1
    b. http://website.com/?id=1

2. Through input elements (usually using POST):
    a. Search boxes.
    b. Login boxes
    c. .....etc

# Vulnerability_scanner

How to discover a vulnerability in a web application?

1. Go into every possible page.
2. Look for ways to send data to the web application (URL + Forms).
3. Send payloads to discover vulnerabilities.
4. Analyse the response to check of the website is vulnerable.

→ General steps are the same regardless of the vulnerability.