# Chapter 1

# Web and Browser Security

## 1.1 Introduction

**Definition 1.1.1 ▶ Domain Name**

A ***domain name*** is a human-readable name for an internet service.

For simplicity, we will assume that a domain name can only have one associated service/server, and vice versa (i.e. a one-to-one mapping between domain names and servers).

**Definition 1.1.2 ▶ Domain Name System (DNS)**

A ***Domain Name System*** maps domain names to their corresponding server's IP address. We say the DNS ***resolves*** a domain name to mean the DNS converts a domain name into an IP address.

It's common for computers to store the IP addresses for one or two DNS services. By default, a DNS has no integrity (that is, an attacker on the network could ). DNSSEC solves this issue but has low adoption.

what could they do tho

**Definition 1.1.3 ▶ Top-Level Domain (TLD), Second-Level Domain, Subdomain, Fully-Qualified Domain Name (FQDN)**

Consider the following domain name:

`userlab.utk.edu`

- `edu` constitutes the ***top-level domain (TLD)***
- `utk` constitutes the ***second-level domain***
- `userlab` constitutes the ***subdomain*** which provides further naming specification for the server. It can have several "levels" and is assumed to be controlled by its parent (sub)domain.
- `userlab.utk.edu` consitutes the ***fully-qualified domain name (FQDN)***, which is sent to the DNS to be resolved into an IP address

In addition, a domain name is the combination of a second-level domain and a top-level domain.

Machines typically parse domain names from right to left, starting from the most broad identifiers and working towards the most specific ones.

`www` is a common subdomain which is sort of an artifact of the internet's inception. In the early days, people wanted to separate the website from other services from the domain name. There is no guarantee that `www.website.com` hosts the same data as `website.com`. Now, most websites specify `website.com` to simply redirect to `www.website.com`.

**Technique 1.1.4 ▸ Resolving Domain Names**

Suppose we have some servers with domain name `amazon.com` and Alice who wants to connect to the servers.

- Alice connects to her intermediate access point that actually connects to the internet (most commonly a router).
- Alice prompts the DNS for the IP to `amazon.com`.
- The DNS resolves the IP address and gives it to Alice.
- Alice connects to the resolved IP address.

**Definition 1.1.5 ▸ Uniform Resource Locator (URL)**

A ***URL*** identifies specific resources on a server. This contrasts from a domain which only specifies which server to connect to.

**Definition 1.1.6 ▸ Scheme, Host, Port, Pathname, Query**

Consider the following URL:

```
scheme://host[:port]/pathname[?query]
```

The square brackets denote optional fields.

- The ***scheme*** is the protocol used to communicate with the server (e.g. http, https, ftp, ftps, ws, etc.).
- The ***host*** is usually the domain name or IP address of the server.
- The ***port*** specifies the port to connect to. If omitted, it uses default values for known schemes (e.g. 80 for http, 443 for https).
- The ***pathname*** identifies a specific resource on the server. It resembles a UNIX file system path, but it does not actually need to map to the underlying file system.
- The ***query*** is addition data that further specifies the requested resource. It always starts with `?` and is a collection of key-value pairs, separated by `&`.

When encoding URLs, there are only a certain set of characters that is allowed. Non-allowed characters are encoded as their hexadecimal representation prepended with a `%`.

**Definition 1.1.7 ▸ Hypertext Transfer Protocol (HTTP)**

***HTTP*** is a common request-response protocol for serving web content. It is built on TCP (ensures reliable data transfer) and is a stateless protocol (that is, connections are all independent of each other at the protocol level; no data is stored about the request itself).

HTTP was originally designed to only serve Hypertext Markup Language (HTML), but is now used to serve general content.

**Definition 1.1.8 ▸ HTTP Methods**

HTTP requests are sent similar to the following string:

$$\texttt{METHOD /pathname?query HTTP VER Host:}$$

$$\texttt{hostname:port} < keyword >\texttt{:}< value >< requestbody >$$

- ***GET:*** retrieve data from the URL

- ***POST:*** push data to the URL
- Others include HEAD, PUT, DELETE, TRACE, CONNECT, OPTIONS, PATCH
-

Although the HTTP spec specifies the intended uses of each method, servers can implement and handle these methods however they choose. So there is generally no standardization of these methods between different servers.