

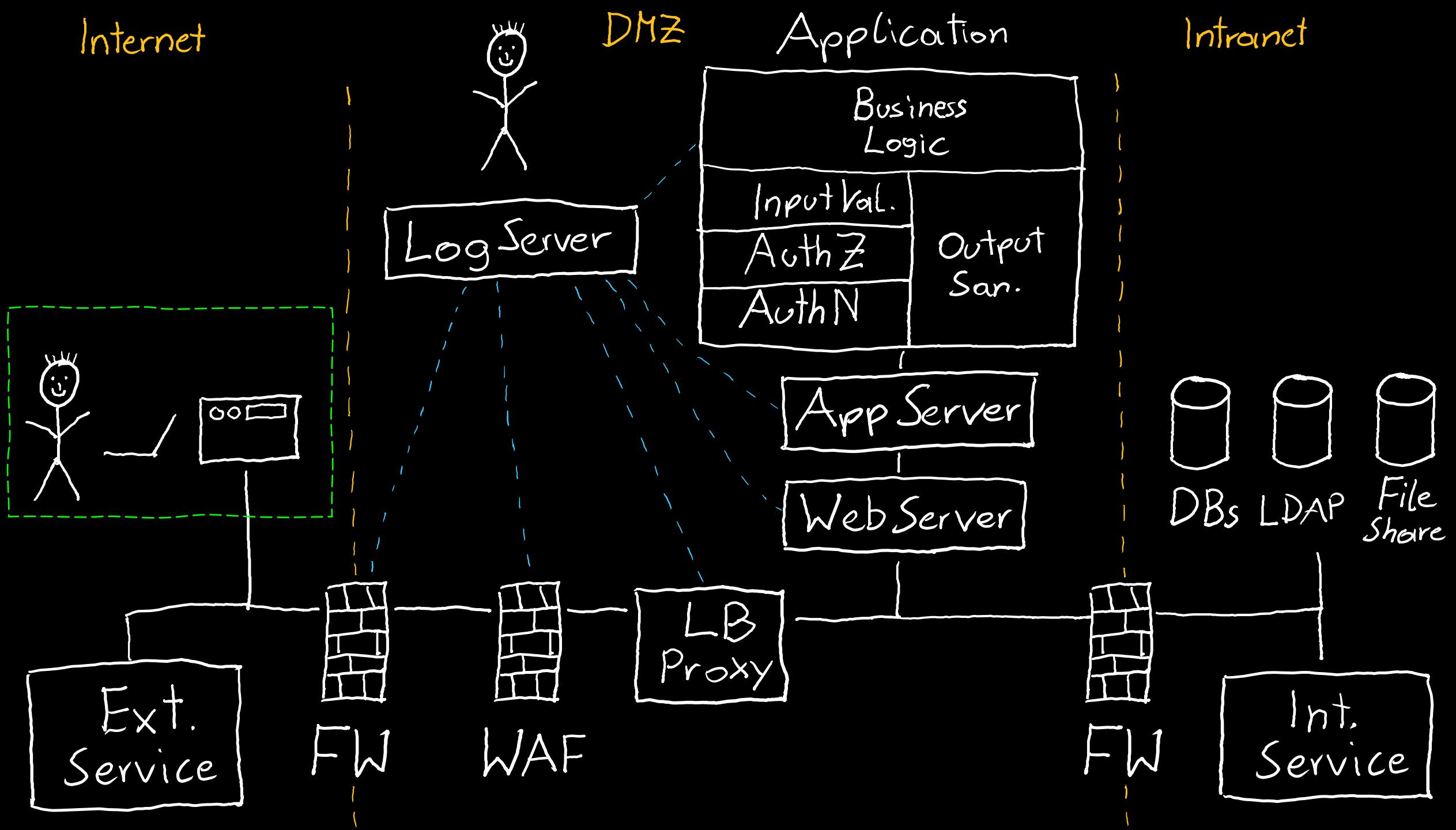
# Architecture & Basic Web Procedure

DNS Hijacking, Information Disclosure,  
Plaintext Transmission

# Rough Overview

1. Introduction
2. Basic Principles and Resources
3. >> Architecture & Basic Web Procedure <<
4. Authentication and Session Management
5. Authorization
6. Server and Backend Attacks
7. Remaining Client Attacks
8. General Topics
9. Conclusions

How does the  
typical architecture of a web  
application look like?



How do you access a website?  
(except for googling it...)

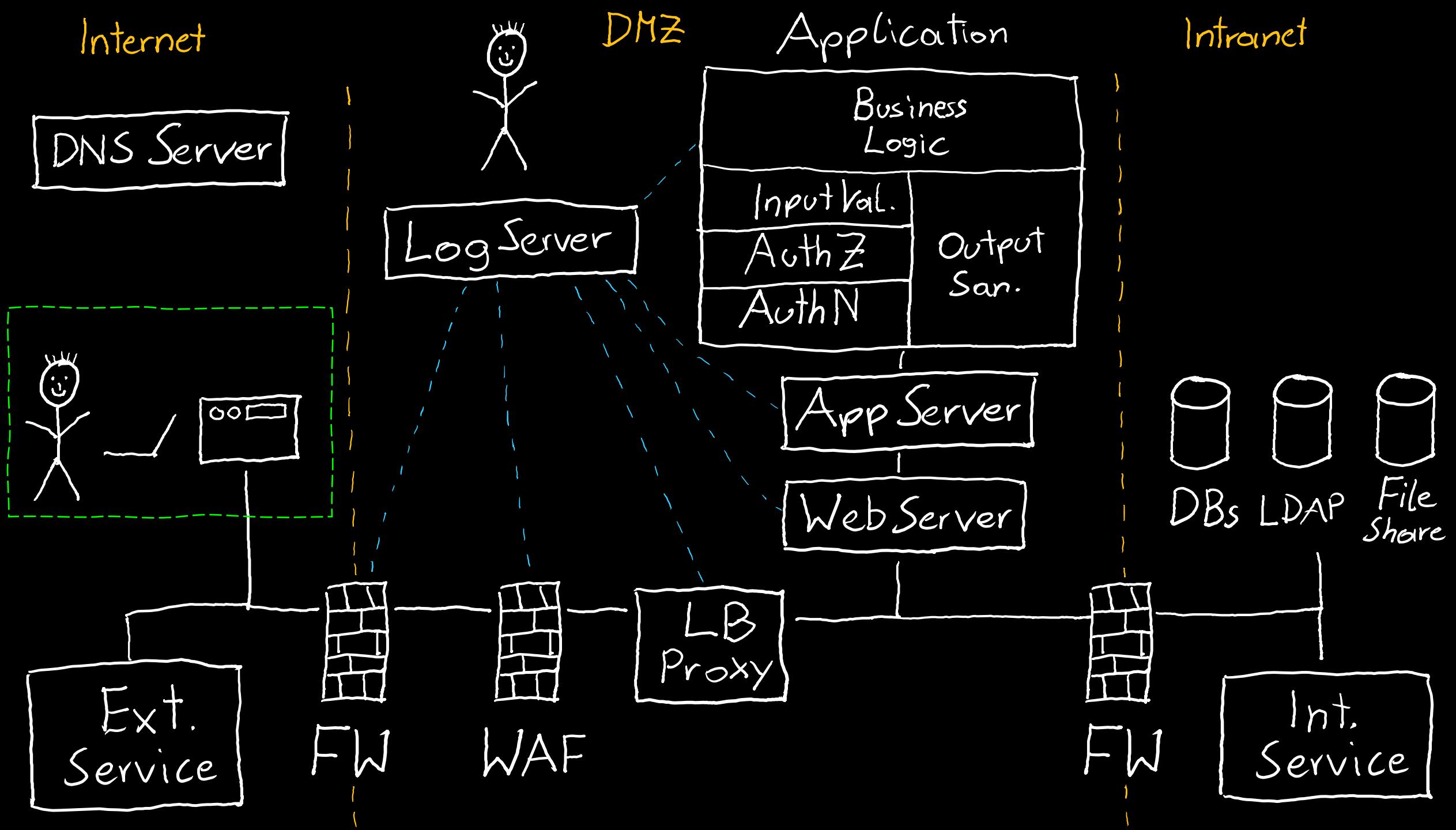
You enter a URL  
*Uniform Resource Locator*

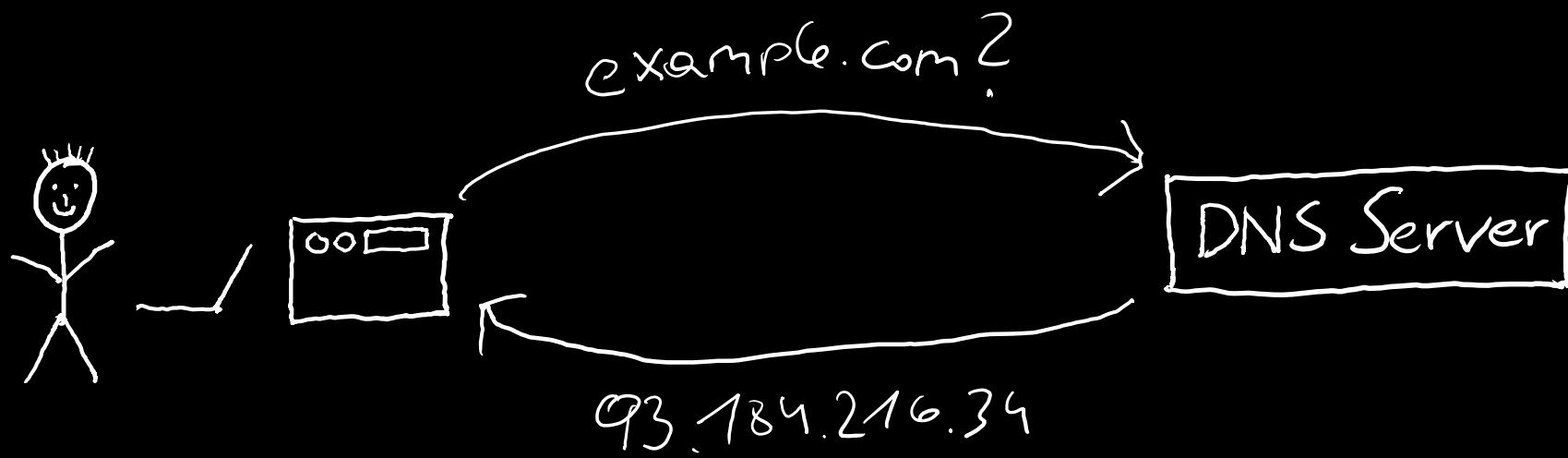
`http://example.com:8080/demos/demo1.html?id=123&name=Luke#ref1`

Protocol	Hostname	Port	Path	Query / Parameters	Fragment
----------	----------	------	------	--------------------	----------

*and hit enter...*

What's next ?

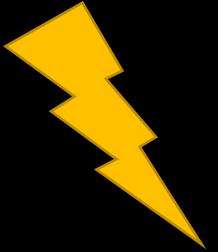




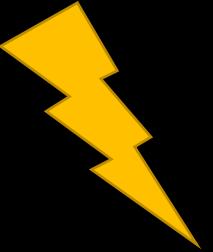
<http://example.com:8080/demos/demo1.html?id=123&name=Luke#ref1>

You already know DNS  
What can go wrong?

# DNS Hijacking

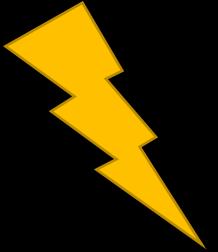


Goal	Redirect user to a rogue server by manipulation of DNS records
How	
Solution	
OWASP Top 10	
(Primary) Violated Principle	



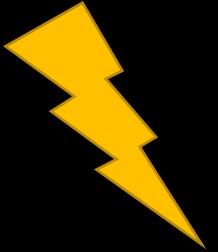
# DNS Hijacking

Goal	Redirect user to a rogue server by manipulation of DNS records
How	Malware on client DNS server attack, e.g. DNS spoofing / cache poisoning MitM between client and DNS server Subdomain takeover
Solution	
OWASP Top 10	
(Primary) Violated Principle	



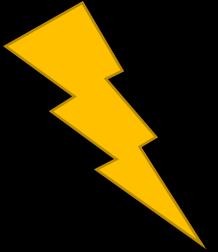
# DNS Hijacking

Goal	Redirect user to a rogue server by manipulation of DNS records
How	Malware on client DNS server attack, e.g. DNS spoofing / cache poisoning MitM between client and DNS server Subdomain takeover
Solution	Hardening DNSSEC / TSIG Be aware of your subdomains
OWASP Top 10	
(Primary) Violated Principle	



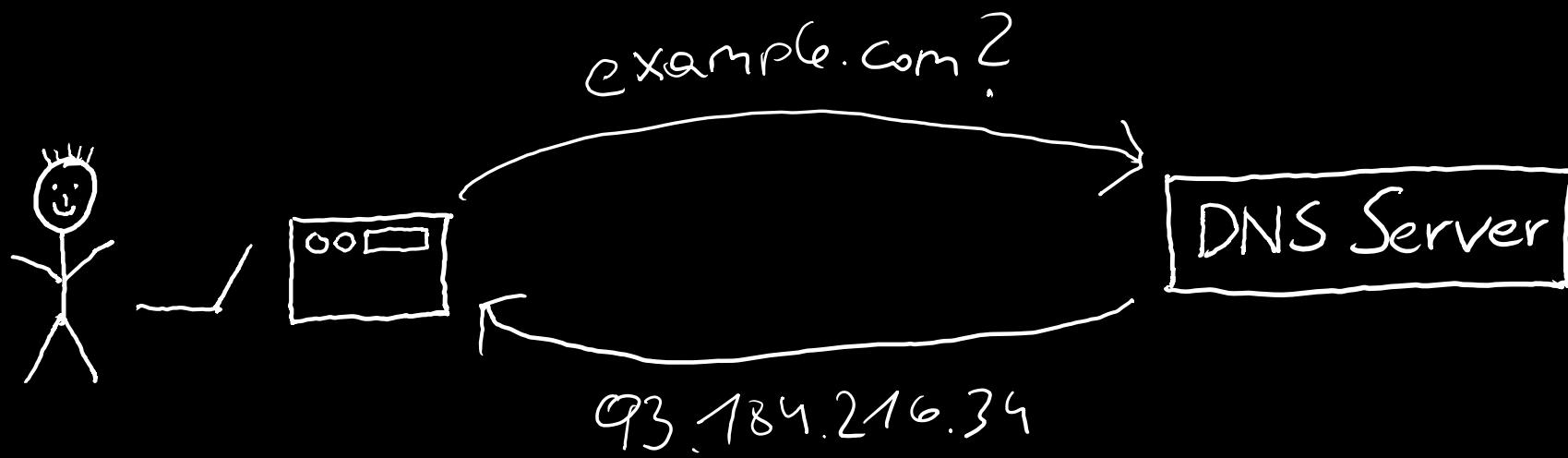
# DNS Hijacking

Goal	Redirect user to a rogue server by manipulation of DNS records
How	Malware on client DNS server attack, e.g. DNS spoofing / cache poisoning MitM between client and DNS server Subdomain takeover
Solution	Hardening DNSSEC / TSIG Be aware of your subdomains
OWASP Top 10	-
(Primary) Violated Principle	



# DNS Hijacking

Goal	Redirect user to a rogue server by manipulation of DNS records
How	Malware on client DNS server attack, e.g. DNS spoofing / cache poisoning MitM between client and DNS server Subdomain takeover
Solution	Hardening DNSSEC / TSIG Be aware of your subdomains
OWASP Top 10	-
(Primary) Violated Principle	„Earn or give, but never assume, trust.“



<http://example.com:8080/demos/demo1.html?id=123&name=Luke#ref1>

# ISO/OSI Model

Application

Presentation

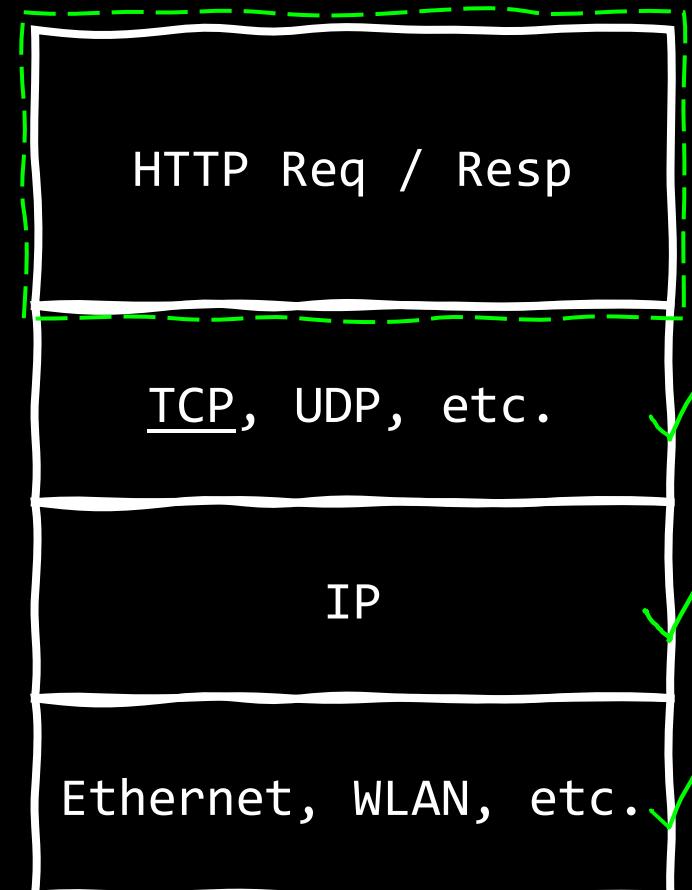
Session

Transport

Network

Data Link  
Physical

# Internet Protocol Suite (TCP/IP Model)

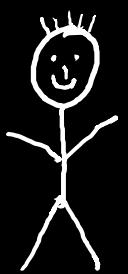


Internet

DMZ

Application

Intranet



Log Server

Business  
Logic

Input Val.

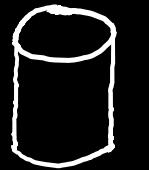
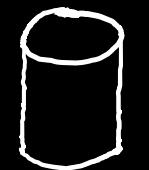
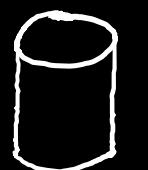
Auth Z

Auth N

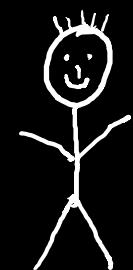
Output  
San.

App Server

Web Server



DBs LDAP File  
Share



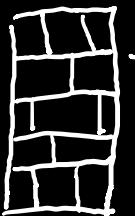
Request



Ext.  
Service

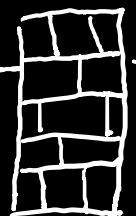


FW



WAF

LB  
Proxy



FW

Int.  
Service



Request

```
> curl -v http://example.com/demos/demo1.html
```

```
-----  
GET /demos/demo1.html HTTP/1.1  
Host: example.com  
User-Agent: curl/7.55.1
```

...

Does every request start like that?

# HTTP Request Verbs / Methods

		classic.com/customers?id=12	restapi.com/customer/12
	Safe / Idempotent	Intended Use	Intended RESTful Use
GET	Y / Y	Read	Read Resource
HEAD	Y / Y	Read (no body)	Read Resource Metadata
POST	N / N	Send something to the server	Create Resource
PUT	N / Y	Update something	Create/Update Resource
DELETE	N / Y	Delete something	Delete Resource
PATCH	N / Y	Update a part of something	Update Resource
CONNECT	N / N	Establish a tunnel	-
OPTIONS	Y / Y	List allowed verbs/methods	-
TRACE	Y / Y	Message loop-back test	-



Request

```
> curl -v http://example.com/demos/demo1.html
```

```
GET /demos/demo1.html HTTP/1.1  
Host: example.com  
User-Agent: curl/7.55.1  
...
```



just the path you specified  
we will talk much more about that later



Request

```
> curl -v http://example.com/demos/demo1.html
```

```
GET /demos/demo1.html [HTTP/1.1]
```

```
Host: example.com
```

```
User-Agent: curl/7.55.1
```

```
...
```

### Protocol version

HTTP/1.0	1996
----------	------

HTTP/1.1	1999
----------	------

<b>HTTP/2.0</b>	<b>2015</b>
-----------------	-------------

<b>HTTP/3.0</b>	<b>in dev.</b>
-----------------	----------------



Request

```
> curl -v http://example.com/demos/demo1.html
```

```
GET /demos/demo1.html HTTP/1.1
```

```
Host: example.com
```

```
User-Agent: curl/7.55.1
```

...

used by the target webserver  
e.g. for name-based virtual hosting  
since HTTP protocol version 1.1



Request

```
> curl -v http://example.com/demos/demo1.html
```

```
GET /demos/demo1.html HTTP/1.1
```

```
Host: example.com
```

```
User-Agent: curl/7.55.1
```

```
...
```

Indicates which program made the request

e.g. to enable mobile view

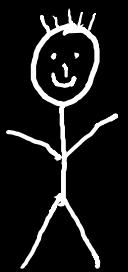
Never rely on it!

Internet

DMZ

Application

Intranet



Log Server

Business Logic

Input Val.

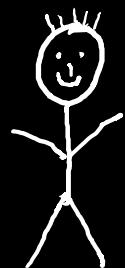
Auth Z

Output San.

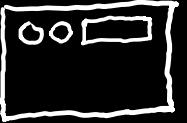
Auth N

App Server

Web Server



Request



Ext. Service



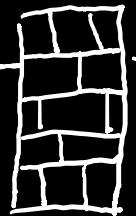
FW



WAF

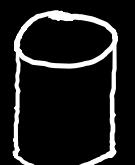
LB Proxy

Request

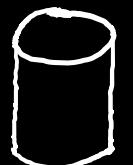


FW

Int. Service



DBs



LDAP



File Share

## Web Server

serves “static” content only  
e.g. pictures, data-files, client-side code (HTML, CSS, JavaScript)

Examples: Apache, nginx, MS IIS, Google Web Server

but no dynamic server-side actions  
e.g. process user input, store/load data from database, etc.

needs server-side programming/scripting languages and a suitable interpreter/runtime  
e.g. PHP, Python, Ruby, ASP.NET, Java ...

this interpreter/runtime can be e.g.:

included in the webserver

e.g. Apache modules (mod\_php / mod\_python / mod\_...)

standalone in an own process and called via

CGI, FastCGI, ISAPI, AJP,...

e.g. Apache Tomcat Java Servlets

standalone in a dedicated application server

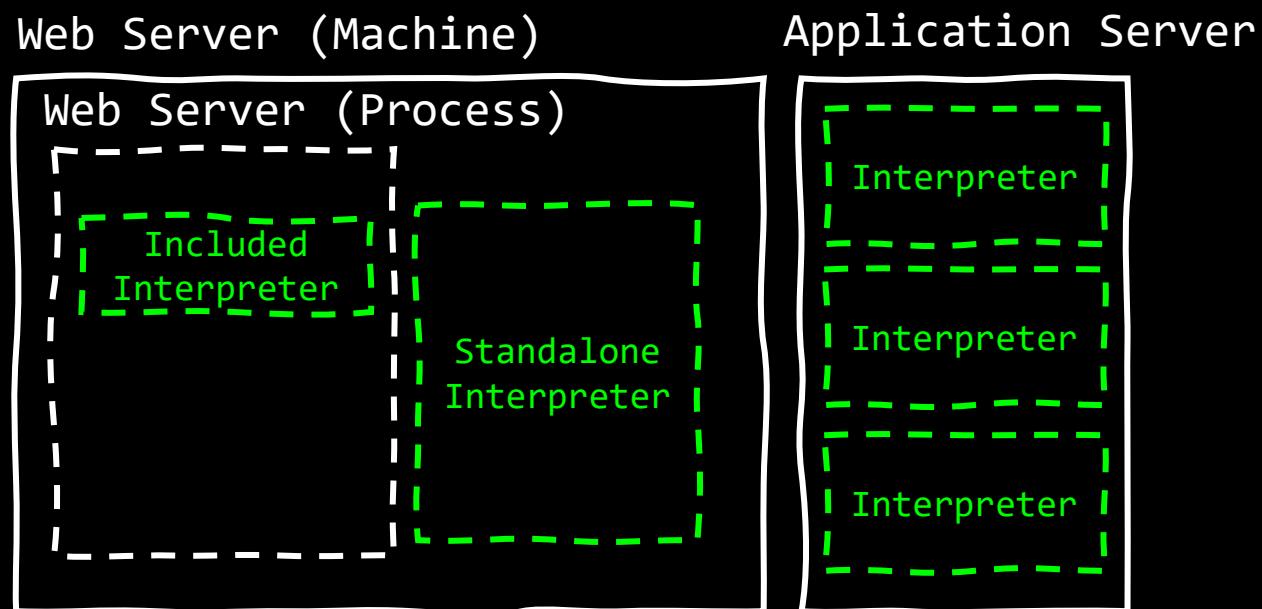
WebLogic

Payara

JBoss

Websphere

...



from an outside view, this is  
completely abstracted

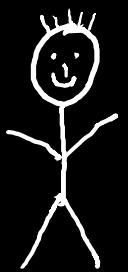
we actually just see „the server“

Internet

DMZ

Application

Intranet



Log Server

Business Logic

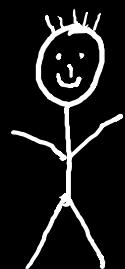
Input Val.

Auth Z

Auth N

Output San.

„the server“



OO

App Server

Web Server

↑✉ Request

DBs LDAP File Share

Ext.  
Service

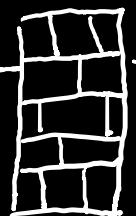


FW



WAF

LB  
Proxy



FW

Int.  
Service

# Request Routing

Classic routing: `http://example.com/showprofile.php`

it's php  
file showprofile.php gets  
executed with php interpreter

Explicit routing: `http://example.com/showprofile`

e.g. Laravel (PHP): `Route::get('/showprofile', [UserController::class, 'showprofile']);`  
<https://laravel.com/docs/8.x/routing>

e.g: Django (Python): `urlpatterns = [ path('/showprofile', views.userprofile)]`  
<https://docs.djangoproject.com/en/3.1/topics/http/urls/>

Internet

DMZ

Application

Intranet

Business  
Logic

Input Val.

Auth Z

Auth N

Output  
San.

Log Server

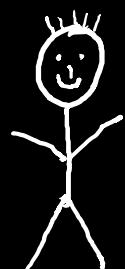
App Server

Web Server

↑✉ Request

„the server“

eventually it  
creates a website  
containing HTML,  
CSS and JS



Ext.  
Service

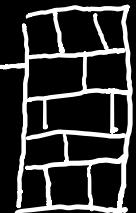


FW



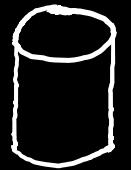
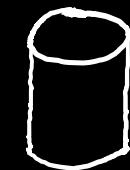
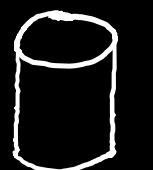
WAF

LB  
Proxy



FW

Int.  
Service



DBs

LDAP

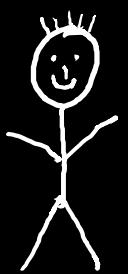
File  
Share

Internet

DMZ

Application

Intranet



Log Server

Business Logic

Input Val.

Auth Z

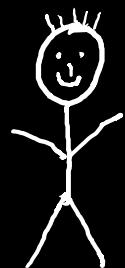
Auth N

Output San.

App Server

Web Server

Response



00 —

Ext.  
Service

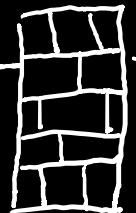


FW



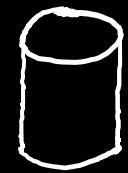
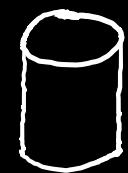
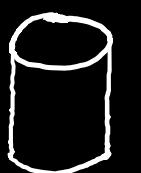
WAF

LB  
Proxy



FW

Int.  
Service



DBs LDAP File  
Share



Response

HTTP/1.1 200 OK

Date: Sat, 10 Oct 2020 18:45:25 GMT

Server: Apache/2.4.38 (Debian) PHP/7.3.19-1~deb10u1

Last-Modified: Sat, 10 Oct 2020 16:49:41 GMT

ETag: "29cd-5b153da05baf0"

Accept-Ranges: bytes

Content-Length: 10701

Vary: Accept-Encoding

Content-Type: text/html

# HTTP Response Status Codes

Code	Meaning
1xx	<b>Informational</b> Everything fine, continue what you're doing
2xx	<b>Success</b> Yes, everything worked out
3xx	<b>Redirection</b> Look at some other place
4xx	<b>Client errors</b> Server doesn't understand you or you try to do something you're not allowed to
5xx	<b>Server errors</b> Something didn't work out

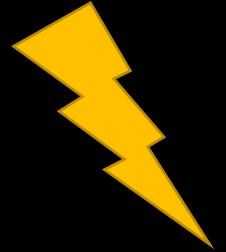


HTTP Header

HTTP/1.1 200 OK  
Date: Sat, 10 Oct 2020 18:45:25 GMT  
Server: Apache/2.4.38 (Debian) PHP/7.3.19-1~deb10u1  
Last-Modified: Sat, 10 Oct 2020 16:49:41 GMT  
ETag: "29cd-5b153da05baf0"  
Accept-Ranges: bytes  
Content-Length: 10701  
Vary: Accept-Encoding  
Content-Type: text/html

do you see any problems?

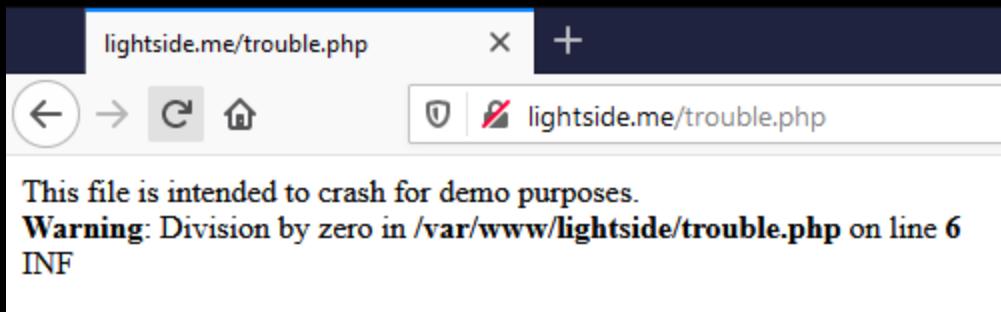
# Information Disclosure



Goal	Gather useful information – either for direct use or for further attacks.
How	two categories: <ul style="list-style-type: none"><li>- technical data<ul style="list-style-type: none"><li>- e.g. HTTP Header, ...</li></ul></li><li>- business data<ul style="list-style-type: none"><li>- authorization problems, SQLi, etc. – we will talk about that later</li></ul></li></ul>
Solution	
OWASP Top 10	
(Primary) Violated Principle	

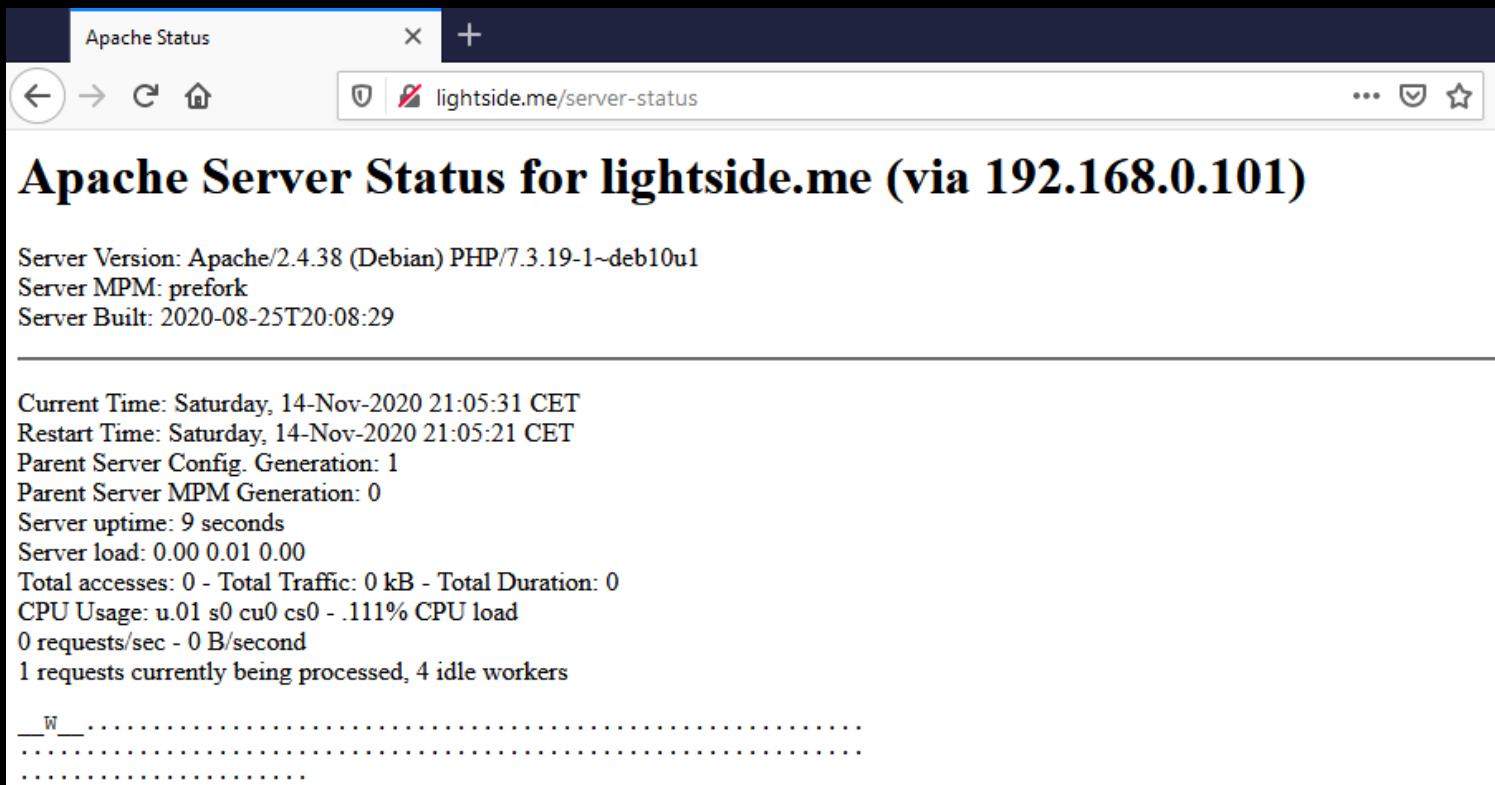
# More technical information disclosures...

- HTTP-Header
- Error messages



# More technical information disclosures...

- HTTP-Header
- Error messages
- Status pages



The screenshot shows a web browser window titled "Apache Status". The address bar displays "lightside.me/server-status". The main content area is titled "Apache Server Status for lightside.me (via 192.168.0.101)". Below the title, it lists server details: "Server Version: Apache/2.4.38 (Debian) PHP/7.3.19-1~deb10u1", "Server MPM: prefork", and "Server Built: 2020-08-25T20:08:29". A horizontal line separates this from the performance metrics. The metrics include: "Current Time: Saturday, 14-Nov-2020 21:05:31 CET", "Restart Time: Saturday, 14-Nov-2020 21:05:21 CET", "Parent Server Config. Generation: 1", "Parent Server MPM Generation: 0", "Server uptime: 9 seconds", "Server load: 0.00 0.01 0.00", "Total accesses: 0 - Total Traffic: 0 kB - Total Duration: 0", "CPU Usage: u.01 s0 cu0 cs0 - .111% CPU load", "0 requests/sec - 0 B/second", and "1 requests currently being processed, 4 idle workers". At the bottom, there is a dotted line followed by a single character "W".

# More technical information disclosures...

- HTTP-Header
- Error messages
- Status pages
- Directory listings
  - unreferenced files
    - e.g. backups, configs etc.



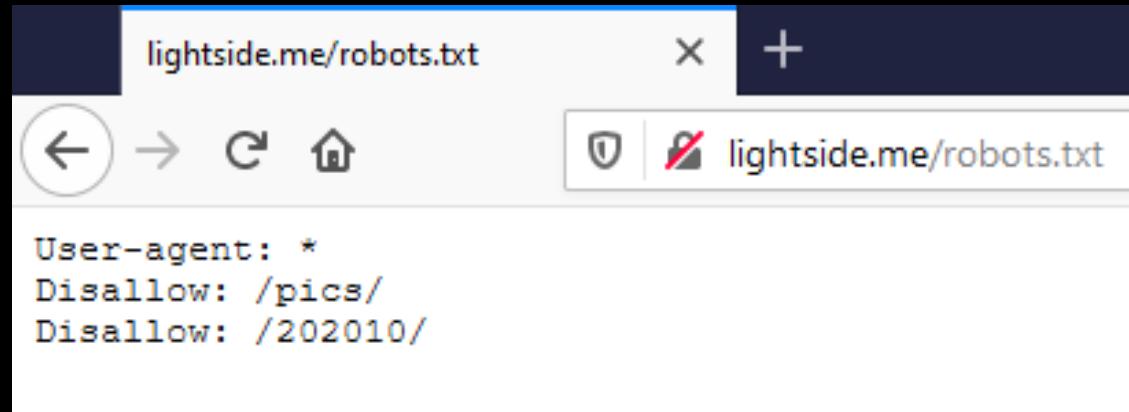
The screenshot shows a web browser window displaying a directory listing titled "Index of /202010". The address bar indicates the URL is "lightside.me/202010/". The page includes standard navigation buttons (back, forward, search, home) and a SSL certificate icon. The main content area is titled "Index of /202010" and lists files and directories:

Name	Last modified	Size	Description
<a href="#">Parent Directory</a>		-	
<a href="#">inv_202010_luke.pdf</a>	2020-10-21 21:44	33K	
<a href="#">test.txt</a>	2020-10-21 21:22	6	

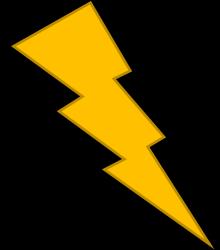
At the bottom of the page, there is a footer note: "Apache/2.4.38 (Debian) PHP/7.3.19-1~deb10u1 Server at lightside.me Port 80"

# More technical information disclosures...

- HTTP-Header
- Error messages
- Status pages
- Directory listings
  - unreferenced files
    - e.g. backups, configs etc.
- robots.txt
- ...

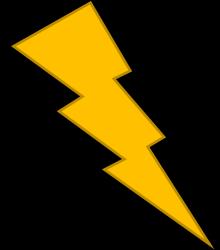


# Information Disclosure



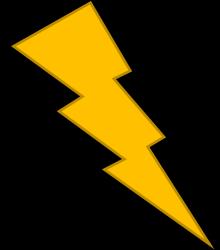
Goal	Gather useful information – either for direct use or for further attacks.
How	two categories: <ul style="list-style-type: none"><li>- technical data<ul style="list-style-type: none"><li>- e.g. HTTP Header, unhandled errors, status, directory index, robots.txt</li></ul></li><li>- business data<ul style="list-style-type: none"><li>- authorization problems, SQLi, etc. – we will talk about that later</li></ul></li></ul>
Solution	Webserver configuration hardening Error-Handling Secure Coding
OWASP Top 10	
(Primary) Violated Principle	

# Information Disclosure



Goal	Gather useful information – either for direct use or for further attacks.
How	two categories: <ul style="list-style-type: none"><li>- technical data<ul style="list-style-type: none"><li>- e.g. HTTP Header, unhandled errors, status, directory index, robots.txt</li></ul></li><li>- business data<ul style="list-style-type: none"><li>- authorization problems, SQLi, etc. – we will talk about that later</li></ul></li></ul>
Solution	Webserver configuration hardening Error-Handling Secure Coding
OWASP Top 10	A6:2017-Security Misconfiguration A3:2017-Sensitive Data Exposure
(Primary) Violated Principle	

# Information Disclosure



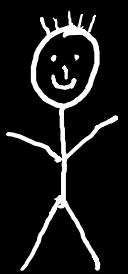
Goal	Gather useful information – either for direct use or for further attacks.
How	two categories: <ul style="list-style-type: none"><li>- technical data<ul style="list-style-type: none"><li>- e.g. HTTP Header, unhandled errors, status, directory index, robots.txt</li></ul></li><li>- business data<ul style="list-style-type: none"><li>- authorization problems, SQLi, etc. – we will talk about that later</li></ul></li></ul>
Solution	Webserver configuration hardening Error-Handling Secure Coding
OWASP Top 10	A6:2017-Security Misconfiguration A3:2017-Sensitive Data Exposure
(Primary) Violated Principle	„Identify sensitive data and how they should be handled.“

Internet

DMZ

Application

Intranet



Log Server

Business Logic

Input Val.

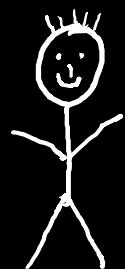
Auth Z

Output San.

Auth N

App Server

Web Server



Response

Ext.  
Service

FW

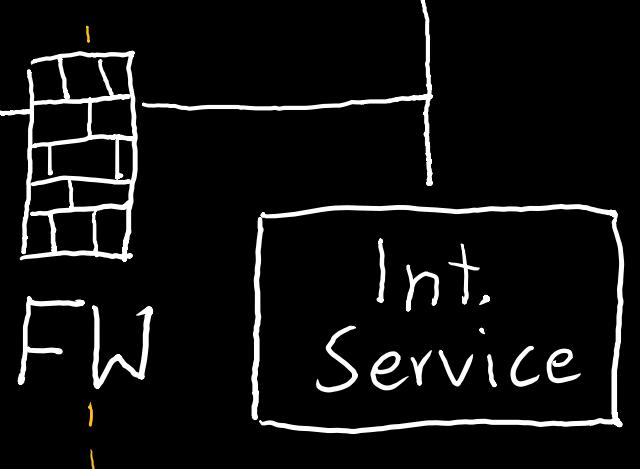
WAF

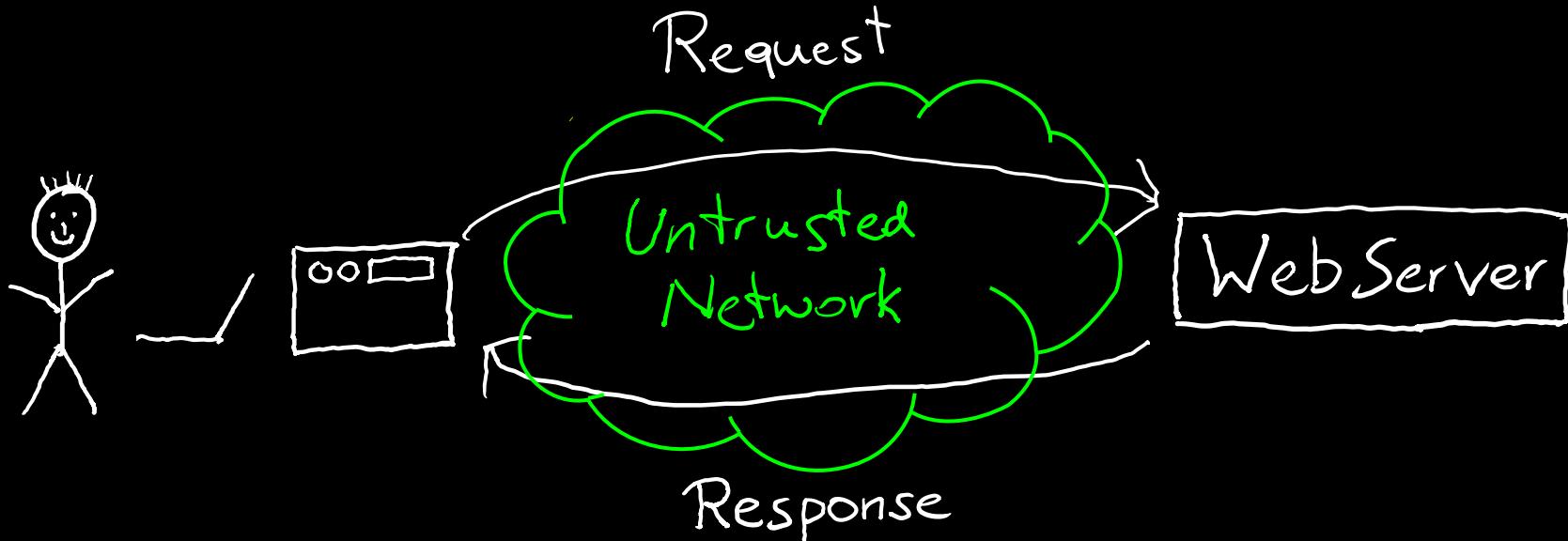
LB  
Proxy

FW

Int.  
Service

DBs LDAP File Share

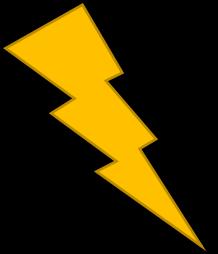




<http://example.com:8080/demos/demo1.html?id=123&name=Luke#ref1>

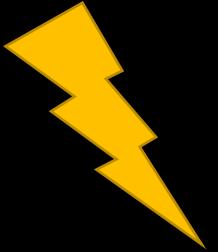
Versions  $\leq$  HTTP/2 are plaintext by default

# Plaintext Transmission (Info-Disclosure)



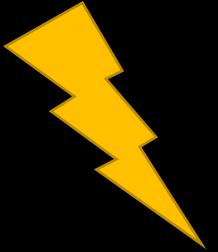
Goal	Read or modify data in transit
How	
Solution	
OWASP Top 10	
(Primary) Violated Principle	

# Plaintext Transmission (Info-Disclosure)



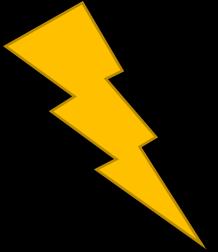
Goal	Read or modify data in transit
How	MitM between client and server <ul style="list-style-type: none"><li>- ARP Spoofing in your network (WIFI) or the servers' network</li><li>- Malicious ISP (employee)</li><li>- Gateways to foreign countries</li><li>- DNS Hijacking</li><li>- ...</li></ul>
Solution	
OWASP Top 10	
(Primary) Violated Principle	

# Plaintext Transmission (Info-Disclosure)



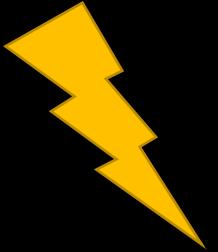
Goal	Read or modify data in transit
How	MitM between client and server <ul style="list-style-type: none"><li>- ARP Spoofing in your network (WIFI) or the servers' network</li><li>- Malicious ISP (employee)</li><li>- Gateways to foreign countries</li><li>- DNS Hijacking</li><li>- ...</li></ul>
Solution	HTTPS (HTTP + SSL/TLS) HTTP/3 uses TLS 1.3 by default
OWASP Top 10	
(Primary) Violated Principle	

# Plaintext Transmission (Info-Disclosure)



Goal	Read or modify data in transit
How	MitM between client and server <ul style="list-style-type: none"><li>- ARP Spoofing in your network (WIFI) or the servers' network</li><li>- Malicious ISP (employee)</li><li>- Gateways to foreign countries</li><li>- DNS Hijacking</li><li>- ...</li></ul>
Solution	HTTPS (HTTP + SSL/TLS) HTTP/3 uses TLS 1.3 by default
OWASP Top 10	A3:2017-Sensitive Data Exposure
(Primary) Violated Principle	

# Plaintext Transmission (Info-Disclosure)



Goal	Read or modify data in transit
How	MitM between client and server <ul style="list-style-type: none"><li>- ARP Spoofing in your network (WIFI) or the servers' network</li><li>- Malicious ISP (employee)</li><li>- Gateways to foreign countries</li><li>- DNS Hijacking</li><li>- ...</li></ul>
Solution	HTTPS (HTTP + SSL/TLS) HTTP/3 uses TLS 1.3 by default
OWASP Top 10	A3:2017-Sensitive Data Exposure
(Primary) Violated Principle	“Earn or give, but never assume, trust.”

# Transport Layer Security (TLS)

- TLS = successor of SSL (Secure Socket Layer)
  - SSL 1.0 ~ 1994/1995
  - ...
  - TLS 1.3 ~ 2018
- Family of multiple protocols
  - Handshake, Change Cipher Spec., Alert, Application Data, Record
- Builds an authenticated encrypted tunnel
  - Authenticates the accessed website (at least)
  - Protects the confidentiality and integrity of the data in transfer
- HTTP over TLS = HTTPS (port 443 for https://)
- PKI needed
  - Chain of trust ( > 1000 CAs & Intermediate CAs )
- HTTPS usage: <https://transparencyreport.google.com/https/overview>

# PFS - Perfect Forward Secrecy

- long-term secret is just used for authentication of session specific Diffie-Hellman Key Exchange
- previous sessions cannot be decrypted, even if
  - one of the old session keys is compromised
  - the long-term secret (server's private key) is compromised
- most current TLS cipher suites implement PFS today

# TLS – Avoid common pitfalls

- Make sure you only allow
  - up to date protocols (TLS 1.2 & TLS 1.3)
  - strong cipher suites
- Of course conflicts with browser compatibility
  - some resources can help you defining appropriate configs
    - e.g. [https://wiki.mozilla.org/Security/Server\\_Side\\_TLS](https://wiki.mozilla.org/Security/Server_Side_TLS)

## Modern compatibility

For services with clients that support TLS 1.3 and don't need backward compatibility, the **Modern** configuration provides an extremely high level of security.

- Cipher suites (TLS 1.3):  
TLS\_AES\_128\_GCM\_SHA256:TLS\_AES\_256\_GCM\_SHA384:TLS\_CHACHA20\_POLY1305\_SHA256
- Cipher suites (TLS 1.2): (none)
- Protocols: TLS 1.3
- Certificate type: ECDSA (P-256)
- TLS curves: X25519, prime256v1, secp384r1
- HSTS: **max-age=63072000** (two years)
- Certificate lifespan: 90 days
- Cipher preference: client chooses

## Intermediate compatibility (recommended)

For services that don't need compatibility with legacy clients, such as Windows XP or old versions of OpenSSL. This is the recommended configuration for the vast majority of services, as it is highly secure and compatible with nearly every client released in the last five (or more) years.

- Cipher suites (TLS 1.3):  
TLS\_AES\_128\_GCM\_SHA256:TLS\_AES\_256\_GCM\_SHA384:TLS\_CHACHA20\_POLY1305\_SHA256
- Cipher suites (TLS 1.2): ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-CHACHA20-POLY1305:DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384
- Protocols: TLS 1.2, TLS 1.3
- TLS curves: X25519, prime256v1, secp384r1
- Certificate type: ECDSA (P-256) (recommended), or RSA (2048 bits)
- DH parameter size: 2048 (ffdhe2048, [RFC 7919](#))
- HSTS: **max-age=63072000** (two years)
- Certificate lifespan: 90 days (recommended) to 366 days
- Cipher preference: client chooses

# TLS - Avoid common pitfalls

deliver all your content via HTTPS only

it can compromise your whole security if  
some parts/methods/etc. are available via  
plain HTTP

(TLS Strip)

# HSTS - HTTP Strict Transport Security

- Tells the browser to always use HTTPS for this site
  - e.g. for one year:
    - `Strict-Transport-Security: max-age=31536000`
- Of course, first visit is still not secure
  - possible solution: HSTS Preload list
    - `Strict-Transport-Security: max-age=31536000; includeSubDomains; preload`
    - <https://hstspreload.org/>

# ISO/OSI Model

Application

Presentation

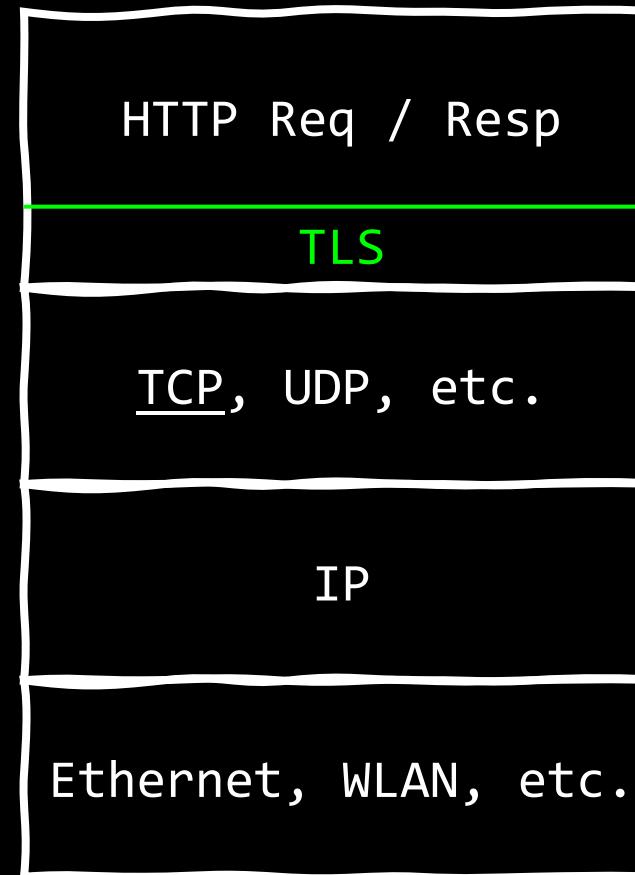
Session

Transport

Network

Data Link  
Physical

# Internet Protocol Suite (TCP/IP Model)

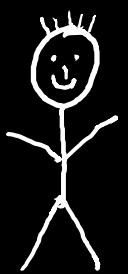


Internet

DMZ

Application

Intranet



Log Server

Business Logic

Input Val.

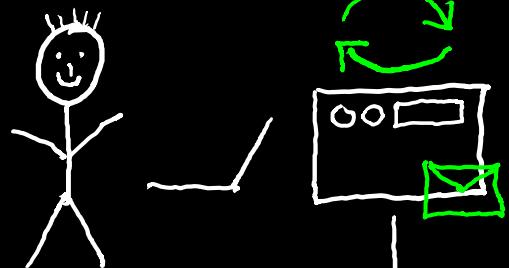
Auth Z

Output San.

Auth N

App Server

Web Server



Ext.  
Service



FW



WAF

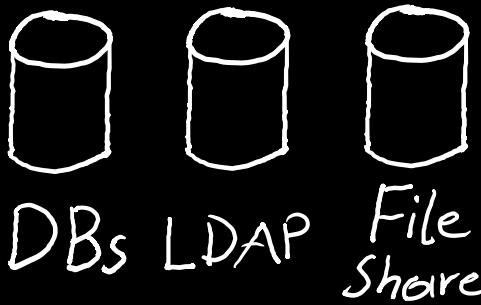


LB  
Proxy

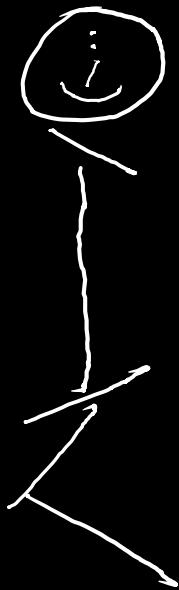


FW

Int.  
Service

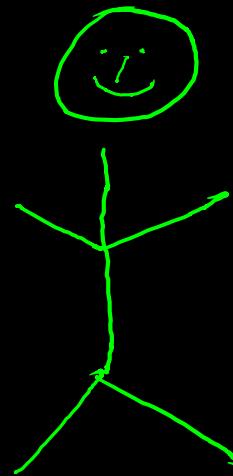


HTML only



gives you building blocks  
and a basic structure

HTML + CSS



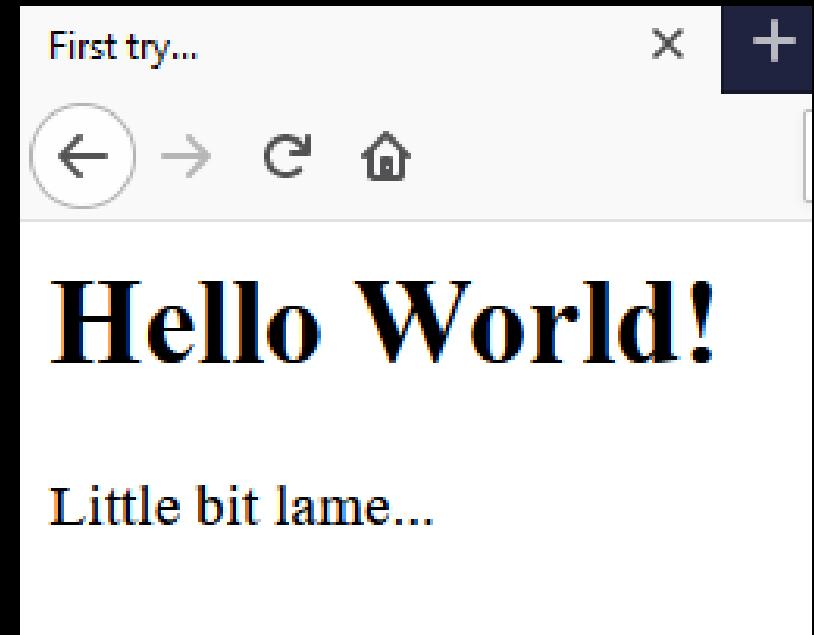
makes it „pretty“

HTML + CSS + JS

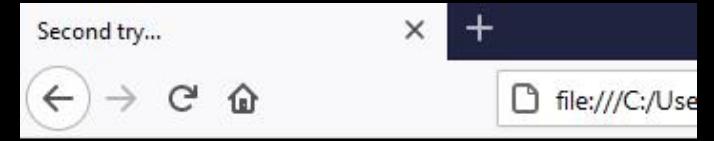


party hard!

```
1 <html>
2   <head>
3     <title>First try...</title>
4   </head>
5   <body>
6     <h1>Hello World!</h1>
7     <p>
8       Little bit lame...
9     </p>
10    </body>
11 </html>
```



```
1 <html>
2   <head>
3     <title>Second try...</title>
4     <style>
5       body{
6         background-color: black;
7       }
8       a{
9         color: greenyellow;
10      }
11     </style>
12   </head>
13   <body>
14     <h1 style="color: greenyellow;">Hello Universe!</h1>
15     <p style="color: white;">
16       better...
17     </p>
18     <a href="https://www.jedipedia.net/wiki/Millennium_Falcon">
19       Millenium Falcon
20     </a><br><br>
21     
23   </body>
24 </html>
```



Hello Universe!

better...

[Millenium Falcon](https://www.jedipedia.net/wiki/Millennium_Falcon)



```
1 <html>
2   <head>
3     <title>Third try...</title>
4     <style>
5       body{
6         background-color: black;
7       }
8       a{
9         color: greenyellow;
10      }
11     </style>
12     <script>
13       var output = "JS interaction";
14     </script>
15   </head>
16   <body>
17     <h1 style="color: greenyellow;">Hello Universe!</h1>
18     <p style="color: white;">
19       now with more action...
20     </p>
21     <a href="https://www.jedipedia.net/wiki/Millennium_Falcon">
22       Millennium Falcon
23     </a><br><br>
24     <img height="200px" width="250px" onmouseover="alert(output)">
25       src="https://www.jedipedia.net/w/images/9/91/Millenium_Falcon_TFA.jpg" />
26   </body>
27 </html>
```

Hello Universe!

now with more action...

Millenium Falcon



JS interaction



OK

# Document Object Model (DOM)

- Interface to treat a HTML/XML document as a tree with objects as it's nodes
- Browser constructs a DOM-Tree of the loaded HTML
- JavaScript can interact with these DOM-Objects through a defined DOM-Interface

# Hello Universe!

now with more action...

[Millenium Falcon](#)



```
Elements Console Sources Network Performance Memory
<html>
  > <head>...</head>
  > <body>
    ...   <h1 style="color: greenyellow;">Hello Universe!</h1> == $0
          <p style="color: white;">
            now with more action...
          </p>
          <a href="https://www.jedipedia.net/wiki/Millennium_Falcon">
            Millenium Falcon
          </a>
          <br>
          <br>
           document.body.childNodes[1].innerHTML = "Modified"
< "Modified"
> document.getElementsByTagName("img")[0].src = "https://upload.wikimedia.org/wikipedia/commons/a/a6/Anonymous_emblem.svg"
< "https://upload.wikimedia.org/wikipedia/commons/a/a6/Anonymous_emblem.svg"
>
```

# Key messages

- Make sure you have control over all your subdomains
- Harden your webserver and TLS configuration properly
- HTTP is a stateless application-layer protocol based on requests and responses

Is it clear now what happens  
if you type in a URL and press  
enter?