



**CYBER
CHALLENGE**
CyberChallenge.it



SPONSOR PLATINUM

accenture security

aizoon AUSTRALIA
EUROPE USA
TECHNOLOGY CONSULTING

B5

EY Building a better
working world

eni

expravia | **ITALTEL**

IBM

KPMG

LEONARDO

NTT DATA
Trusted Global Innovator

NUMERA
SISTEMI E INFORMATICA S.p.A.

Telsy

SPONSOR GOLD

bip.

CISCO

**MONTE
DEI PASCHI
DI SIENA**
BANCA DAL 1472

negg®

NOVANEXT
connecting the future

pwc

SPONSOR SILVER

**DGi
ONE**
the leading
digital company

**ICT
CYBER
CONSULTING**

HTTP Protocol And Web Security Overview

2

Riccardo BONAFEDE
Università di Padova
bonaff@live.it



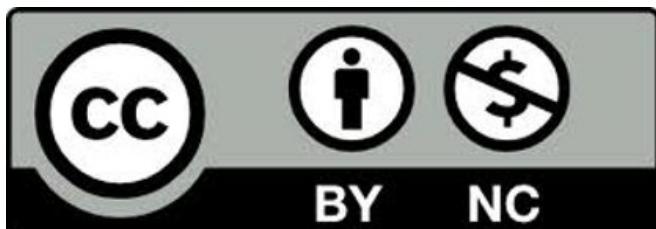
<https://cybersecnatlab.it>

License & Disclaimer

3

License Information

This presentation is licensed under
the Creative Commons BY-NC License



To view a copy of the license, visit:

<http://creativecommons.org/licenses/by-nc/3.0/legalcode>

Disclaimer

- We disclaim any warranties or representations as to the accuracy or completeness of this material.
- Materials are provided “as is” without warranty of any kind, either express or implied, including without limitation, warranties of merchantability, fitness for a particular purpose, and non-infringement.
- Under no circumstances shall we be liable for any loss, damage, liability or expense incurred or suffered which is claimed to have resulted from use of this material.

Outline

4

- HTTP History
- Key Features and Overview of HTTP
- Security and Web Security
- Tooling

Outline

5

- HTTP History
- Key Features and Overview of HTTP
- Security and Web Security
- Tooling

HTTP History

6

- HTTP was introduced at the beginning of the '90s
- The first version of the protocol, **HTTP 0.9**, was released under the World Wide Web initiative
 - Extremely simple
 - Released in 1991
- <https://www.w3.org/Protocols/HTTP/AsImplemented.html>

HTTP History

7

- HTTP initial goal was to **share documents**
- Every document was (and still is) written in **HTML**
 - The first version of the language, HTML 1.0, is a barebone language whose main goal was to format texts and to connect them through hyperlinks
- The **first** example of a **browser** for this language was called "WorldWideWeb"

HTTP History

FROM: W3.ORG

World Wide Web

8

The WorldWideWeb (W3) is a wide-area [hypermedia](#) information retrieval initiative aiming to give universal access to a large universe of documents.

Everything there is online about W3 is linked directly or indirectly to this document, including an [executive summary](#) of the project, [Mailing lists](#) , [Policy](#) , November's [W3 news](#) , [Frequently Asked Questions](#) .

[What's out there?](#)

Pointers to the world's online information, [subjects](#) , [W3 servers](#), etc.

[Help](#)

on the browser you are using

[Software Products](#)

A list of W3 project components and their current state. (e.g. [Line Mode](#) ,[X11 Viola](#) ,[NeXTStep](#) ,[Servers](#) ,[Tools](#) ,
[Mail robot](#) ,[Library](#))

[Technical](#)

Details of protocols, formats, program internals etc

[Bibliography](#)

Paper documentation on W3 and references.

[People](#)

A list of some people involved in the project.

[History](#)

A summary of the history of the project.

[How can I help ?](#)

If you would like to support the web..

[Getting code](#)

Getting the code by [anonymous FTP](#) , etc.

HTTP

9



format
a

The screenshot displays the WorldWideWeb browser/editor application running on a Mac OS X-like environment. The main window shows the homepage of the GNU Project (<http://www.gnu.org/>). The sidebar contains links to FSF Europe and FSF India Translations. The main content area features a logo and text about the GNU Project's mission to promote freedom. The 'Info' window provides technical details about the browser, including its version (2.02 with libwww 2.16pre 1) and author (Tim Berners-Lee). The application's menu bar includes 'WorldWideWeb', 'Style', 'Document', 'Navigate', 'Find', 'Mark/Inspect', 'Links', 'Selection', 'Link destination', 'Image', 'Change', 'Link selection to marked', 'Insert image', and 'Relationship (none)'. The toolbar includes buttons for Back, Next, Previous, Home, Find Panel..., Find Next, Find Previous, Enter Selection, and Jump to Selection. The status bar at the bottom indicates the URL (<http://www.gnu.org/>) and the date (© CINI – 2020 Rel. 26.03.2020).

HTTP - History

10

- Through the years, **browsers became more and more complex**
- **Mosaic** in mid-1993 brought new features, such as the possibility to embed images into web pages
- Several software houses started to develop their own browser, adding new features to defeat the concurrence
 - HTML enchantments
 - JavaScript
 - Plugins such as Java/Flash

HTTP - History

11

- This race (called the "**browsers war**") led to a vast diversity of standards
- In order to maintain compatibility, each browser implemented its own (often undocumented) heuristics to circumvent compatibility issues
 - Often ignoring all the security implications

HTTP - Present

12

- In an effort to mitigate this anarchy, in **1994** the **W3C consortium** was created
 - The goal was to set mandatory web standards for vendors
- Eventually, vendors started to follow these standards, and by now the vast majority of the problems introduced by the "browsers war" are solved

Outline

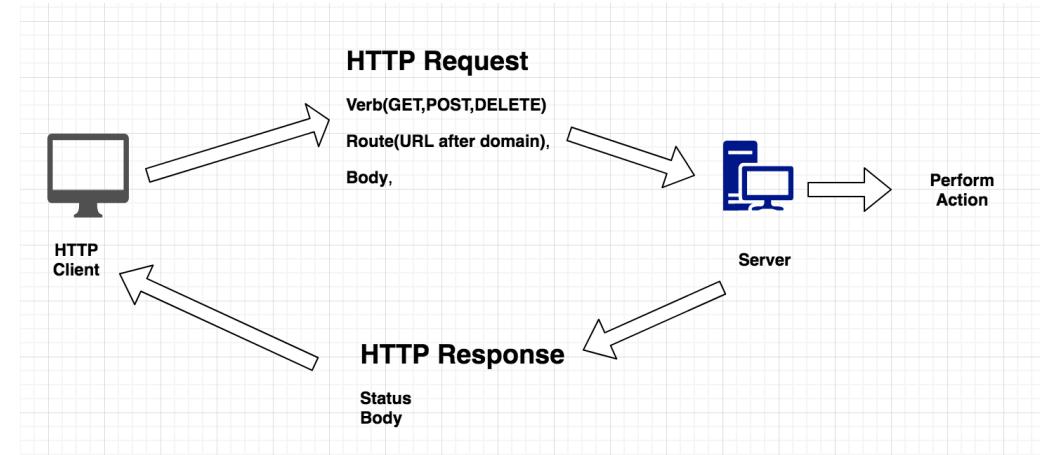
13

- HTTP History
- Key Features and Overview of HTTP
- Security and Web Security
- Tooling

HTTP Overview

14

- Defined in **RFC2616¹**
- High level protocol (Application level in the ISO/OSI stack)
- Mainly on TCP
 - Ports 80/443
 - HTTPS <-- HTTP over TLS
- Human readable
- Client/Server architecture
- Stateless



1: <https://tools.ietf.org/html/rfc2616>

HTTP Overview

15

- HTTP is about **resources**
- A resource is an asset that a client requests to access, and it can be
 - A HTML file
 - An image
 - An information
 - ...

HTTP Overview

16

- Resources are uniquely represented with **URLs**, acronym of
 - Uniform Resource Locator
- URLs are defined in RFCs **1738¹** (URLs) and **3986²** (URIs)

1: <https://tools.ietf.org/html/rfc1738>

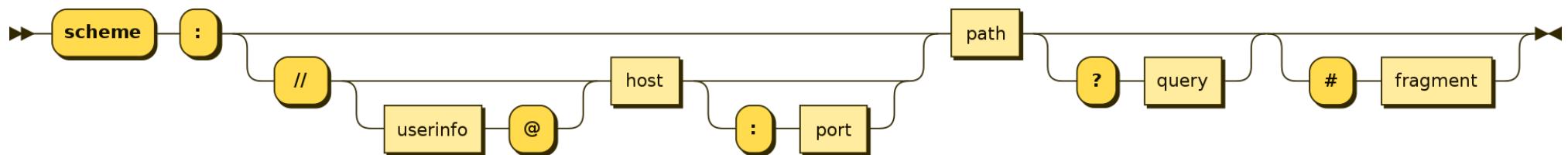
2: <https://tools.ietf.org/html/rfc3986>

HTTP Overview - URLs

17

- HTTP URLs are written with the following syntax

<scheme>://<user>:<password>@<host>:<port>/<url-path>



HTTP Overview - URLs

18

- <scheme>://<user>:<password>@<host>:<port>/<url-path>
 - <**scheme**> specifies the protocol used, i.e. http or https
 - <**user**> and <**password**> are the credentials needed to access the resource
 - <**host**> and <**port**> represent the address to which the client should connect to send the request
 - <**url-path**> is the resource requested to the server
 - <*userpassword*> and <*port*> are optional

HTTP Overview - URLs

19

- The <url-path> has its own syntax in http, but since its interpretation is reserved to the backend, this syntax may change from web app to web app
- Usually it is in the form of

/<directory>/<file>?<query>#fragment

HTTP Overview - URLs

20

- The <directory>/<file> part is used to represent the "physical" location of the resource on the filesystem of the server
- Right now it is more a "virtual" location

<https://twitter.com/Twitter/status/1212237037631352832>

HTTP Overview - URLs

21

- The <query> is optional and it can be used to send information to the backend
- It is a dictionary with a **key-value representation**, generally in the form of

`varname1=value1&varname2=value2`

HTTP Overview - URLs

22

- The fragment is a piece of information that is reserved for clients
- Clients never send it to the server
- If the fragment appears in a request, the server will ignore it

HTTP Overview - URLs

23

- `https://example.com/file.html?foo=bar#someinfos`
 - Scheme: https
 - Host-port: example.com:443 (443 is the default port for https)
 - Path: file.html
 - Query: a variable named foo with value bar
 - Fragment: someinfos

HTTP Overview - URLencoding

24

- Some characters are **prohibited** in URLs
 - What happens if a variable contains a "#" ?
 - The server will think that the information after the "#" is the fragment, and it will happily ignore everything after it (or throw an error)

HTTP Overview - URLencoding

25

- This problem is solved using a **particular encoding**, that converts every character in a "not harmful" representation
- This encoding is called "**urlencoding**"

HTTP Overview - URLencoding

26

- This encoding is very simple
 - Take the hex value of a character you want to encode, and then prepend a "%" symbol
- # == %23
- Every reserved character in a URL must be urlencoded
- Every non-printable character must be urlencoded
- Spaces can be represented either with %20, or with the plus sign (+)

HTTP Overview

27

- **Requests and Responses** are HTTP messages composed of three different parts
 - The Request/Response line
 - The Header Fields
 - A Body (optional)
- Every line in the Request/Response is terminated by the "CR;LF" sequence: \r\n or 0x0d0a in binary
- An empty line separates the last header field from the body

HTTP Overview - Requests

28

GET / HTTP/1.1

Host: www.google.com

User-Agent: curl/7.64.0

Accept: */*

Request Line

Header Fields

Body Field

Body, empty this time

HTTP Overview - Requests

29

- The Request line is composed of
 - A **method**
 - "we want to do X with the resource"
 - The **resource** for which we are doing the request
 - And, finally, the **protocol version**

GET / HTTP/1.1

HTTP Overview - Methods

30

- Tell the server "what we are doing" with the resource
- Standard methods
 - GET
 - POST
 - OPTIONS
 - HEAD
- It is also possible to define custom methods

HTTP Overview - Body

31

- **Generic data** sent to the server
- Its type (or encoding) is defined by the **Content-Type** header
- It can be **encoded in different ways**
 - application/x-www-form-urlencoded
 - text/plain
 - ...
- Can also have a **custom encoding**
 - application/json
 - foo/bar
 - ...

HTTP Overview - Headers

32

- Headers are used to send **additional data** to the server
- Serialized in the form **name: value**
- Some are mandatory:
 - Host
 - Content-Encoding/Content-Length if there is a body

HTTP Overview - Responses

33

- A Response is very similar to a Request
- It differs only for the **first line**, which is called "status-line"
- This line is mandatory, and tells the client the type of the response and the version of the protocol used to make the response

HTTP Overview - Responses

34

HTTP/1.1 200 Ok

Status-Line

Host: 127.0.0.1:5000

Date: Thu, 12 Mar 2020 10:31:38 GMT

Header Fields

Connection: close

X-Powered-By: PHP/7.4.3

< b > Hello World! < /b >

Body field

HTTP Overview – Status Line

35

- The status-line composed by the **version of the protocol**, an **integer number**, and a **string**
- The number is called **status code**. Status codes are divided into five categories:
 - 1**: Informational Response
 - 2**: Success
 - 3**: Location change
 - 4**: Client Error
 - 5**: Server Error

HTTP Overview – Status code

36

- Some common status codes are
 - 200: The request was successful
 - 400: The request was malformed
 - 404: The requested resource could not be found
 - 500: The server had a critical error, and could not complete the request

HTTP Overview - Cookies

37

- In order to make HTTP stateful, **cookies** were introduced
- Cookies are **text information** that a web client receives and stores from a server, and sends back within every request to the same server
- They are used mainly for
 - Session management
 - Personalization
 - Tracking



HTTP Overview - Cookies

38

- HTTP servers can set cookies with the response header field **Set-Cookie**
- Cookies can also be set client-side via JavaScript
- Cookies are composed by a name, a value, and some meta-information
 - The origin (e.g., the server which sends the cookie)
 - The expire date
 - Some security policies

HTTP Overview - Cookies

39

▲ Response Headers

Access-Control-Allow-Credentials: **true**

Access-Control-Allow-Headers: **X-Requested-With, Content-Type, X-Codingpedia**

Access-Control-Allow-Methods: **GET, POST, DELETE, PUT**

Access-Control-Allow-Origin: *

Content-Length: **65**

Content-Type: **application/json**

Date: **Tue, 23 May 2017 08:44:06 GMT**

Server: **GlassFish Server Open Source Edition 4.1**

Set-Cookie: **token=y9cHZIjGqSlipT;Version=1;Comment=;Domain=;Path=/;Max-Age=3600;Expires=Tue, 23 May 2017 09:44:06 GMT**

X-Powered-By: **Servlet/3.1 JSP/2.3 (GlassFish Server Open Source Edition 4.1 Java/Oracle Corporation/1.8)**

- The expire date
- Some security policies

HTTP Overview - Cookies

40

- Browsers will send back cookies to the server in accordance with its scope
- The scope is the "origin" in which each cookie was created
 - If a cookie named "foo" is set by "www.google.com", it cannot be sent to "www.microsoft.com", but only to "www.google.com"

HTTP Overview - Cookies

41

- In addition to the origin, other security policies can be set
 - Secure and HTTPOnly
 - SameSite
 - None
 - Strict
 - Lax <-- The default on Chrome

Outline

42

- HTTP History
- Key Features and Overview of HTTP
- **Security and Web Security**
- Tooling

Security

43

- As the name suggests, security is about the protection of "something"
- When dealing with computers, we normally identify this "something" with "information"
- Security wants to ensure three main properties of information
 - Integrity
 - Confidentiality
 - Availability

Security

44

- **Integrity**
 - Maintaining the accuracy and completeness of data
- **Confidentiality**
 - Data must be accessible only to whom is authorized to
- **Availability**
 - Data must be accessible when needed

Security

45

- A vulnerability is a **weakness in a system** that permits an attacker to violate one or more of the three previous properties
- Every vulnerability must have an impact
 - How this vulnerability of this system violates one or more of the three principles?
- The best way to find how a vulnerability impact a system is to assume the point of view of an attacker.

Security

46

- The best way to find how a vulnerability impact a system is to assume the point of view of an attacker
- This is done testing the application in an offensive maner
- This activity is called penetration testing

Web Security

47

- Web security apply to vulnerabilities that affect web applications.
- Typically web applications are the most exposed assets to an attacker
- And http is really fragile..

Web Security

48

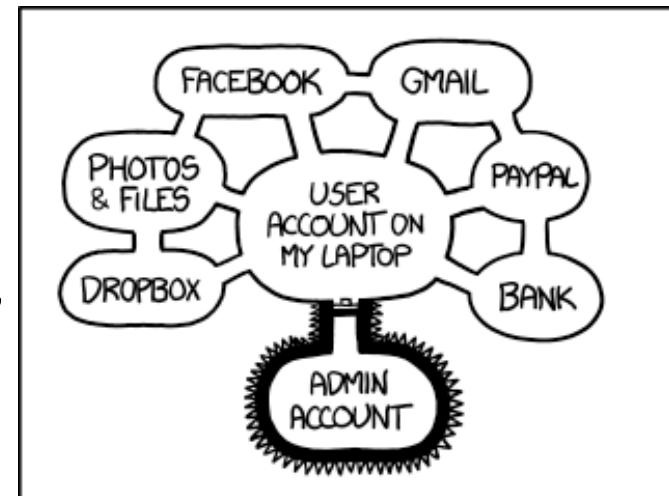
- HTTP was created with the intent to serve **static documents**
- The protocol is **simple by design**
 - It is a **stateless** protocol, since there was no need to keep track of the current client
 - **Documents were simple**, there was no need for animations
 - The **security was not a big concern**, at the beginning there was not much to protect on the web

Web Security

49

- But now we have
 - Dynamic generated pages, e.g. scripts that generate pages on-the-fly
 - Exceptionally complex pages: HTML CSS, JavaScript, WebAssembly, plugins... and a lot more
 - A lot of **secrets to protect**
- Such complexity leads to a **huge attack surface**

The web is a mess...



IF SOMEONE STEALS MY LAPTOP WHILE I'M LOGGED IN, THEY CAN READ MY EMAIL, TAKE MY MONEY, AND IMPERSONATE ME TO MY FRIENDS,
BUT AT LEAST THEY CAN'T INSTALL DRIVERS WITHOUT MY PERMISSION.

Web Security

50

- Web security is about the security of web assets, e.g. everything that runs over HTTP
 - **Server-Side Security:** The impact affect in the remote server
 - **Client-Side Security:** The impact affect the client
 - Please note: this does not mean that it is a vulnerability of the browser!
 - Rule of thumb: *If you need to send a link to the victim, then probably it is a client-side vulnerability*

Web Security

51

- In order to find vulnerabilities, there are two main methodologies
 - **Blackbox**
 - We do not know anything about the system we are attacking
 - **Whitebox**
 - We know everything about the system, we have the source code, we can debug it, ...

Web Security - BlackBox

52

- **Enumeration:** the **more information** we have about the system **the better**
 - Look at the functions an application implements
 - Try to input random things. If you have to insert a number, try to insert some letters, and look at what happens
- **Try and error:** because we do not know anything about the system, we **need to try attacks in order to undercover problems**

Web Security – WhiteBox

53

- In a WhiteBox testing, you can also debug your application:
 - Static Analysis
 - Code review: identify dangerous behaviors of the programmer
- Dynamic analysis
 - You can debug your application

Outline

54

- HTTP History
- Key Features and Overview of HTTP
- Security and Web Security
- Tooling

Web Security – Some useful tools

55

- Browser
- Curl/wget
 - A Command line utility to make http requests
- Python requests
 - A useful python library to do http requests
- Burp suite/zap proxy
 - live edit raw http requests and response
- Test server (php dev & httpsimplepython)
- Ngrok
 - creates a public http/tcp tunnel to your machine

On-the-fly HTTP server

56

- PHP
 - A very fast-to deploy test server
 - Serves every file inside the directory it was launched, and executes .php scripts
 - `php -S 127.0.0.1:5000`

Launch it from a test directory! You don't want to leak your .ssh directory !

ngrok

57

- What if you need a public server?
 - VPS
 - Ngrok: <https://ngrok.com/>
 - Serveo : <http://serveo.net/>
 - (this seems to be down at the time of writing)

ngrok

58

- Ngrok allows one to create tunnels
- You can run it with the command
 - `$ ngrok http 5000`
 - Creates a http tunnel and redirects every request to the local port 5000
 - `$ ngrok tcp 5000`
 - Creates a tcp tunnel and redirects every connection to the local port 5000

```
ngrok by @inconschreivable                                         (Ctrl+C to quit)

Session Status          online
Account                 bonaff (Plan: Free)
Update                  update available (version 2.3.35, Ctrl-U to update)
Version                2.3.34
Region                 United States (us)
Web Interface          http://127.0.0.1:4040
Forwarding              http://1da23a85.ngrok.io -> http://localhost:5000
Forwarding              https://1da23a85.ngrok.io -> http://localhost:5000

Connections            ttl     opn      rt1      rt5      p50      p90
                        0       0       0.00    0.00    0.00    0.00
```

ngrok - Inspect x +

127.0.0.1:4040/inspect/http

ngrok online Inspect Status Documentation

All Requests Clear

GET / 200 OK 1.34ms

half a minute ago Duration 1.34ms IP 109.115.64.3

GET /

Summary Headers Raw Binary Replay ▾

Headers

Accept	text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Encoding	gzip, deflate
Accept-Language	en-US,en;q=0.5
Dnt	1
Host	85804868.ngrok.io
Upgrade-Insecure-Requests	1
User-Agent	Mozilla/5.0 (X11; Linux x86_64; rv:73.0) Gecko/20100101 Firefox/73.0
X-Forwarded-For	109.115.64.3

200 OK

Summary Headers Raw Binary

Ask a question

HTTP Protocol And Web Security Overview

61

Riccardo BONAFEDE
Università di Padova
bonaff@live.it



<https://cybersecnatlab.it>



**CYBER
CHALLENGE**
CyberChallenge.it



SPONSOR PLATINUM

accenture security

aizoon AUSTRALIA
EUROPE USA
TECHNOLOGY CONSULTING

B5

EY Building a better
working world



expravia | **ITALTEL**

IBM

KPMG

LEONARDO

NTT DATA
Trusted Global Innovator

NUMERA
SISTEMI E INFORMATICA S.p.A.

Telsy

SPONSOR GOLD

bip.

CISCO

**MONTE
DEI PASCHI
DI SIENA**
BANCA DAL 1472

negg®

NOVANEXT
connecting the future

pwc

SPONSOR SILVER

**DGi
ONE**
the leading
digital company

**ICT
CYBER
CONSULTING**