HTTP Protocol And Web Security Overview

Riccardo Bonafede
Università di Padova
bonaff@live.it

CYBER CHALLENGE
CyberChallenge.IT

cini
Cybersecurity National Lab

https://cybersecnatlab.it

License & Disclaimer

License Information

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

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

Outline

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

HTTP History

- 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

- 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

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.

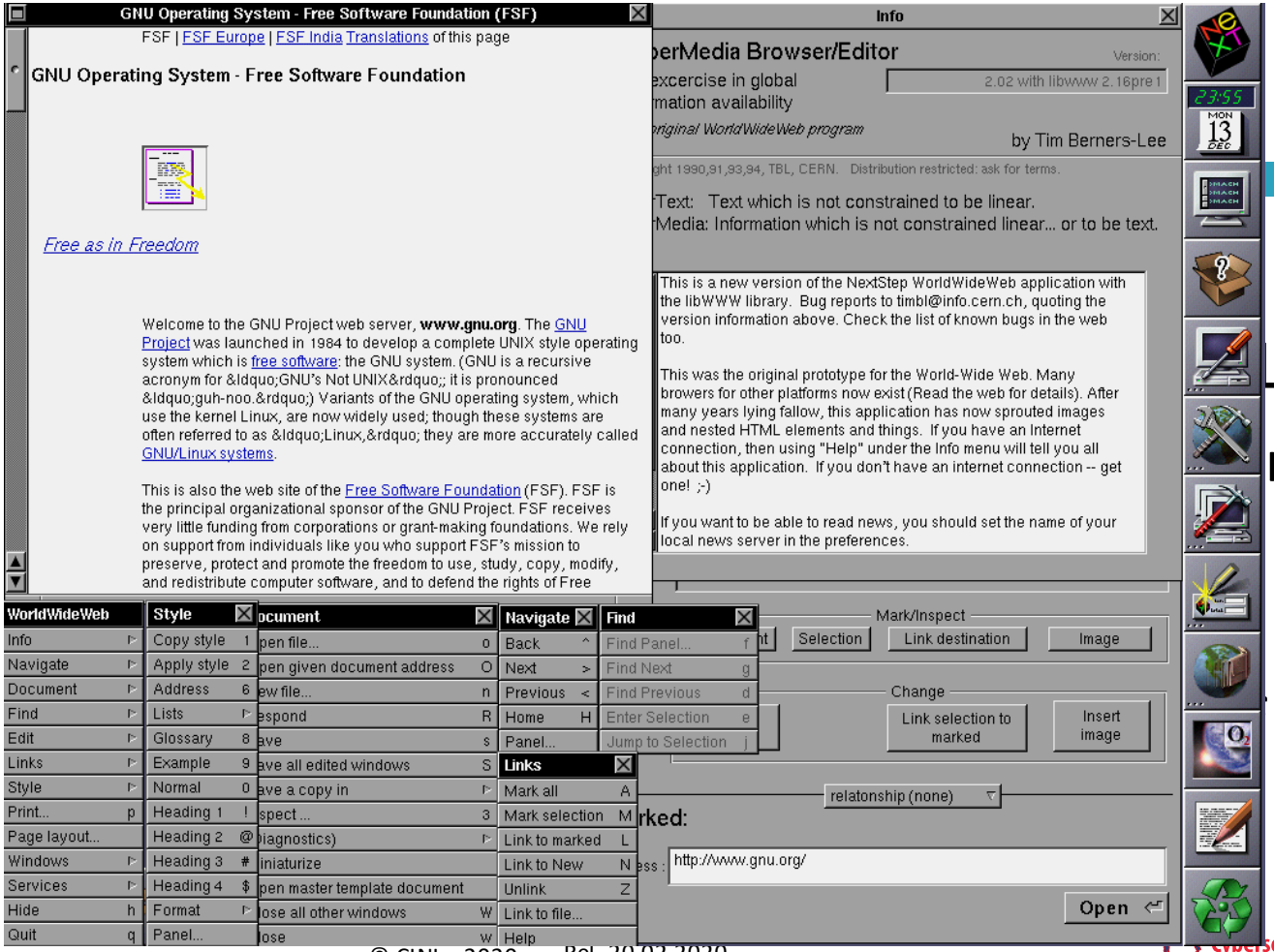
- H
- H
- -
- I
- k

L

rmat

f a

HTTP


GNU Operating System - Free Software Foundation (FSF)
FSF | FSF Europe | FSF India Translations of this page
GNU Operating System - Free Software Foundation
Free as in Freedom
Welcome to the GNU Project web server, www.gnu.org. The GNU Project was launched in 1984 to develop a complete UNIX style operating system which is free software: the GNU system. (GNU is a recursive acronym for “GNU's Not UNIX”; it is pronounced “guh-noo.”) Variants of the GNU operating system, which use the kernel Linux, are now widely used; though these systems are often referred to as “Linux,” they are more accurately called GNU/Linux systems.
This is also the web site of the Free Software Foundation (FSF). FSF is the principal organizational sponsor of the GNU Project. FSF receives very little funding from corporations or grant-making foundations. We rely on support from individuals like you who support FSF's mission to preserve, protect and promote the freedom to use, study, copy, modify, and redistribute computer software, and to defend the rights of Free
Info
berMedia Browser/Editor
Version:
2.02 with libwww 2.16pre 1
by Tim Berners-Lee
Text: Text which is not constrained to be linear.
Media: Information which is not constrained linear... or to be text.
This is a new version of the NextStep WorldWideWeb application with the libWWW library. Bug reports to timbl@info.cern.ch, quoting the version information above. Check the list of known bugs in the web too.
This was the original prototype for the World-Wide Web. Many browers for other platforms now exist (Read the web for details). After many years lying fallow, this application has now sprouted images and nested HTML elements and things. If you have an Internet connection, then using "Help" under the Info menu will tell you all about this application. If you don't have an internet connection -- get one! ;-)
If you want to be able to read news, you should set the name of your local news server in the preferences.
WorldWideWeb
Info
Navigate
Document
Find
Edit
Links
Style
Print... p
Page layout...
Windows
Services
Hide h
Quit q
Style
Copy style 1
Apply style 2
Address 6
Lists
Glossary 8
Example 9
Normal 0
Heading 1 !
Heading 2 @
Heading 3 #
Heading 4 $
Format
Panel...
Navigate
Back ^
Next >
Previous <
Home H
Panel...
Find
Find Panel... f
Find Next g
Find Previous d
Enter Selection e
Jump to Selection j
Links
Mark all A
Mark selection M
Link to marked L
Link to New N
Unlink Z
Link to file...
Help
Mark/Inspect
Selection
Link destination
Image
Change
Link selection to marked
Insert image
relatonship (none)
http://www.gnu.org/
Open

HTTP - History

- 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

- 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

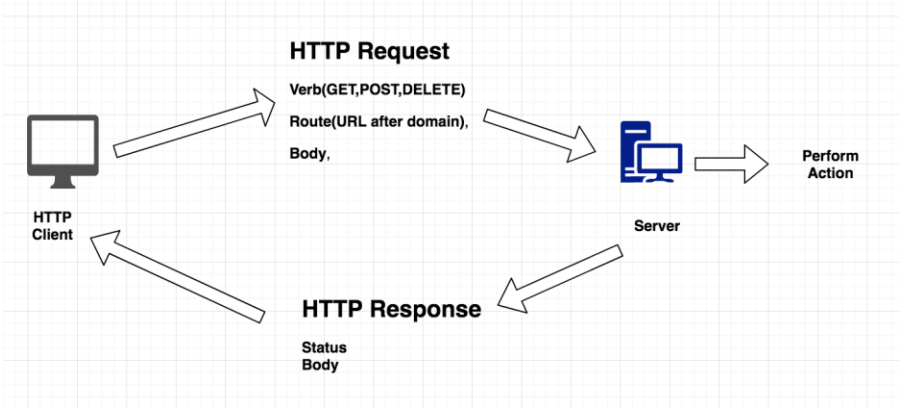
- In an effort to mitigate this anarchy, in **1994** the **W3C consortium** was created
  - The goal was to set mandatory web standards for vendors
- Eventually, vendors started to follow these standards, and by now the vast majority of the problems introduced by the "browsers war" are solved

Outline

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

HTTP Overview

- Defined in **RFC2616**[1]
- 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

- 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

- Resources are uniquely represented with **URLs**, acronym of
  - Uniform Resource Locator
- URLs are defined in RFCs **1738**[1] (URLs) and **3986**[2] (URIs)

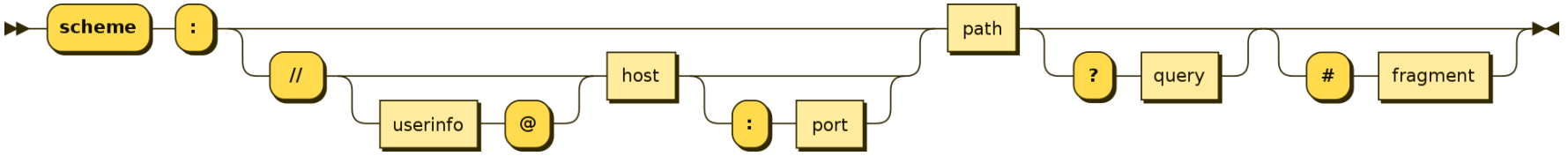
1: https://tools.ietf.org/html/rfc1738

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

HTTP Overview - URLs

- HTTP URLs are written with the following syntax

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



HTTP Overview - URLs

- <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
  - *<user>, <password> and <port> are optional*

HTTP Overview - URLs

- 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

- 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

- 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

- 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

- 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

- 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

- 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

- 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

- **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


```
GET / HTTP/1.1
Host: www.google.com
User-Agent: curl/7.64.0
Accept: */*


*Body, empty this time*
```

Request Line

Header Fields

Body Field

HTTP Overview - Requests

- 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

- 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

- **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

- 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

- 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


```
HTTP/1.1 200 Ok
Host: 127.0.0.1:5000
Date: Thu, 12 Mar 2020 10:31:38 GMT
Connection: close
X-Powered-By: PHP/7.4.3

<b>Hello World!</b>
```

Status-Line

Header Fields

Body field

HTTP Overview – Status Line

- 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

- 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

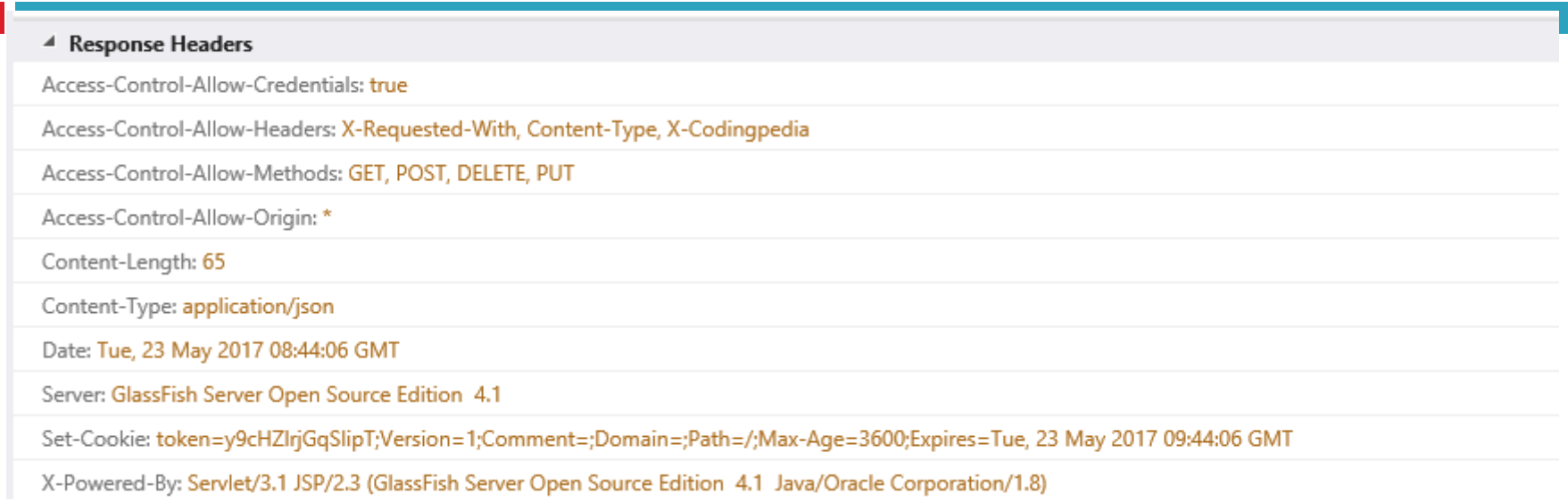
HTTP Overview - Cookies

- 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

- 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



- The expire date
- Some security policies

HTTP Overview - Cookies

- 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

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

Outline

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

Security

- 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

- **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

- 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

- 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

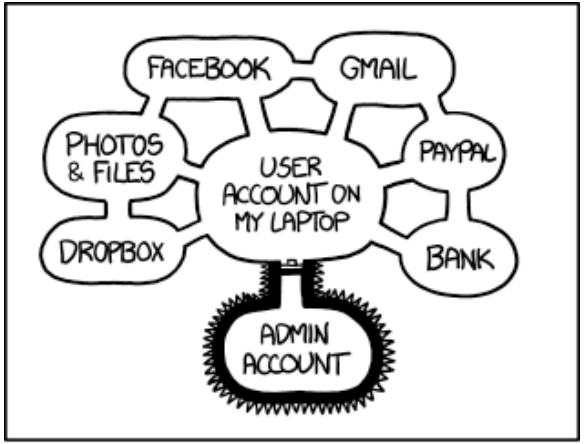
- Web security apply to vulnerabilities that affect web applications.
- Typically web applications are the most exposed assets to an attacker
- And http is really fragile..

Web Security

- 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

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



The web is a mess...

Web Security

- 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 brower!
  - Rule of thumb: *If you need to send a link to the victim, then probably it is a client-side vulnerability*

Web Security

- 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

- **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

- 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

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

Web Security – Some useful tools

- 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

- 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`

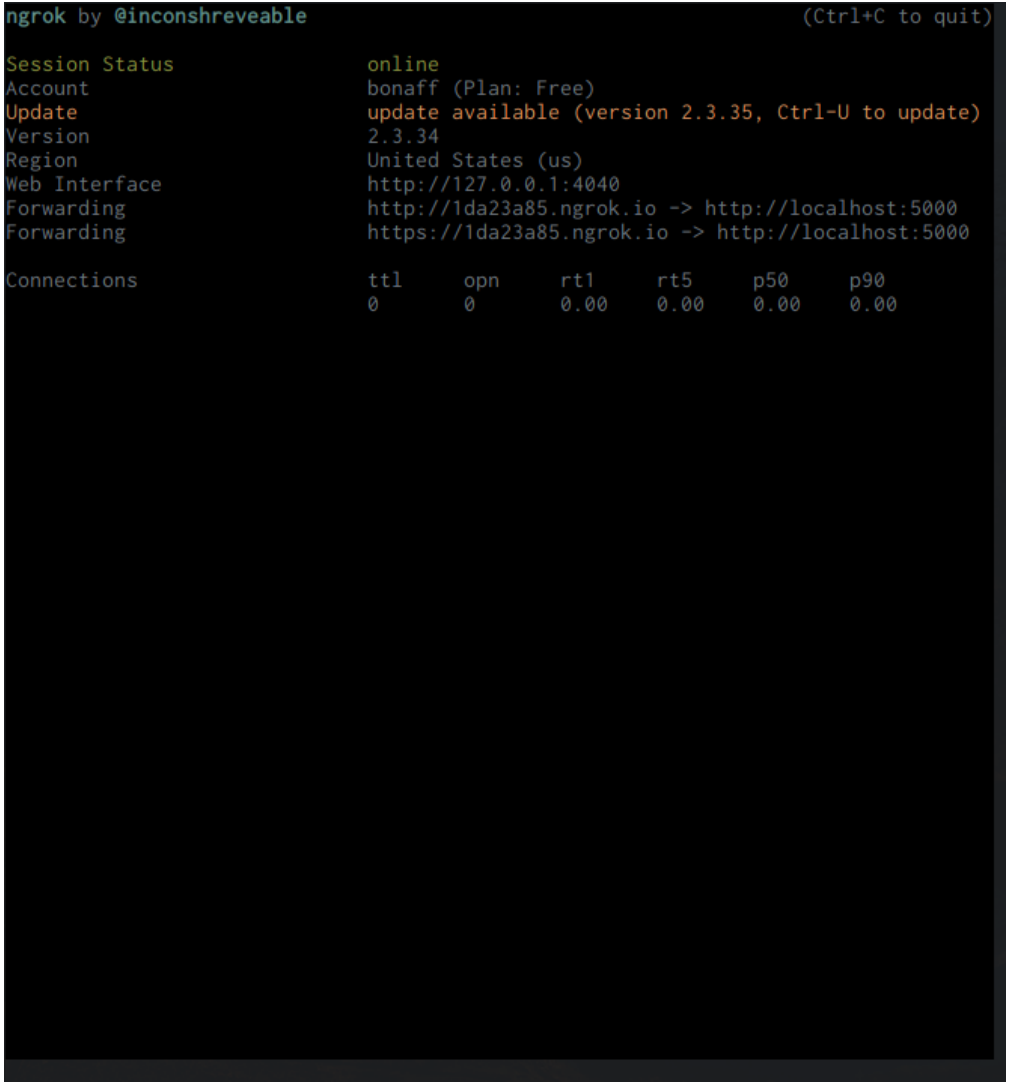
Launch it from a test directory! You don't want to leak your .ssh directory !

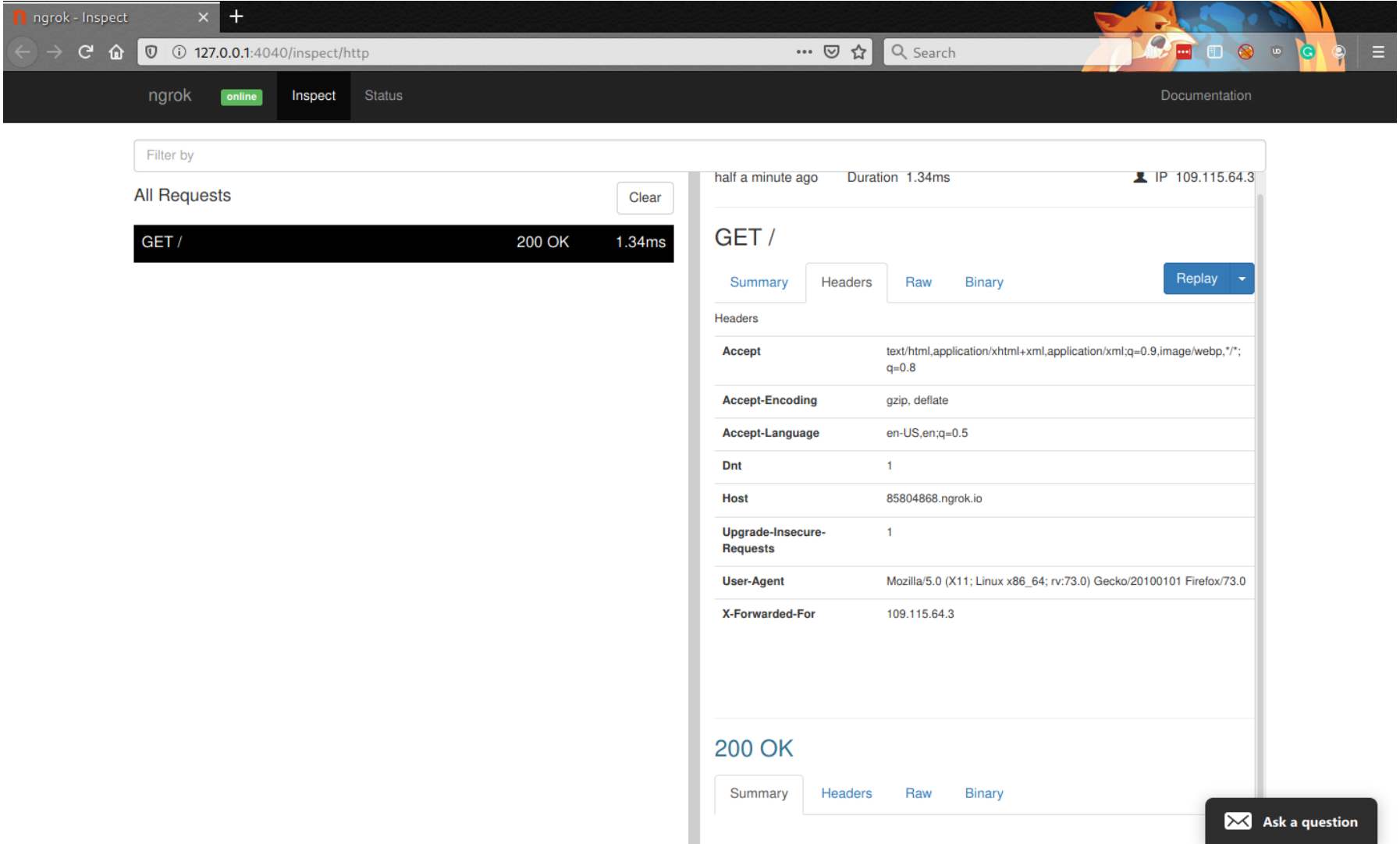
ngrok

- 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

- 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 @inconshreveable (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
127.0.0.1:4040/inspect/http
Search
ngrok
online
Inspect
Status
Documentation
Filter by
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