v4

Penetration Testing Student

Web Applications

Section 01 | Module 03

Table of Contents

Module 3 | Web Applications

Learning Objectives

By the end of this module, you should have a better understanding of:

- HTTP Protocol
- Burp Suite basics

Introduction

3.1 Introduction

Web applications are applications running on **web servers** and accessible via web browsers.

Many people interact with web applications every day, as currently, almost every website on the Internet includes some kind of **intelligence** in its web pages.

The intelligence could either be on the client side or the server side.

3.1 Introduction

The web app world is extremely **heterogeneous**. Every web application is different from others because developers have many ways to accomplish the same task.

As paraphrased from Stan Lee, “*with great flexibility comes great power of messing things up*".

In other words, having flexibility in web app development also means having flexibility in creating insecure code.

3.1 Introduction

To understand web application security, you need to know some web application fundamental aspects:

- HTTP Protocol Basics
- Cookies
- Sessions
- Same Origin Policy

HTTP Protocol Basics

3.2 HTTP Protocol Basics

How does this support my pentesting career?

- The ability to exploit web applications and find vulnerabilities in web servers and services
- Web applications technology is used market-wide also by desktop or mobile applications

3.2 HTTP Protocol Basics

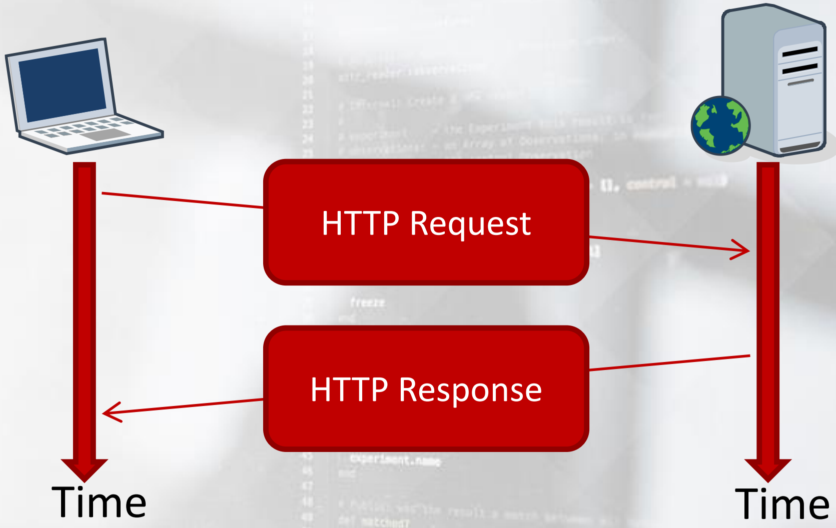
Hypertext Transfer Protocol (HTTP) is the most used application protocol on the Internet. It is the client-server protocol used to transfer web pages and web application data.

In HTTP, the client, usually a web browser, connects to a web server such as **MS IIS** or **Apache HTTP Server**. HTTP is also used under the hood by many mobile and modern applications.

3.2 HTTP Protocol Basics

During an HTTP communication, the client and the server exchange **messages.**

The client sends **requests** to the server and gets back **responses**.

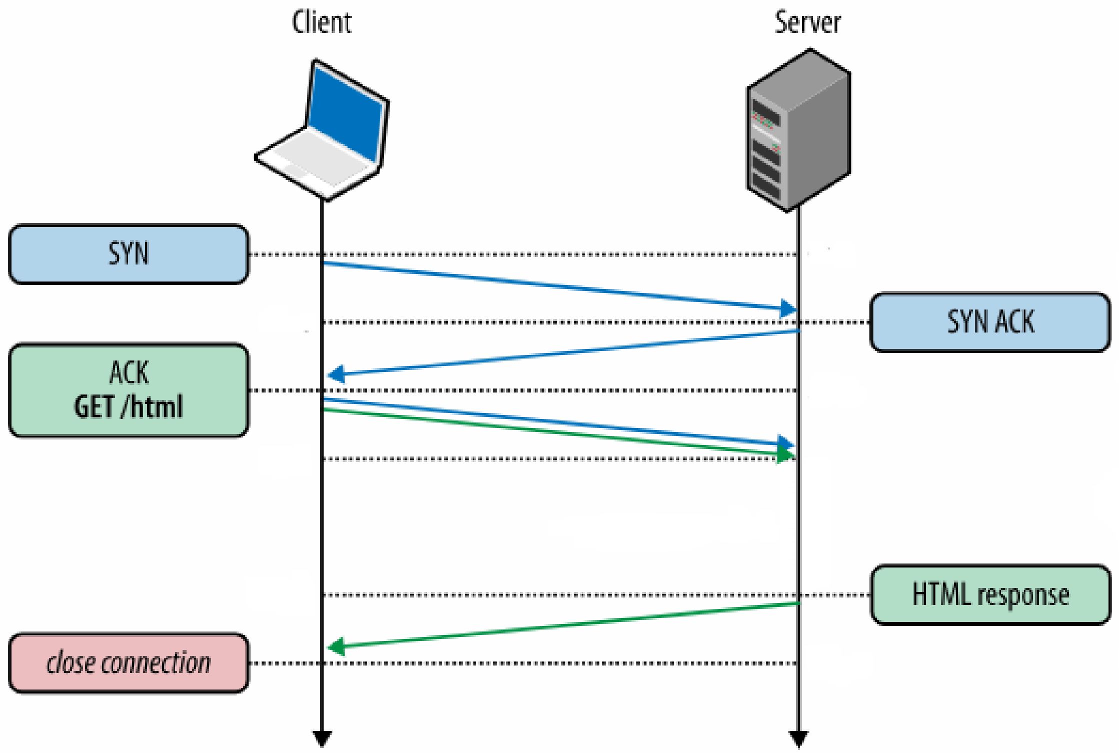


3.2 HTTP Protocol Basics

HTTP works on top of TCP protocol.

That means, first a TCP connection is established, and then the client sends its request, and waits for the answer. The server processes the request and sends back its answer, providing a status code and appropriate data.

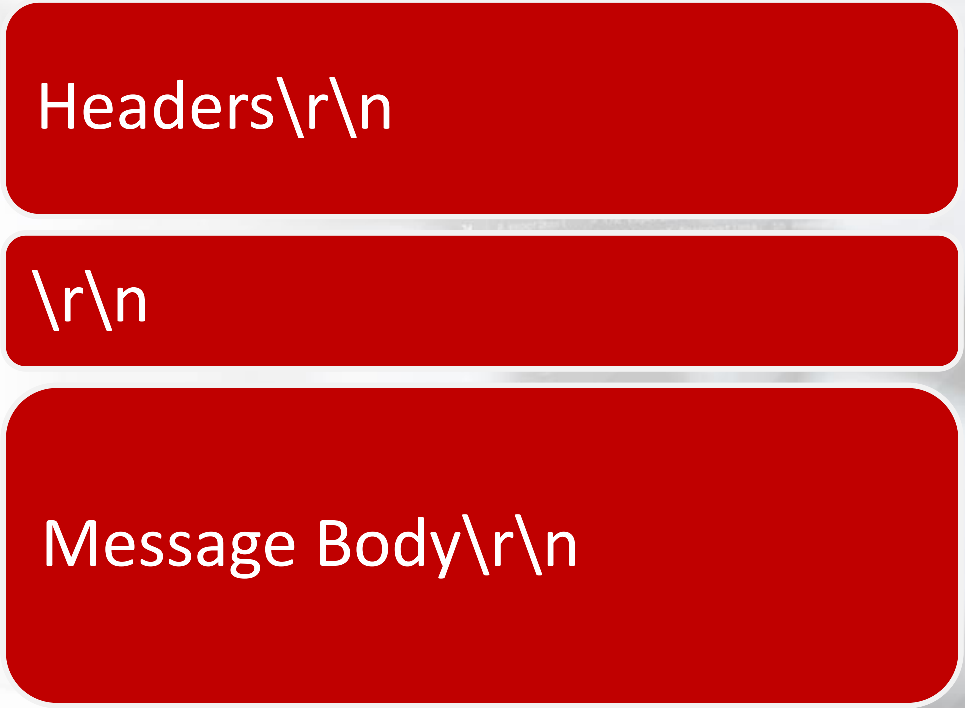
3.2 HTTP Protocol Basics



https://hpbn.co/http1x/

3.2 HTTP Protocol Basics

The format of an HTTP message is:



3.2 HTTP Protocol Basics

To end lines in HTTP, you have to use the `\r` (carriage return) and the `\n` (newline) characters.

The header contains a request followed by some header fields. Every header field has the following format:

- `Header-name: header value`

3.2.1 HTTP Requests

The following is an HTTP request example.

```
GET / HTTP/1.1
Host: www.elearnsecurity.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:31.0) Gecko/20100101 Firefox/31.0
Iceweasel/31.2.0
Accept: text/html
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

3.2.1 HTTP Requests

This request has an empty body, as there is nothing after the two empty lines following the headers.

```
GET / HTTP/1.1
Host: www.elearnsecurity.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:31.0) Gecko/20100101 Firefox/31.0
Iceweasel/31.2.0
Accept: text/html
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive


```

3.2.1 HTTP Requests

This is the **HTTP verb** of the request.

The HTTP verb, or request method, states the type of the request.

```
GET / HTTP/1.1
Host: www.elearnsecurity.com
User-Agent: Mozilla/5.0 (X11; Linux
x86_64; rv:31.0) Gecko/20100101
Firefox/31.0 Iceweasel/31.2.0
Accept: text/html
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

3.2.1 HTTP Requests

GET is used when opening web resources.

If you open a browser and type www.elearnsecurity.com in the address bar, your browser will send this very same request to the server.

```
GET / HTTP/1.1
Host: www.elearnsecurity.com
User-Agent: Mozilla/5.0 (X11; Linux
x86_64; rv:31.0) Gecko/20100101
Firefox/31.0 Iceweasel/31.2.0
Accept: text/html
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

3.2.1 HTTP Requests

After the HTTP VERB you can see the **path** (/) and the **protocol version** (HTTP 1.1).

The path tells the server which resource the browser is asking for. The protocol version tells the server how to communicate with the browser.

```
GET / HTTP/1.1
Host: www.elearnsecurity.com
User-Agent: Mozilla/5.0 (X11; Linux
x86_64; rv:31.0) Gecko/20100101
Firefox/31.0 Iceweasel/31.2.0
Accept: text/html
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

3.2.1 HTTP Requests

There are many HTTP methods, like:

- PUT
- TRACE
- HEAD
- POST

These are only a few, but know that there are many more out there.

```
GET / HTTP/1.1
Host: www.elearnsecurity.com
User-Agent: Mozilla/5.0 (X11; Linux
x86_64; rv:31.0) Gecko/20100101
Firefox/31.0 Iceweasel/31.2.0
Accept: text/html
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

3.2.1 HTTP Requests

The **Host** header field specifies the Internet hostname and port number of the resource being requested.

A web server can host multiple websites. This header field tells the server which site the client is asking for.

```
GET / HTTP/1.1
Host: www.elearnsecurity.com
User-Agent: Mozilla/5.0 (X11; Linux
x86_64; rv:31.0) Gecko/20100101
Firefox/31.0 Iceweasel/31.2.0
Accept: text/html
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

3.2.1 HTTP Requests

The host value is obtained from the URI of the resource.

In this case:

`www.elearnsecurity.com`

```
GET / HTTP/1.1
Host: www.elearnsecurity.com
User-Agent: Mozilla/5.0 (X11; Linux
x86_64; rv:31.0) Gecko/20100101
Firefox/31.0 Iceweasel/31.2.0
Accept: text/html
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

http://www.w3.org/TR/uri-clarification/

3.2.1 HTTP Requests

User-Agent tells the server what client software is issuing the request.

A client could be:

- **Firefox**
- **Internet Explorer**
- **Safari**
- **Opera**
- **Chrome**
- **A mobile app...**

```
GET / HTTP/1.1
Host: www.elearnsecurity.com
User-Agent: Mozilla/5.0 (X11; Linux
x86_64; rv:31.0) Gecko/20100101
Firefox/31.0 Iceweasel/31.2.0
Accept: text/html
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

3.2.1 HTTP Requests

It also reveals to the server the operating system version.

```
GET / HTTP/1.1
Host: www.elearnsecurity.com
User-Agent: Mozilla/5.0 (X11; Linux
x86_64; rv:31.0) Gecko/20100101
Firefox/31.0 Iceweasel/31.2.0
Accept: text/html
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

3.2.1 HTTP Requests

The browser sends the Accept header field to specify which document type it is expecting in the response.

```
GET / HTTP/1.1
Host: www.elearnsecurity.com
User-Agent: Mozilla/5.0 (X11; Linux
x86_64; rv:31.0) Gecko/20100101
Firefox/31.0 Iceweasel/31.2.0
Accept: text/html
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

3.2.1 HTTP Requests

Similarly, with **Accept-Language**, the browser can ask for a specific (human) language in the response.

```
GET / HTTP/1.1
Host: www.elearnsecurity.com
User-Agent: Mozilla/5.0 (X11; Linux
x86_64; rv:31.0) Gecko/20100101
Firefox/31.0 Iceweasel/31.2.0
Accept: text/html
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

3.2.1 HTTP Requests

Accept-Encoding works similarly to Accept but restricts the content encoding, not the content itself.

In this case, the browser accepts two types of compression:

- gzip
- deflate

```
GET / HTTP/1.1
Host: www.elearnsecurity.com
User-Agent: Mozilla/5.0 (X11; Linux
x86_64; rv:31.0) Gecko/20100101
Firefox/31.0 Iceweasel/31.2.0
Accept: text/html
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

3.2.1 HTTP Requests

The **Connection** header field allows the sender to specify options that are desired for that particular connection.

Future communications with the server will reuse the current connection.

```
GET / HTTP/1.1
Host: www.elearnsecurity.com
User-Agent: Mozilla/5.0 (X11; Linux
x86_64; rv:31.0) Gecko/20100101
Firefox/31.0 Iceweasel/31.2.0
Accept: text/html
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

3.2.2 HTTP Responses

When the server receives a request, it processes it and then sends an **HTTP response** to the client. The response has its own header format.

```
HTTP/1.1 200 OK
Date: Wed, 19 Nov 2014 10:06:45 GMT
Cache-Control: private, max-age=0
Content-Type: text/html; charset=UTF-8
Content-Encoding: gzip
Server: Apache/2.2.15 (CentOS)
Content-Length: 99043



< PAGE CONTENT > ...
```

3.2.2 HTTP Responses

As you can see, this response has a message body (`< PAGE CONTENT >`). The header and message body are separated by two empty lines (`\r\n\r\n`).

```
HTTP/1.1 200 OK
Date: Wed, 19 Nov 2014 10:06:45 GMT
Cache-Control: private, max-age=0
Content-Type: text/html; charset=UTF-8
Content-Encoding: gzip
Server: Apache/2.2.15 (CentOS)
Content-Length: 99043


< PAGE CONTENT > ...
```

3.2.2 HTTP Responses

The first line of a Response message is the **Status-Line**, which consists of the **protocol version** (HTTP 1.1) followed by a numeric **status code** (200) and its relative **textual meaning** (OK).

```
HTTP/1.1 200 OK
Date: Wed, 19 Nov 2014 10:06:45 GMT
Cache-Control: private, max-age=0
Content-Type: text/html;
charset=UTF-8
Content-Encoding: gzip
Server: Apache/2.2.15 (CentOS)
Content-Length: 99043


< PAGE CONTENT > ...
...
```

3.2.2 HTTP Responses

The more common status codes are:

- **200 OK**: the resource is found.
- **301 Moved Permanently**: the requested resource has been assigned a new permanent URI.
- **302 Found**: the resource is temporarily under another URI.

```
HTTP/1.1 200 OK
Date: Wed, 19 Nov 2014 10:06:45 GMT
Cache-Control: private, max-age=0
Content-Type: text/html;
charset=UTF-8
Content-Encoding: gzip
Server: Apache/2.2.15 (CentOS)
Content-Length: 99043


< PAGE CONTENT > ...
...
```

3.2.2 HTTP Responses

- **403 Forbidden**: the client does not have enough privileges, and the server refuses to fulfill the request.
- **404 Not Found**: the server cannot find a resource matching the request.
- **500 Internal Server Error**: the server does not support the functionality required to fulfill the request.

```
HTTP/1.1 200 OK
Date: Wed, 19 Nov 2014 10:06:45 GMT
Cache-Control: private, max-age=0
Content-Type: text/html;
charset=UTF-8
Content-Encoding: gzip
Server: Apache/2.2.15 (CentOS)
Content-Length: 99043



< PAGE CONTENT > ...
...
```

3.2.2 HTTP Responses

Date represents the date and time at which the message was originated.

```
HTTP/1.1 200 OK
Date: Wed, 19 Nov 2014 10:06:45 GMT
Cache-Control: private, max-age=0
Content-Type: text/html;
charset=UTF-8
Content-Encoding: gzip
Server: Apache/2.2.15 (CentOS)
Content-Length: 99043


< PAGE CONTENT > ...
...
```

3.2.2 HTTP Responses

With the **Cache-Control** header, the server informs the client about cached content.

Using cached content saves bandwidth, as it prevents the client from re-requesting unmodified content.

```
HTTP/1.1 200 OK
Date: Wed, 19 Nov 2014 10:06:45 GMT
Cache-Control: private, max-age=0
Content-Type: text/html;
charset=UTF-8
Content-Encoding: gzip
Server: Apache/2.2.15 (CentOS)
Content-Length: 99043


< PAGE CONTENT > ...
...
```

3.2.2 HTTP Responses

Content-Type lets the client know how to interpret the body of the message.

```
HTTP/1.1 200 OK
Date: Wed, 19 Nov 2014 10:06:45 GMT
Cache-Control: private, max-age=0
Content-Type: text/html;
charset=UTF-8
Content-Encoding: gzip
Server: Apache/2.2.15 (CentOS)
Content-Length: 99043


< PAGE CONTENT > ...
...
```

3.2.2 HTTP Responses

Content-Encoding extends Content-Type.

In this case, the message body is compressed with gzip.

```
HTTP/1.1 200 OK
Date: Wed, 19 Nov 2014 10:06:45 GMT
Cache-Control: private, max-age=0
Content-Type: text/html;
charset=UTF-8
Content-Encoding: gzip
Server: Apache/2.2.15 (CentOS)
Content-Length: 99043


< PAGE CONTENT > ...
...
```

3.2.2 HTTP Responses

The **Server** header field simply contains the header of the server that generated the content.

This (optional) field is very useful during a pentest to identify the software running on a server.

```
HTTP/1.1 200 OK
Date: Wed, 19 Nov 2014 10:06:45 GMT
Cache-Control: private, max-age=0
Content-Type: text/html;
charset=UTF-8
Content-Encoding: gzip
Server: Apache/2.2.15 (CentOS)
Content-Length: 99043


< PAGE CONTENT > ...
...
```

3.2.2 HTTP Responses

Content-Length indicates the length, in bytes, of the message body.

```
HTTP/1.1 200 OK
Date: Wed, 19 Nov 2014 10:06:45 GMT
Cache-Control: private, max-age=0
Content-Type: text/html;
charset=UTF-8
Content-Encoding: gzip
Server: Apache/2.2.15 (CentOS)
Content-Length: 99043


< PAGE CONTENT > ...
...
```

3.2.3 HTTPS

Now that you know how HTTP works, let’s see how to **protect it**!

HTTP content, as in every clear-text protocol, can be easily intercepted or mangled by an attacker on the path. Moreover, HTTP does not provide strong authentication between the parties.

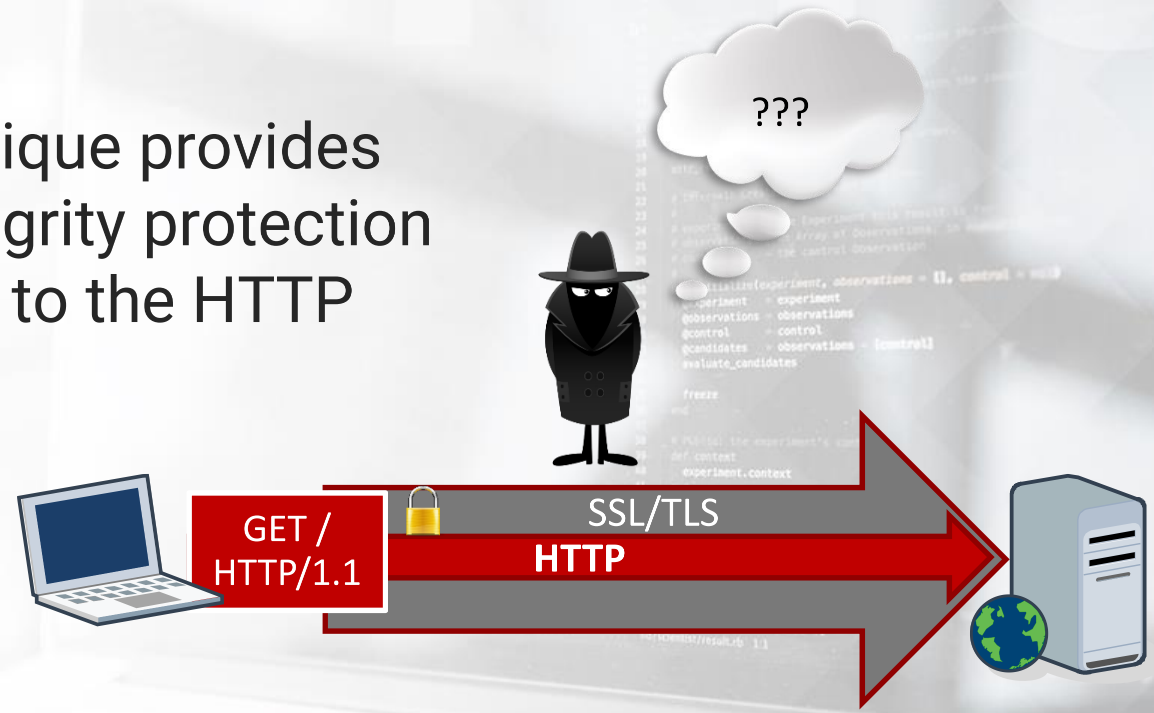
3.2.3 HTTPS

In the following slides, you will see how to **protect HTTP** using an **encryption layer**.

HTTP Secure (HTTPS), or HTTP over SSL/TLS, is a method to run HTTP which is a clear-text protocol over SSL/TLS, a cryptographic protocol.

3.2.3 HTTPS

This layering technique provides confidentiality, integrity protection and authentication to the HTTP protocol.



3.2.3 HTTPS

In other words, when using HTTPS:

- An attacker on the path cannot sniff the application layer communication.
- An attacker on the path cannot alter the application layer data.
- The client can tell the real identity of the server and, sometimes, vice-versa.

3.2.3 HTTPS

HTTPS offers encryption, which means that a network adjacent user is able to sniff the traffic, but he will not know:

- HTTP Request headers, body, target domain
- HTTP Response headers, body

On the other hand, when inspecting HTTPS, one cannot know what domain is contacted and what data is exchanged.

3.2.3 HTTPS

A network adjacent user might recognize:

- Target IP address
- Target port
- DNS or similar protocols may disclose which domain user tries to resolve

3.2.3 HTTPS

HTTPS **does not protect against web application flaws**! All the attacks against an application happen regardless of SSL/TLS.

The extra encryption layer just protects data exchanged between the client and the server. It does not protect from an attack against the application itself.

3.2.3 HTTPS

Attacks such as XSS and SQL injections will still work.

Understanding how HTTP and web applications work is fundamental to mount stealthy and effective attacks!

3.2.4 Video – HTTP and HTTPS Protocols Basics

HTTP and HTTPS Protocols Basics

In this video, you will see HTTP and HTTPS in action both via command line tools or with Burp Proxy.

You will learn how to:

- Perform requests, receive responses and analyze HTTP(s) messages both manually and with an automated tool.
- Use Burp at the end of this module.

**Videos are only available in Full or Elite Editions of the course. To upgrade, click HERE. To access, go to the course in your members area and click the resources drop-down in the appropriate module line.*

3.2.5 References

HTTP Status Code Reference: a document referencing HTTP status codes and their meaning.

SSL/TLS Strong Encryption: An Introduction: Apache's introduction on protecting HTTP by means of SSL/TLS.

HTTP Overview, History, Versions and Standards: History and evolution of HTTP.

http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html
http://httpd.apache.org/docs/2.2/ssl/ssl_intro.html
http://www.tcpipguide.com/free/t_HTTPOverviewHistoryVersionsandStandards.htm

3.3

HTTP Cookies

3.3 HTTP Cookies

HTTP is a **stateless** protocol; this means that websites cannot keep the state of a visit across different HTTP requests.

In other words, every HTTP request is **completely unrelated** to the ones preceding and following it.

3.3 HTTP Cookies

To overcome this limitation, sessions and **cookies** were invented in 1994.

Netscape, a leading company at that time, invented cookies **to make HTTP stateful**.

3.3 HTTP Cookies

How does this support my pentesting career?

- Cookies are foundation of authorization of many applications
- Often exploits rely on stealing cookies

3.3 HTTP Cookies

Cookies are not rocket science. They are just textual information installed by a website into the "cookie jar" of the web browser.

The **cookie jar** is the storage space where a web browser stores the cookies.

3.3.1 Cookies Format

A server can set a cookie via the **Set-Cookie** HTTP header field in a response message. A cookie contains the following attributes:

- The actual content
- An expiration date
- A path
- The domain
- Optional flags:
  - Http only flag
  - Secure flag

```
HTTP/1.1 200 OK
Date: Wed, 19 Nov 2014 10:06:45 GMT
Cache-Control: private, max-age=0
Content-Type: text/html;
charset=UTF-8
Content-Encoding: gzip
Server: Apache/2.2.15 (CentOS)
Set-Cookie: ID=Value; expires=Thu,
21-May-2015 15:25:20 GMT; path=/;
domain=.example.site; HttpOnly
Content-Length: 99043
```

3.3.1 Cookies Format

Cookie content	`ID=Value;`
Expiration date	**`expires`**`=Thu, 21-May-2015 15:25:20 GMT;`
Path	**`path`**`=/example/path;`
Domain	**`domain`**`=.example.site;`
Flag-setting attributes	**`HttpOnly`**`;` **`Secure`**

3.3.2 Cookies Handling

Browsers use domain, path, expires and flags attributes to choose whether or not to send a cookie in a request.

Cookies are sent only to the **valid domain/path** when they are **not expired** and according to their **flags**.

3.3.3 Cookie Domain

The **domain** field and the **path** field set the **scope** of the cookie. The browser sends the cookie only if the request is for the right domain.

When a web server installs a cookie, it sets the domain field, e.g., `elearnsecurity.com`. Then, the browser will use the cookie for every request sent to that domain and **all its subdomains**.

EXAMPLE

3.3.3 Cookie Domain

When a cookie has the domain attribute set to:

- **domain**=elearnsecurity.com

or

- **domain**=.elearnsecurity.com

The browser will send the cookie to:

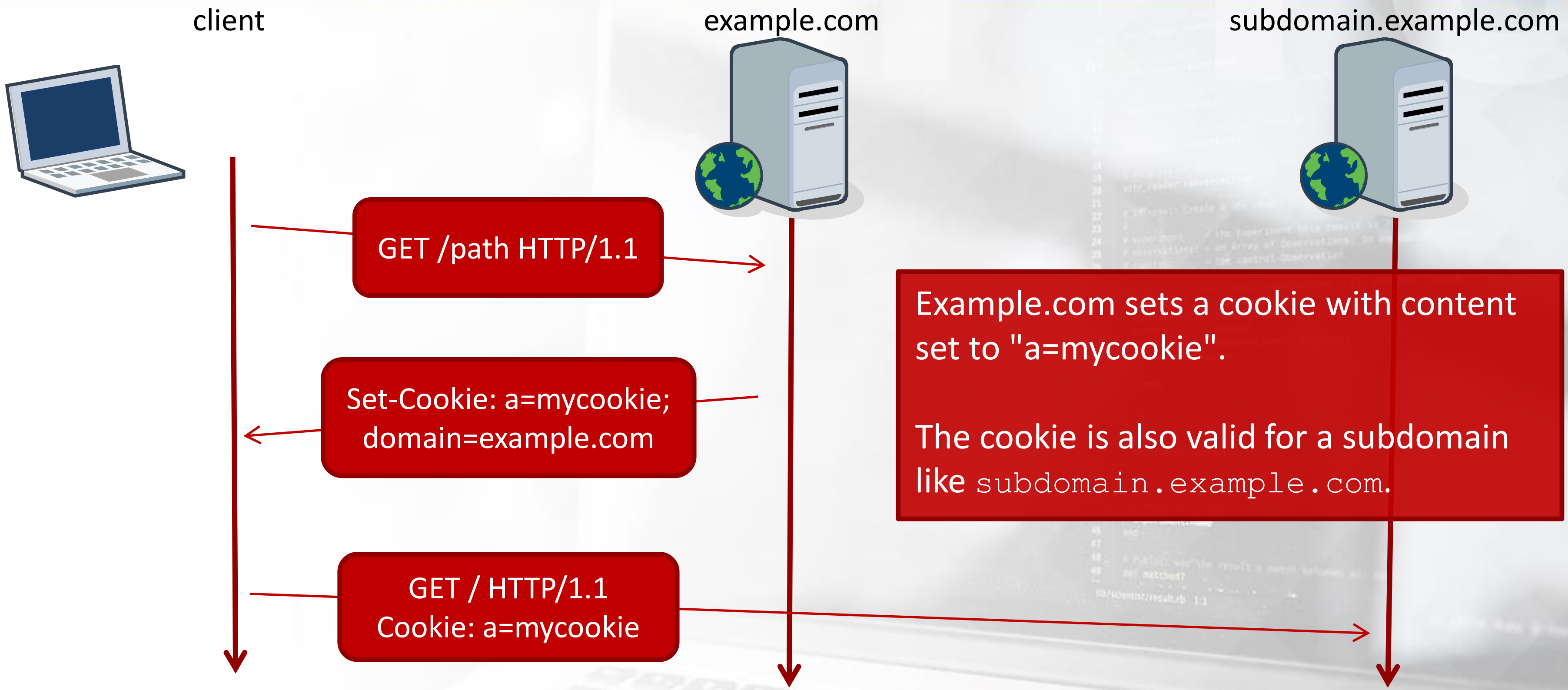
- www.elearnsecurity.com
- whatever.subdomain.elearnsecurity.com
- elearnsecurity.com

3.3.3 Cookie Domain

If the **server does not specify** the domain attribute, the browser will automatically set the domain as the server domain and set the cookie **host-only** flag; this means that the cookie will be sent **only to that precise hostname**.

In the following examples, you will see how a browser chooses to send or not to send a cookie.

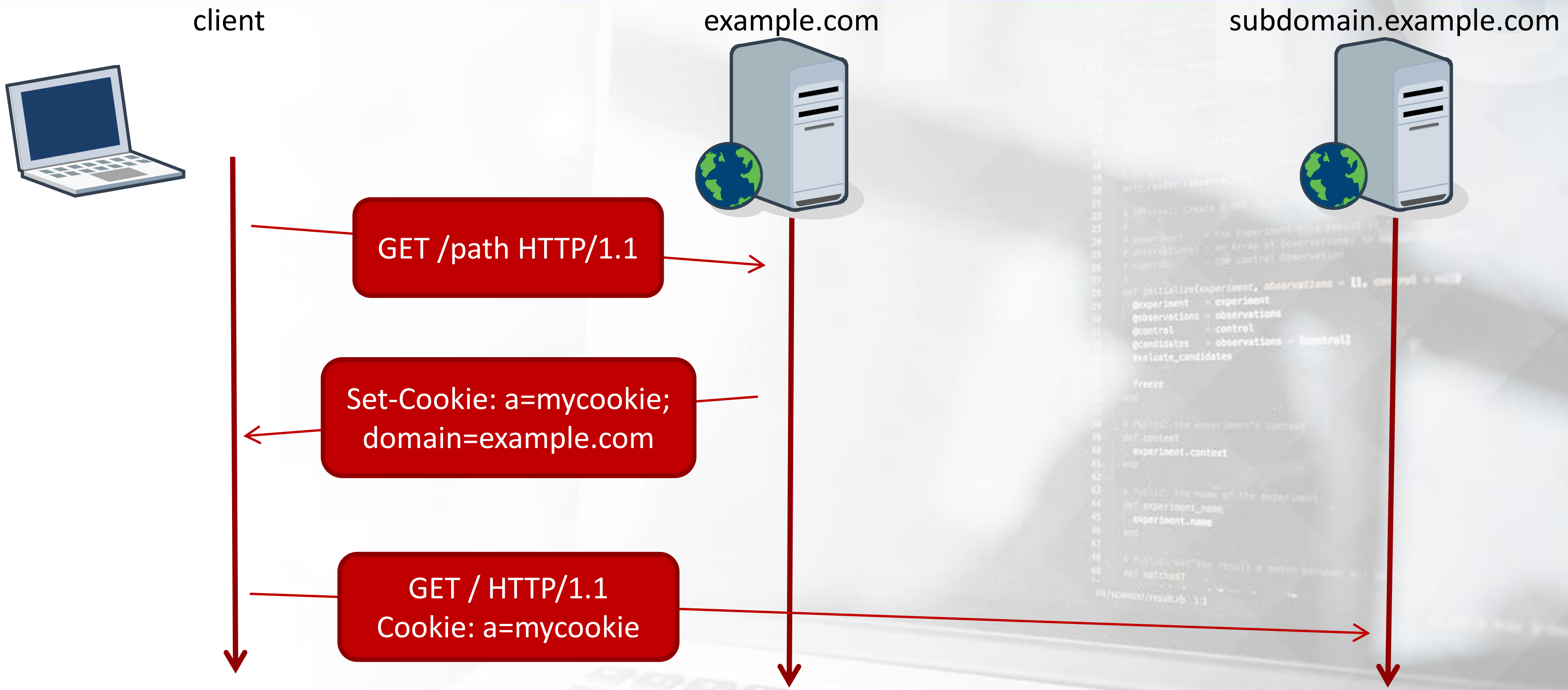
3.3.3.1 Cookie Domain Examples



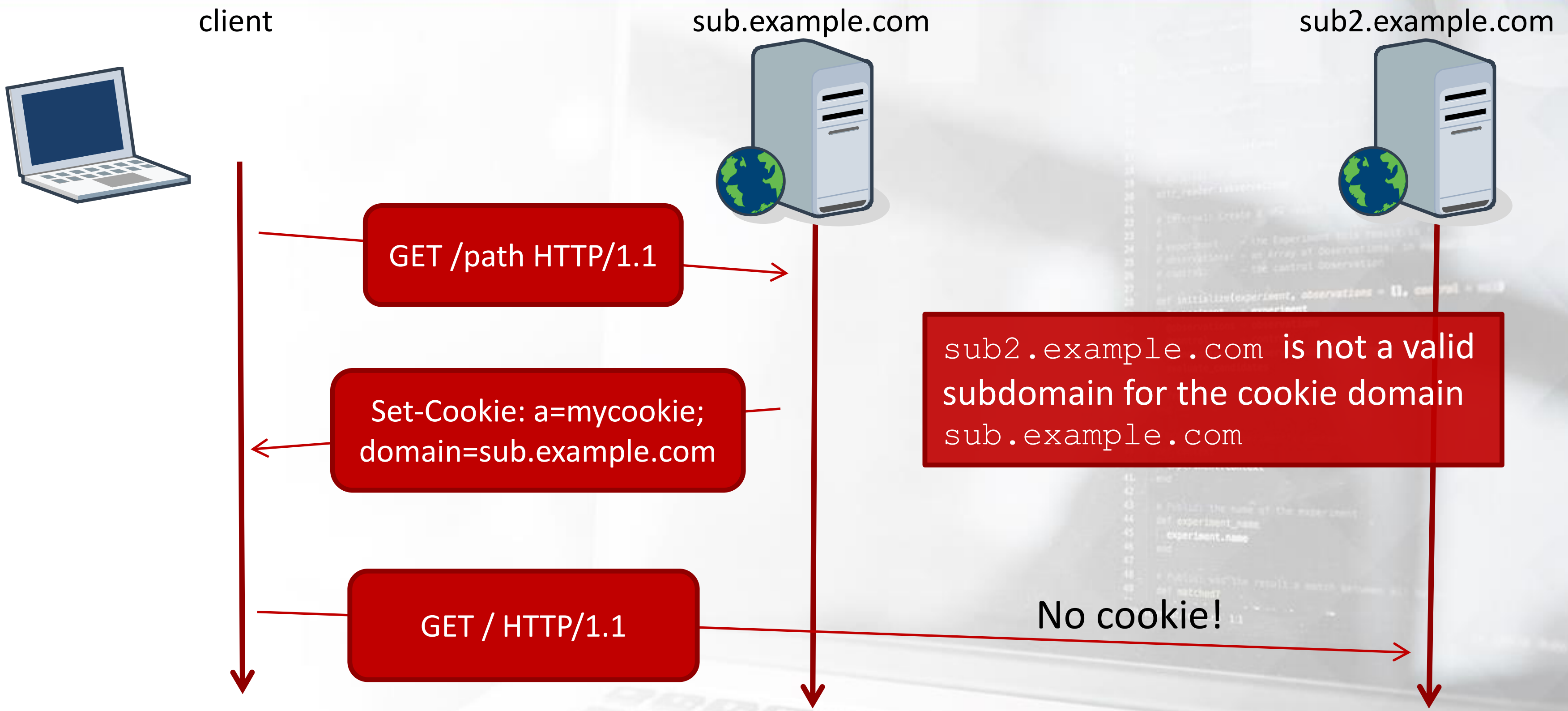
Example.com sets a cookie with content set to "a=mycookie".

The cookie is also valid for a subdomain like `subdomain.example.com`.

3.3.3.1 Cookie Domain Examples

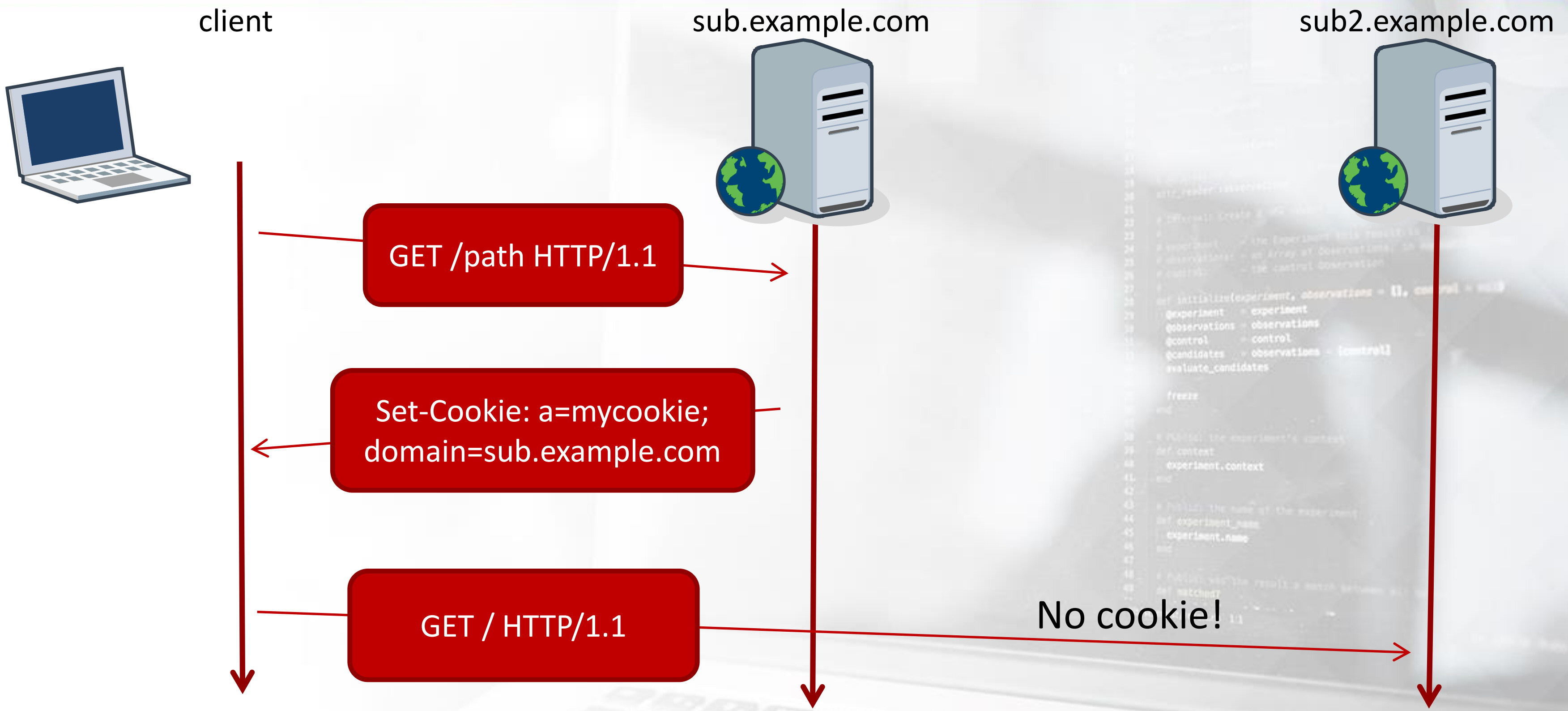


3.3.3.1 Cookie Domain Examples

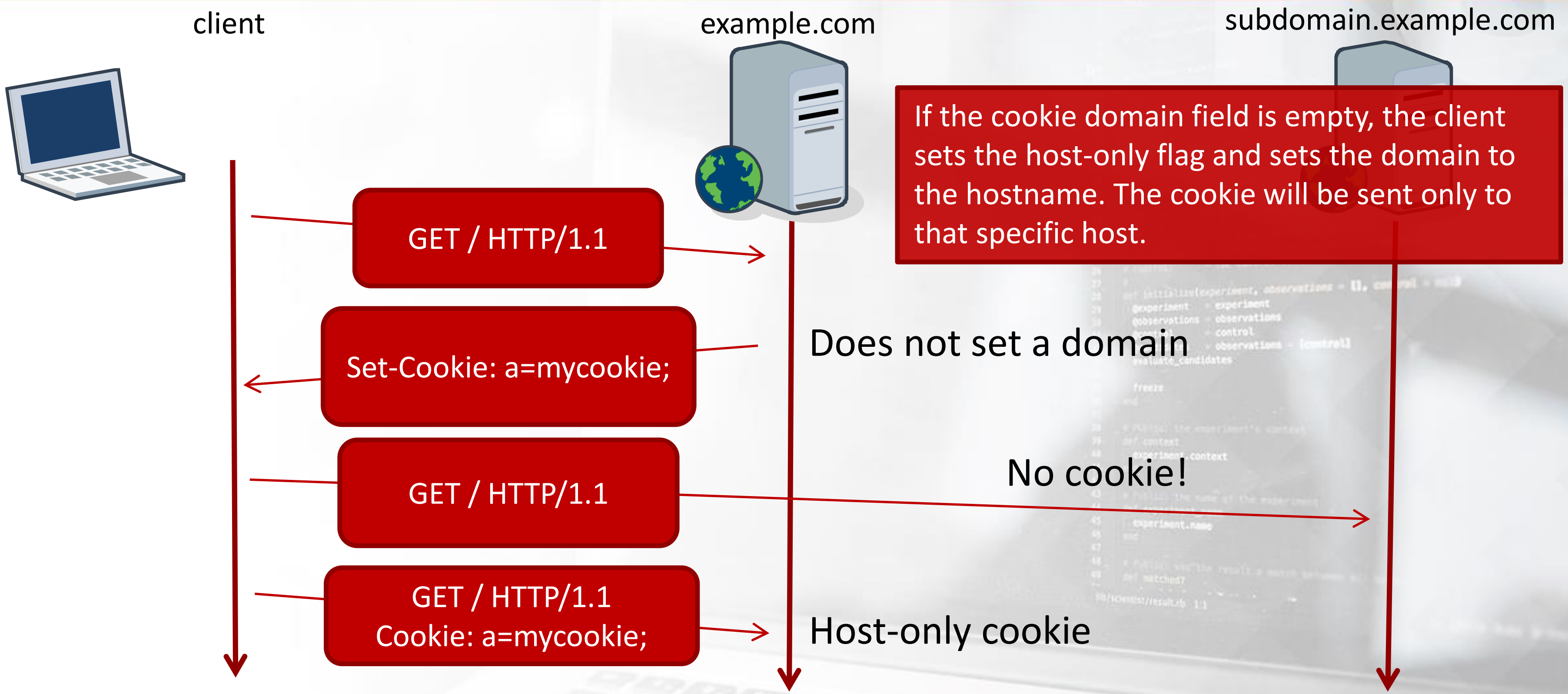


`sub2.example.com` is not a valid subdomain for the cookie domain `sub.example.com`

3.3.3.1 Cookie Domain Examples

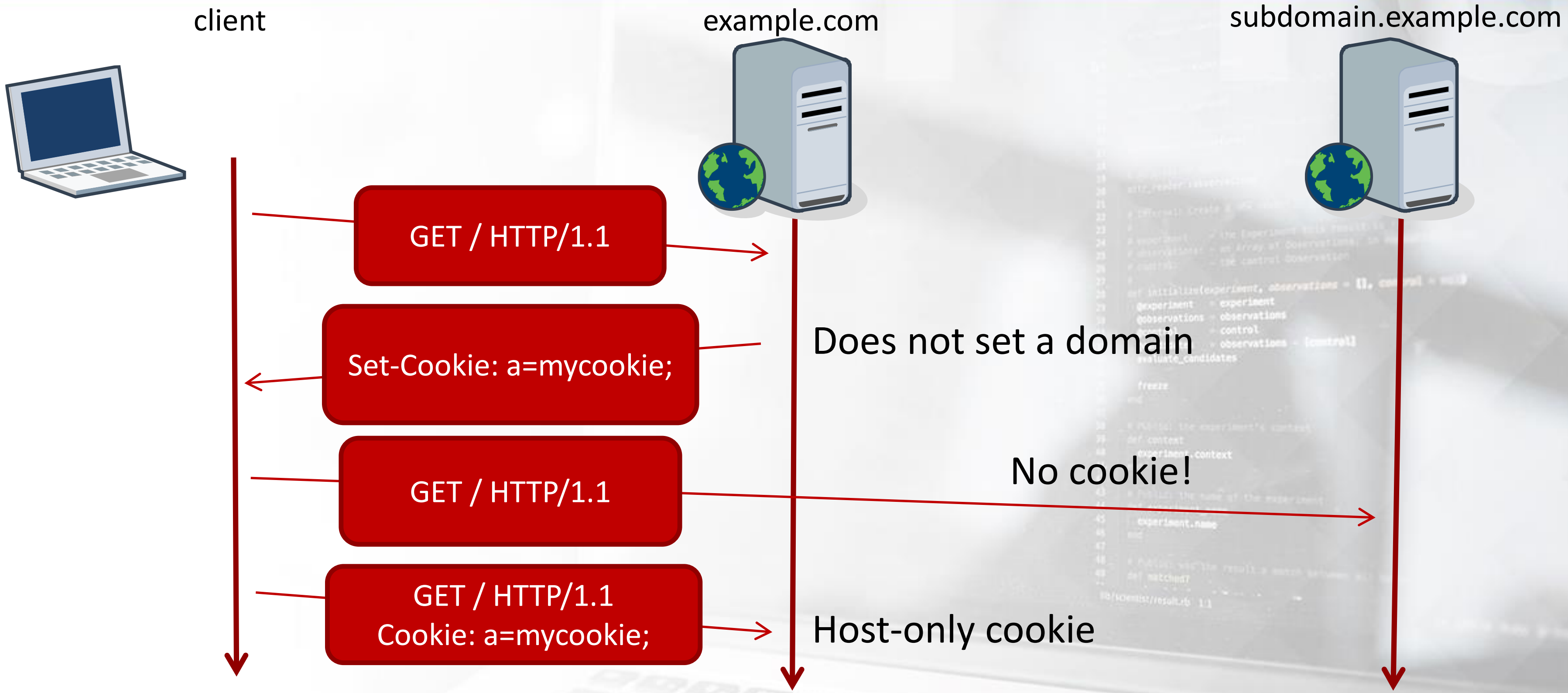


3.3.3.1 Cookie Domain Examples



If the cookie domain field is empty, the client sets the host-only flag and sets the domain to the hostname. The cookie will be sent only to that specific host.

3.3.3.1 Cookie Domain Examples



3.3.4 Cookie Path

As we saw previously, the **path** and the **domain** attributes set the **scope** of a cookie.

The browser will send a cookie to the **right domain and to any subpath of the path** field value.

EXAMPLE

3.3.4 Cookie Path

When a cookie has the path attribute set to:

- **path**=/the/path

The browser will send the cookie to the right domain and to the resources in:

- /the/path
- /the/path/sub
- /the/path/sub/sub/sub/path

But, it will not send it to /otherpath.

3.3.5 Cookie Expires Attribute

The **expires** attribute sets the **validity time window** of a cookie.

A browser will not send an expired cookie to the server. Session cookies expire with the HTTP session; you will see more about that later in this module.

3.3.6 Cookie Http-Only Attribute

When a server installs a cookie into a client with the **http-only attribute**, the client will set the **http-only flag** for that cookie. This mechanism prevents JavaScript, Flash, Java and any other non-HTML technology from reading the cookie, thus preventing cookie stealing via XSS.

You will see how to exploit XSS vulnerabilities in the *Web Application Attacks* module.

3.3.7 Cookie Secure Attribute

Secure flag creates secure cookies that will only be sent over an HTTP**S** connection (they will not be sent over HTTP).

EXAMPLE

3.3.8 Cookie Content

A cookie can carry a number of values. A server can set multiple values with a single `Set-Cookie` header by specifying multiple `KEY=Value` pairs.

```
Set-Cookie: Username="john"; auth=1
```

Sets two values: one for username and one for auth.

3.3.9 Cookie Protocol

RFC6265 states cookies format, how a server can install cookies and how a client can use them.

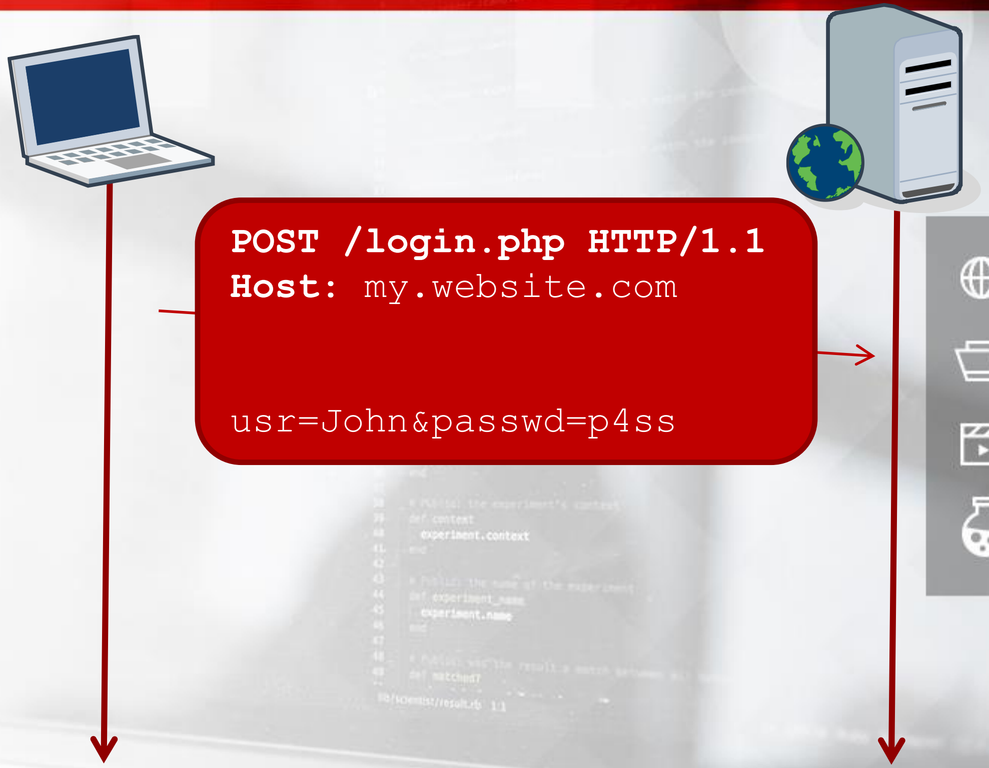
Let‘s see cookies in action with a simple example.

http://tools.ietf.org/html/rfc6265

3.3.9 Cookie Protocol

Cookies are often installed during a login.

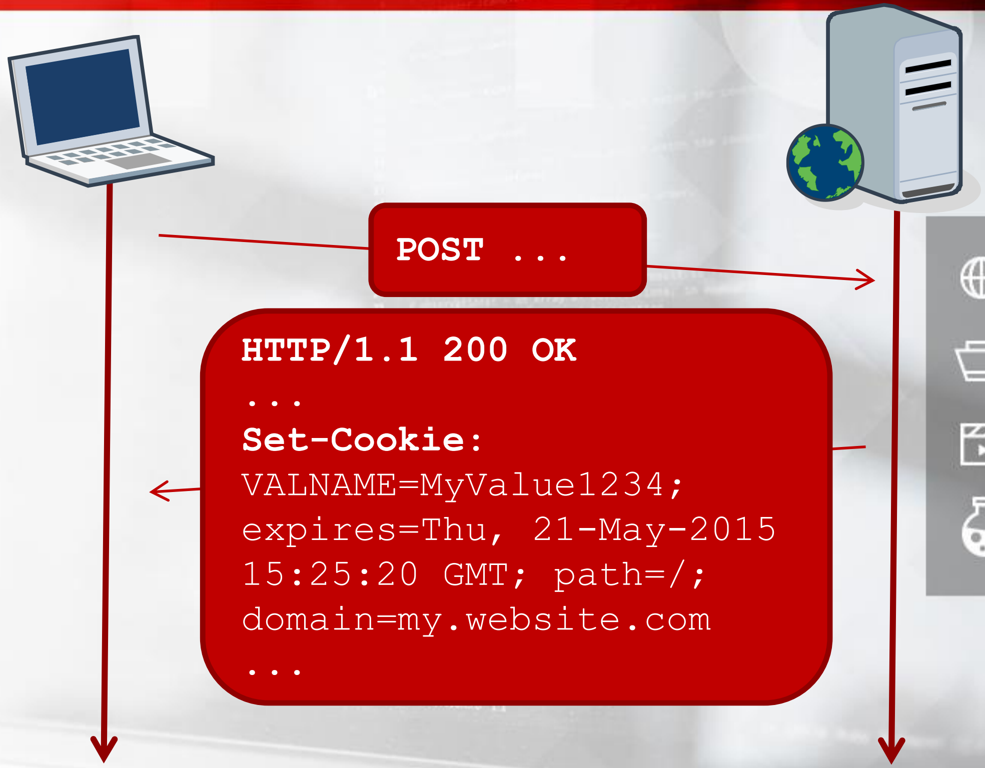
In this example, the browser sends a **POST** request with the username and password.



```
POST /login.php HTTP/1.1
Host: my.website.com


usr=John&passwd=p4ss
```

3.3.9 Cookie Protocol

The server sends a response with a **Set-cookie** header field, thus telling the browser to install the cookie.



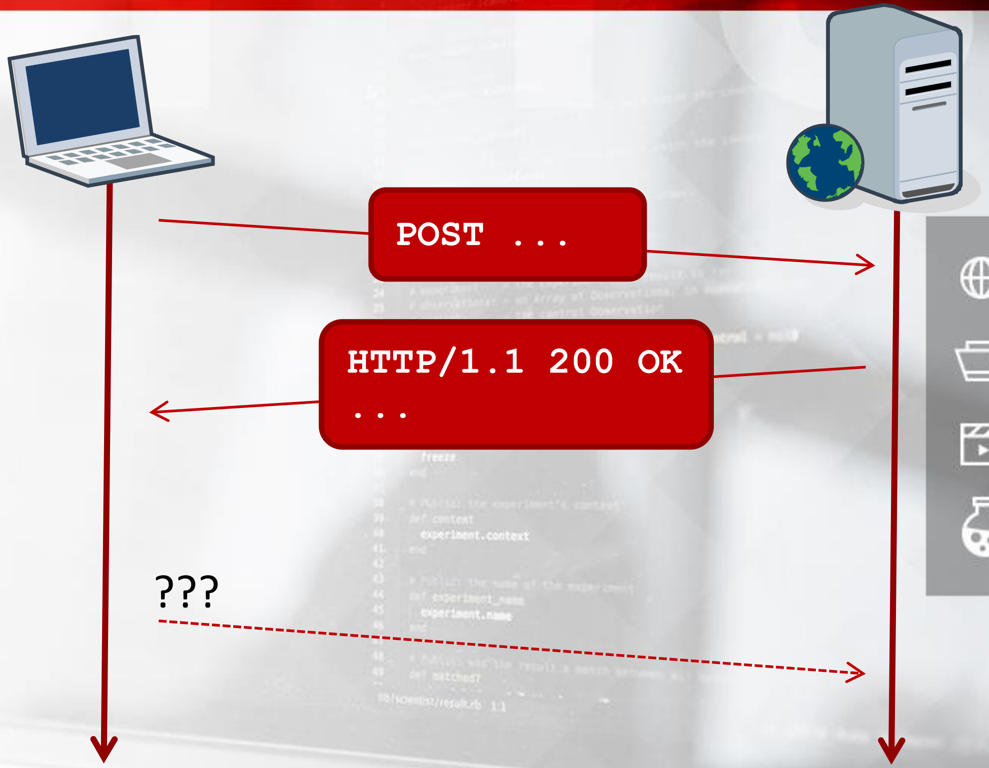
```
POST ...
```

```
HTTP/1.1 200 OK
...
Set-Cookie:
VALNAME=MyValue1234;
expires=Thu, 21-May-2015
15:25:20 GMT; path=/;
domain=my.website.com
...
```

3.3.9 Cookie Protocol

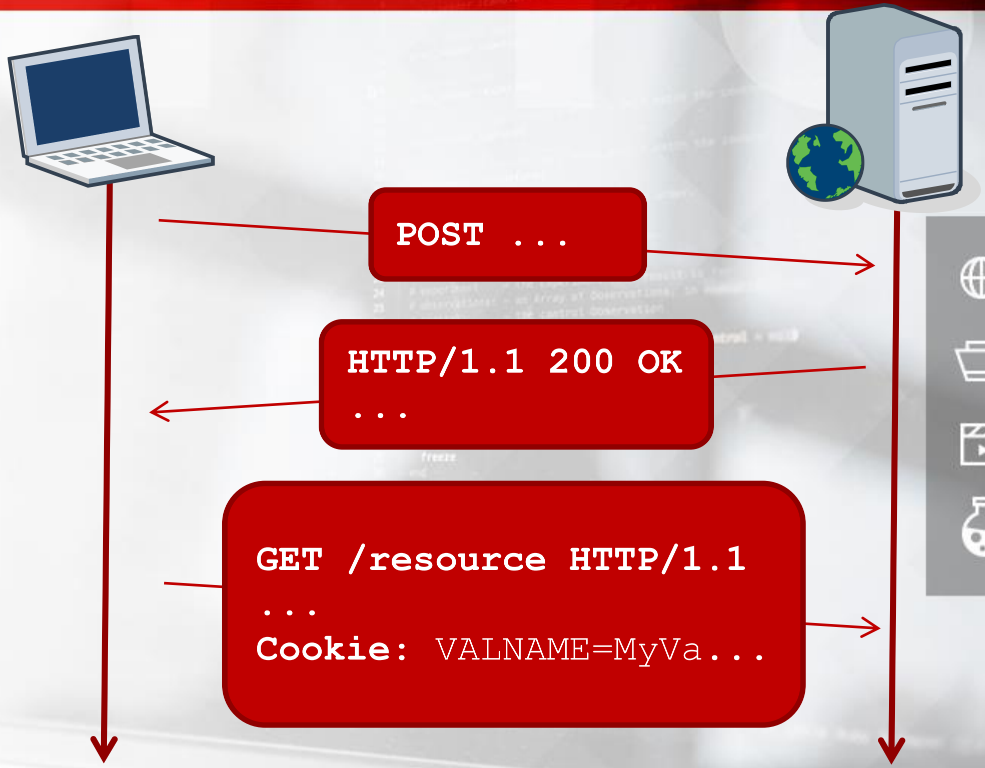
For every subsequent request, the browser considers:

- Domain
- Path
- Expiration
- Flags



3.3.9 Cookie Protocol

If all the checks pass, the browser will insert a **cookie:** header in the request.



```
POST ...
```

```
HTTP/1.1 200 OK
...
```

```
GET /resource HTTP/1.1
...
Cookie: VALNAME=MyVa...
```

Sessions

3.4 Sessions

Sometimes the web developer prefers to store some information on the **server side** instead of the client side; this happens to hide the application logic or just to avoid the back and forth data transmission typical of cookies.

3.4 Sessions

Sessions are a mechanism that lets the website store variables specific for a given visit on the **server side**.

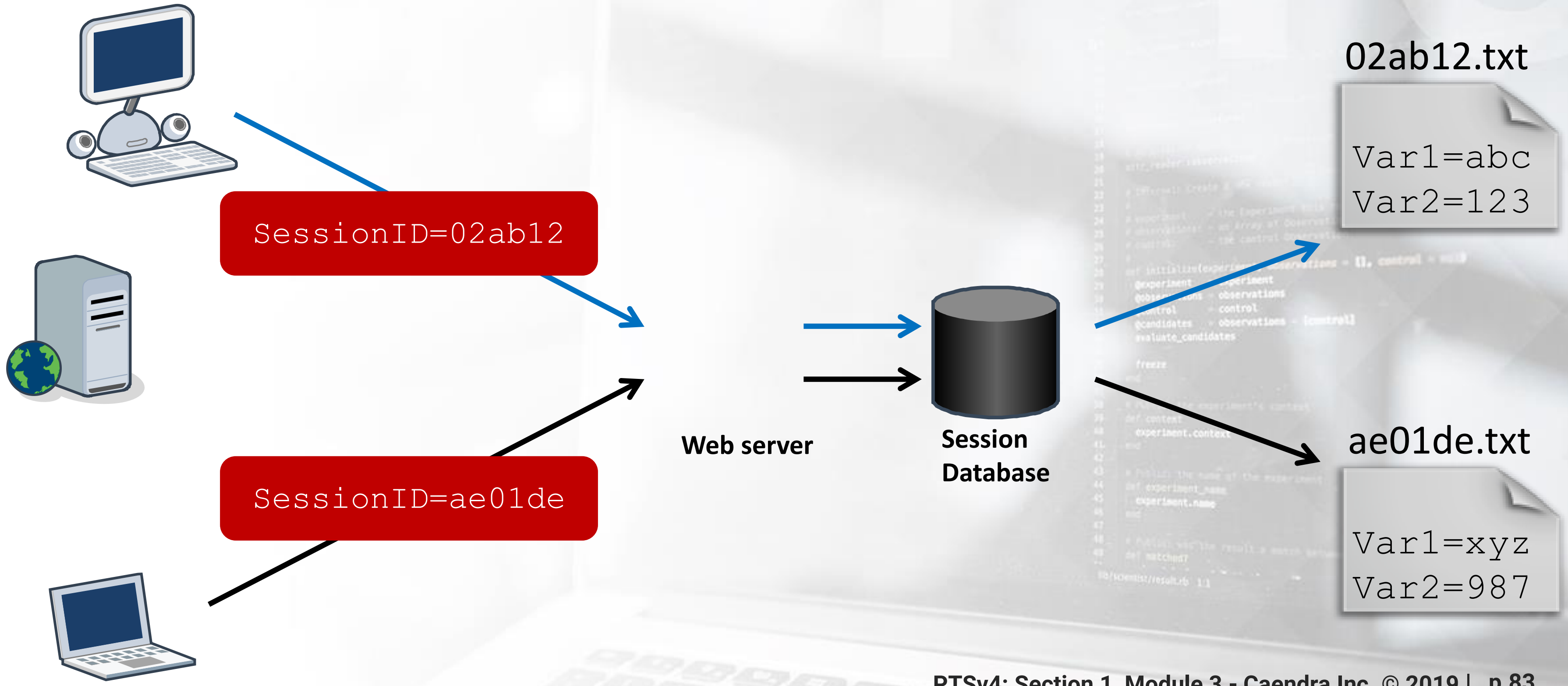
Each user session is identified by a **session id**, or token, which the server assigns to the client.

3.4 Sessions

The client then presents this ID for each subsequent request, thus being recognized by the server.

By means of the session ID, the **server retrieves the state of the client** and all its associated variables. The server stores Session IDs inside text files in its storage. You can find an example in the next slide.

3.4.1 Sessions Example



3.4.2 Session Cookies

Q

How does a web application install session IDs on a web browser?

A

*By using **session cookies**.*

It’s now time to see how to use **HTTP sessions**!

3.4.2 Session Cookies

Session cookies just contain a single parameter value pair referring to the session.

```
•    SESSION=0wvCtOBWDH8w
•    PHPSESSID=l3Kn5Z6Uo4pH
•    JSESSIONID=W7DPUBgh7kTM
```

3.4.2 Session Cookies

Websites running PHP install session cookies by using the "`PHPSESSID`" parameter name, while JSP websites use "`JSESSIONID`". Each development language has its own default session parameter name.

Of course, the web developer can also choose to use a custom parameter name.

3.4.2 Session Cookies

If needed, servers install session cookies after a browser performs some kind of activity, like:

- Opening a specific page
- Changing settings in the web application
- Logging in

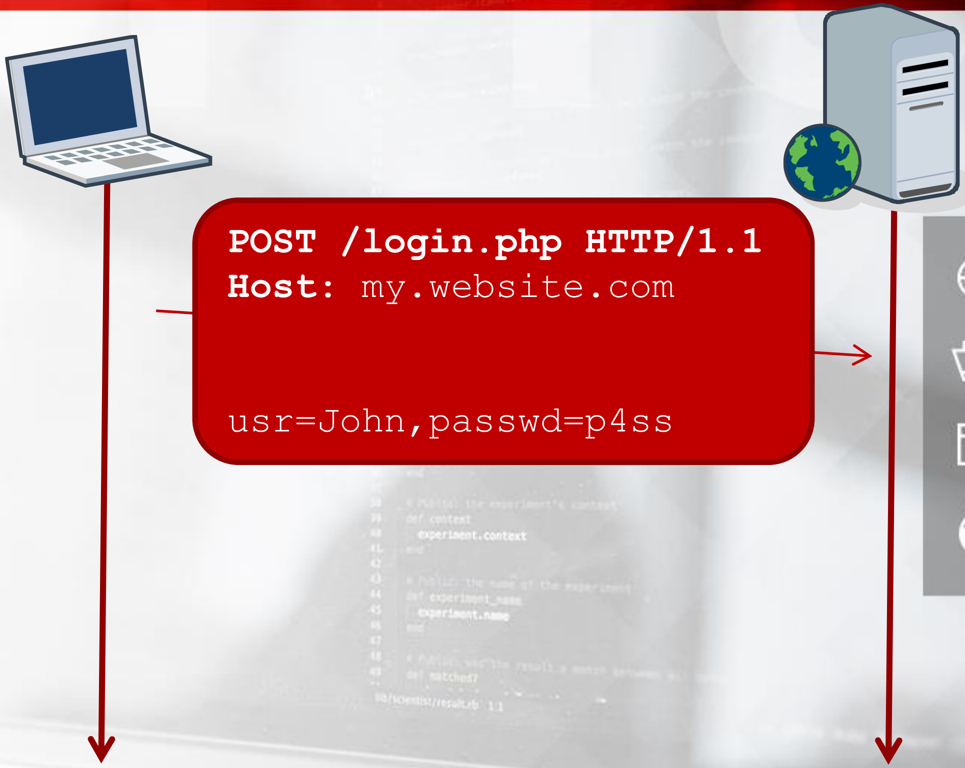
3.4.2 Session Cookies

The browser then uses the cookie in subsequent requests. A session could contain many variables, so sending a small cookie keeps the bandwidth usage low.

In the following example, you can see a session cookie in action.

3.4.2.1 Session Cookies Example

The client uses a login form to POST the user's credentials.

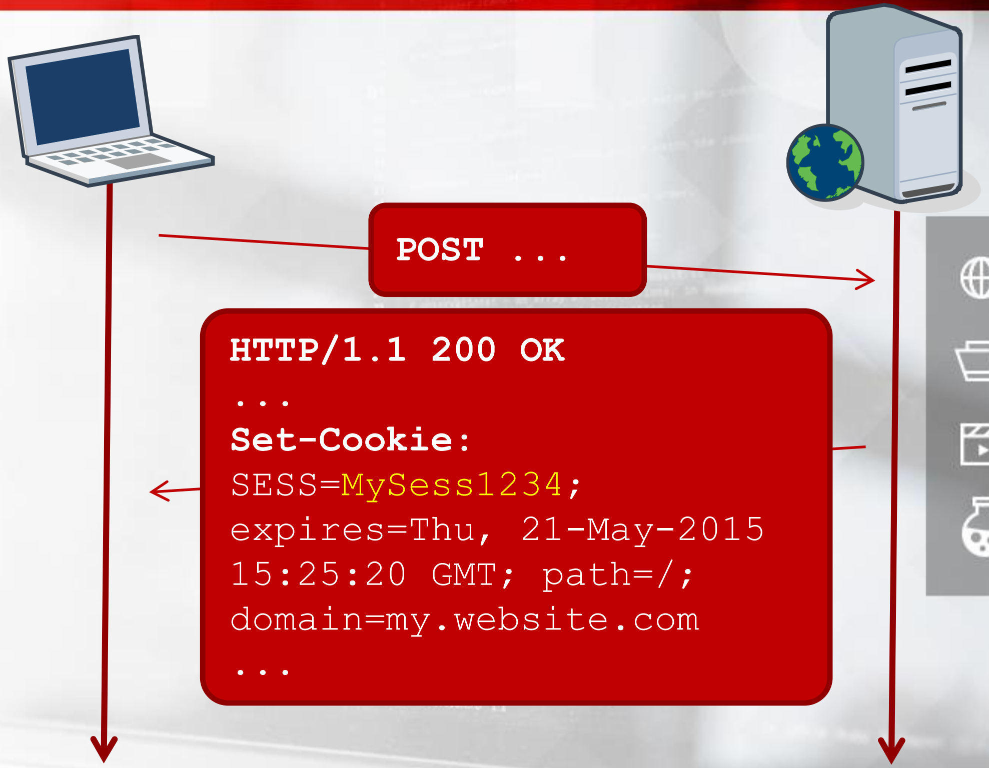


```
POST /login.php HTTP/1.1
Host: my.website.com


usr=John,passwd=p4ss
```

3.4.2.1 Session Cookies Example

The server sends back a response with a Set-cookie header field.

The cookie contains the **session ID**.

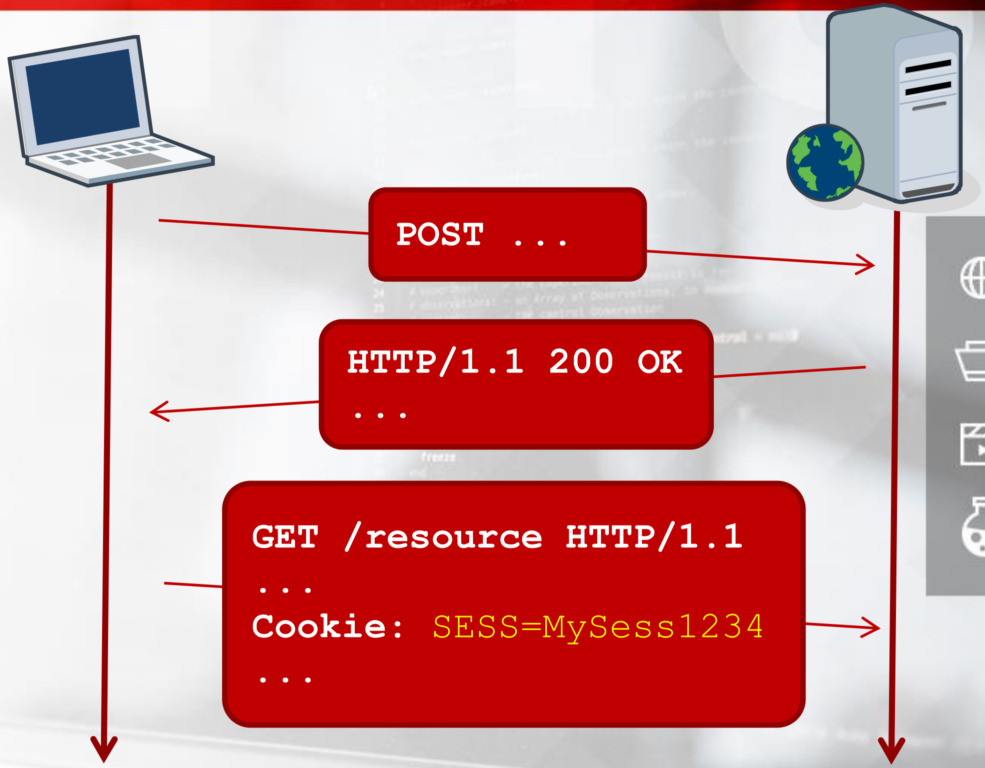


```
POST ...
```

```
HTTP/1.1 200 OK
...
Set-Cookie:
SESS=MySess1234;
expires=Thu, 21-May-2015
15:25:20 GMT; path=/;
domain=my.website.com
...
```

3.4.2.1 Session Cookies Example

The browser will send back the cookie according to the cookie protocol, thus sending the **session ID**.



```
POST ...
```

```
HTTP/1.1 200 OK
...
```

```
GET /resource HTTP/1.1
...
Cookie: SESS=MySess1234
...
```

EXAMPLE

3.4.3 GET Requests

Session IDs can also be transmitted via **GET requests**.

```
http://example.site/resource.php?sessid=k27rds7h8w
```

3.4.4 Video – HTTP Cookies and Sessions

HTTP Cookies & Sessions

In this video, you will see how the cookie protocol works (installation of cookies, manipulation of cookies, and dissecting a cookie, as well as how HTTP sessions work and how they use cookies.

You will see reference to a tool called Firebug. At the time of recording, it was an extension for the Iceweasel browser. Modern Kali Linux comes preinstalled with the Firefox browser instead of Iceweasel, but technically, it is almost the same browser as Iceweasel. Additionally, functionalities of firebug are now part of Developer Tools of Firefox, so instead of downloading the addon, you can simply turn on the Developer Tools by visiting Options -> Web Developer -> Toogle Tools, or just press Ctrl + Shift + I in your browser window.

**Videos are only available in Full or Elite Editions of the course. To upgrade, click HERE. To access, go to the course in your members area and click the resources drop-down in the appropriate module line.*

Same Origin Policy

3.5 Same Origin Policy

Same Origin Policy (SOP) is a critical point of web application security.

This policy prevents JavaScript code from getting or setting properties on a resource coming from a **different origin**.

3.5 Same Origin Policy

The browser uses:

Protocol

Hostname

Port

To determine if JavaScript can access a resource:
Hostname, *port*, and *protocol* **must match**.

EXAMPLE

3.5 Same Origin Policy

A JavaScript script on

$$\underbrace{\texttt{https}}_{\text{protocol}}\underbrace{\texttt{://www.elearnsecurity.com}}_{\text{hostname}}\texttt{:}\underbrace{\texttt{345/}}_{\text{port}}$$

can read resources from:

- `https://www.elearnsecurity.com:345/path`
- `https://www.elearnsecurity.com:345/path/2`

EXAMPLE

3.5 Same Origin Policy

But not from:

- `https://www.elearnsecurity.com/path` (same protocol and domain but different port)
- `http://www.elearnsecurity.com:345/path` (same port and domain but different protocol)
- `https://www.heralab.net:345/path` (same port and protocol but different domain)

3.5.1 HTML Tags

Note that SOP applies only to the **actual code of a script**.

It is still possible to include external resources by using HTML tags like `img`, `script`, `iframe`, `object`, etc.

3.5 Same Origin Policy

The entire web application security is based on Same Origin Policy.

If a script on domain A was able to read content on domain B, it would be possible to steal clients' information and mount a number of very dangerous attacks.

3.6

Burp Suite

3.6 Burp Suite

How does this support my pentesting career?

- Web application analysis
- Finding vulnerabilities
- Attacks
- Burp Suite is one of most used pentesting tools

3.6.1 Intercepting Proxies

Any web application contains many objects like scripts, images, style sheets, client and server-side intelligence.

Having **tools** that help in the **study** and **analysis** of web application behavior is critical.

3.6.1 Intercepting Proxies

An **intercepting proxy** is a tool that lets you analyze and modify any request, and any response exchanged between an HTTP client and a server.

By intercepting HTTP messages, a pentester can study a web application behavior and manually test for vulnerabilities.

3.6.1 Intercepting Proxies

The most used web application proxies are:

- The intercepting proxy feature of Burp Suite
- ZAP

Proxies are fundamental while analyzing web applications and will become your best friend for web-app testing.

http://portswigger.net/burp/
https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project

3.6.1 Intercepting Proxies

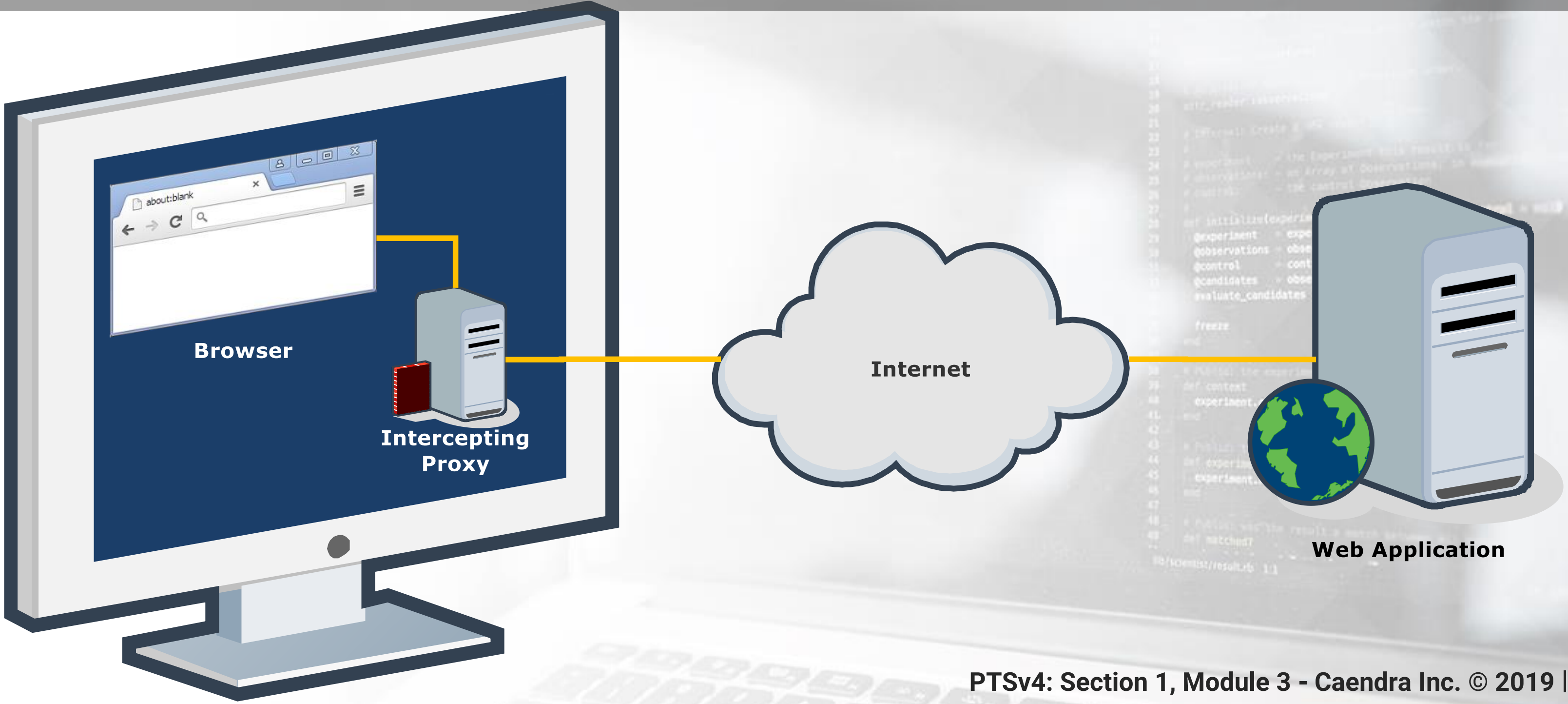
Do not confuse intercepting proxies with common web proxy servers like Squid. Proxy servers have different purposes: bandwidth optimization, content filtering and more.

The next two images will make that clearer.

http://www.squid-cache.org/

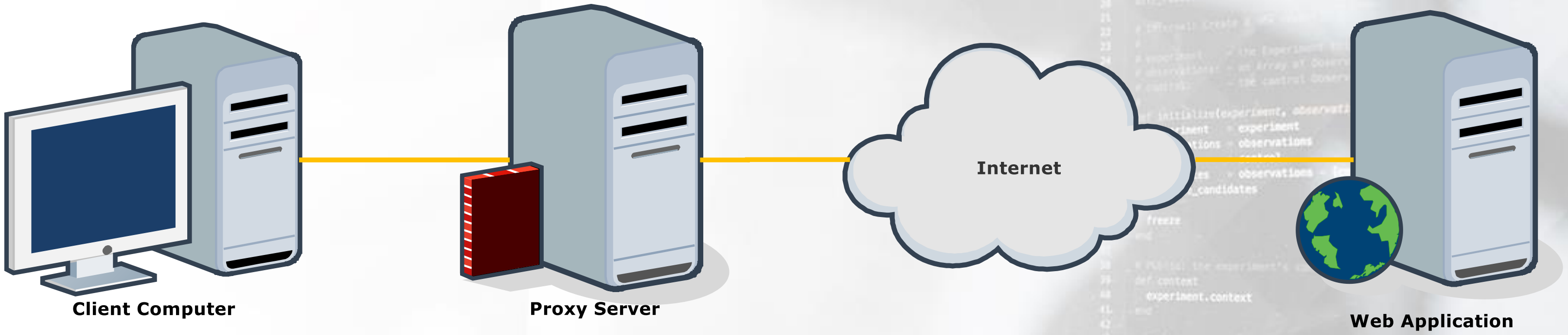
3.6.1.1 Intercepting Proxy Example

Here the proxy is an application which intercepts the penetration tester's browser traffic.



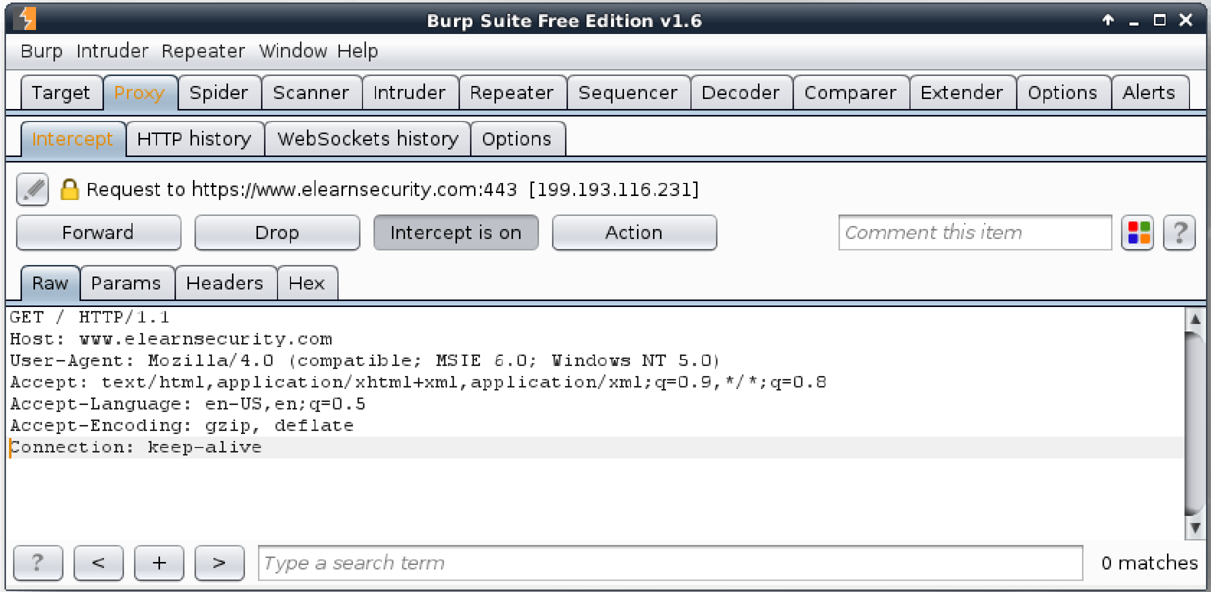
3.6.1.2 Proxy Server Example

Here the proxy server filters all the traffic coming from the internal network.



3.6.2 Burp Proxy

Burp suite offers one of the best proxies available. You can download the Free Edition here.



Burp Suite Free Edition v1.6
Burp Intruder Repeater Window Help
Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Options Alerts
Intercept HTTP history WebSockets history Options
Forward Drop Intercept is on Action
Comment this item
Raw Params Headers Hex
? < + >
Type a search term
0 matches

Request to https://www.elearnsecurity.com:443 [199.193.116.231]

```
GET / HTTP/1.1
Host: www.elearnsecurity.com
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0)
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

https://portswigger.net/burp/communitydownload

3.6.2 Burp Proxy

Burp suite will let you:

- Intercept requests and responses between your browser and the web server.
- Build requests manually.
- Crawl a website by automatically visiting every page in a website.
- Fuzz web applications by sending them patterns of valid and invalid inputs to test their behavior.

3.6.2 Burp Proxy

By using Burp, you can **intercept and modify** requests coming from your browsers **before** they are sent to the remote server.

You can modify the **header** and the **body** of a message **by hand or automatically**.

3.6.2.1 Burp Proxy Configuration

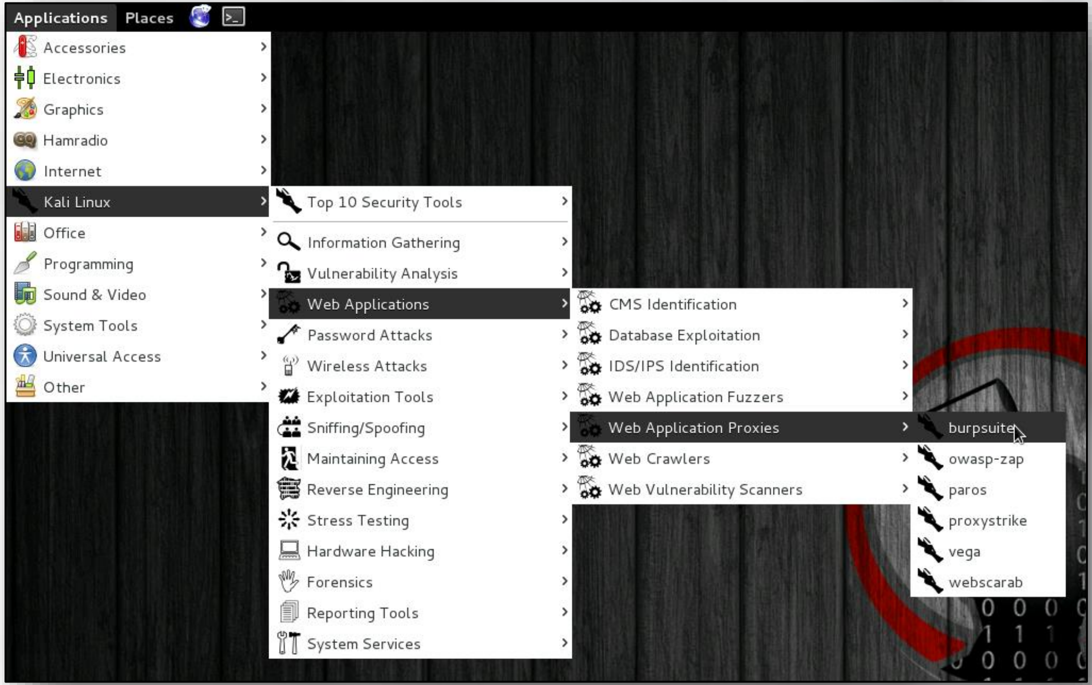
In the following slides, you will see how to launch, configure and use Burp Suite with your browser.

Try to understand all the settings by trying them on your computer!

3.6.2.1 Burp Proxy Configuration

Launch Burp Suite:

In Kali you will find it under *Kali Linux > Web Applications > Web Application Proxies > burpsuite*.



3.6.2.1 Burp Proxy Configuration

If you want to run it on another operating system, you can download it from the Portswigger website.

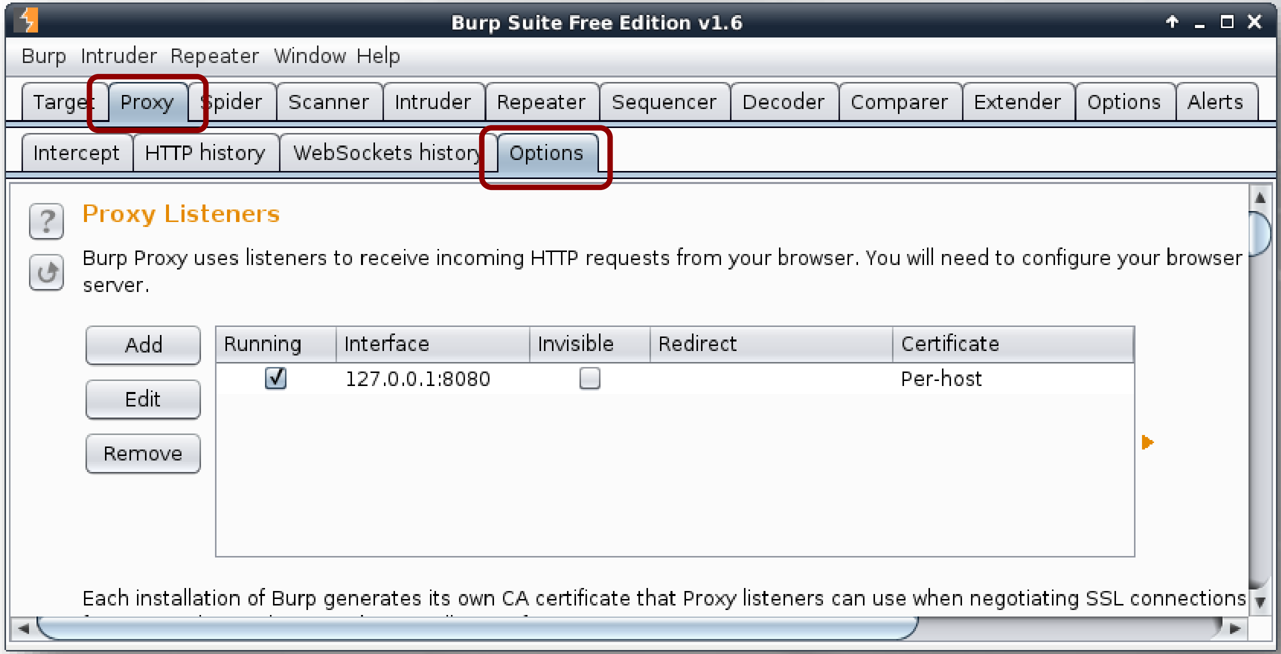
To run Burp, double click on the jar file you downloaded or run the below from the console:

```
java -jar burpsuite_free_v1.6.jar
```

https://portswigger.net/burp/communitydownload

3.6.2.1 Burp Proxy Configuration

Now, go to the *Proxy* tab and then to the *Options* sub-tab.


Burp Suite Free Edition v1.6
Burp Intruder Repeater Window Help
Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Options Alerts
Intercept HTTP history WebSockets history Options
Add
Edit
Remove

Proxy Listeners

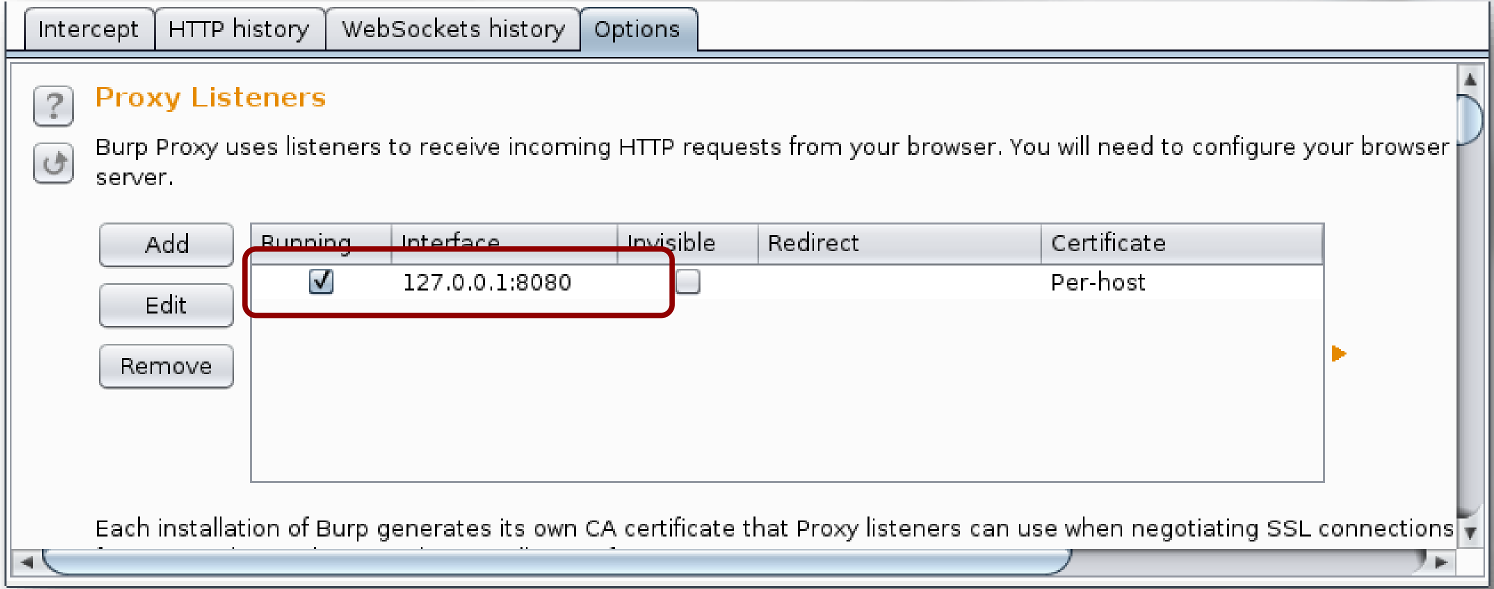
Burp Proxy uses listeners to receive incoming HTTP requests from your browser. You will need to configure your browser server.

Running	Interface	Invisible	Redirect	Certificate
✓	127.0.0.1:8080	☐		Per-host

Each installation of Burp generates its own CA certificate that Proxy listeners can use when negotiating SSL connections

3.6.2.1 Burp Proxy Configuration

Here you can start and stop the proxy and configure the *host:port* pair on which burp will listen.



Proxy Listeners

Burp Proxy uses listeners to receive incoming HTTP requests from your browser. You will need to configure your browser server.

Running	Interface	Invisible	Redirect	Certificate
✓	127.0.0.1:8080	☐		Per-host

Each installation of Burp generates its own CA certificate that Proxy listeners can use when negotiating SSL connections

3.6.2.1 Burp Proxy Configuration

Scrolling down you can find other configuration items to fine tune, which messages to intercept, how to automatically change message content and more.

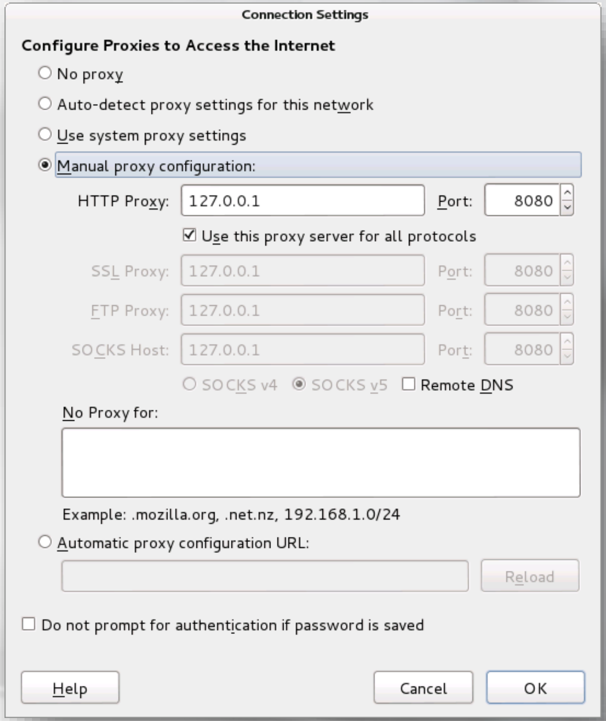
For now, just leave the default options as they are, you will see how to use those features later on.

3.6.2.1 Burp Proxy Configuration

Once Burp Proxy is configured, you have to configure your browser to use it as the proxy for every protocol.

3.6.2.1 Burp Proxy Configuration

In Firefox, you have to open the *Preferences* window, go to the advanced tab, click on the *Network* sub-tab and finally open the *Connection settings* window.



Connection Settings

Configure Proxies to Access the Internet

- ○ No proxy
- ○ Auto-detect proxy settings for this network
- ○ Use system proxy settings
- ● Manual proxy configuration:

HTTP Proxy: 127.0.0.1 Port: 8080

☑ Use this proxy server for all protocols

SSL Proxy: 127.0.0.1 Port: 8080

FTP Proxy: 127.0.0.1 Port: 8080

SOCKS Host: 127.0.0.1 Port: 8080

○ SOCKS v4 ● SOCKS v5 ☐ Remote DNS

No Proxy for:

Example: .mozilla.org, .net.nz, 192.168.1.0/24

- ○ Automatic proxy configuration URL:

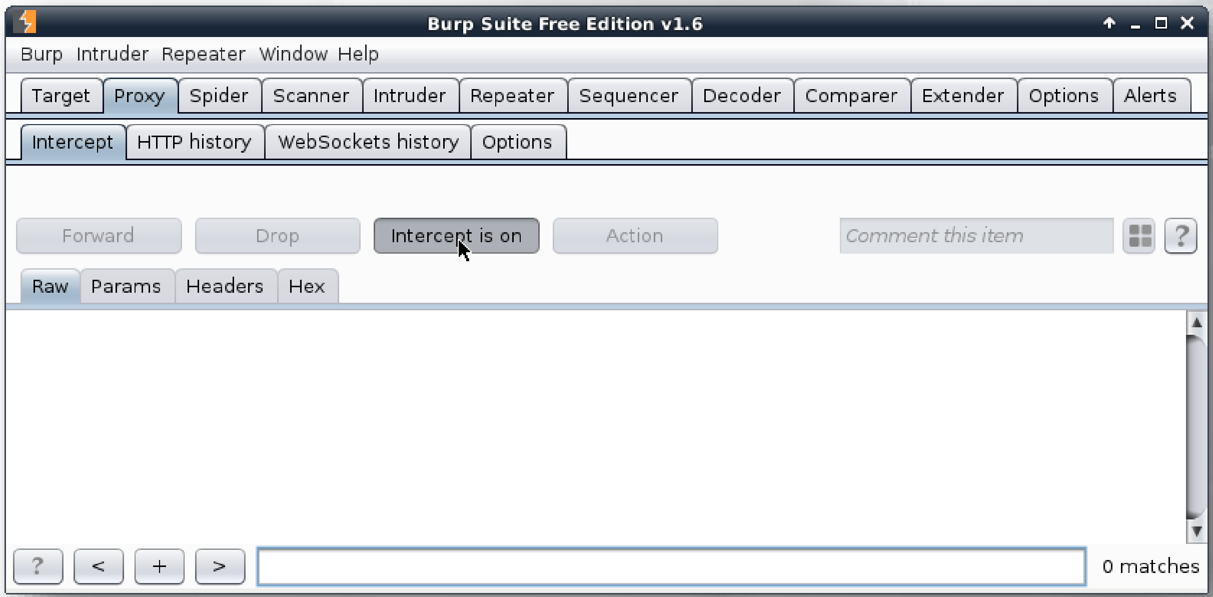
Reload

☐ Do not prompt for authentication if password is saved

Help Cancel OK

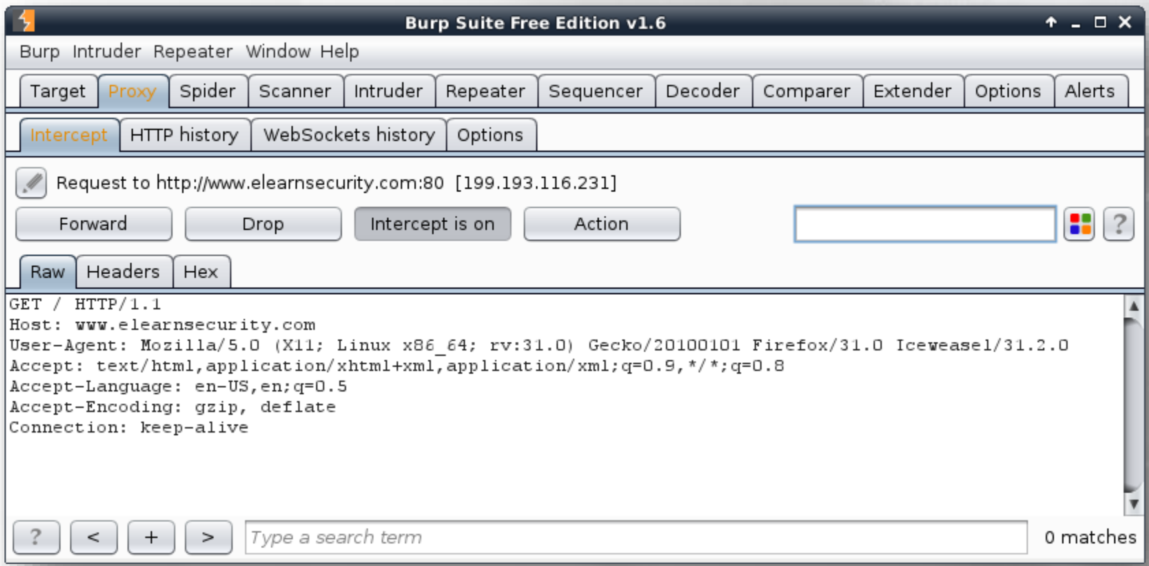
3.6.2.1 Burp Proxy Configuration

To intercept traffic, switch to Burp and go to *Proxy > Intercept* and click on the *Intercept is off* button to enable interception.


Burp Suite Free Edition v1.6
Burp Intruder Repeater Window Help
Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Options Alerts
Intercept HTTP history WebSockets history Options
Forward
Drop
Intercept is on
Action
Comment this item
Raw Params Headers Hex
0 matches

3.6.2.1 Burp Proxy Configuration

Now open a website with your browser; Burp will pop up intercepting the request. Optionally, you can modify and then forward it by clicking on *Forward.*

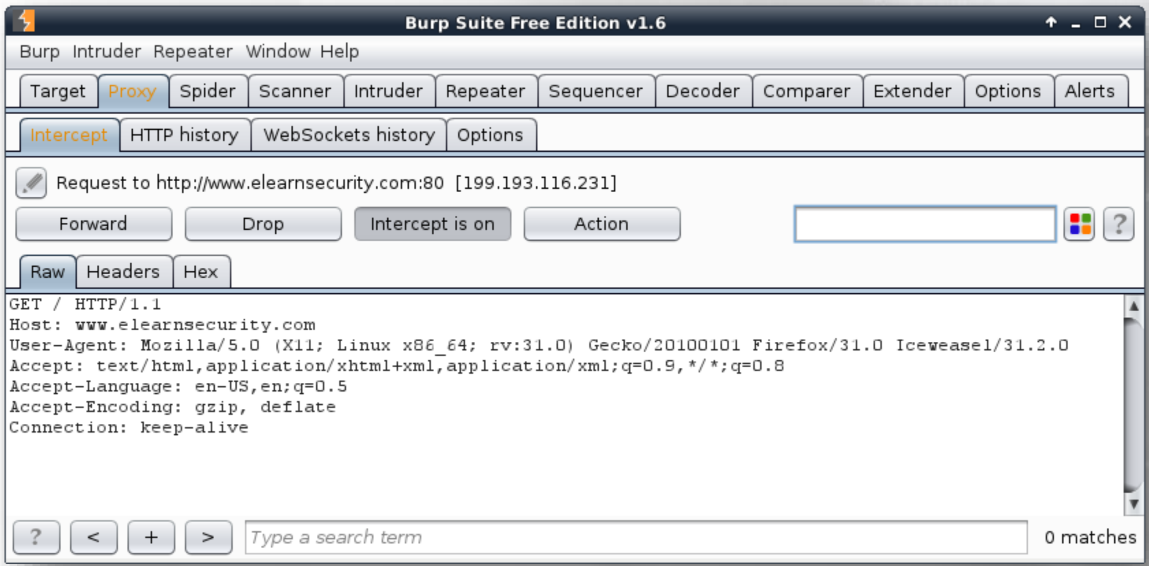

Burp Suite Free Edition v1.6
Burp Intruder Repeater Window Help
Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Options Alerts
Intercept HTTP history WebSockets history Options
Forward Drop Intercept is on Action
Raw Headers Hex
Type a search term
0 matches

Request to http://www.elearnsecurity.com:80 [199.193.116.231]

```
GET / HTTP/1.1
Host: www.elearnsecurity.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:31.0) Gecko/20100101 Firefox/31.0 Iceweasel/31.2.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

3.6.2.1 Burp Proxy Configuration

When *Intercept* is on, every browser request stops at Burp Proxy. You can modify the entire request or just its headers.

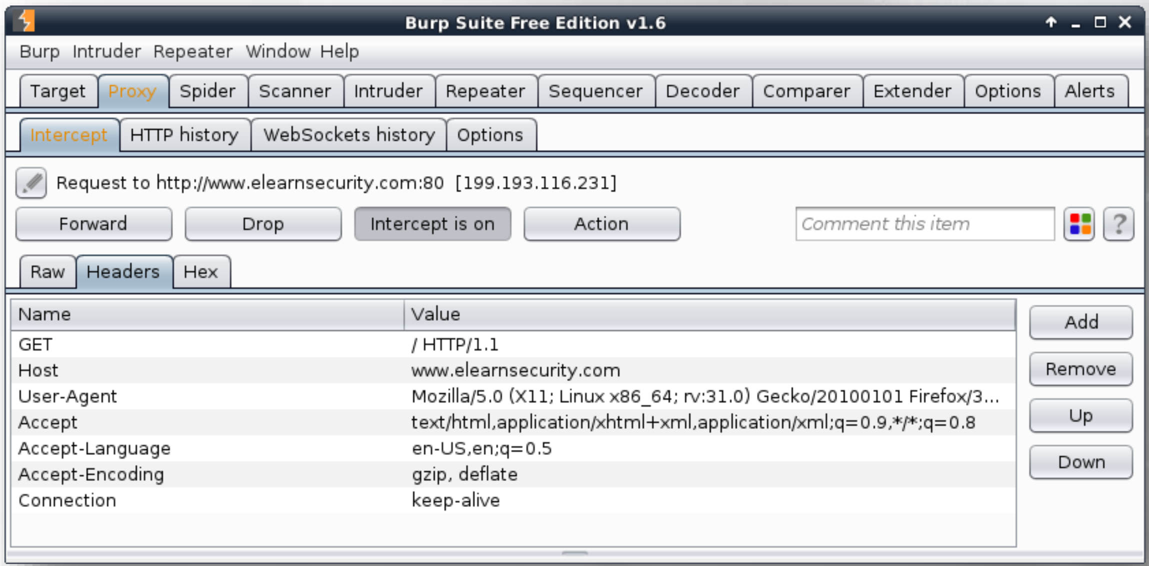

Burp Suite Free Edition v1.6
Burp Intruder Repeater Window Help
Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Options Alerts
Intercept HTTP history WebSockets history Options
Forward
Drop
Intercept is on
Action
Raw Headers Hex
Type a search term
0 matches

Request to http://www.elearnsecurity.com:80 [199.193.116.231]

```
GET / HTTP/1.1
Host: www.elearnsecurity.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:31.0) Gecko/20100101 Firefox/31.0 Iceweasel/31.2.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

3.6.2.1 Burp Proxy Configuration

You can modify the headers both in the *Raw* tab or in the *Headers* tab. Remember to forward the request after editing it!


Burp Suite Free Edition v1.6
Burp Intruder Repeater Window Help
Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Options Alerts
Intercept HTTP history WebSockets history Options
Forward
Drop
Intercept is on
Action
Comment this item
Raw Headers Hex
Add
Remove
Up
Down

Request to http://www.elearnsecurity.com:80 [199.193.116.231]

Name	Value
GET	/ HTTP/1.1
Host	www.elearnsecurity.com
User-Agent	Mozilla/5.0 (X11; Linux x86_64; rv:31.0) Gecko/20100101 Firefox/3...
Accept	text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language	en-US,en;q=0.5
Accept-Encoding	gzip, deflate
Connection	keep-alive

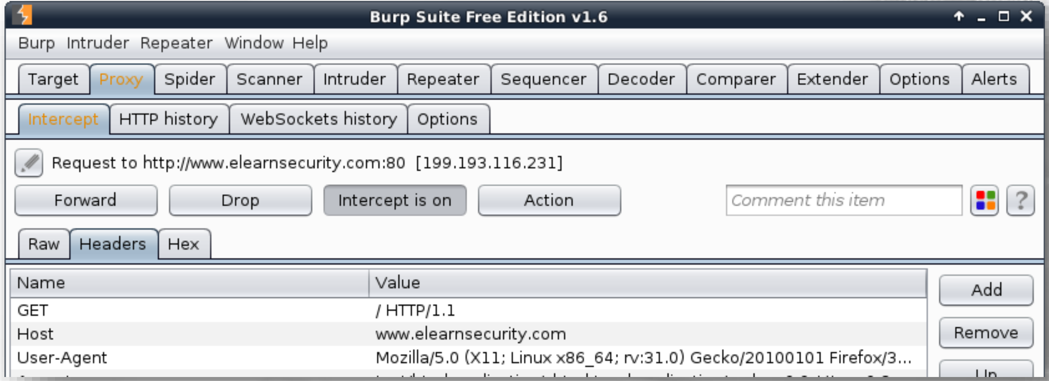
3.6.2.1 Burp Proxy Configuration

Q

What is the difference between the **Raw** *and the* **Headers** *tab?*

A

They present the very same information with a different format. The **Headers** *tab simply presents the headers in a tabular manner.*



Burp Suite Free Edition v1.6
Burp Intruder Repeater Window Help
Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Options Alerts
Intercept HTTP history WebSockets history Options
Forward Drop Intercept is on Action
Comment this item
Raw Headers Hex
Add Remove

Request to http://www.elearnsecurity.com:80 [199.193.116.231]

Name	Value
GET	/ HTTP/1.1
Host	www.elearnsecurity.com
User-Agent	Mozilla/5.0 (X11; Linux x86_64; rv:31.0) Gecko/20100101 Firefox/3...

3.6.2.1 Burp Proxy Configuration

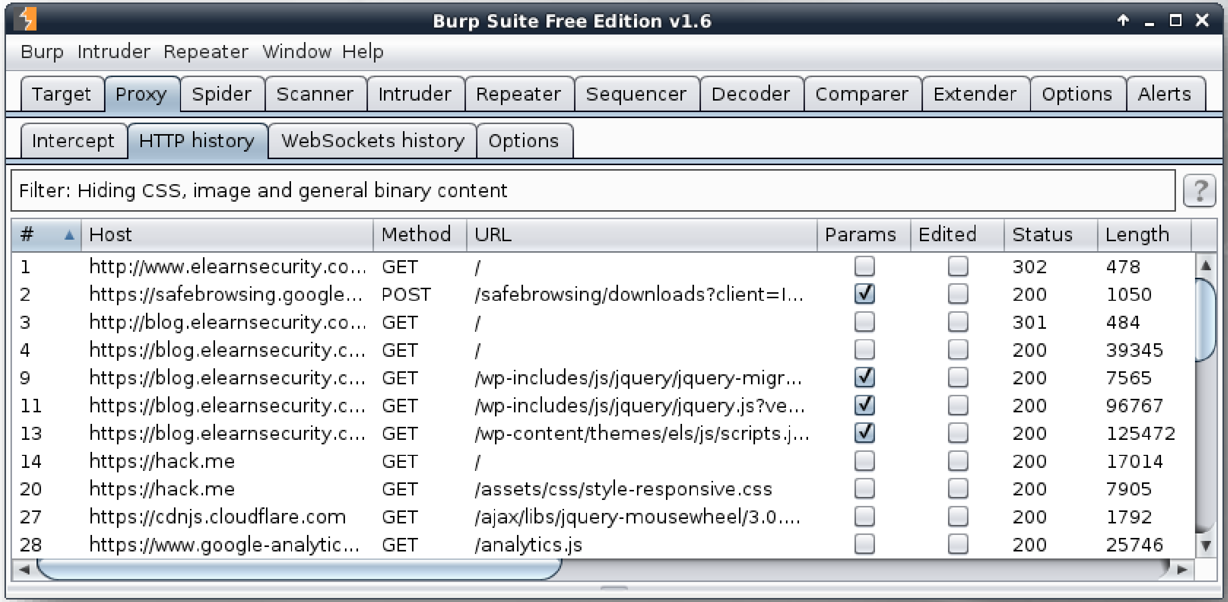
You do not need to manually intercept and forward every request though. Even if you leave the master interception off, Burp will still collect information on the HTTP traffic coming to and from your browser.

You can check what Burp is collecting in two ways:

- On the *Proxy > History* tab
- In the *Target > Site Map* tab

3.6.2.1 Burp Proxy Configuration

Burp Suite *Proxy* tab contains an *HTTP history* sub tab.

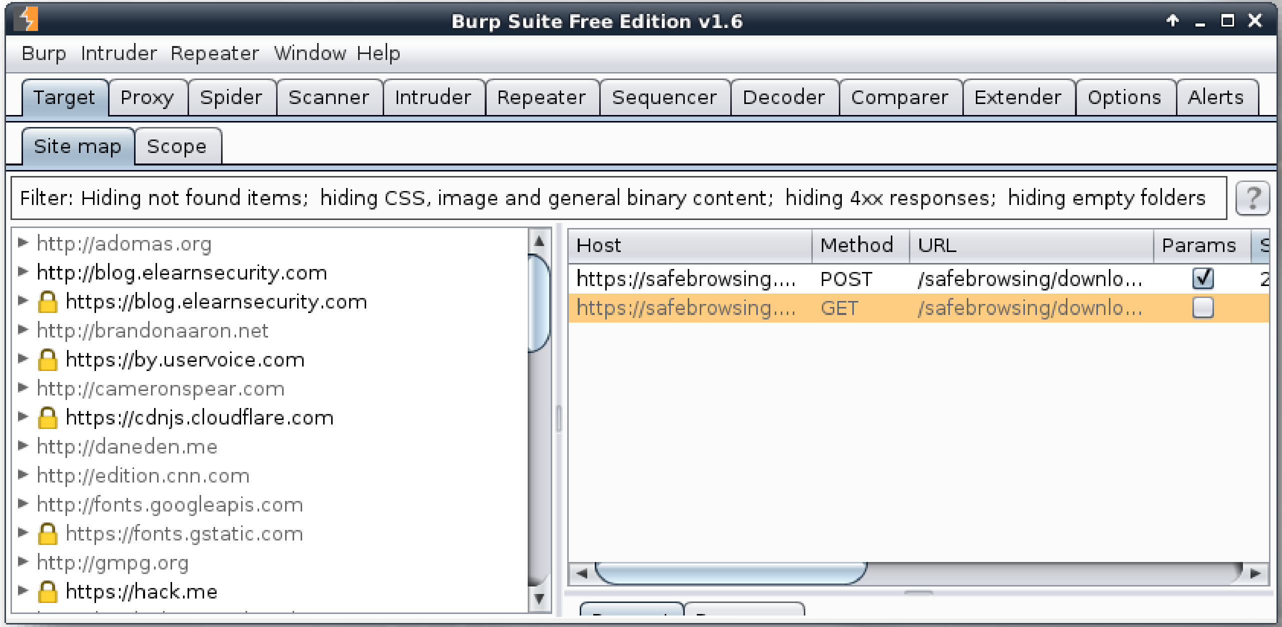

Burp Suite Free Edition v1.6
Burp Intruder Repeater Window Help
Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Options Alerts
Intercept HTTP history WebSockets history Options

Filter: Hiding CSS, image and general binary content

#	Host	Method	URL	Params	Edited	Status	Length
1	http://www.elearnsecurity.co...	GET	/	☐	☐	302	478
2	https://safebrowsing.google...	POST	/safebrowsing/downloads?client=I...	✓	☐	200	1050
3	http://blog.elearnsecurity.co...	GET	/	☐	☐	301	484
4	https://blog.elearnsecurity.c...	GET	/	☐	☐	200	39345
9	https://blog.elearnsecurity.c...	GET	/wp-includes/js/jquery/jquery-migr...	✓	☐	200	7565
11	https://blog.elearnsecurity.c...	GET	/wp-includes/js/jquery/jquery.js?ve...	✓	☐	200	96767
13	https://blog.elearnsecurity.c...	GET	/wp-content/themes/els/js/scripts.j...	✓	☐	200	125472
14	https://hack.me	GET	/	☐	☐	200	17014
20	https://hack.me	GET	/assets/css/style-responsive.css	☐	☐	200	7905
27	https://cdnjs.cloudflare.com	GET	/ajax/libs/jquery-mousewheel/3.0....	☐	☐	200	1792
28	https://www.google-analytic...	GET	/analytics.js	☐	☐	200	25746

3.6.2.1 Burp Proxy Configuration

You can also check what Burp is collecting on *Target > Site Map.*


Burp Suite Free Edition v1.6
Burp Intruder Repeater Window Help
Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Options Alerts
Site map Scope

Filter: Hiding not found items; hiding CSS, image and general binary content; hiding 4xx responses; hiding empty folders

- http://adomas.org
- http://blog.elearnsecurity.com
- https://blog.elearnsecurity.com
- http://brandonaaron.net
- https://by.uservoice.com
- http://cameronspear.com
- https://cdnjs.cloudflare.com
- http://daneden.me
- http://edition.cnn.com
- http://fonts.googleapis.com
- https://fonts.gstatic.com
- http://gmpg.org
- https://hack.me

Host	Method	URL	Params	S
https://safebrowsing....	POST	/safebrowsing/downlo...	✓	2
https://safebrowsing....	GET	/safebrowsing/downlo...	☐	

3.6.3 Burp Repeater

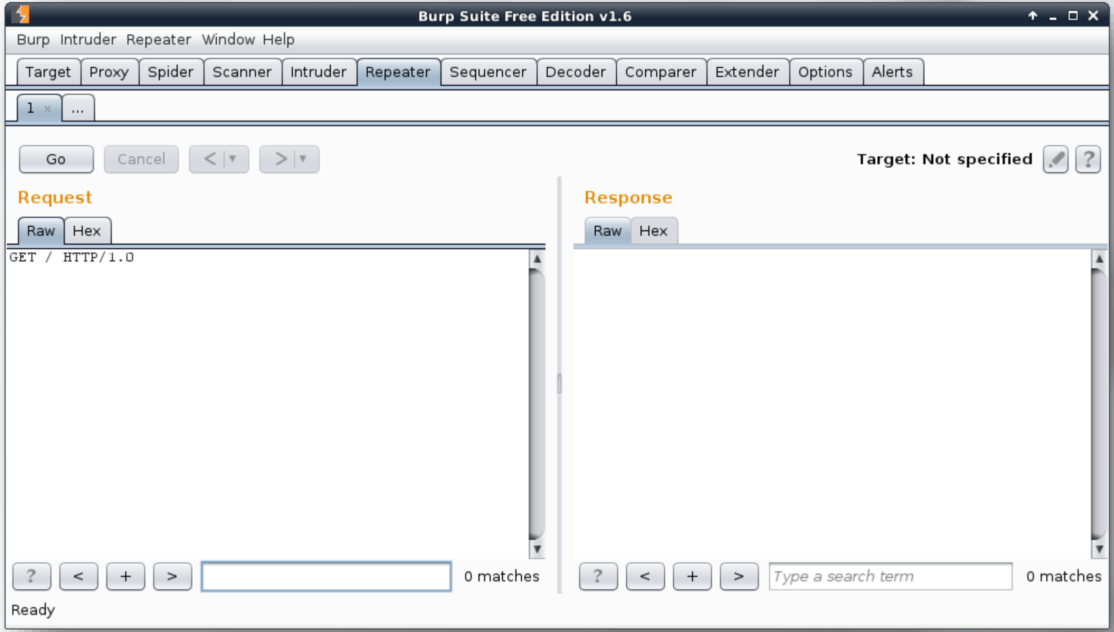
Another feature is **Burp Repeater**, which lets you manually build raw HTTP requests.

You can do the same thing by using other tools such as *netcat* or *telnet*, but *Burp* provides you:

- Syntax highlighting
- Raw and rendered responses
- Integration with other Burp tools

3.6.3 Burp Repeater

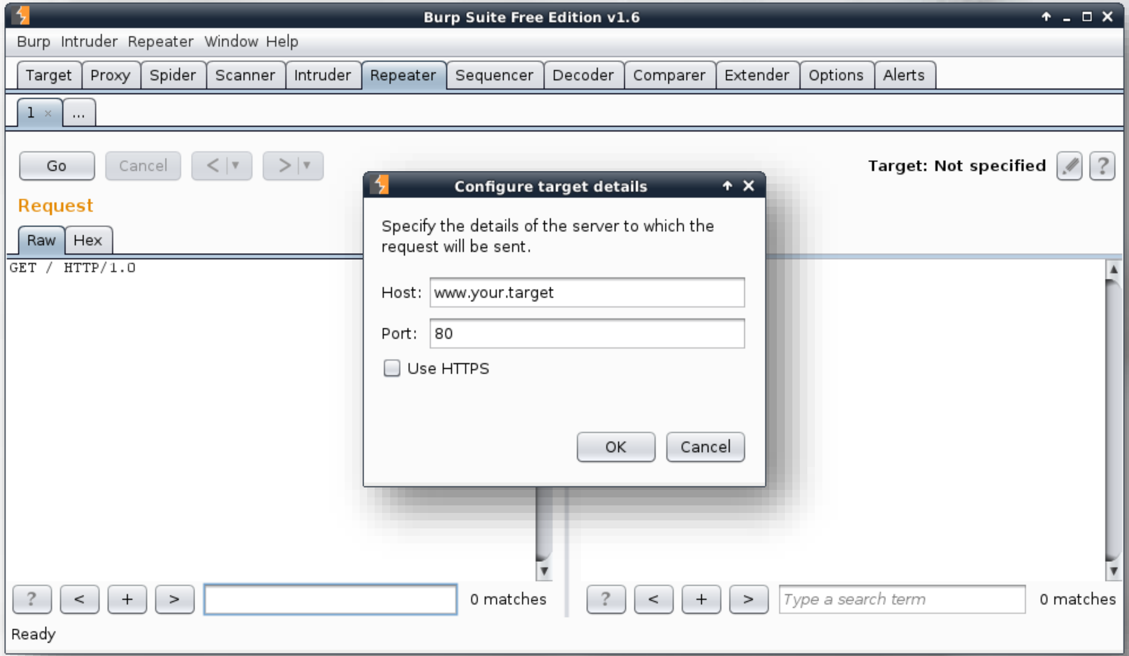
To create a request, first set your target by clicking on the pencil icon in the upper right corner of the tab.


Burp Suite Free Edition v1.6
Burp Intruder Repeater Window Help
Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Options Alerts
1 × ...
Go Cancel < >
Target: Not specified
Request
Raw Hex
Response
Raw Hex
? < + > 0 matches
? < + > Type a search term 0 matches
Ready

```
GET / HTTP/1.0
```

3.6.3 Burp Repeater

You can then set your target host and port.


Burp Suite Free Edition v1.6
Burp Intruder Repeater Window Help
Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Options Alerts
1 ...
Go Cancel
Target: Not specified
Request
Raw Hex
Configure target details
OK Cancel
0 matches
Type a search term
0 matches
Ready

```
GET / HTTP/1.0
```

Specify the details of the server to which the request will be sent.

Host: www.your.target

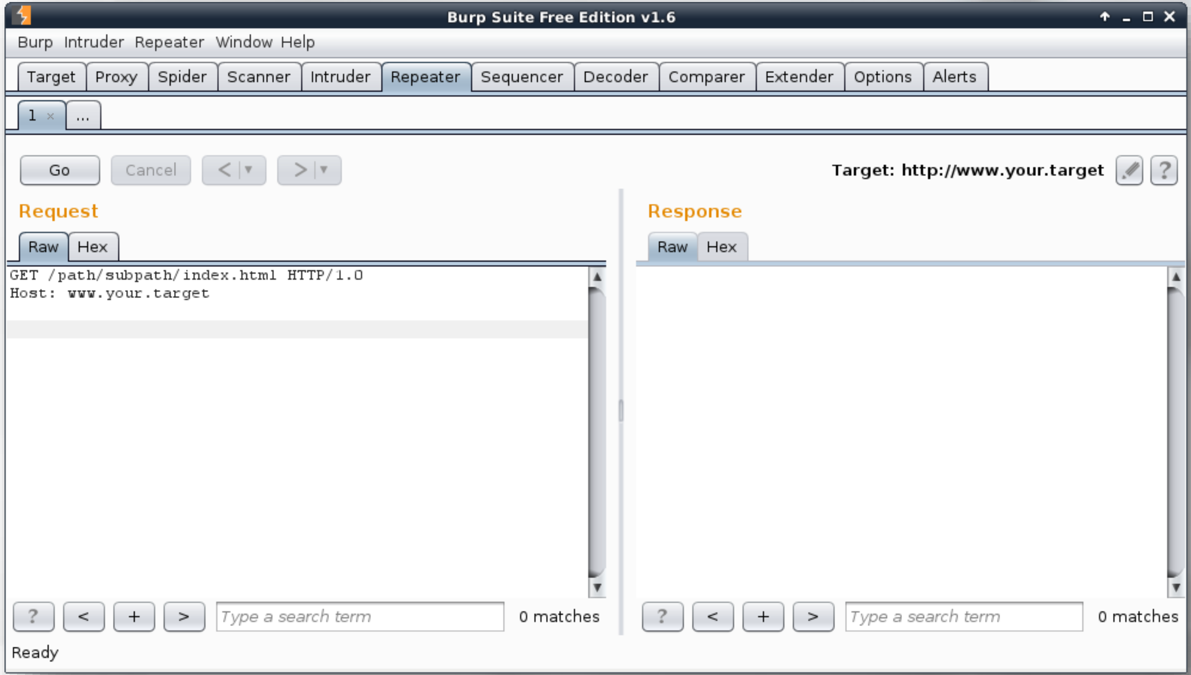
Port: 80

[ ] Use HTTPS

3.6.3 Burp Repeater

You can define your request by using this text area.

Every request must have at least an **HTTP VERB** (GET, POST, HEAD, ...)

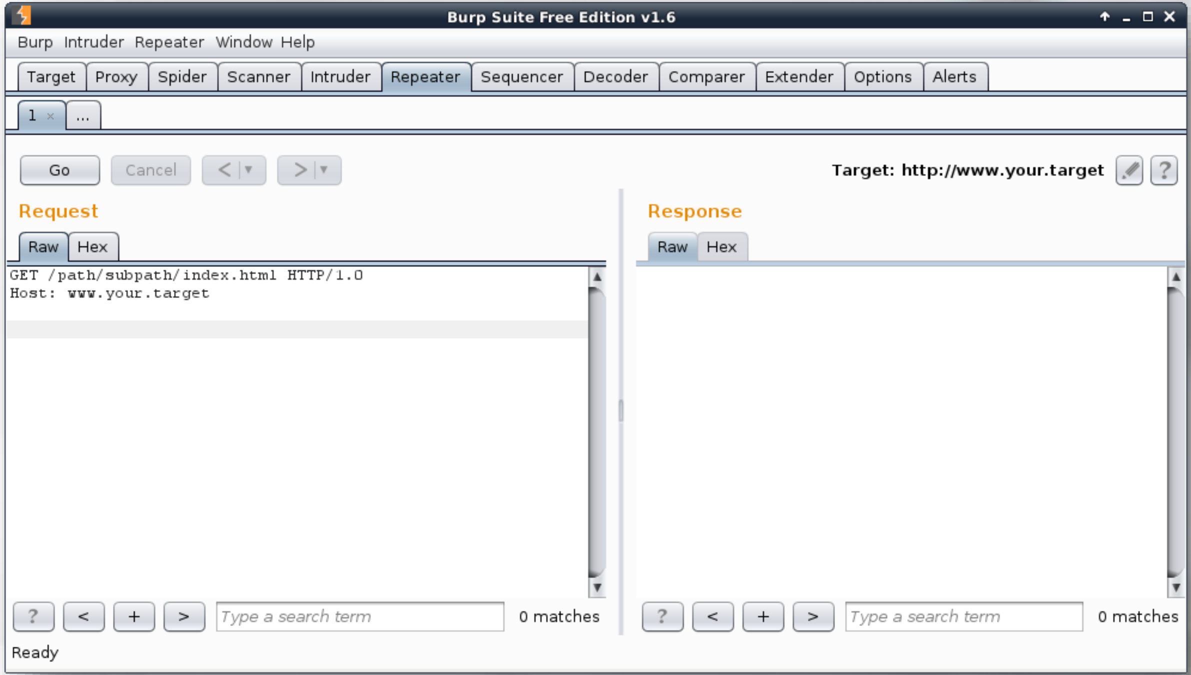


Burp Suite Free Edition v1.6
Burp Intruder Repeater Window Help
Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Options Alerts
Go
Cancel
Target: http://www.your.target
Request
Raw Hex
Response
Raw Hex
Type a search term
0 matches
Ready

```
GET /path/subpath/index.html HTTP/1.0
Host: www.your.target
```

3.6.3 Burp Repeater

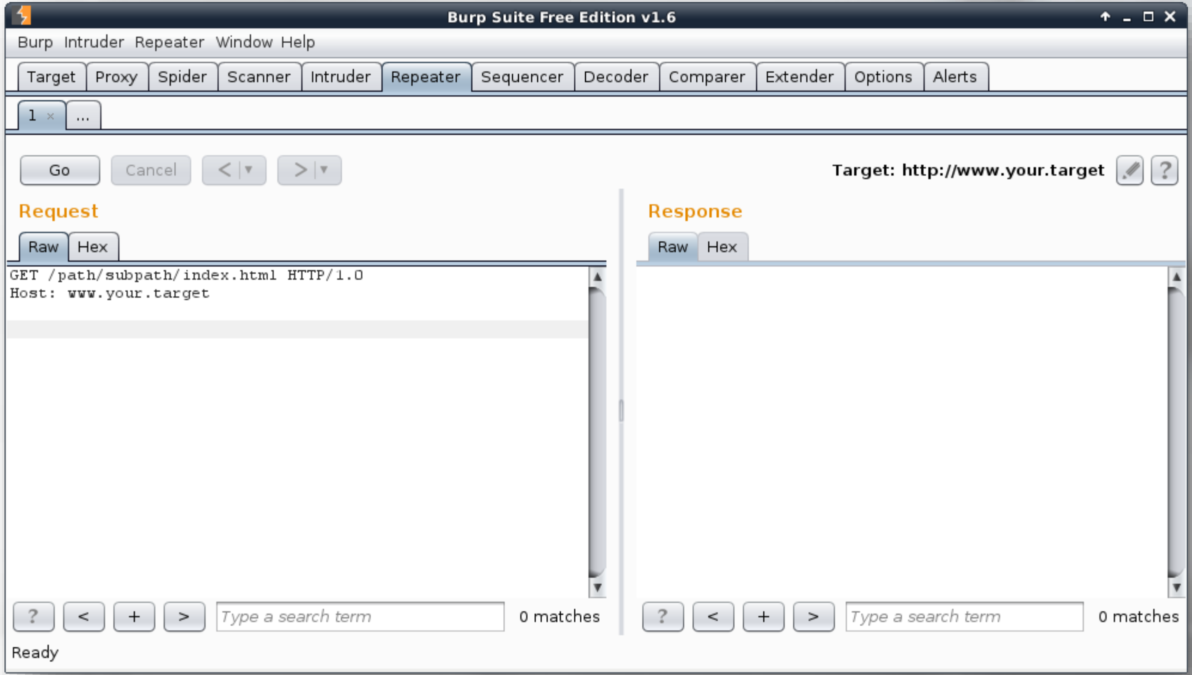
Here is the **Host** header.


Burp Suite Free Edition v1.6
Burp Intruder Repeater Window Help
Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Options Alerts
1
...
Go
Cancel
Target: http://www.your.target
Request
Response
Raw
Hex
Type a search term
0 matches
Ready

```
GET /path/subpath/index.html HTTP/1.0
Host: www.your.target
```

3.6.3 Burp Repeater

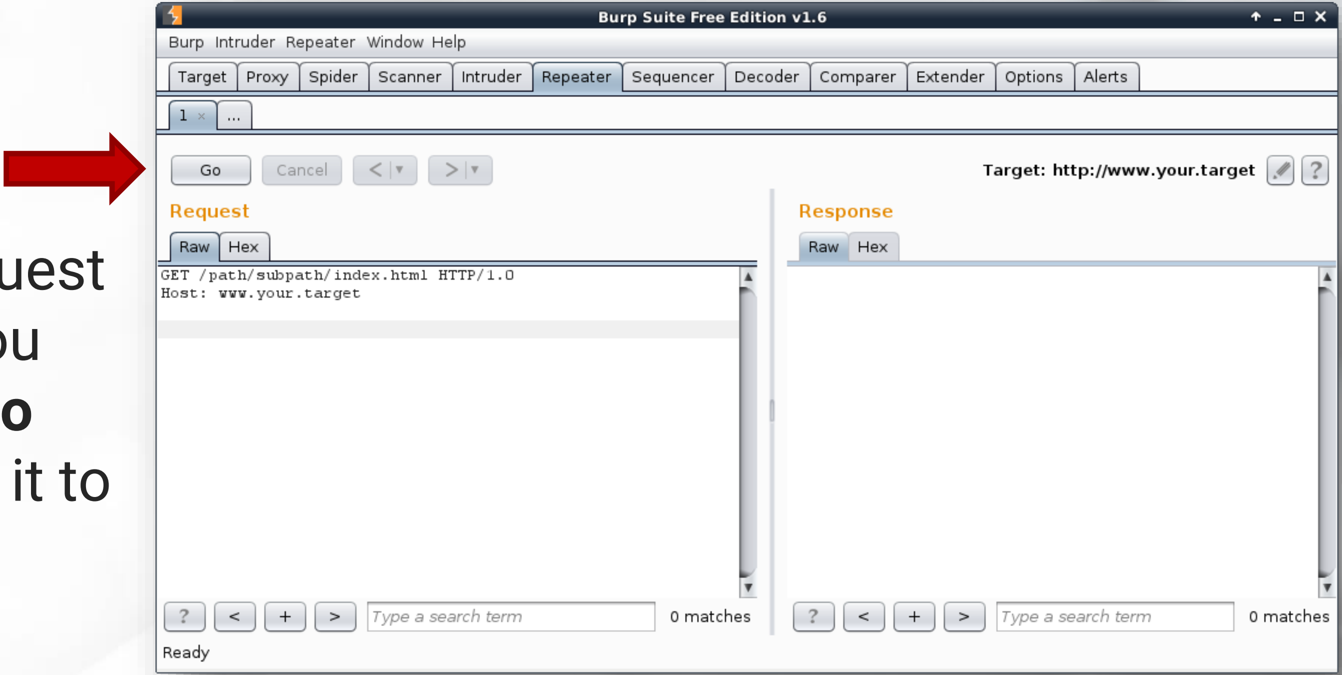
And here we see **two empty lines** after the headers.


Burp Suite Free Edition v1.6
Burp Intruder Repeater Window Help
Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Options Alerts
1 ...
Go Cancel
Target: http://www.your.target
Request
Raw Hex
Response
Raw Hex
Type a search term
0 matches
Type a search term
0 matches
Ready

```
GET /path/subpath/index.html HTTP/1.0
Host: www.your.target


```

3.6.3 Burp Repeater

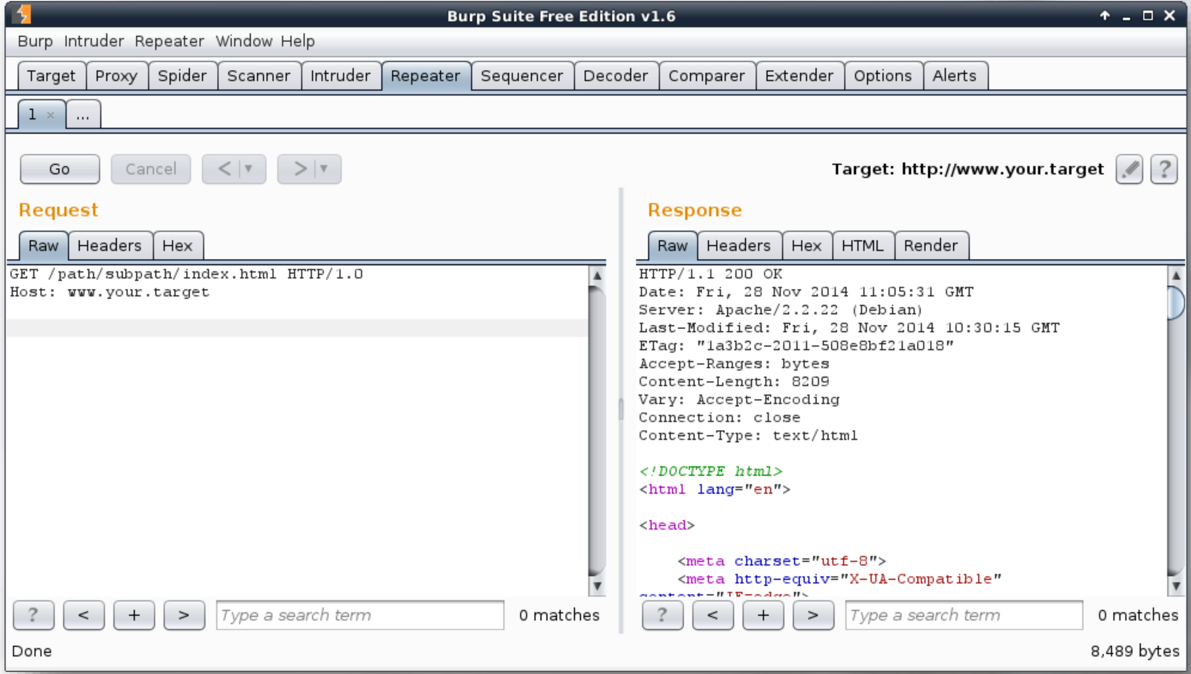
When your request is complete, you can click the **Go** button to send it to the server.


Burp Suite Free Edition v1.6
Burp Intruder Repeater Window Help
Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Options Alerts
1 ...
Go Cancel
Target: http://www.your.target
Request
Raw Hex
Response
Raw Hex
Type a search term
0 matches
Type a search term
0 matches
Ready

```
GET /path/subpath/index.html HTTP/1.0
Host: www.your.target
```

3.6.3 Burp Repeater

Burp will then display the response in the **Response** panel.

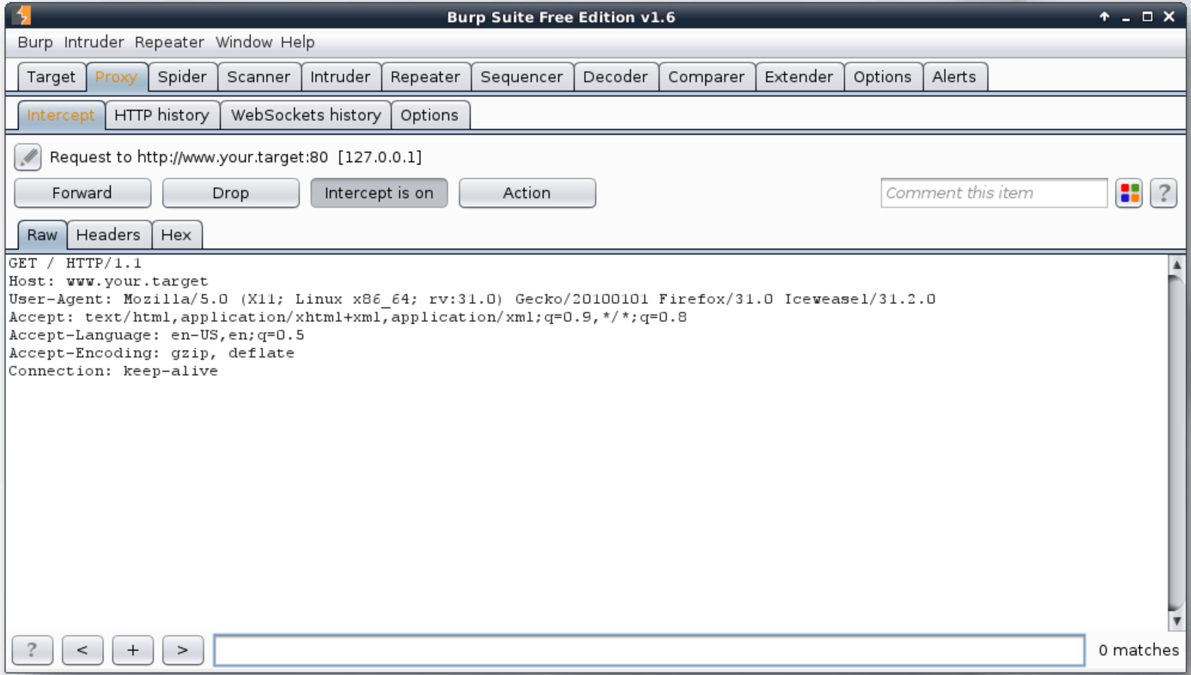

Burp Suite Free Edition v1.6
Burp Intruder Repeater Window Help
Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Options Alerts
1 ...
Go Cancel
Target: http://www.your.target
Request
Raw Headers Hex
Response
Raw Headers Hex HTML Render
Type a search term
0 matches
Done
8,489 bytes

```
GET /path/subpath/index.html HTTP/1.0
Host: www.your.target
```

```
HTTP/1.1 200 OK
Date: Fri, 28 Nov 2014 11:05:31 GMT
Server: Apache/2.2.22 (Debian)
Last-Modified: Fri, 28 Nov 2014 10:30:15 GMT
ETag: "1a3b2c-2011-508e8bf21a018"
Accept-Ranges: bytes
Content-Length: 8209
Vary: Accept-Encoding
Connection: close
Content-Type: text/html

<!DOCTYPE html>
<html lang="en">

<head>

    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible"
```

3.6.3 Burp Repeater

An easier method to build requests is to intercept a browser request with the proxy and send it to the Repeater function.

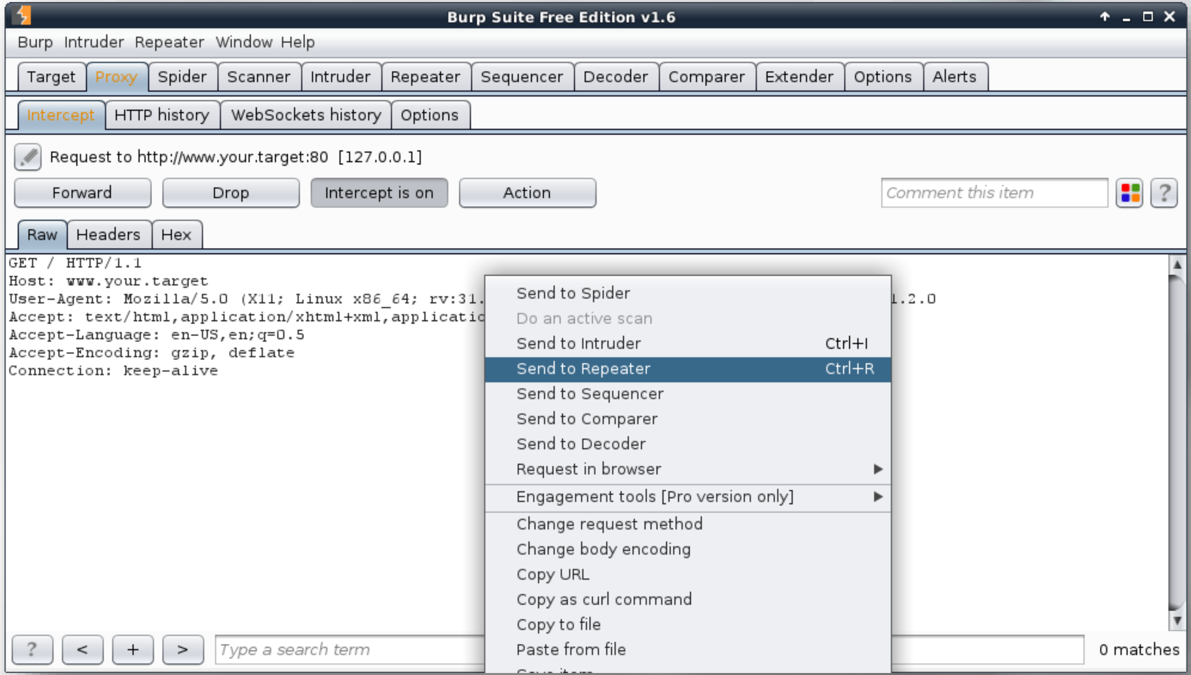

Burp Suite Free Edition v1.6
Burp Intruder Repeater Window Help
Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Options Alerts
Intercept HTTP history WebSockets history Options
Forward Drop Intercept is on Action
Comment this item
Raw Headers Hex
0 matches

Request to http://www.your.target:80 [127.0.0.1]

```
GET / HTTP/1.1
Host: www.your.target
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:31.0) Gecko/20100101 Firefox/31.0 Iceweasel/31.2.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

3.6.3 Burp Repeater

You can do that with the **Ctrl+R** shortcut or by right-clicking in the request body and selecting **Send to Repeater**.


Burp Suite Free Edition v1.6
Burp Intruder Repeater Window Help
Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Options Alerts
Intercept HTTP history WebSockets history Options
Request to http://www.your.target:80 [127.0.0.1]
Forward Drop Intercept is on Action
Comment this item
Raw Headers Hex
Send to Spider
Do an active scan
Send to Intruder Ctrl+I
Send to Repeater Ctrl+R
Send to Sequencer
Send to Comparer
Send to Decoder
Request in browser
Engagement tools [Pro version only]
Change request method
Change body encoding
Copy URL
Copy as curl command
Copy to file
Paste from file
Type a search term
0 matches

```
GET / HTTP/1.1
Host: www.your.target
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:31.                                              1.2.0
Accept: text/html,application/xhtml+xml,applicatic
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

3.6.4 Video – Burp Suite

Burp Suite

In this video, you will learn how to define the target scope in Burp and how to configure the Proxy to intercept only the traffic in scope.

Moreover, you will see how to configure and use other features of Burp according to the penetration test engagement goals.

**Videos are only available in Full or Elite Editions of the course. To upgrade, click HERE. To access, go to the course in your members area and click the resources drop-down in the appropriate module line.*

3.6.5 Hera Lab – Burp Suite Basics

Try to test your knowledge with a real task to be done with help of Burp Suite.

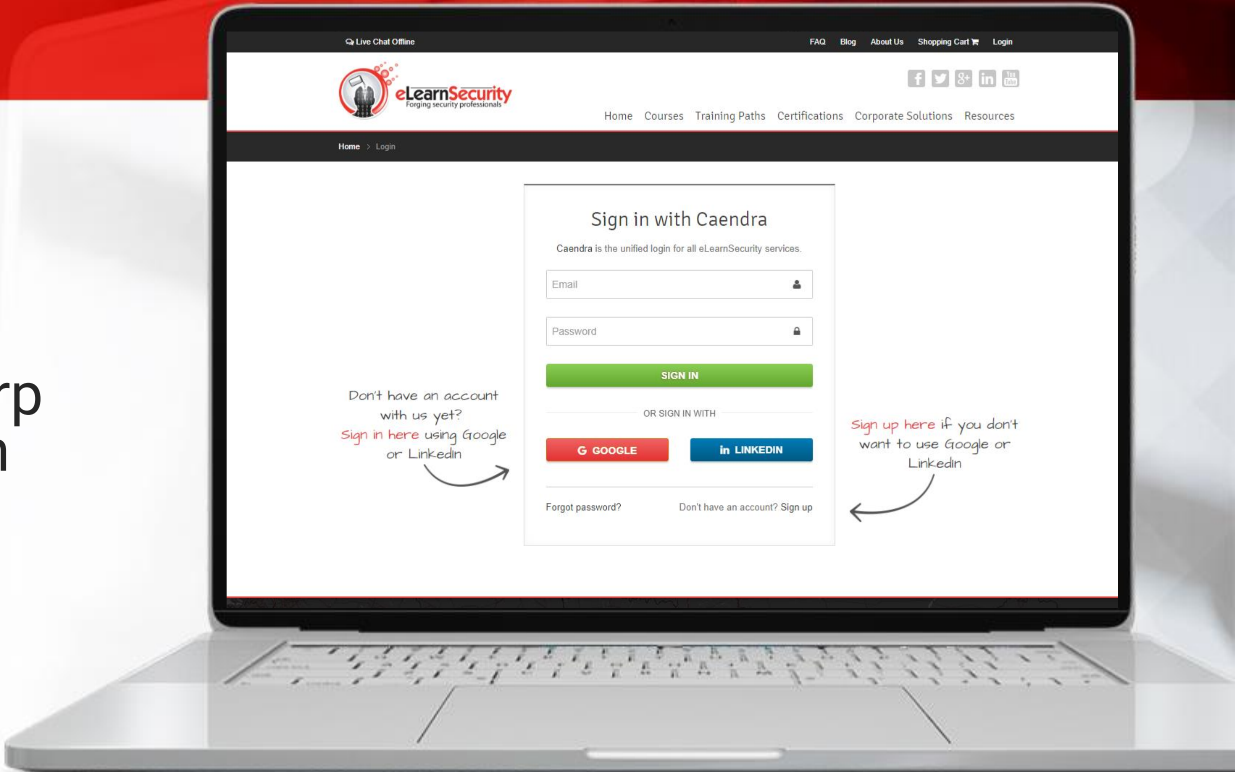
The task you are about to face is very similar to what pentesters do everyday when inspecting a web application using Burp Suite.

3.6.5 Hera Lab – Burp Suite Basics

Burp Suite Basics

In this lab you will:

- Inspect a web application using Burp Suite and find hidden resources.
- Use Burp Intruder to attack a web application


Live Chat Offline
FAQ
Blog
About Us
Shopping Cart
Login
eLearnSecurity
Forging security professionals
Home
Courses
Training Paths
Certifications
Corporate Solutions
Resources
Home > Login
Sign in with Caendra
Caendra is the unified login for all eLearnSecurity services.
Email
Password
SIGN IN
OR SIGN IN WITH
GOOGLE
LINKEDIN
Forgot password?
Don't have an account? Sign up
Don't have an account with us yet? Sign in here using Google or Linkedin
Sign up here if you don't want to use Google or Linkedin

**Labs are only available in Full or Elite Editions of the course. To upgrade, click HERE. To access, go to the course in your members area and click the labs drop-down in the appropriate module line or to the virtual labs tabs on the left navigation.*

3.6.6 Hera Lab – Burp Suite

The best way to learn how to use a tool is through practice. Analyze a web application with Burp and hack it!

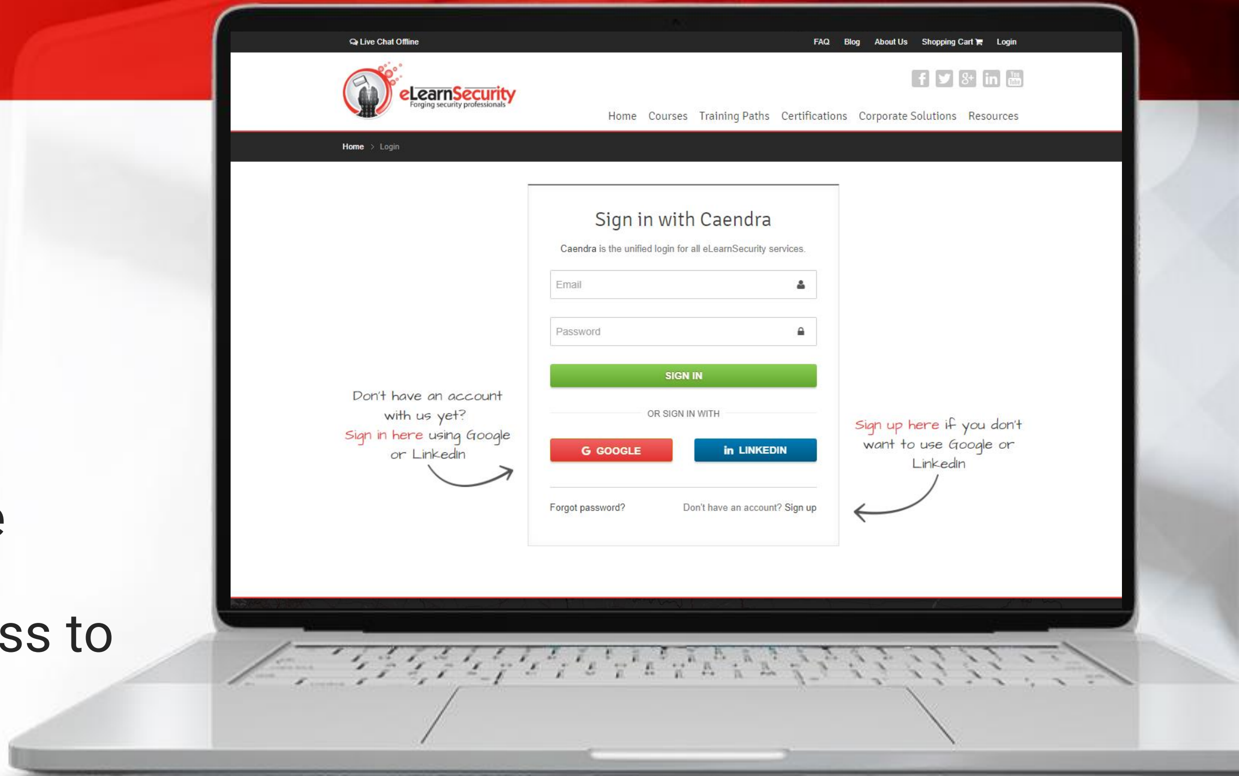
Try to reach the lab goal by yourself. If you really get stuck, you can always check the solutions in the lab manual.

3.6.6 Hera Lab – Burp Suite

Burp Suite

In this lab you will:

- Configure Burp Suite
- Crawl a web site
- Intercept requests and responses to study the web application
- Get unauthorized access to the development area


Live Chat Offline
FAQ Blog About Us Shopping Cart Login
eLearnSecurity
Forging security professionals
Home Courses Training Paths Certifications Corporate Solutions Resources
Home > Login
Sign in with Caendra
Caendra is the unified login for all eLearnSecurity services.
Email
Password
SIGN IN
OR SIGN IN WITH
GOOGLE
LINKEDIN
Forgot password?
Don't have an account? Sign up
Don't have an account with us yet? Sign in here using Google or LinkedIn
Sign up here if you don't want to use Google or LinkedIn

**Labs are only available in Full or Elite Editions of the course. To upgrade, click HERE. To access, go to the course in your members area and click the labs drop-down in the appropriate module line or to the virtual labs tabs on the left navigation.*

References

References

[URIs, URLs, and URNs: Clarifications and Recommendations 1.0](http://www.w3.org/TR/uri-clarification/)

http://www.w3.org/TR/uri-clarification/

[SSL/TLS Strong Encryption: An Introduction](http://httpd.apache.org/docs/2.2/ssl/ssl_intro.html)

http://httpd.apache.org/docs/2.2/ssl/ssl_intro.html

[HTTP State Management Mechanism (cookies)](http://tools.ietf.org/html/rfc6265)

http://tools.ietf.org/html/rfc6265

[ZAP](https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project)

https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project

References

HTTP Status Code Reference

http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html

HTTP Overview, History, Versions and Standards

http://www.tcpipguide.com/free/t_HTTPOverviewHistoryVersionsandStandards.htm

Burp Suite

https://portswigger.net/burp/communitydownload

Portswigger Burp Suite Editions

http://portswigger.net/burp/

References

Squid: Optimising Web Delivery

http://www.squid-cache.org/

HTTP/1.X

https://hpbn.co/http1x/

Videos

HTTP and HTTPS Protocols Basics

In this video, you will see HTTP and HTTPS in action both via command line tools or with Burp Proxy. You will learn how to perform requests, receive responses and analyze HTTP(s) messages both manually and with an automated tool. Additionally, you will learn how to use Burp at the end of this module.

HTTP Cookies and Sessions

In this video, you will see how the cookie protocol works; specifically, installation of cookies, manipulation of cookies, and dissecting a cookie. Moreover, you will see how HTTP sessions work and how they use cookies.

Videos are only available in Full or Elite Editions of the course. To upgrade, click HERE. To access, go to the course in your members area and click the resources drop-down in the appropriate module line.

Videos

Burp Suite

In this video, you will learn how to define the target scope in Burp and how to configure the Proxy to intercept only the traffic in scope. Moreover, you will see how to configure and use other features of Burp according to the penetration test engagement goals.

**Videos are only available in Full or Elite Editions of the course. To upgrade, click HERE. To access, go to the course in your members area and click the resources drop-down in the appropriate module line.*

Labs

Burp Suite Basics

Learn how to use Burp Intruder and Repeater in order to attack a web application. In this lab you will use applications' own resources in order to hack it using Burp Intruder.

Burp Suite

Learn how to use Burp by hacking a web application! In this lab you will configure Burp Suite, crawl a web site, intercept requests and responses to study the web application, and get unauthorized access to the development area.

**Labs are only available in Full or Elite Editions of the course. To upgrade, click HERE. To access, go to the course in your members area and click the labs drop-down in the appropriate module line or to the virtual labs tabs on the left navigation.*