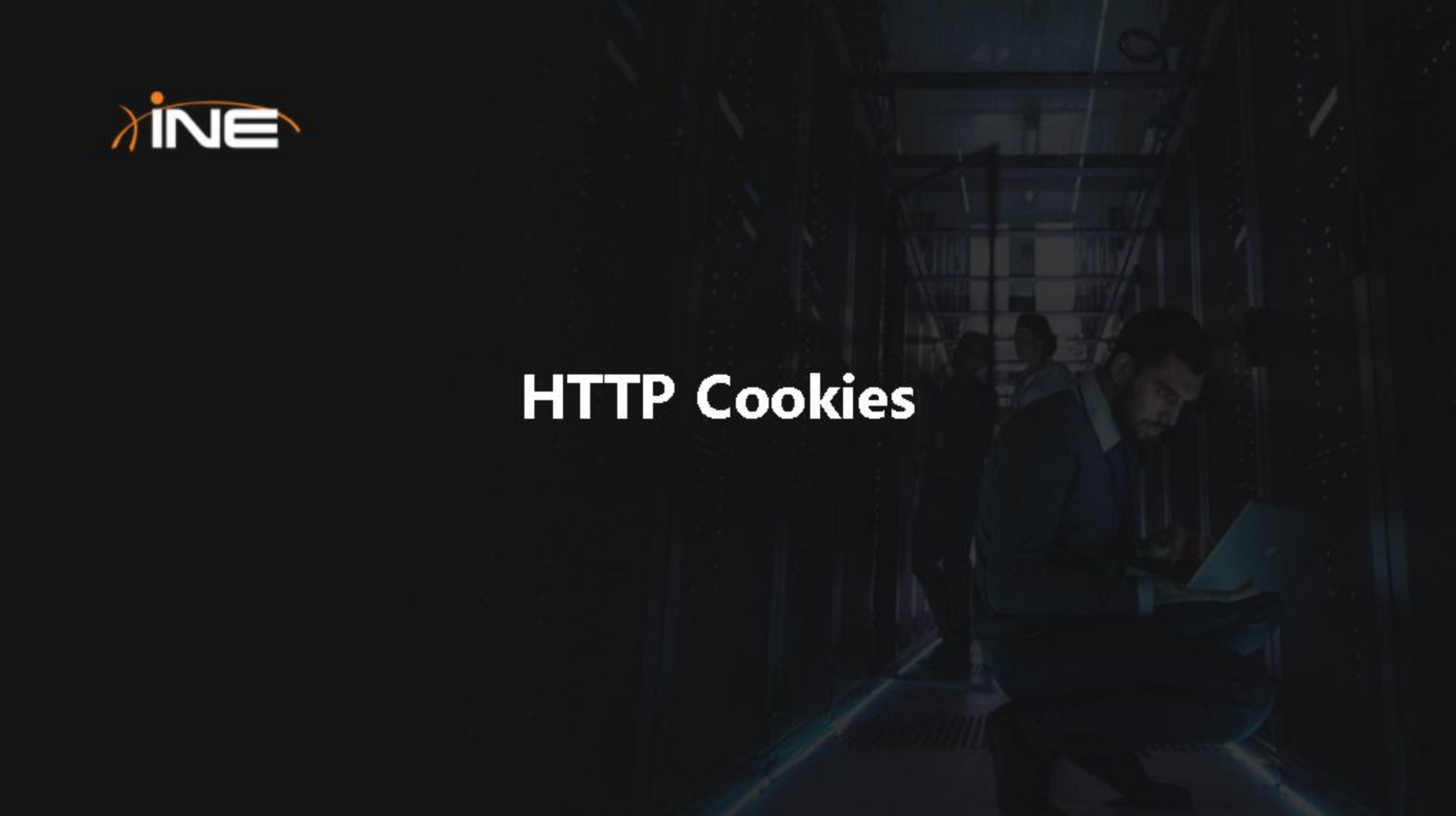


HTTP Cookies

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**.

3.3.3 Cookie Domain

EXAMPLE

+ When a cookie has the domain attribute set to:
  - `domain=elearnsecurity.com`

    or
  - `domain=.elearnsecurity.com`

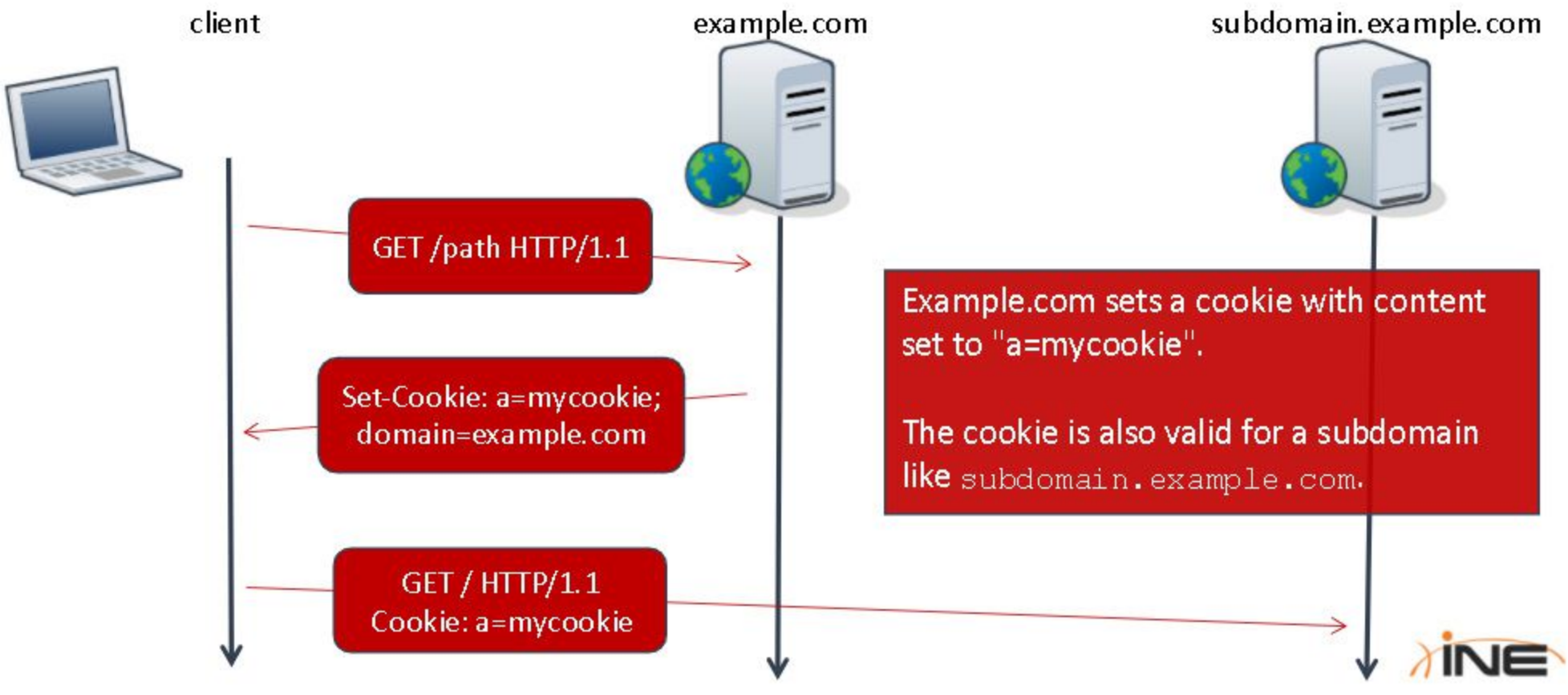
+ The browser will send the cookie to:
  - `www.elearnsecurity.com`
  - `whatever.subdomain.elearnsecurity.com`
  - `elearnsecurity.com`

3.3.3 Cookie Domain

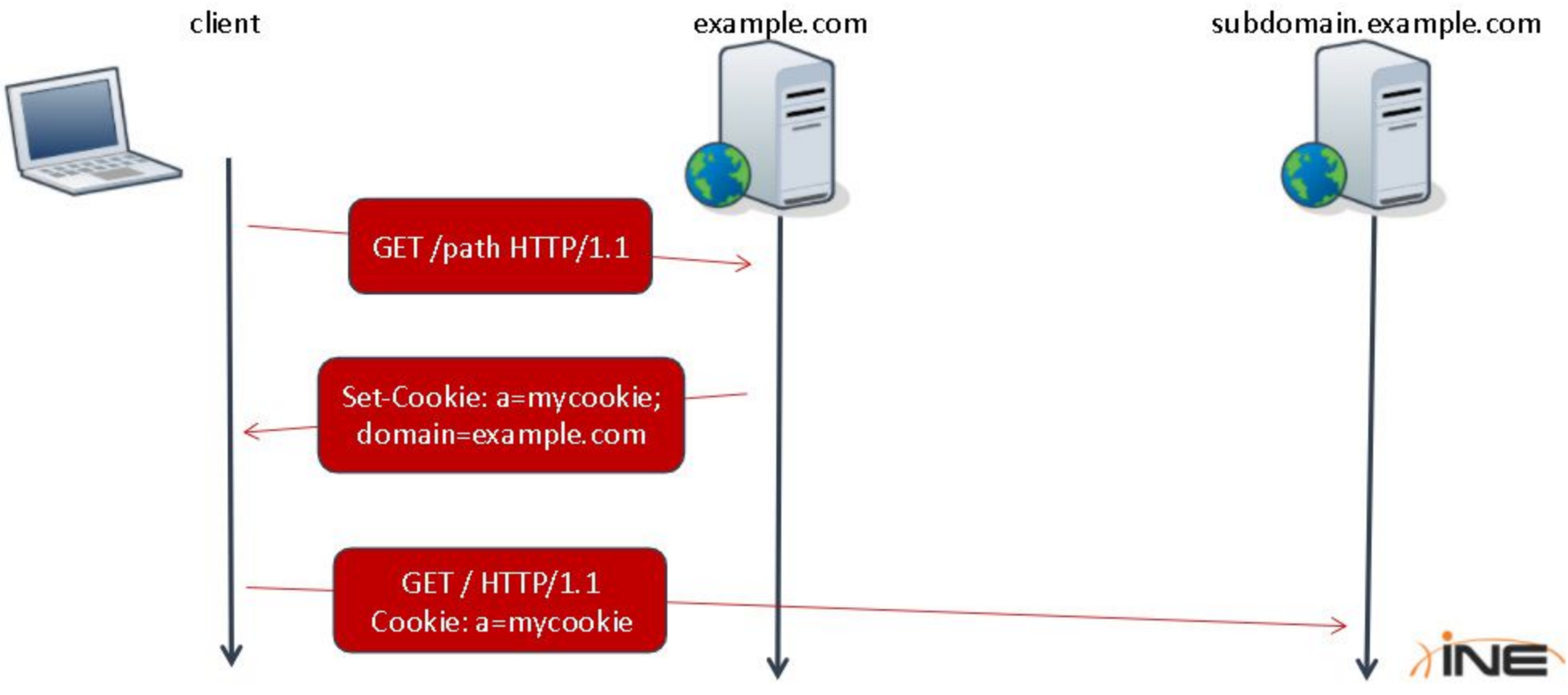
- If the **server does not specify** the domain attribute, the browser will automatically set the domain as the server domain and set the cookie **host-only** flag; this means that the cookie will be sent **only to that precise hostname**.

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

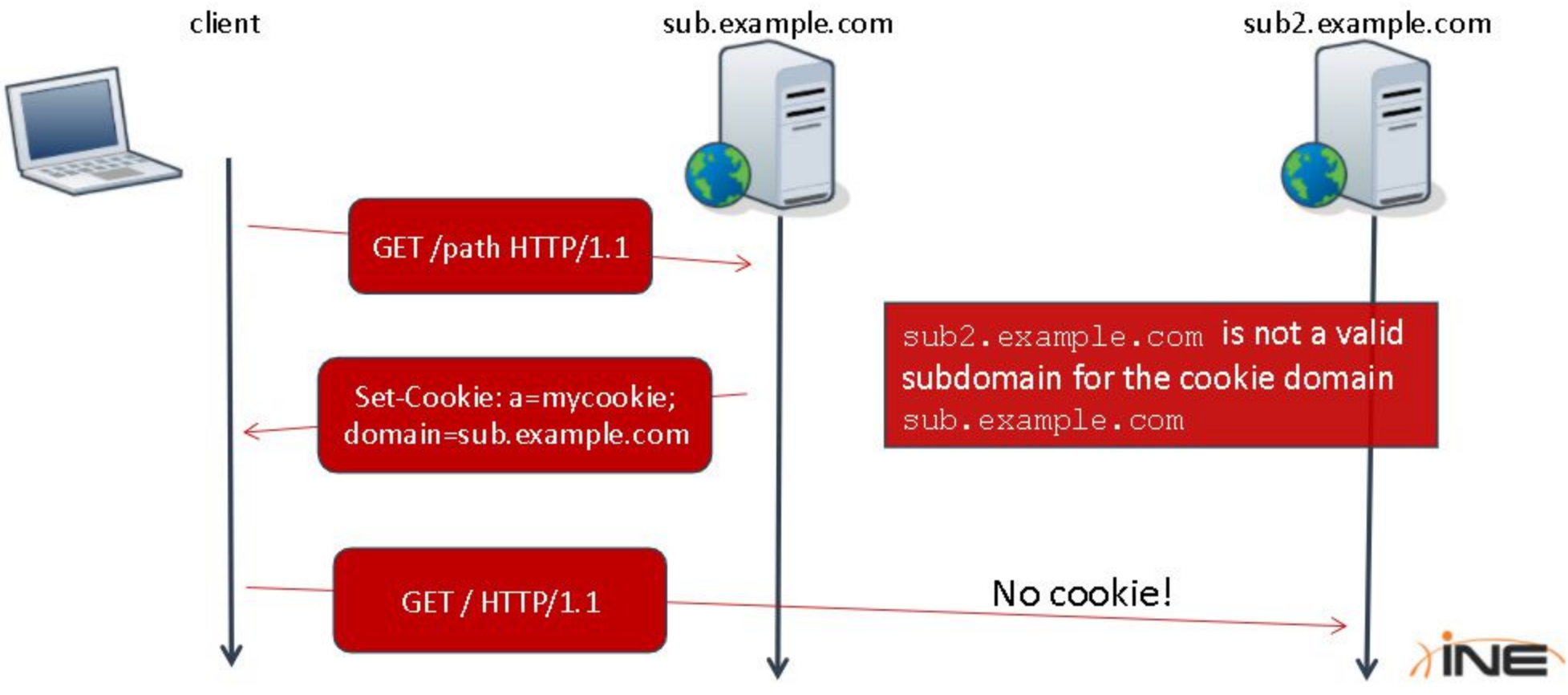
3.3.3.1 Cookie Domain Examples



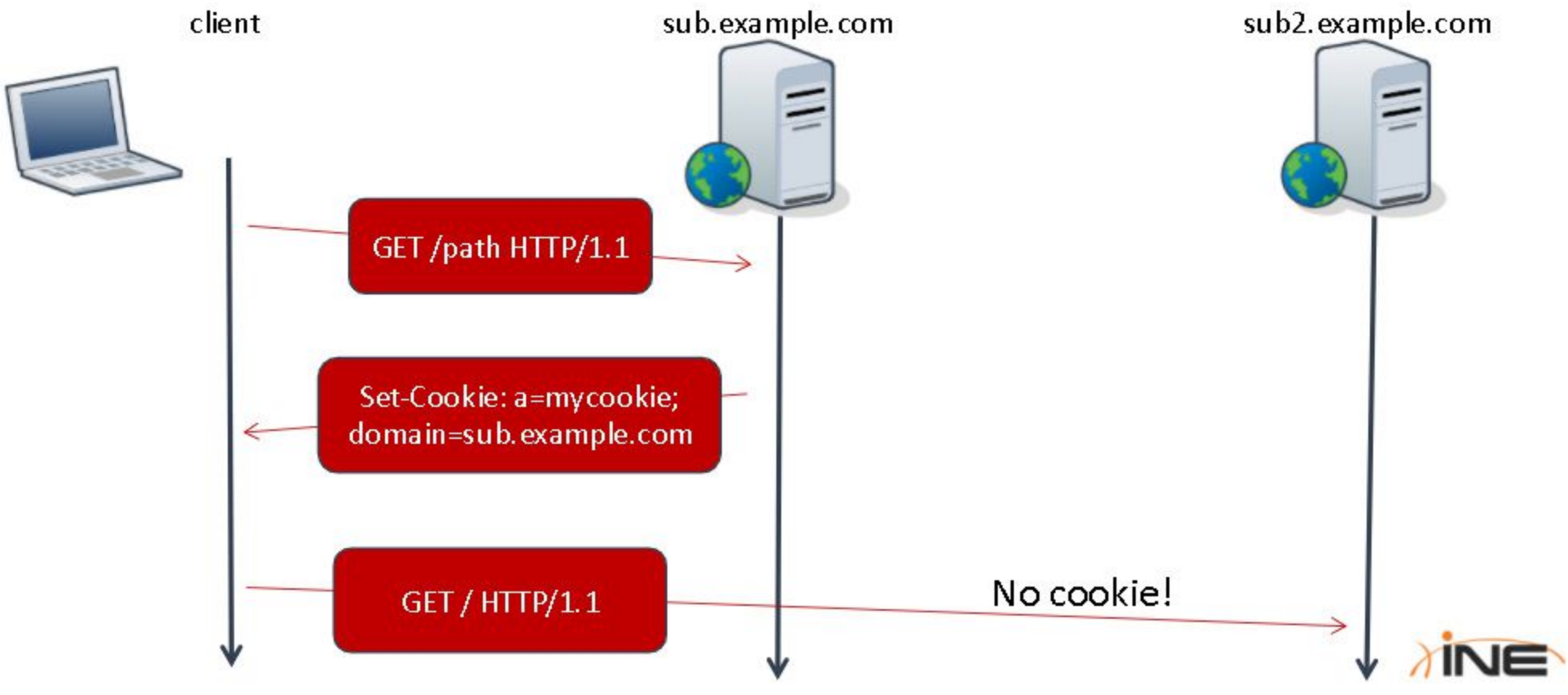
3.3.3.1 Cookie Domain Examples



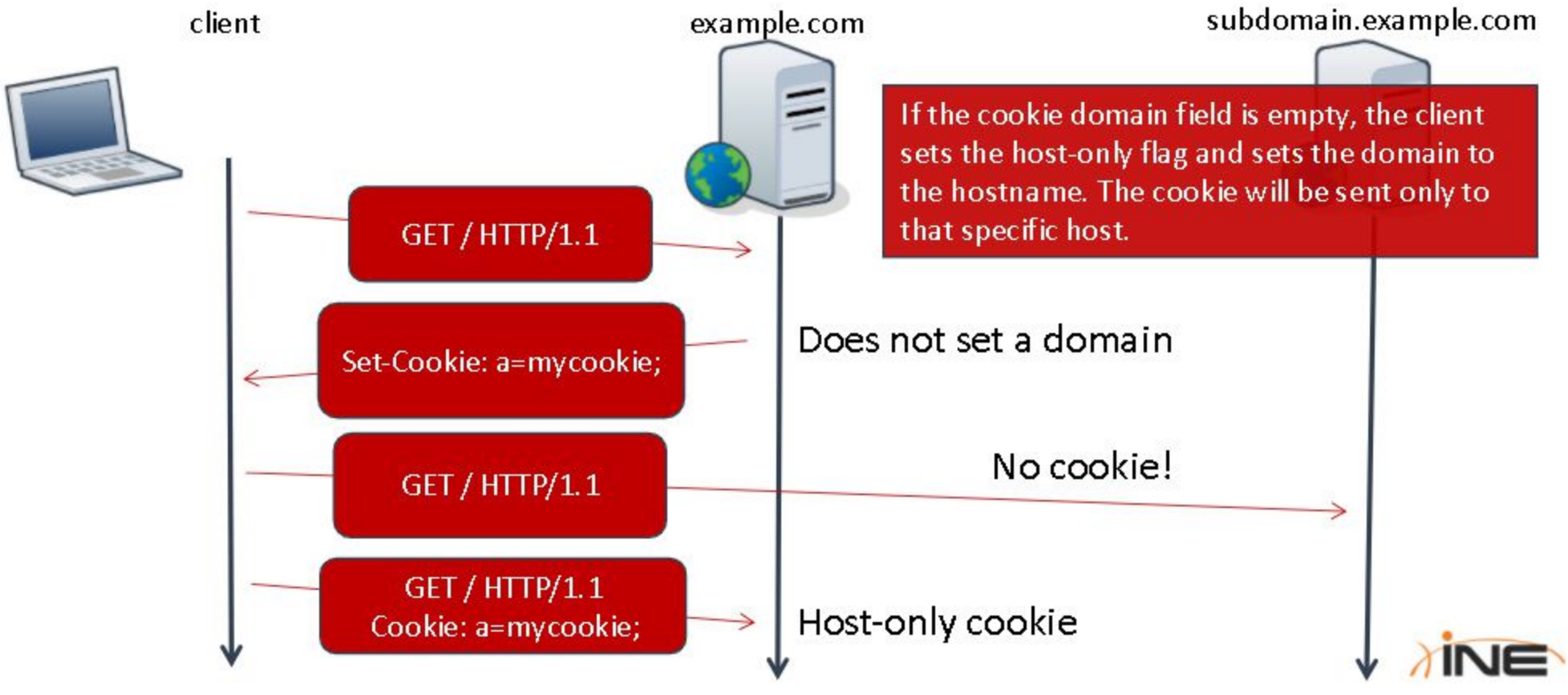
3.3.3.1 Cookie Domain Examples



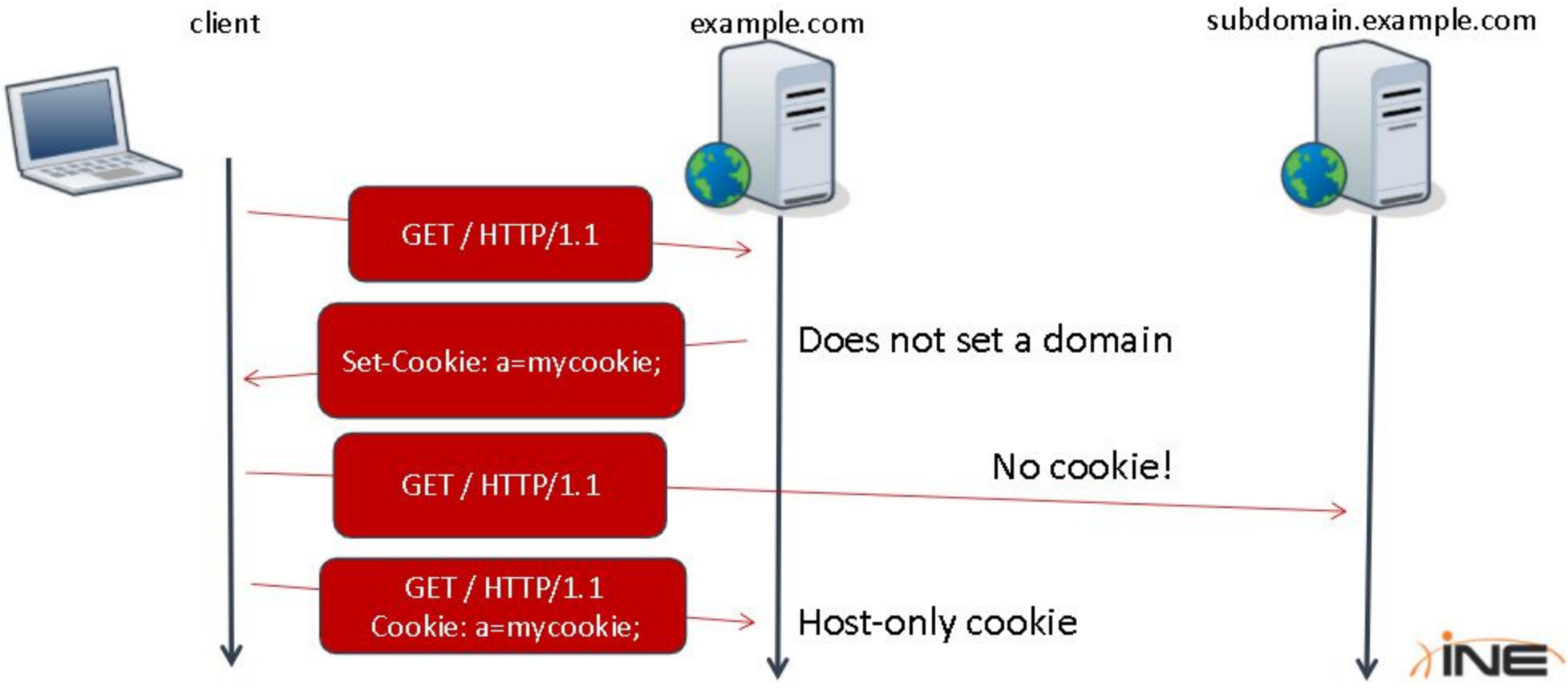
3.3.3.1 Cookie Domain Examples



3.3.3.1 Cookie Domain Examples



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.

3.3.4 Cookie Path

EXAMPLE

- 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 HTTPS connection (they will not be sent over HTTP).

3.3.8 Cookie Content

EXAMPLE

+ 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.

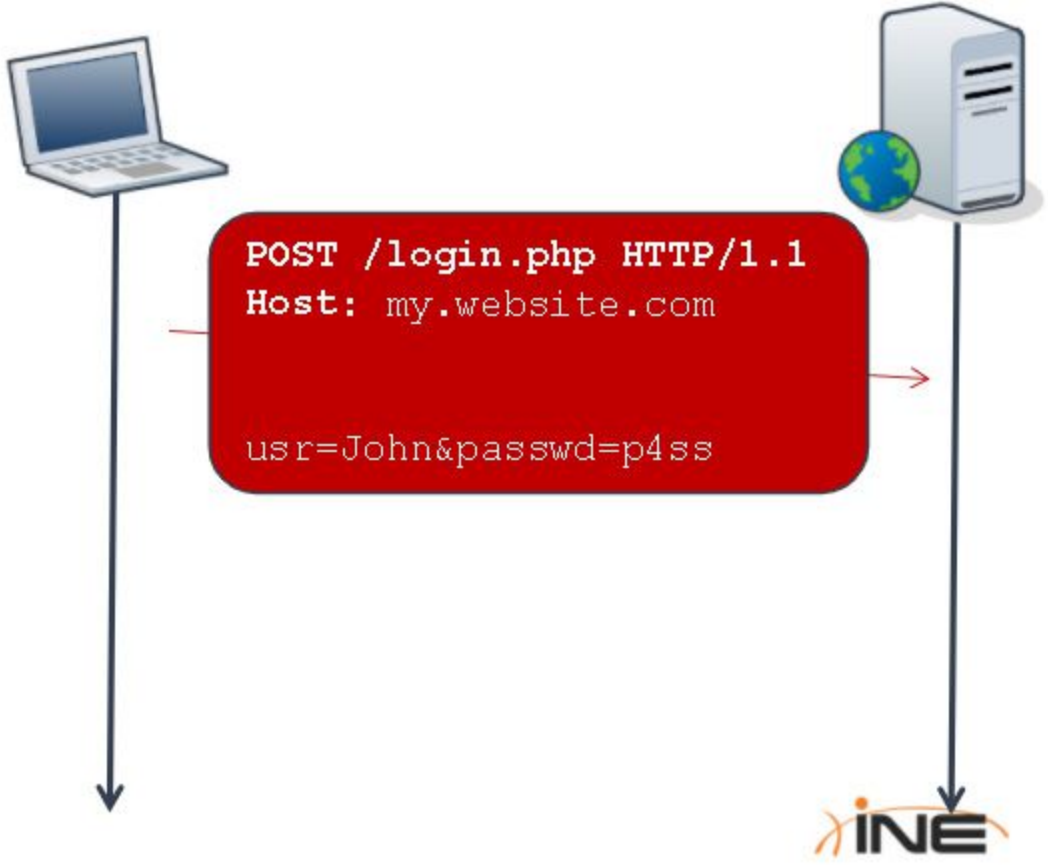
3.3.9 Cookie Protocol

- 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

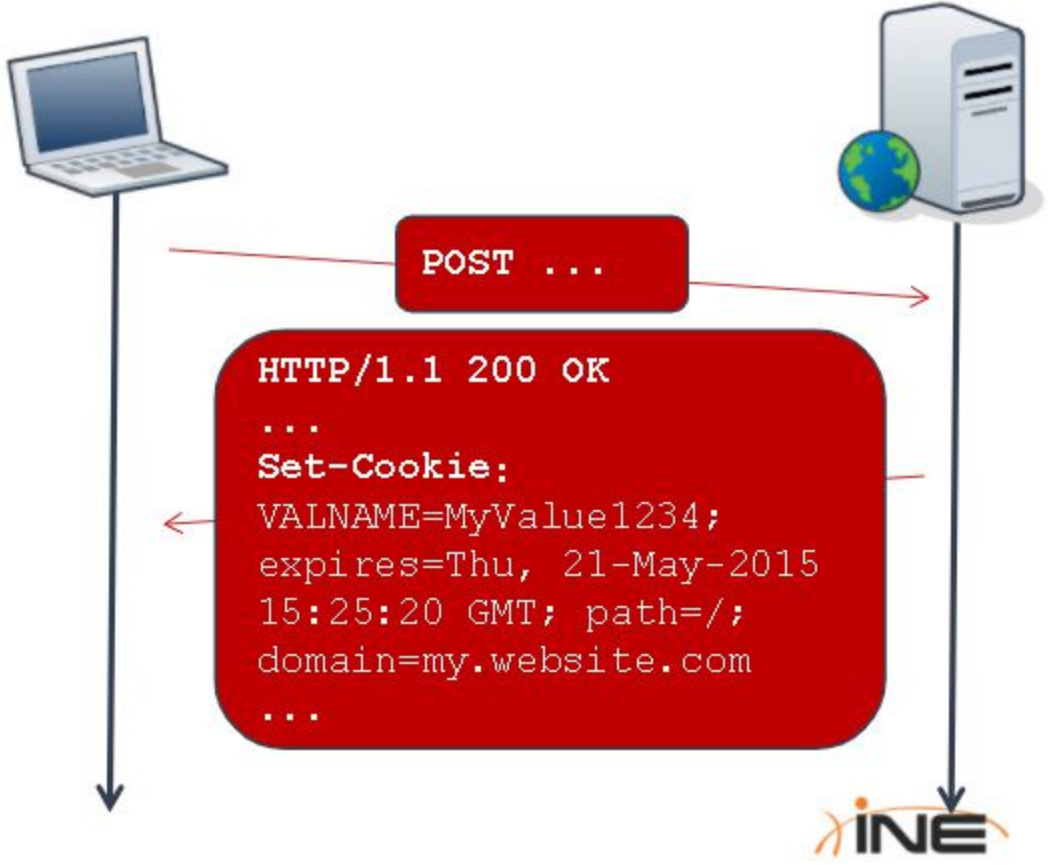
3.3.9 Cookie Protocol

- Cookies are often installed during a login.
- In this example, the browser sends a **POST** request with the username and password.



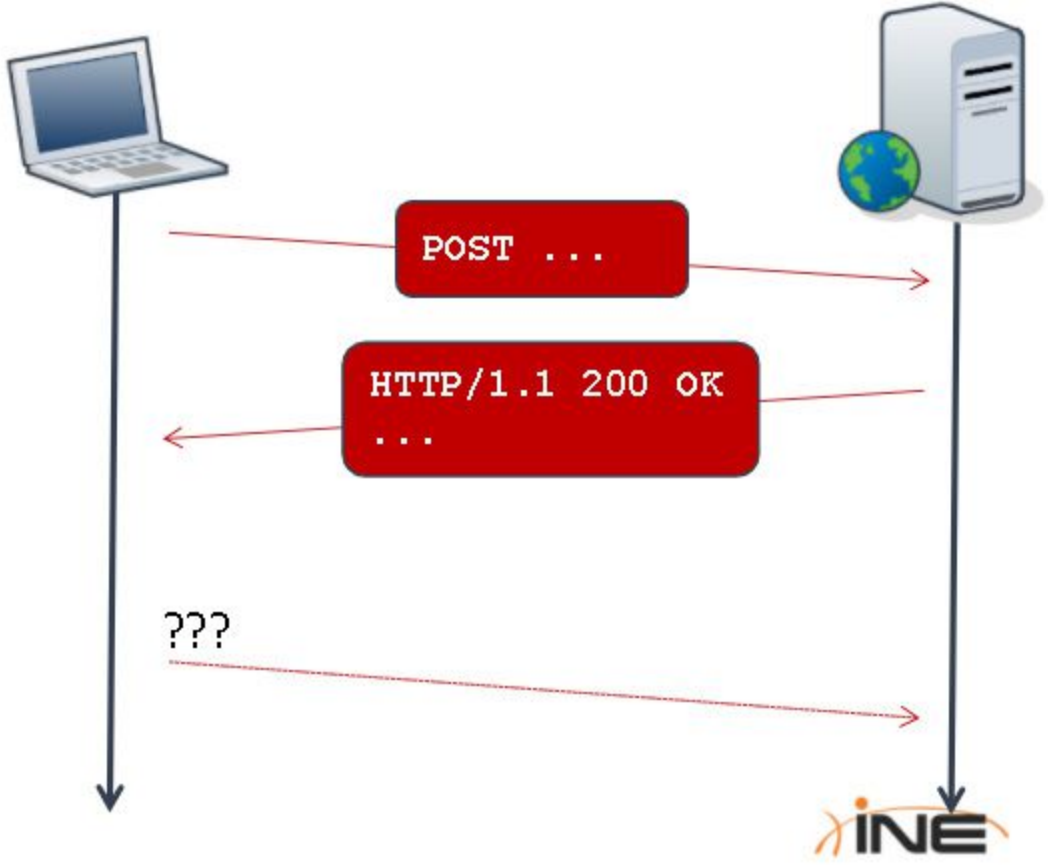
3.3.9 Cookie Protocol

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



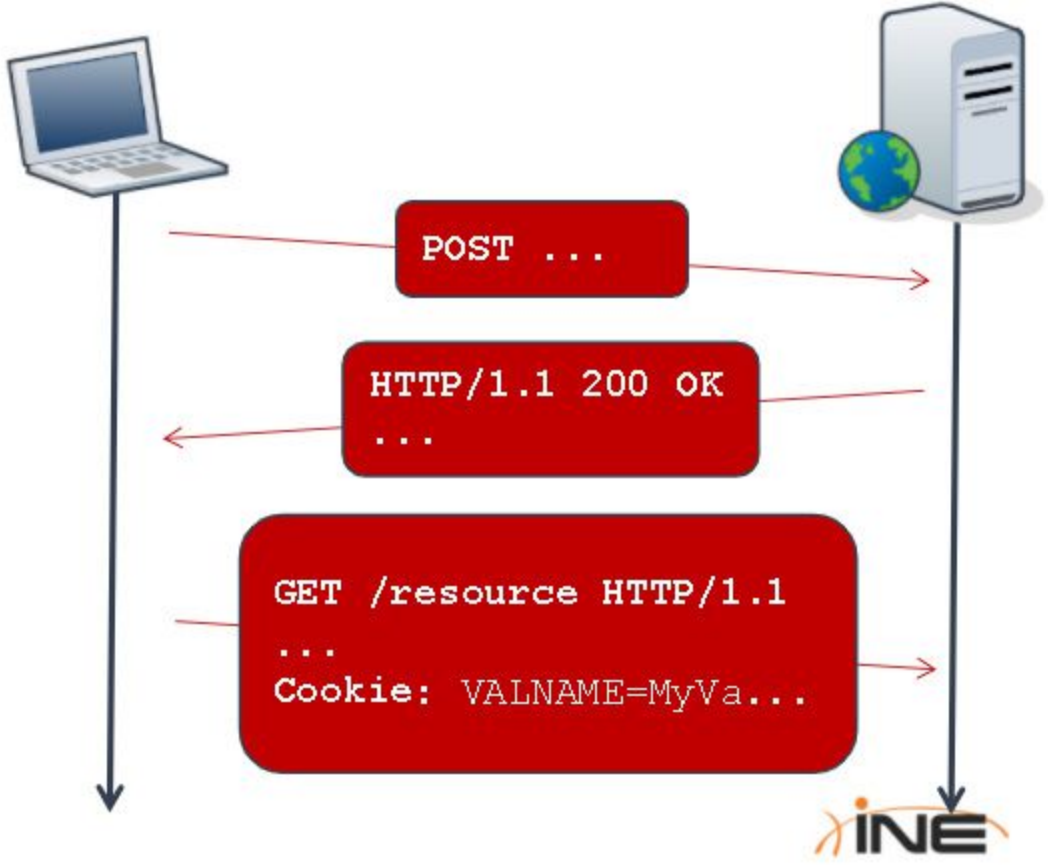
3.3.9 Cookie Protocol

- 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.



References

+ RFC6265: http://tools.ietf.org/html/rfc6265