

Web Security

Week 2 - DNS, Cookies, Sessions

Old Dominion University

Department of Computer Science

CS 495/595 Spring 2022

Michael L. Nelson <mln@cs.odu.edu>

2022-01-24

DNS

Resolving hostnames

```
scorpii:/home/mln % nslookup stanford.edu  
Server:      127.0.0.53  
Address:    127.0.0.53#53
```

Non-authoritative answer:
Name: stanford.edu
Address: 171.67.215.200

```
scorpii:/home/mln % nslookup odu.edu  
Server:      127.0.0.53  
Address:    127.0.0.53#53
```

Non-authoritative answer:
Name: odu.edu
Address: 128.82.112.29

```
mln2@Michaels-MacBook-Air ~ % nslookup stanford.edu  
Server:      2001:578:3f::30  
Address:    2001:578:3f::30#53
```

Non-authoritative answer:
Name: stanford.edu
Address: 171.67.215.200

```
mln2@Michaels-MacBook-Air ~ % nslookup odu.edu  
Server:      2001:578:3f::30  
Address:    2001:578:3f::30#53
```

Non-authoritative answer:
Name: odu.edu
Address: 128.82.112.29

Resolving hostnames

```
scorpii:/home/mln % cd /etc  
  
scorpii:/etc % more hosts  
127.0.0.1 localhost  
172.18.12.33scorpii.infra.cs.odu.eduscorpii  
  
# The following lines are desirable for IPv6 capable  
hosts  
::1 localhost ip6-localhost ip6-loopback  
ff02::1 ip6-allnodes  
ff02::2 ip6-allrouters
```

```
scorpii:/etc % more /etc/resolv.conf  
# This file is managed by man:systemd-resolved(8). Do  
not edit.  
#  
# This is a dynamic resolv.conf file for connecting  
local clients to the  
# internal DNS stub resolver of systemd-resolved.  
This file lists all  
# configured search domains.  
#  
# Run "systemd-resolve --status" to see details about  
the uplink DNS servers  
# currently in use.  
#  
# Third party programs must not access this file  
directly, but only through the  
# symlink at /etc/resolv.conf. To manage  
man:resolv.conf(5) in a different way,  
# replace this symlink by a static file or a  
different symlink.  
#  
# See man:systemd-resolved.service(8) for details  
about the supported modes of  
# operation for /etc/resolv.conf.  
  
nameserver 127.0.0.53  
options edns0  
search infra.cs.odu.edu cs.odu.edu
```

On my iPhone

AT&T 10:47 PM

Wi-Fi mln-airport-5

IP Address 192.168.0.208

Subnet Mask 255.255.255.0

Router 192.168.0.1

[Renew Lease](#)

IPV6 ADDRESS

IP Address 3 Addresses >

Router fe80::c2a0:dff:fe11:7a7b

DNS

Configure DNS Automatic >

HTTP PROXY

Configure Proxy Off >

AT&T 10:47 PM

Back Configure DNS Save

Automatic ✓

Manual

DNS SERVERS

68.105.28.11

68.105.29.11

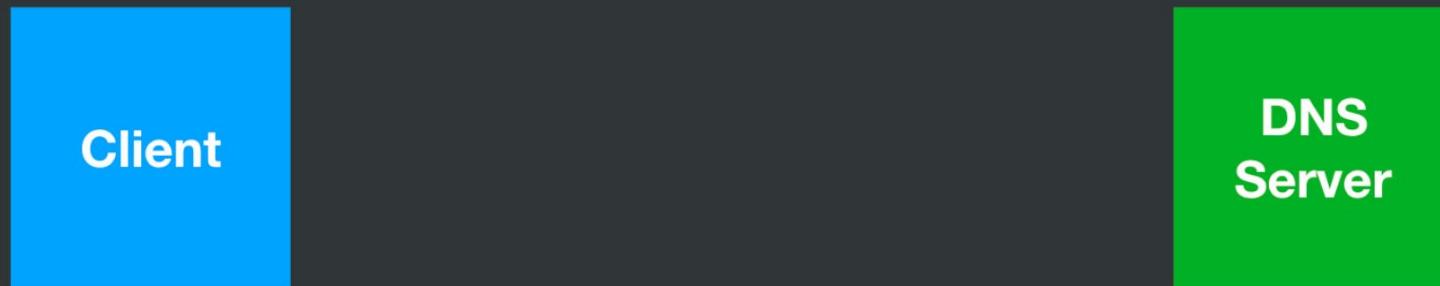
68.105.28.12

2001:578:3f::30

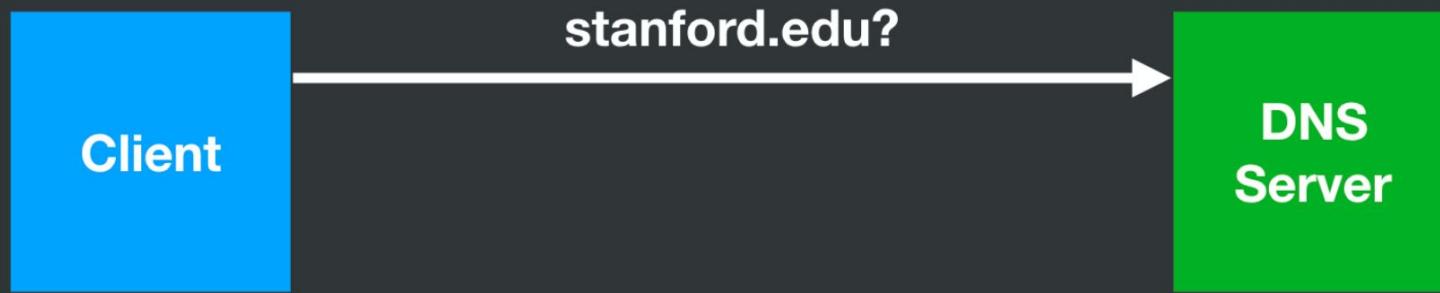
2001:578:3f:1::30

SEARCH DOMAINS

Domain Name System



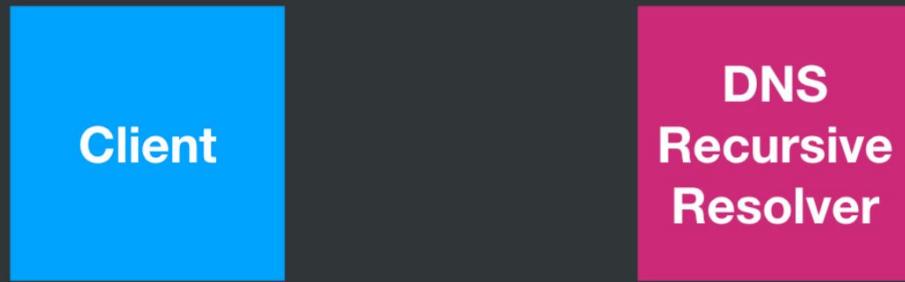
Domain Name System



Domain Name System



DNS Recursive Resolution



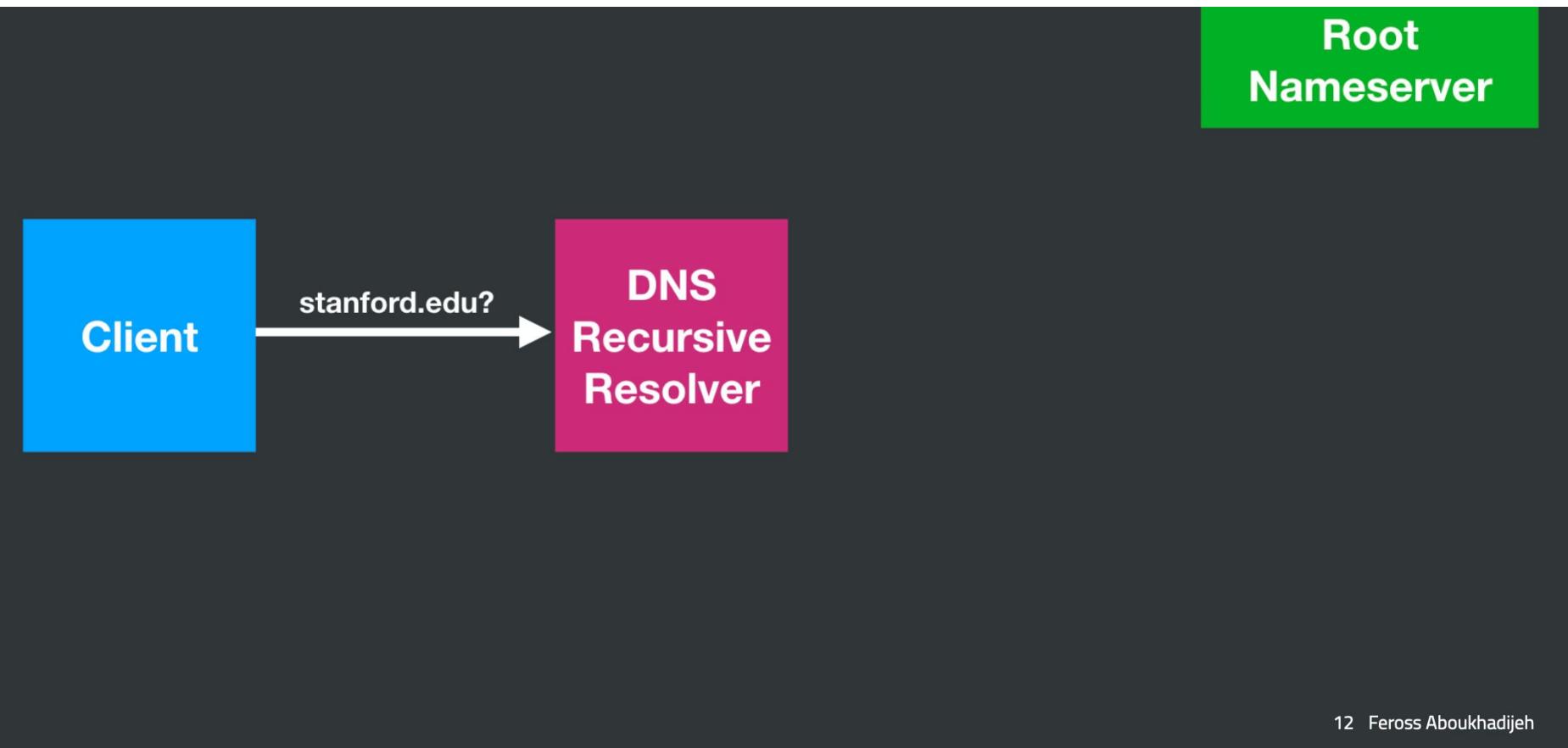
10 Feross Aboukhadijeh

DNS Recursive Resolution



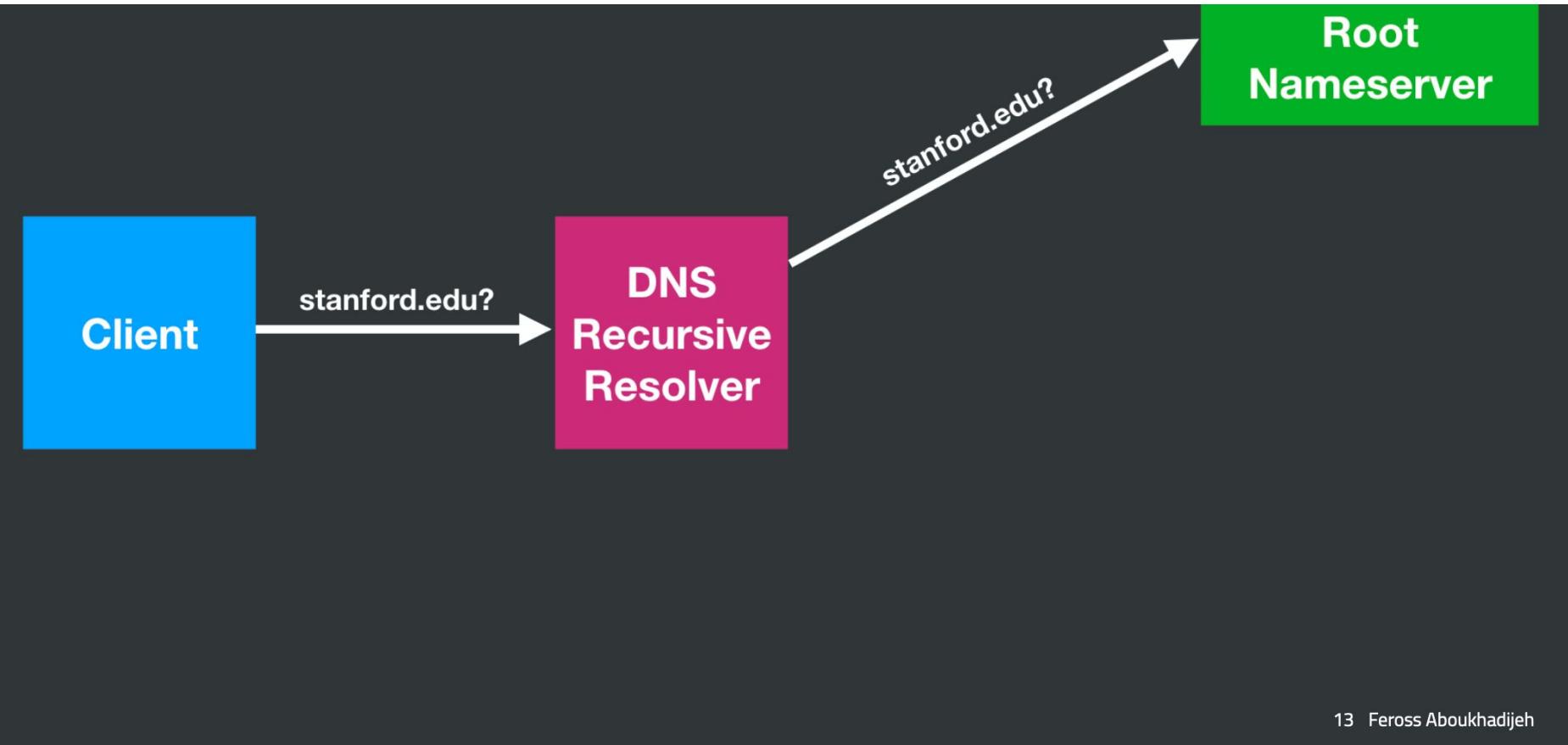
11 Feross Aboukhadijeh

DNS Recursive Resolution



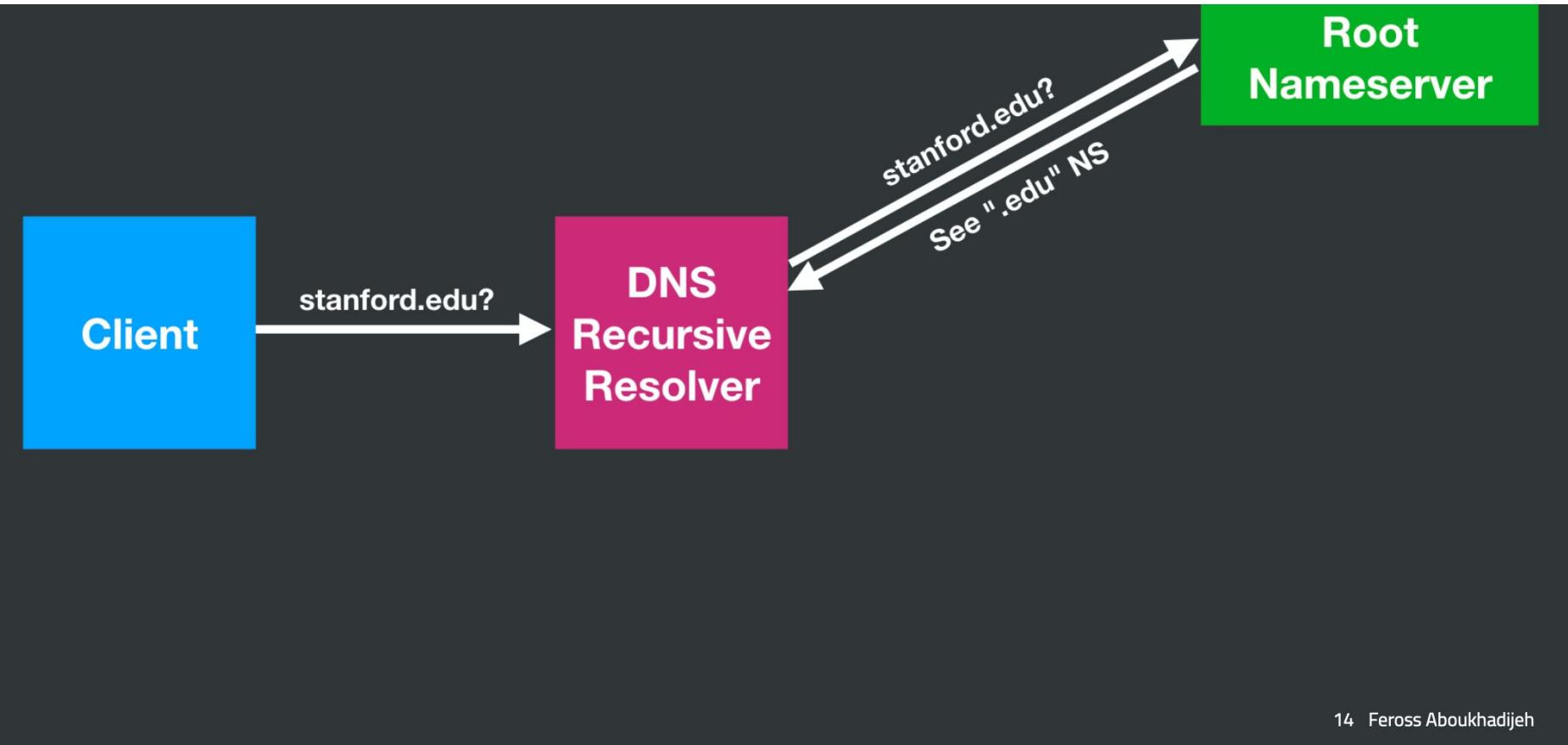
12 Feross Aboukhadijeh

DNS Recursive Resolution



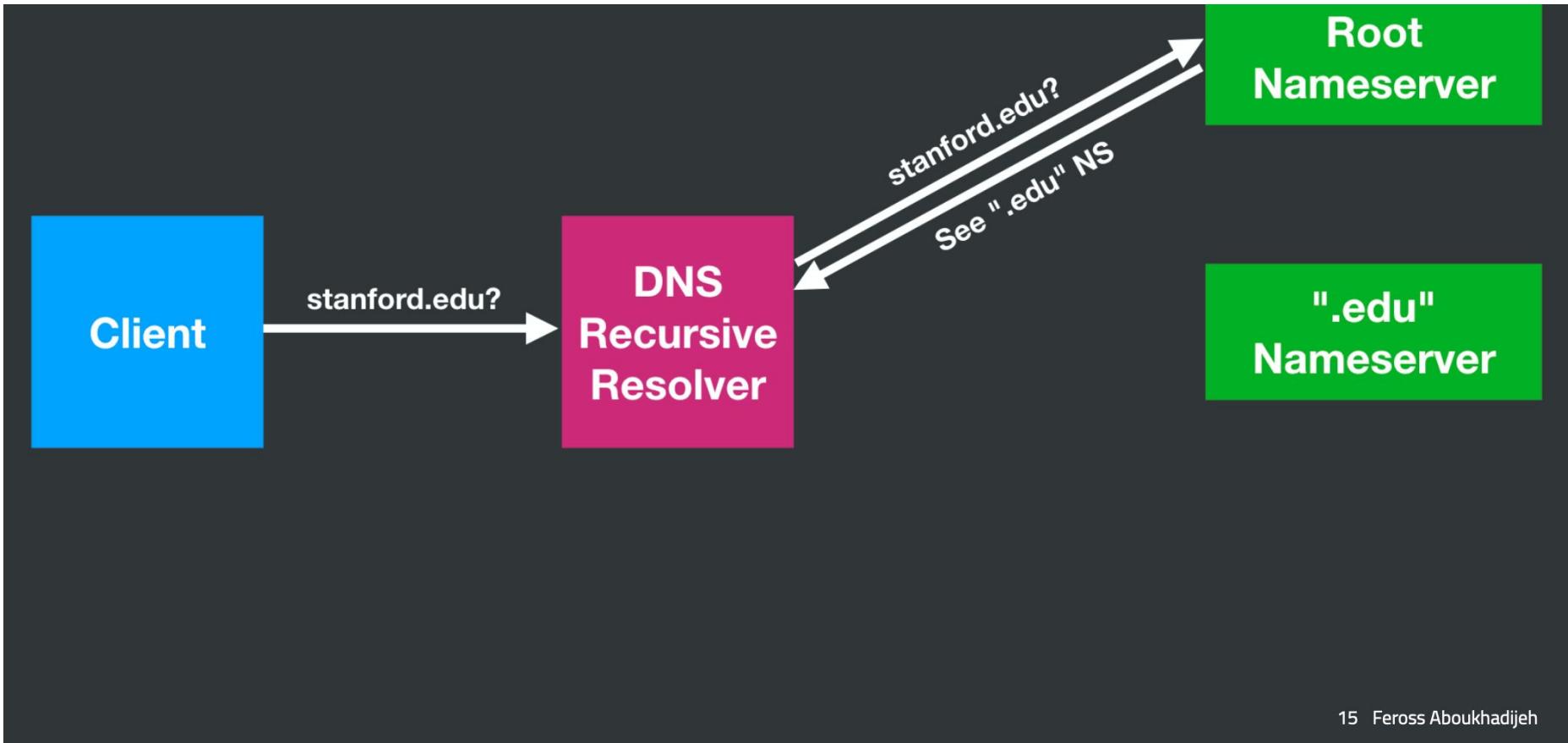
13 Feross Aboukhadijeh

DNS Recursive Resolution



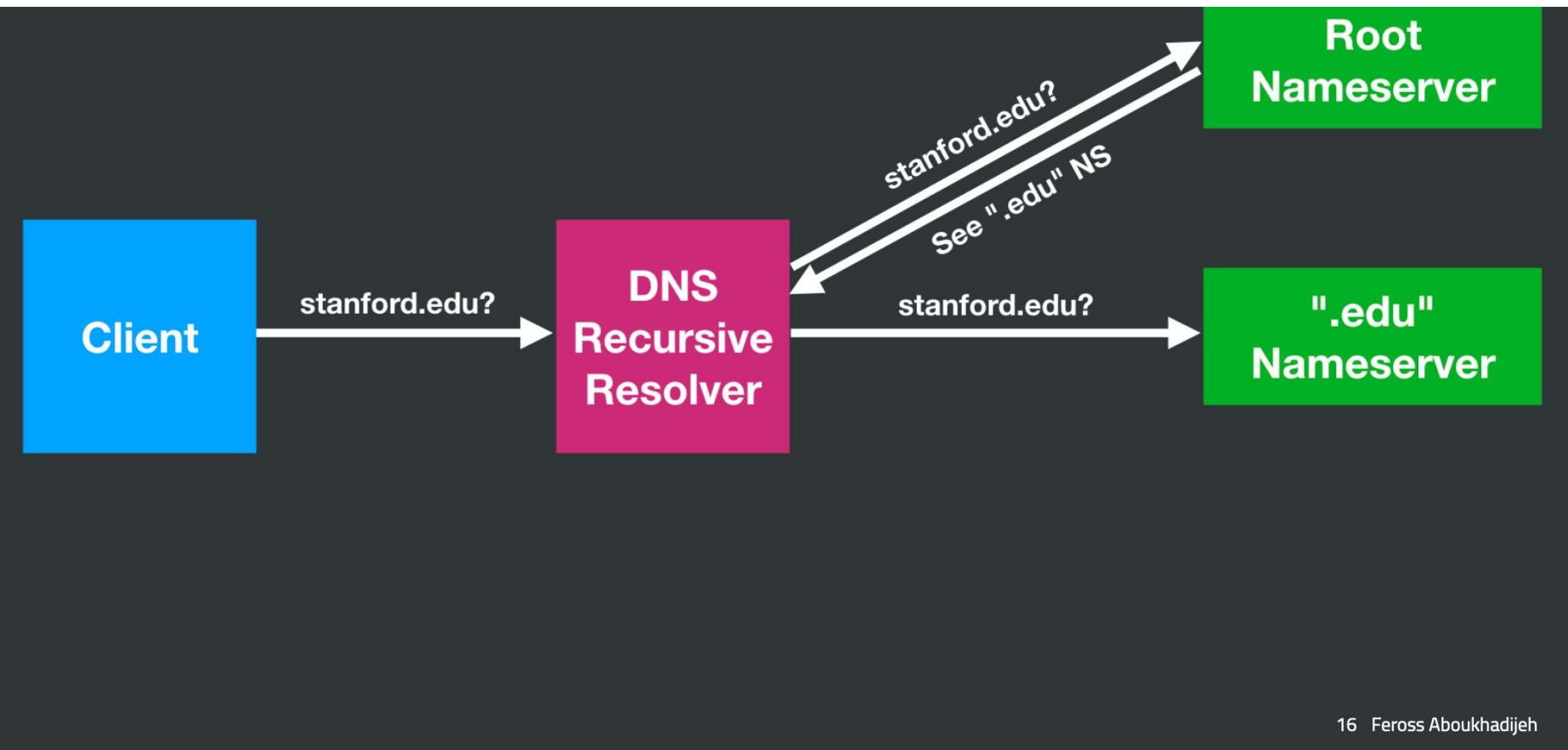
14 Feross Aboukhadijeh

DNS Recursive Resolution



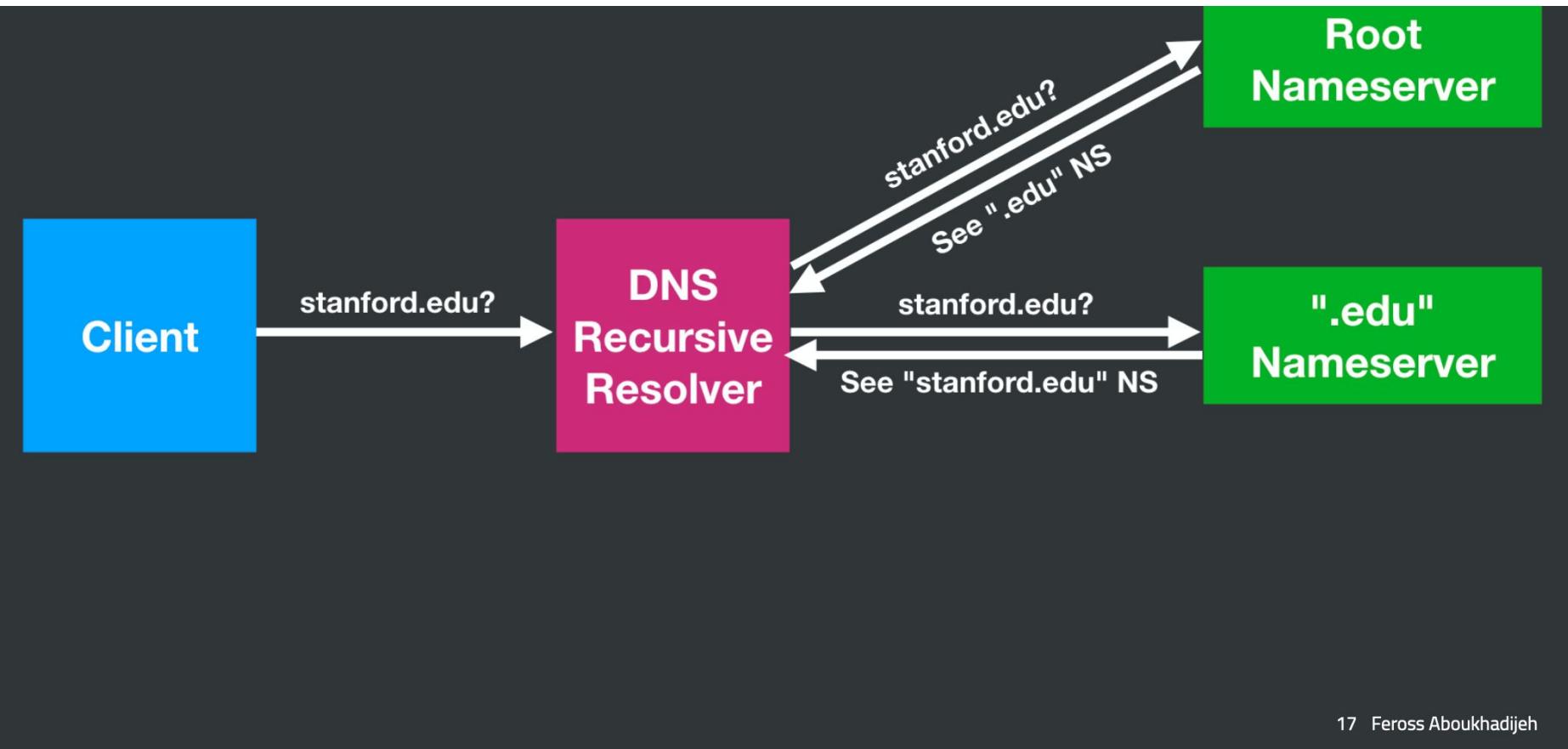
15 Feross Aboukhadijeh

DNS Recursive Resolution



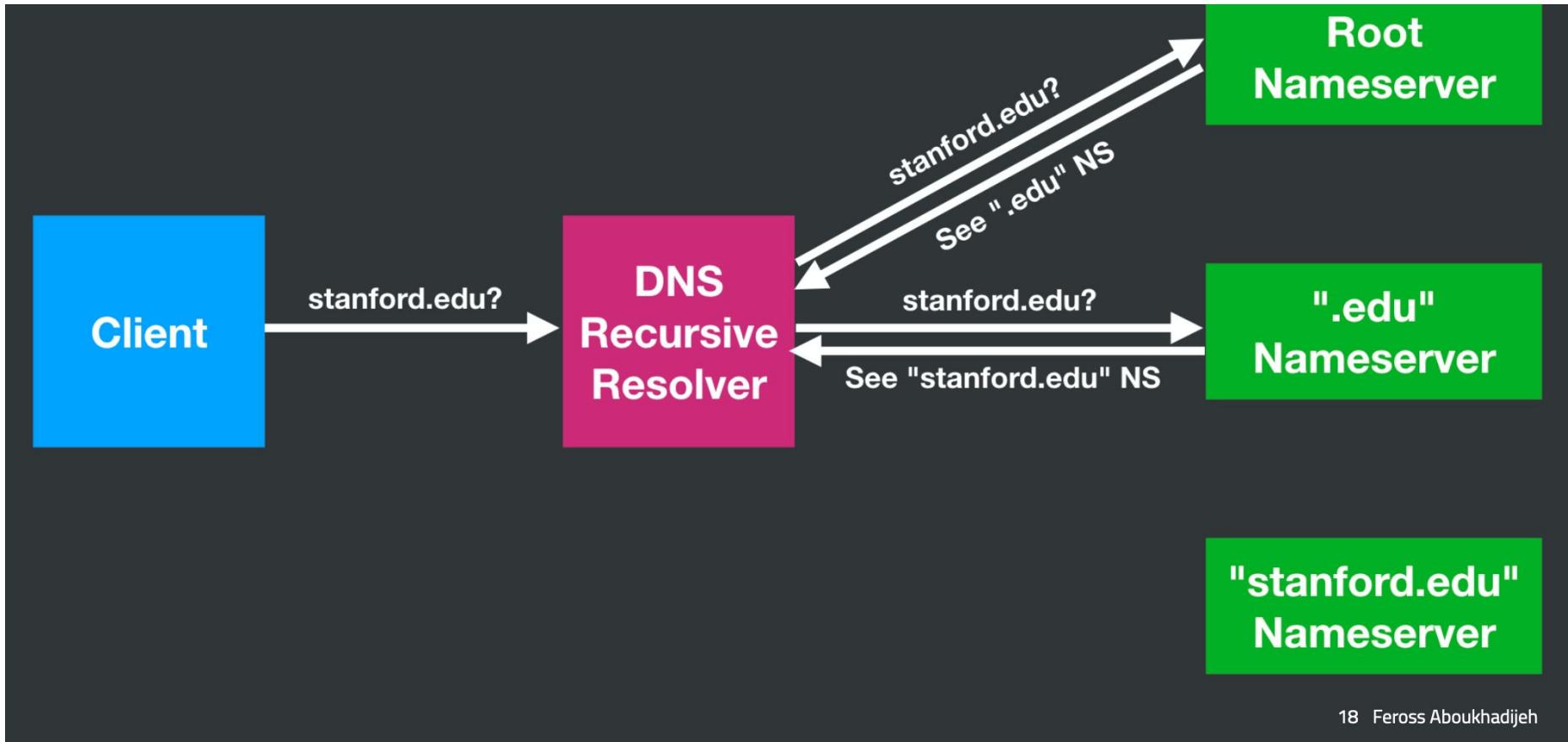
16 Feross Aboukhadijeh

DNS Recursive Resolution



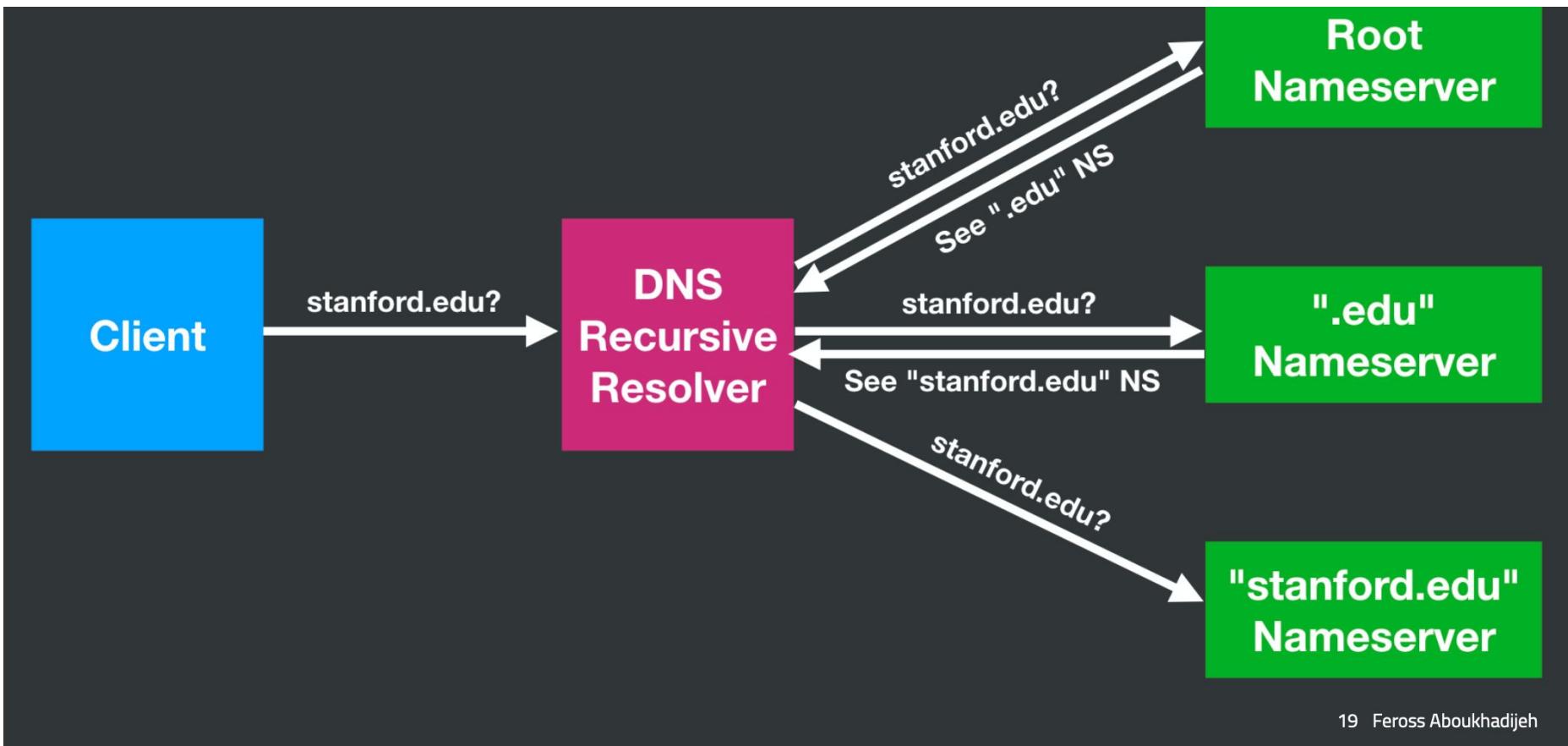
17 Feross Aboukhadijeh

DNS Recursive Resolution



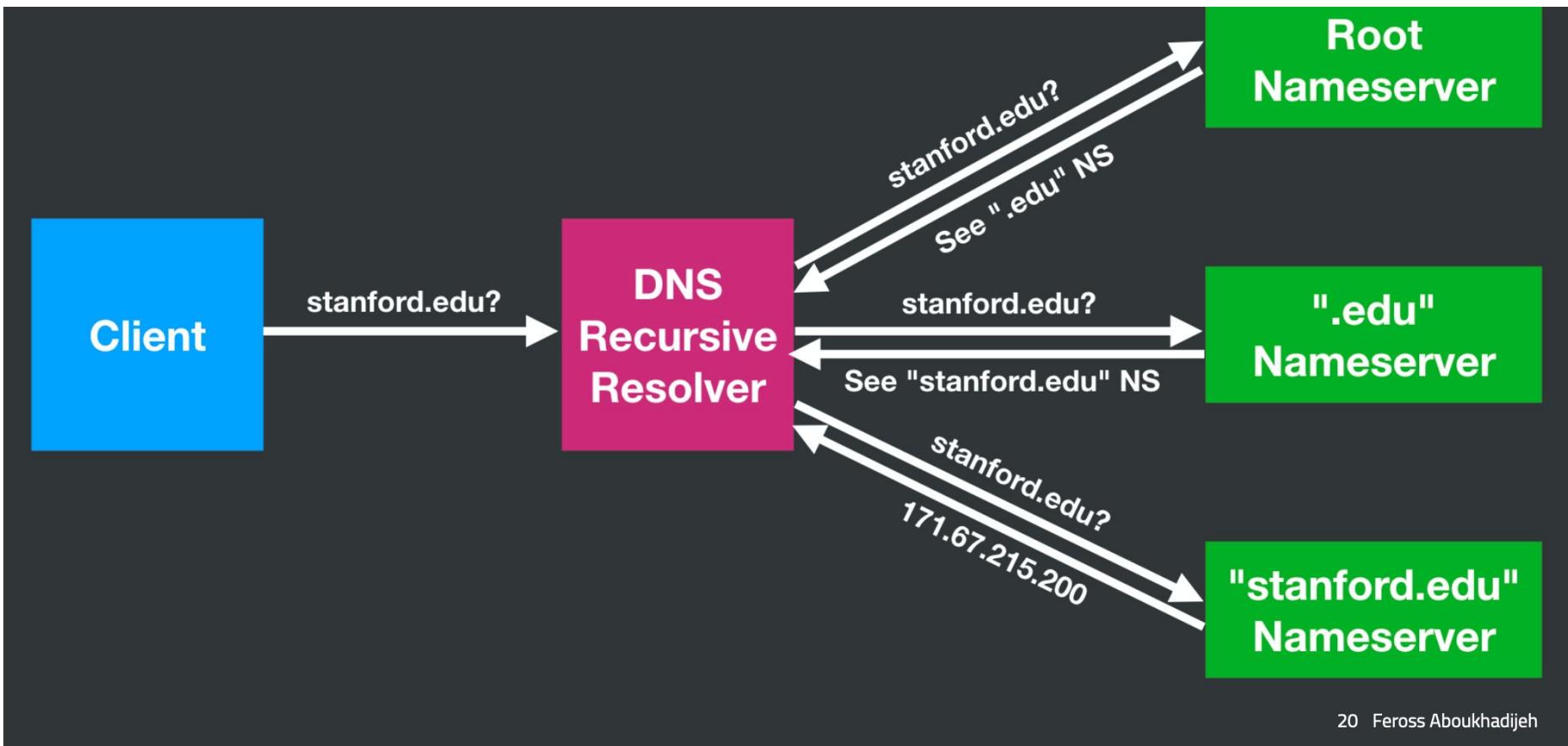
18 Feross Aboukhadijeh

DNS Recursive Resolution



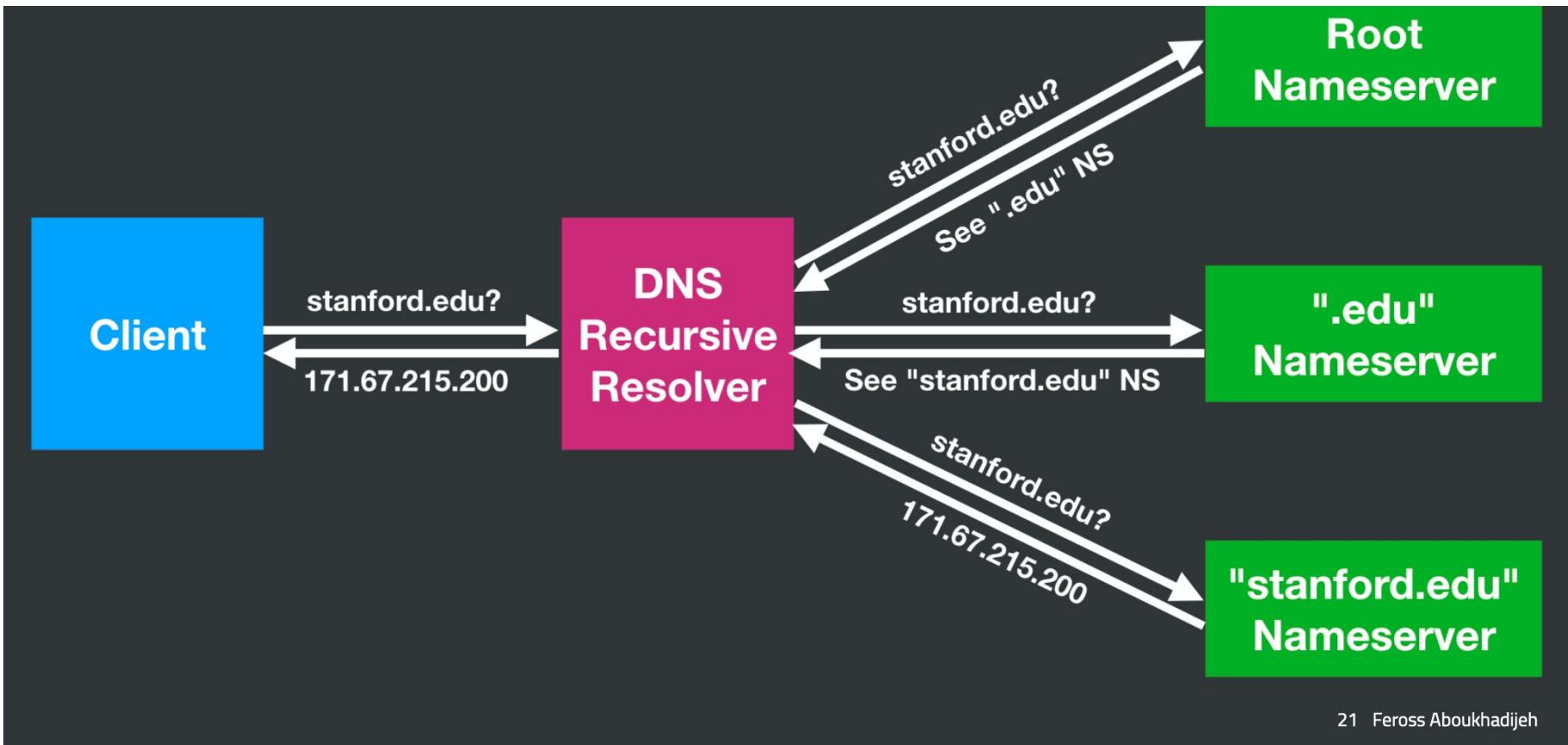
19 Feross Aboukhadijeh

DNS Recursive Resolution



20 Feross Aboukhadijeh

DNS Recursive Resolution



21 Feross Aboukhadijeh

DNS Resolution Before HTTP Connections

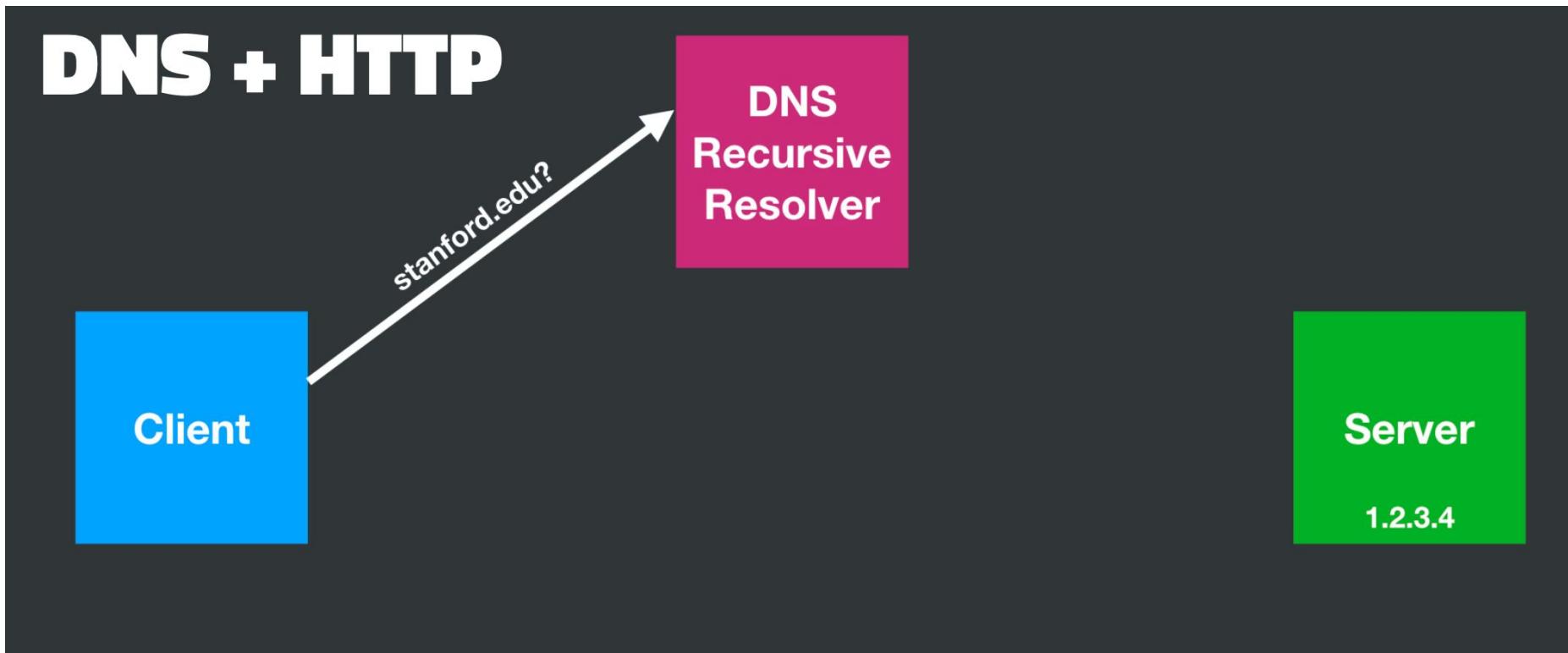
DNS + HTTP

DNS
Recursive
Resolver

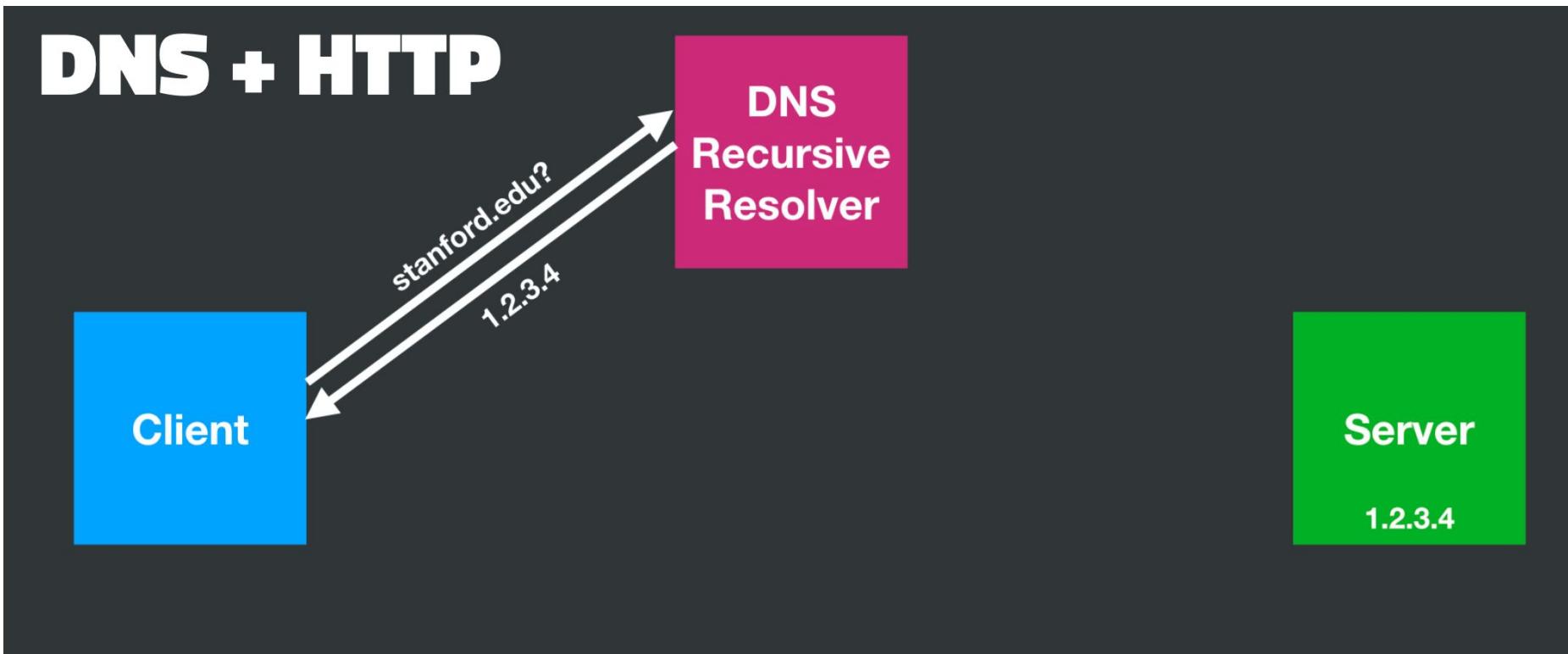
Client

Server
1.2.3.4

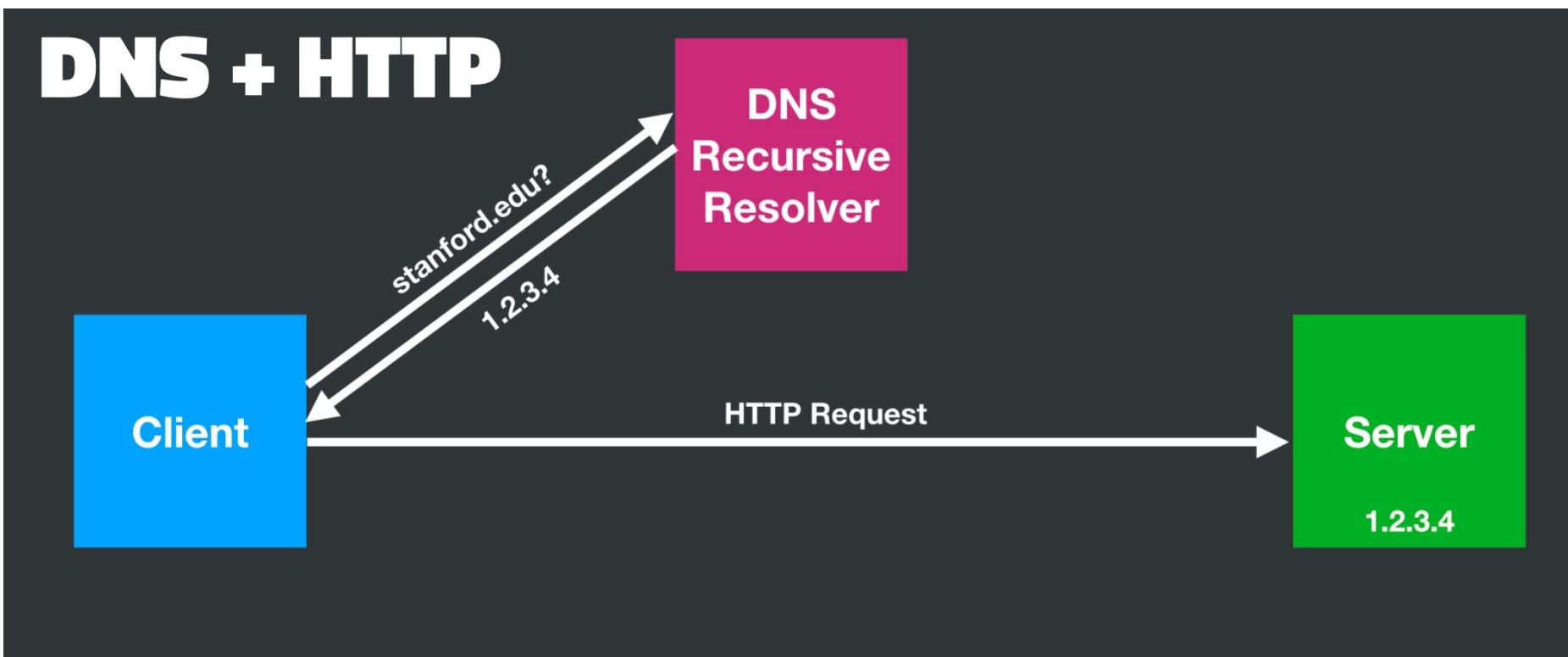
DNS Resolution Before HTTP Connections



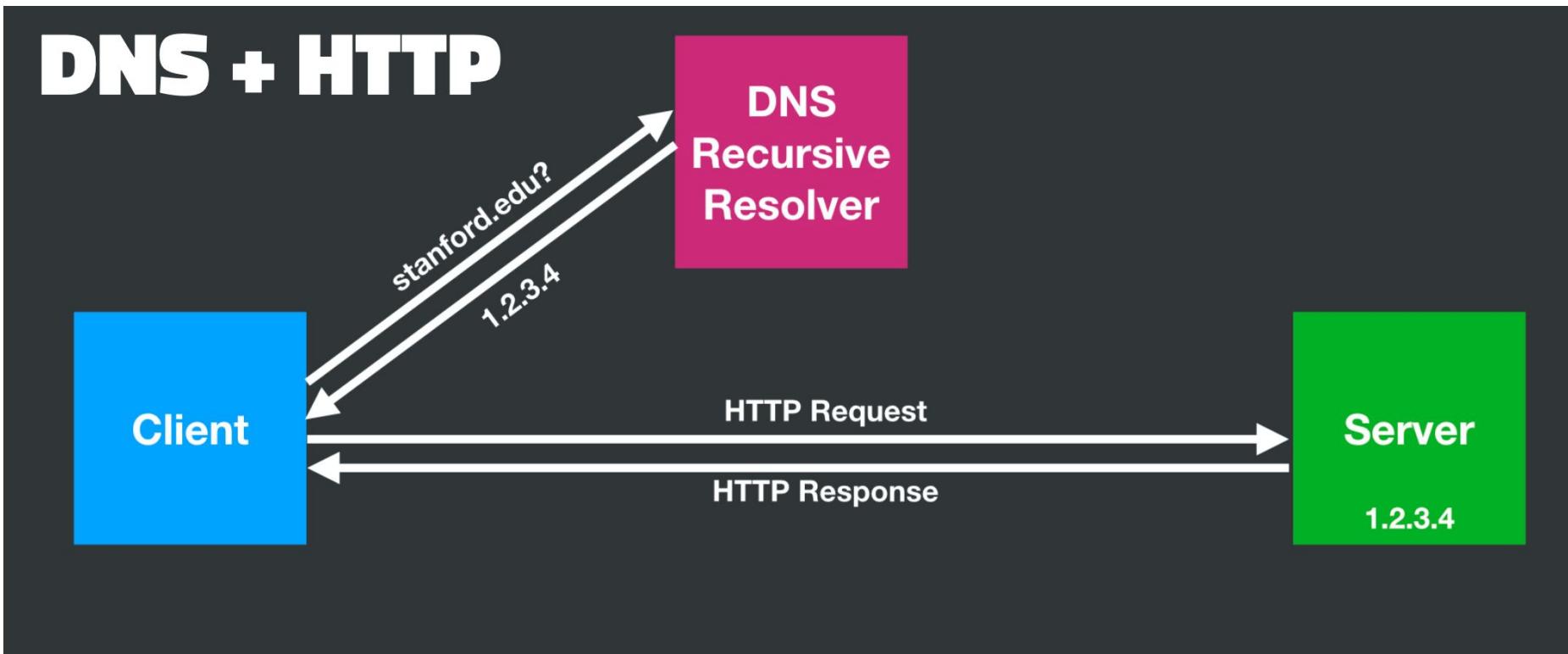
DNS Resolution Before HTTP Connections



DNS Resolution Before HTTP Connections



DNS Resolution Before HTTP Connections

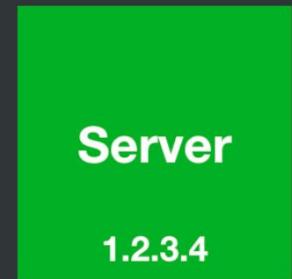
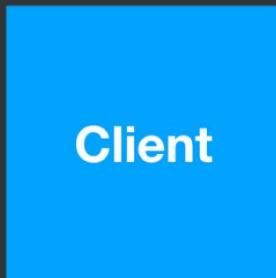


DNS Hijacking

- Attacker changes DNS records of target to point to own IP address
- All site visitors are directed to attacker's web server
- Motivation
 - Phishing
 - Revenue through ads, cryptocurrency mining, etc.
- How do they do it?

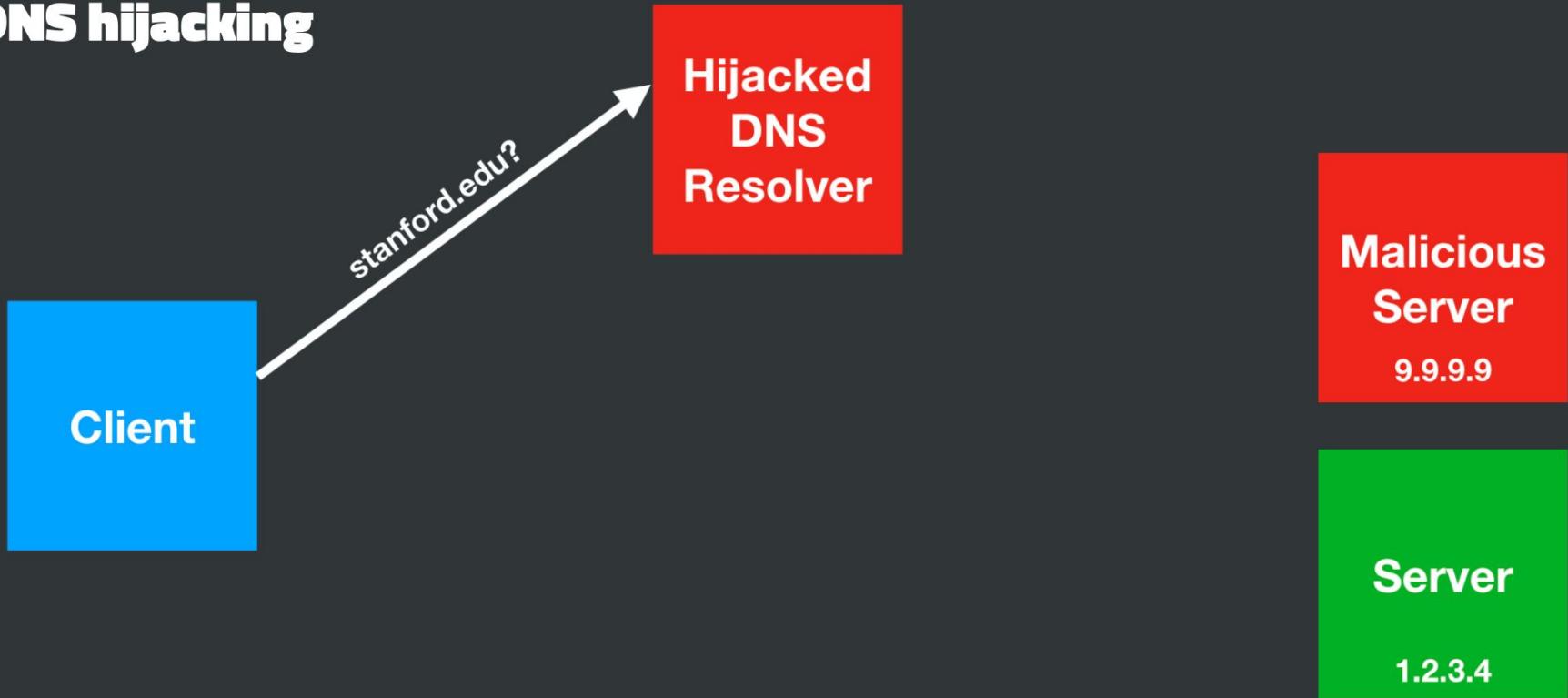
DNS Hijacking

DNS hijacking



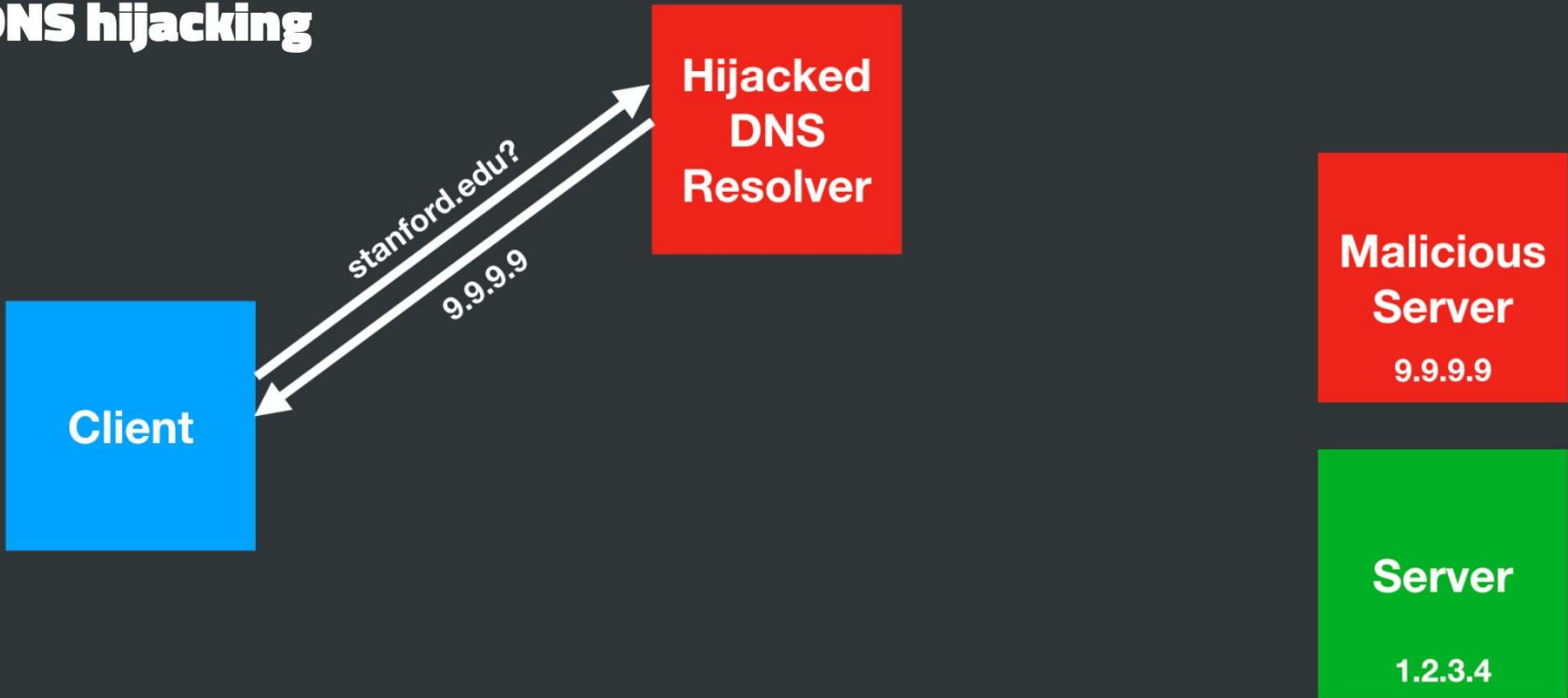
DNS Hijacking

DNS hijacking



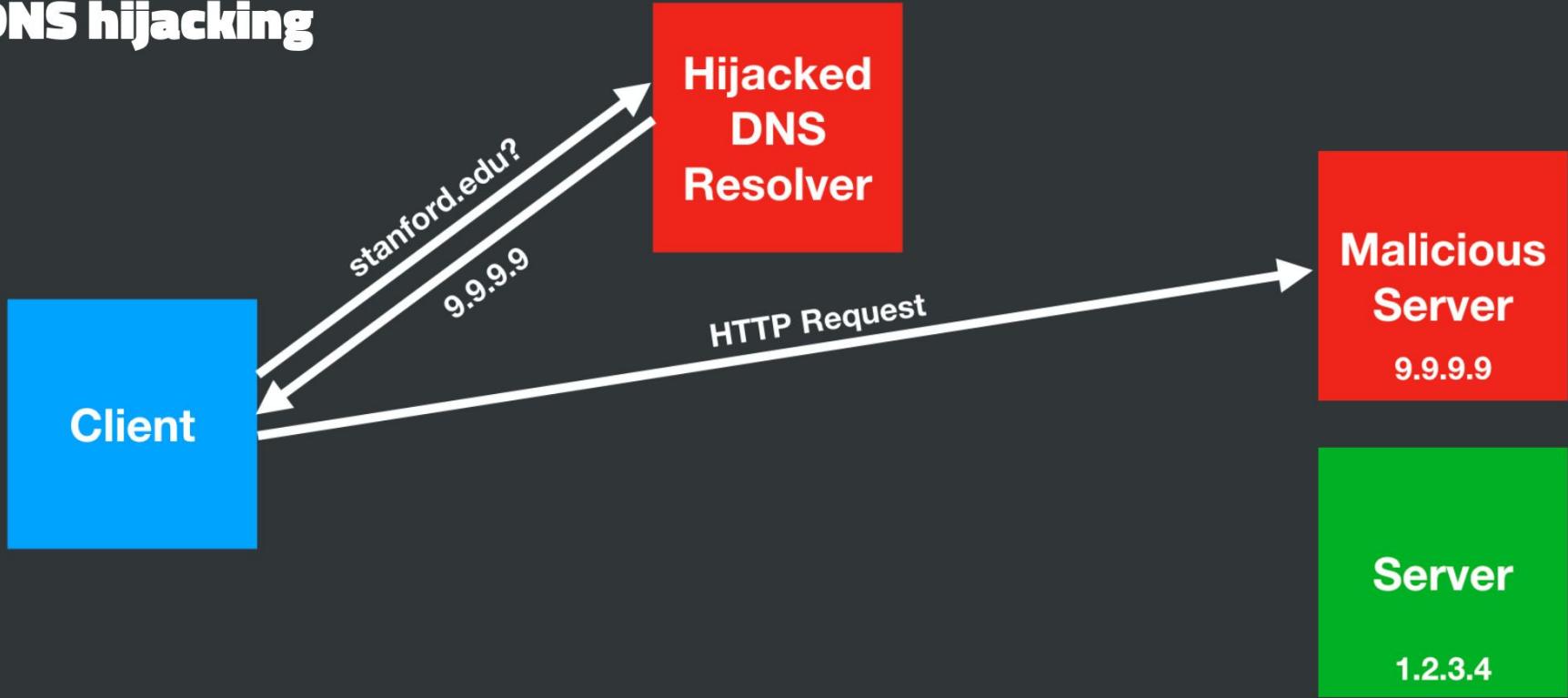
DNS Hijacking

DNS hijacking



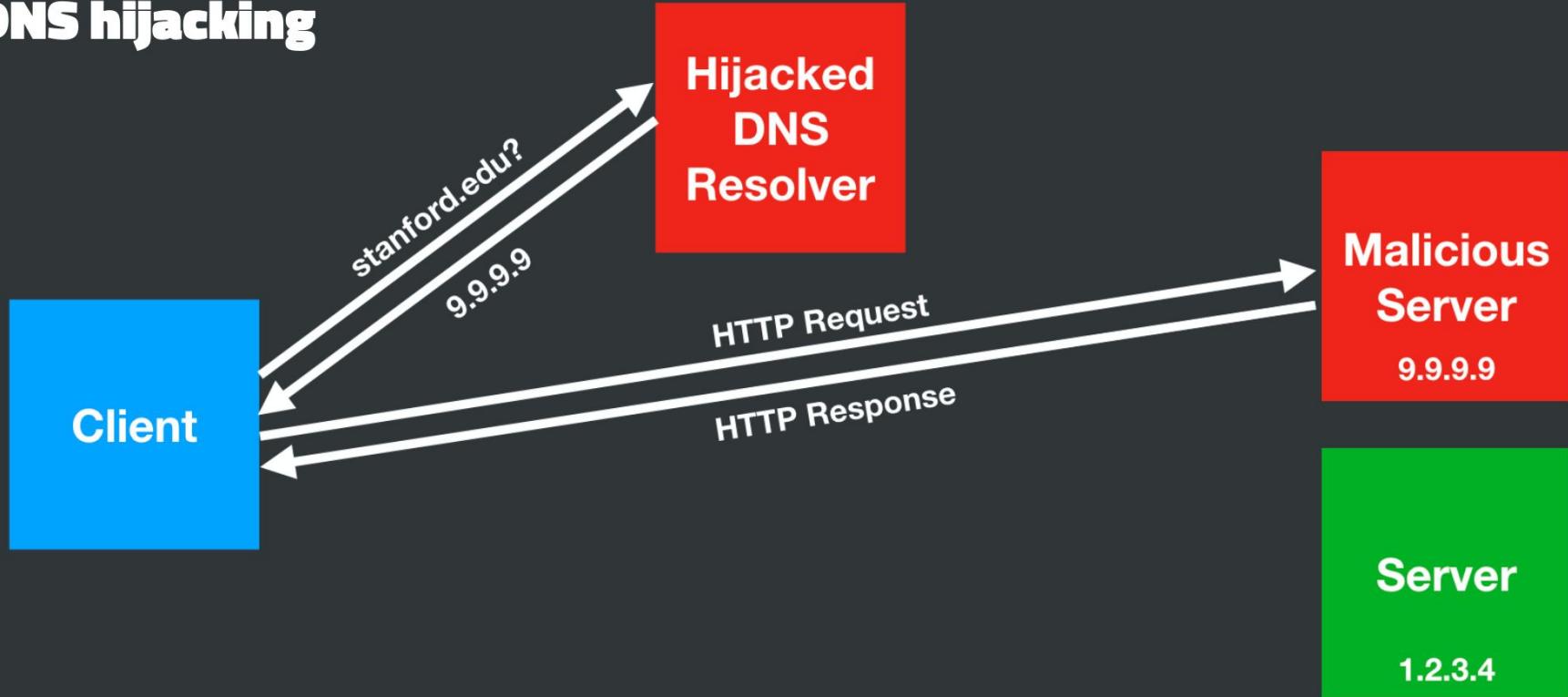
DNS Hijacking

DNS hijacking

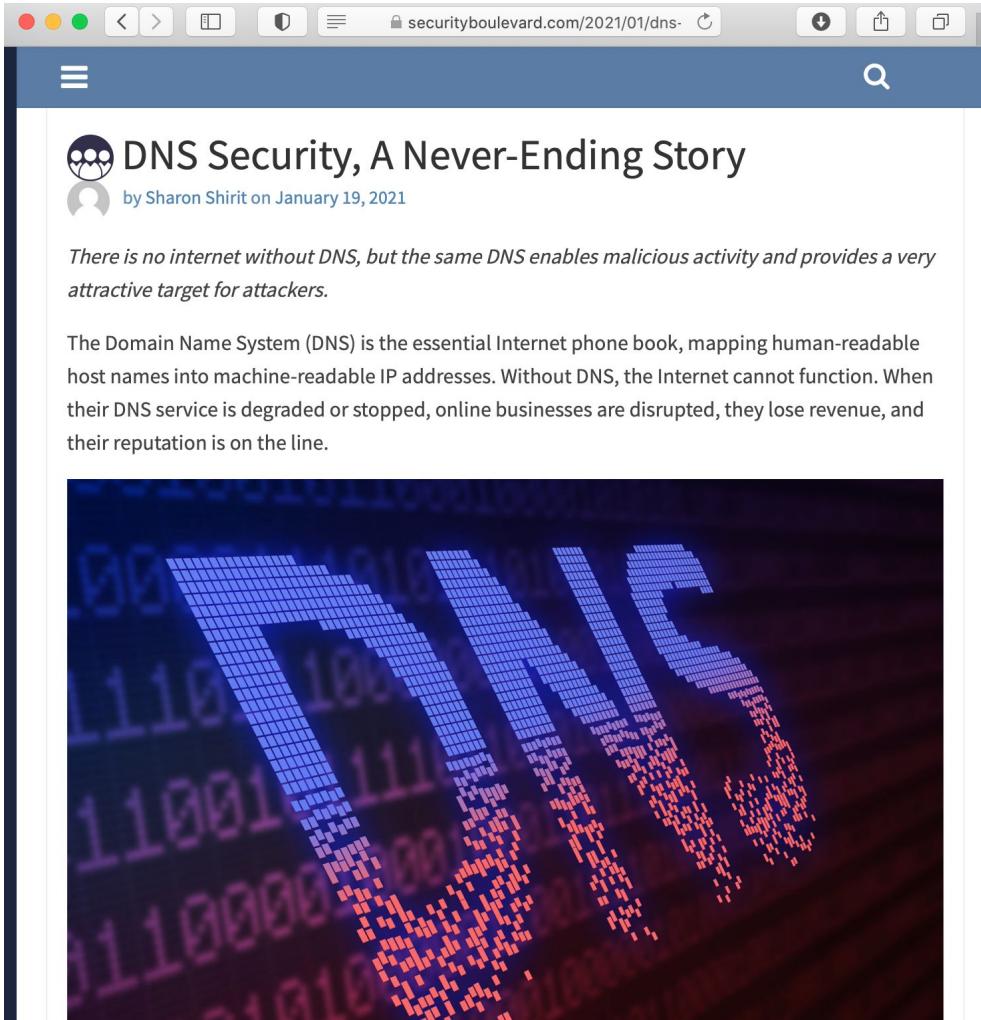


DNS Hijacking

DNS hijacking



DNS attacks won't go away



The screenshot shows a web browser window with the URL `securityboulevard.com/2021/01/dns-security-a-never-ending-story/` in the address bar. The page title is "DNS Security, A Never-Ending Story" with a small icon of three people. Below the title, it says "by Sharon Shirit on January 19, 2021". The main content starts with a quote: "There is no internet without DNS, but the same DNS enables malicious activity and provides a very attractive target for attackers." It then explains that the Domain Name System (DNS) is the essential Internet phone book, mapping human-readable host names into machine-readable IP addresses. Without DNS, the Internet cannot function. When their DNS service is degraded or stopped, online businesses are disrupted, they lose revenue, and their reputation is on the line. At the bottom of the page is a large, abstract graphic composed of binary code (0s and 1s) forming a stylized 'X' shape.

<https://securityboulevard.com/2021/01/dns-security-a-never-ending-story/>

DNS hijack from your ISP

The screenshot shows a web browser window with the URL lookup.t-mobile.com/index.php?origURL=http%3A//questionablecontent.net/&r=&bc=. The page content is a search results page for 'questionablecontent.net' on the T-Mobile website. The results include links for 'Find Answers Fast', 'EVINE Live - Once ShopHQ', 'Compare Prices', 'Q-C - Cheap Prices', 'Questionable Content', 'Questionable Content - definition of Questionable Content by ...', and 'Questionable Content - Questionable Content Wiki'. A sidebar on the right lists 'Related Searches' such as Accstation, Qc, All Comic Book, X Man, Marvel Comic, Marvels, C.c Cosplay, Comic Cartoon, Comic Book Price Guide, The Batman, Find Answers Fast, EVINE Live - Once ShopHQ, Compare Prices, and others.

Make this My Homepage F.A.Q. Customer Support Why am I here?

T-Mobile Search

Web Results

[Find Answers Fast](#)
Search for Information Here Look Up Quick Results Now!
Sponsored by: [www.wow.com/Fast-Answers](#)

[EVINE Live - Once ShopHQ](#)
Huge Selection of Jewelry, Watches, Apparel, Beauty, and Home Decor.
Sponsored by: [www.evine.com](#)

[Compare Prices](#)
Now up to 75% off Compare prices and save up to 75%
Sponsored by: [Compare.salebounty.com](#)

[Q-C - Cheap Prices](#)
See Hot Bargains for Q-C! Update Your Home for Less.
Sponsored by: [www.NexTag.com/Home-and-Garden](#)

[Questionable Content](#)
Centers around an average frustrated 20-something music nerd, his PC and Faye. Includes archive, FAQ and overview.
[questionablecontent.net](#)

[Questionable Content - definition of Questionable Content by ...](#)
By its very nature, it is open sourced -- meaning any one can edit its **contents**, providing **questionable content** that is then taken to be the definitive information ...
[www.thefreedictionary.com/Questionable+Content](#)

[Questionable Content - Questionable Content Wiki](#)
Overview Edit. **Questionable Content** is a slice-of-life webcomic popular for its combination of believable and engaging characters, flights of fancy, and banter. It is ...
[questionablecontent.wikia.com/wiki/Questionable_Content](#)

Related Searches

[Accstation](#)
[Qc](#)
[All Comic Book](#)
[X Man](#)
[Marvel Comic](#)
[Marvels](#)
[C.c Cosplay](#)
[Comic Cartoon](#)
[Comic Book Price Guide](#)
[The Batman](#)
[Find Answers Fast](#)
Search for Information Here Look Up Quick Results Now!
[www.wow.com/Fast-Answers](#)

[EVINE Live - Once ShopHQ](#)
Huge Selection of Jewelry, Watches, Apparel, Beauty, and Home Decor.
[www.evine.com](#)

[Compare Prices](#)
Now up to 75% off Compare prices and save up to 75%
[Compare.salebounty.com](#)

I was going to show an example from cox.net (my home provider), but now it seems like they no longer do this?

DNS Privacy

- Queries are in plaintext
- ISPs have been known to sell this data
- **Pro tip:** Consider switching your DNS settings to Cloudflare (1.1.1.1) or another provider with a good privacy policy

DNS over HTTPS

The screenshot shows a web browser window with the following details:

- Address Bar:** developers.cloudflare.com/1.1.1/dns-over-https
- Page Title:** Cloudflare Docs
- Search Bar:** Search 1.1.1 docs...
- Header Buttons:** Standard browser controls (refresh, back, forward, search).
- Left Sidebar (Table of Contents):**
 - 1. 1.1.1
 - Welcome
 - ▶ Setting up 1.1.1.1
 - What is 1.1.1.1?
 - ▶ 1.1.1.1 for Families
 - ▶ DNS over HTTPS
 - Making requests
 - Using JSON
 - Configure your browser to use DNS over HTTPS
 - Running a DNS over HTTPS client
 - Using DNS Wireformat
 - The nitty gritty
 - DNS over TLS
 - For IPv6-only networks
 - ▶ Fun with DNS
 - ▶ Privacy
 - Terms
- Main Content Area:**

DNS over HTTPS

Even if you are visiting a site using HTTPS, your DNS query is sent over an unencrypted connection. That means that even if you are browsing <https://cloudflare.com>, anyone listening to packets on the network knows you are attempting to visit cloudflare.com.

The second problem with unencrypted DNS is that it is easy for a Man-In-The-Middle to change DNS answers to route unsuspecting visitors to their phishing, malware or surveillance site. DNSSEC solves this problem as well by providing a mechanism to check the validity of a DNS answer, but only a single-digit percentage of domains use DNSSEC.

To combat this problem, Cloudflare offers DNS resolution over an HTTPS endpoint. If you build a mobile application, browser, operating system, IoT device or router, you can choose for your users to use the DNS over HTTPS endpoint instead of sending DNS queries over plaintext for increased security and privacy of your users.
- Page Footer:**

Edit on GitHub · Updated 1 day ago

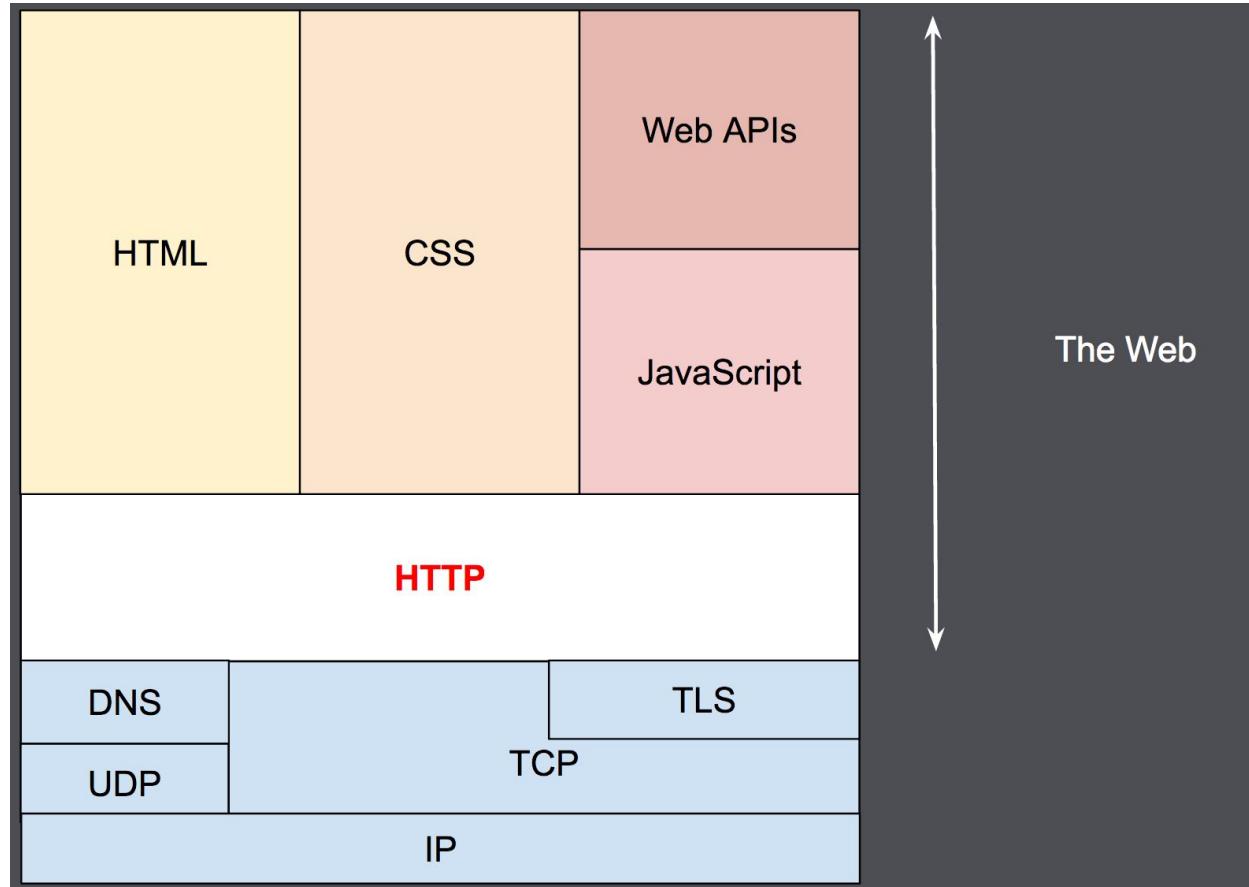
<https://developers.cloudflare.com/1.1.1/dns-over-https>

More curl... curl is important...

```
canis:/home/mln % curl -v -I https://www.odu.edu/
*   Trying 128.82.112.29 ...
* TCP_NODELAY set
* Connected to www.odu.edu ( 128.82.112.29 ) port 443 (#0)
* ALPN, offering h2
* ALPN, offering http/1.1
* successfully set certificate verify locations:
*   CAfile: /etc/ssl/certs/ca-certificates.crt
*   CApth: /etc/ssl/certs
* TLSv1.3 (OUT), TLS handshake, Client hello (1):
* TLSv1.3 (IN), TLS handshake, Server hello (2):
* TLSv1.2 (IN), TLS handshake, Certificate (11):
* TLSv1.2 (IN), TLS handshake, Server key exchange (12):
* TLSv1.2 (IN), TLS handshake, Server finished (14):
* TLSv1.2 (OUT), TLS handshake, Client key exchange (16):
* TLSv1.2 (OUT), TLS handshake, Finished (20):
* TLSv1.2 (OUT), TLS handshake, Finished (20):
* SSL connection using TLSv1.2 / ECDHE-RSA-AES128-GCM-SHA256
* ALPN, server did not agree to a protocol
* Server certificate:
*   subject: serialNumber=232982; jurisdictionC=US; jurisdictionST=Virginia; businessCategory=Government Entity;
C=US; postalCode=23529; ST=VA; L=Norfolk; street=4600 Elkhorn Ave; O=Old Dominion University; OU=ITS; OU=COMODO
EV Multi-Domain SSL; CN=www.odu.edu
*   start date: Jun  5 00:00:00 2019 GMT
*   expire date: Jun  4 23:59:59 2021 GMT
*   subjectAltName: host "www.odu.edu" matched cert's "www.odu.edu"
*   issuer: C=GB; ST=Greater Manchester; L=Salford; O=COMODO CA Limited; CN=COMODO RSA Extended Validation Secure
Server CA
*   SSL certificate verify ok.
> HEAD / HTTP/1.1
> Host: www.odu.edu
> User-Agent: curl/7.58.0
> Accept: */*
>
< HTTP/1.1 200 OK
HTTP/1.1 200 OK
< Date: Thu, 28 Jan 2021 16:52:04 GMT
Date: Thu, 28 Jan 2021 16:52:04 GMT
< Server: Apache/2.4.6 (Red Hat Enterprise Linux)
Server: Apache/2.4.6 (Red Hat Enterprise Linux)
< Vary: Host
Vary: Host
< Accept-Ranges: bytes
Accept-Ranges: bytes
< Connection: close
Connection: close
```

TCP
DNS
TLS
HTTP Request
HTTP Response

We now have a better definition of what is “the web”



<https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview>

HTTP Proxies

- Can cache content
- Can block content (e.g., malware, adult content)
- Can modify content
- Can sit in front of many servers ("reverse proxy")

HTTP with a proxy server



59 Feross Aboukhadijeh

HTTP with a proxy server



60 Feross Aboukhadijeh

40

HTTP with a proxy server



61 Feross Aboukhadijeh

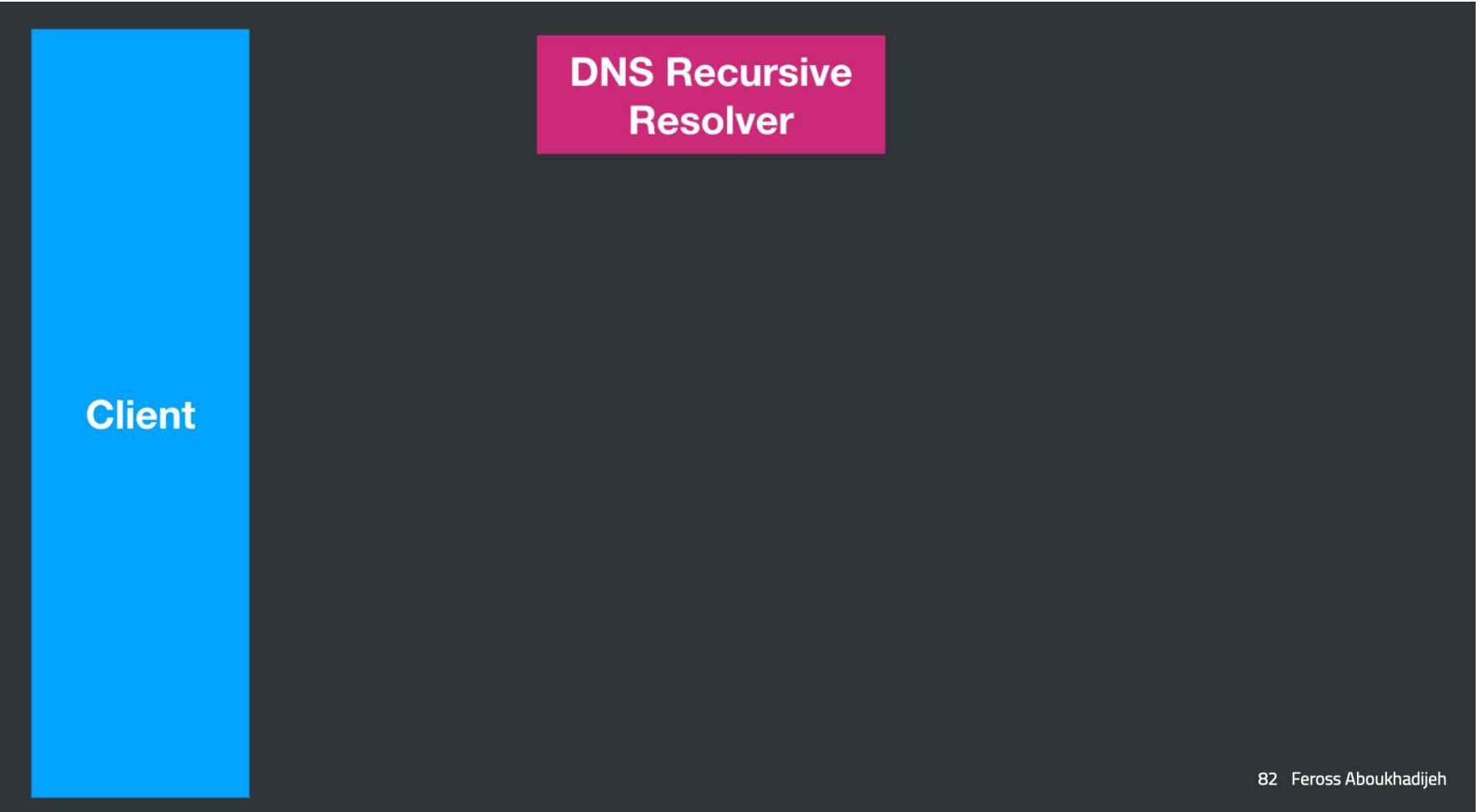
HTTP with a proxy server



62 Feross Aboukhadijeh

Web stack

Putting it together

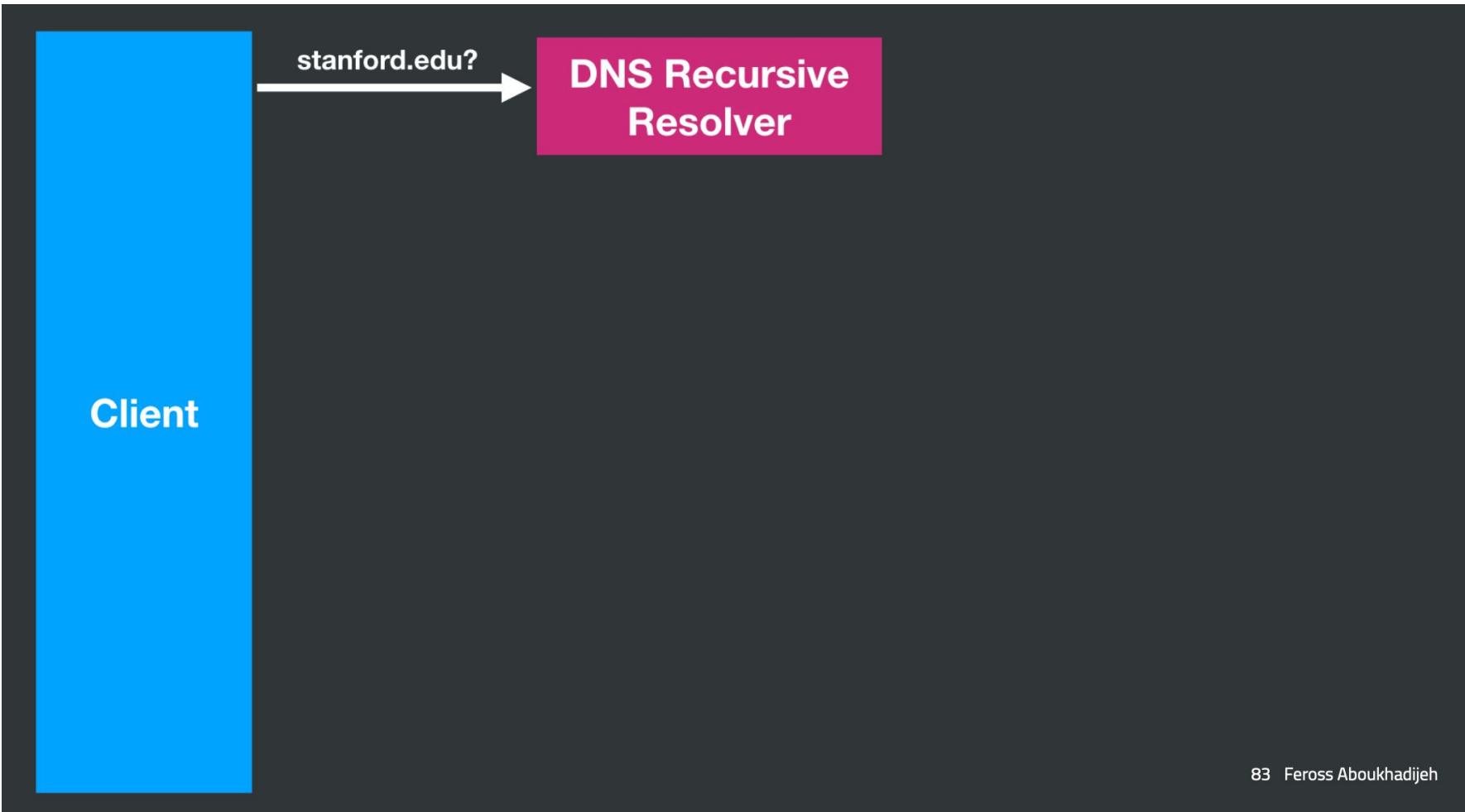


DNS Recursive
Resolver

Client

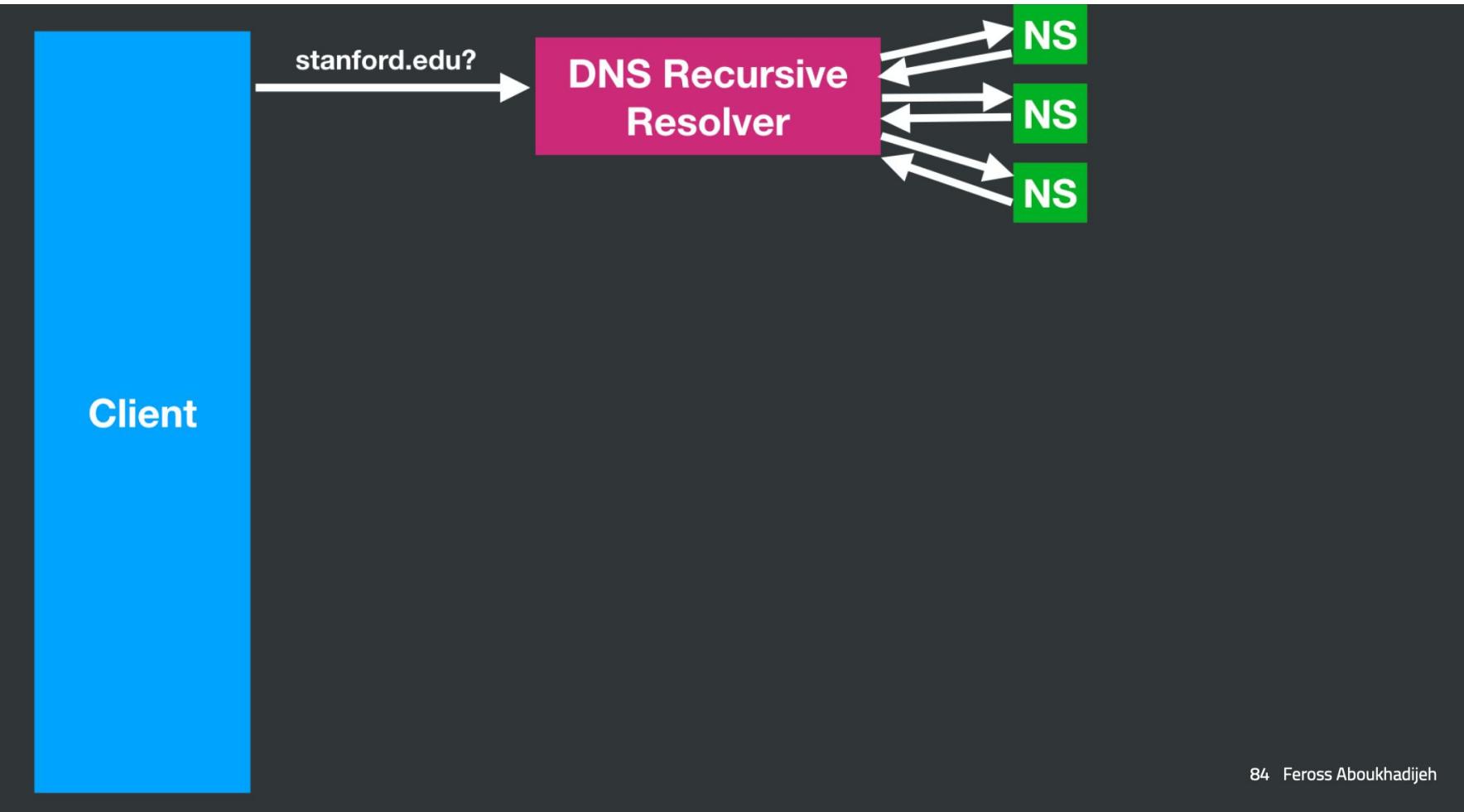
82 Feross Aboukhadijeh

Putting it together



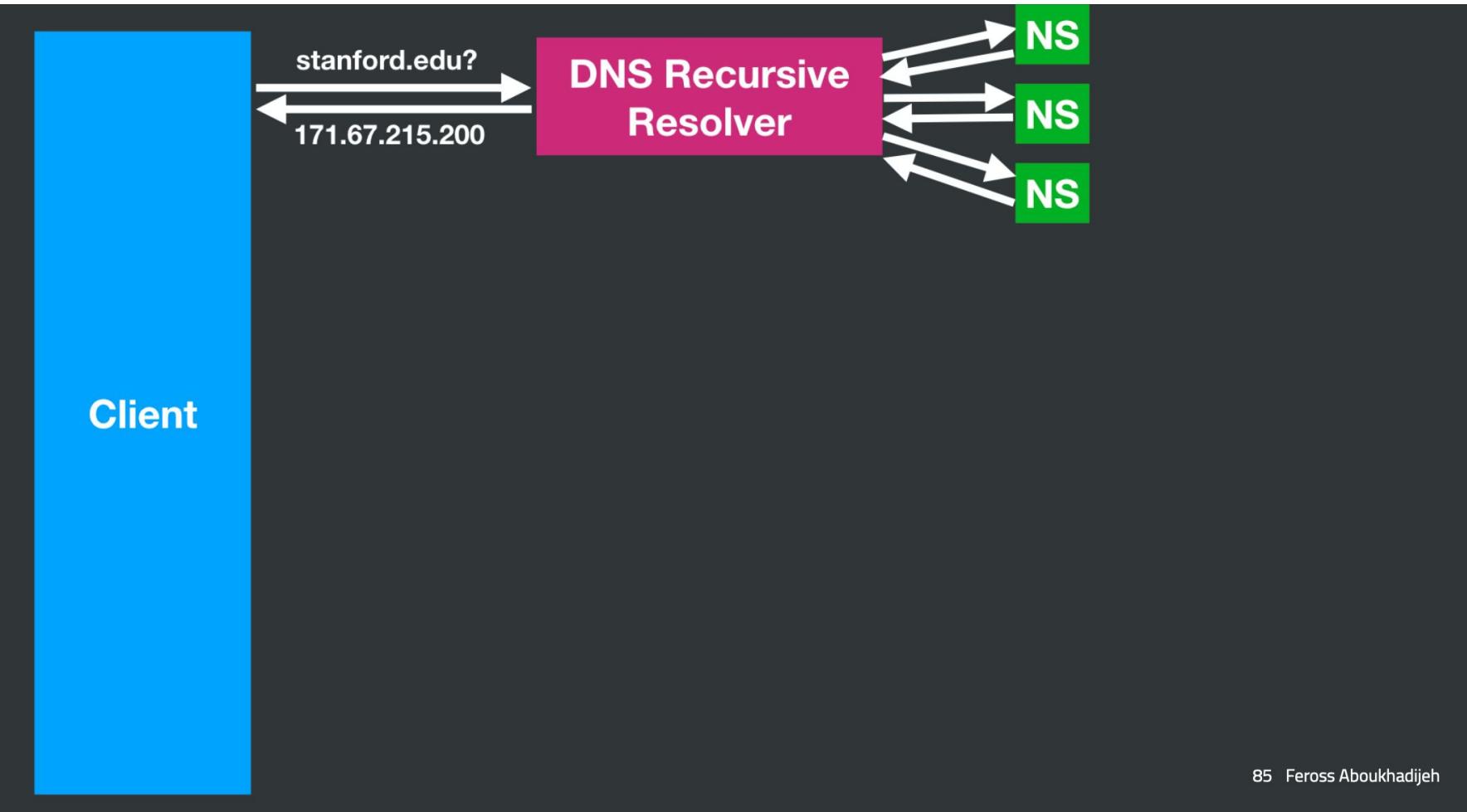
83 Feross Aboukhadijeh

Putting it together



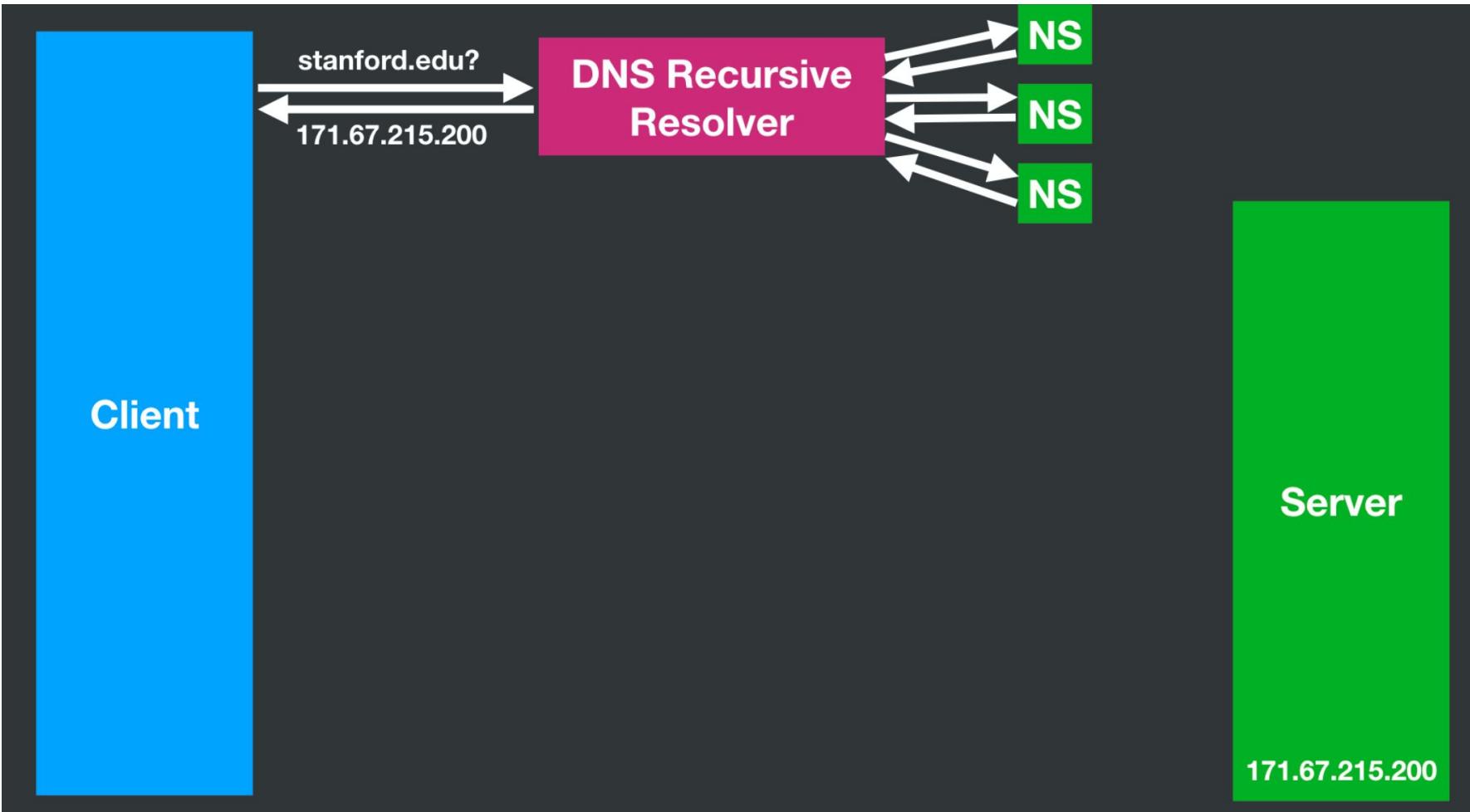
84 Feross Aboukhadijeh

Putting it together

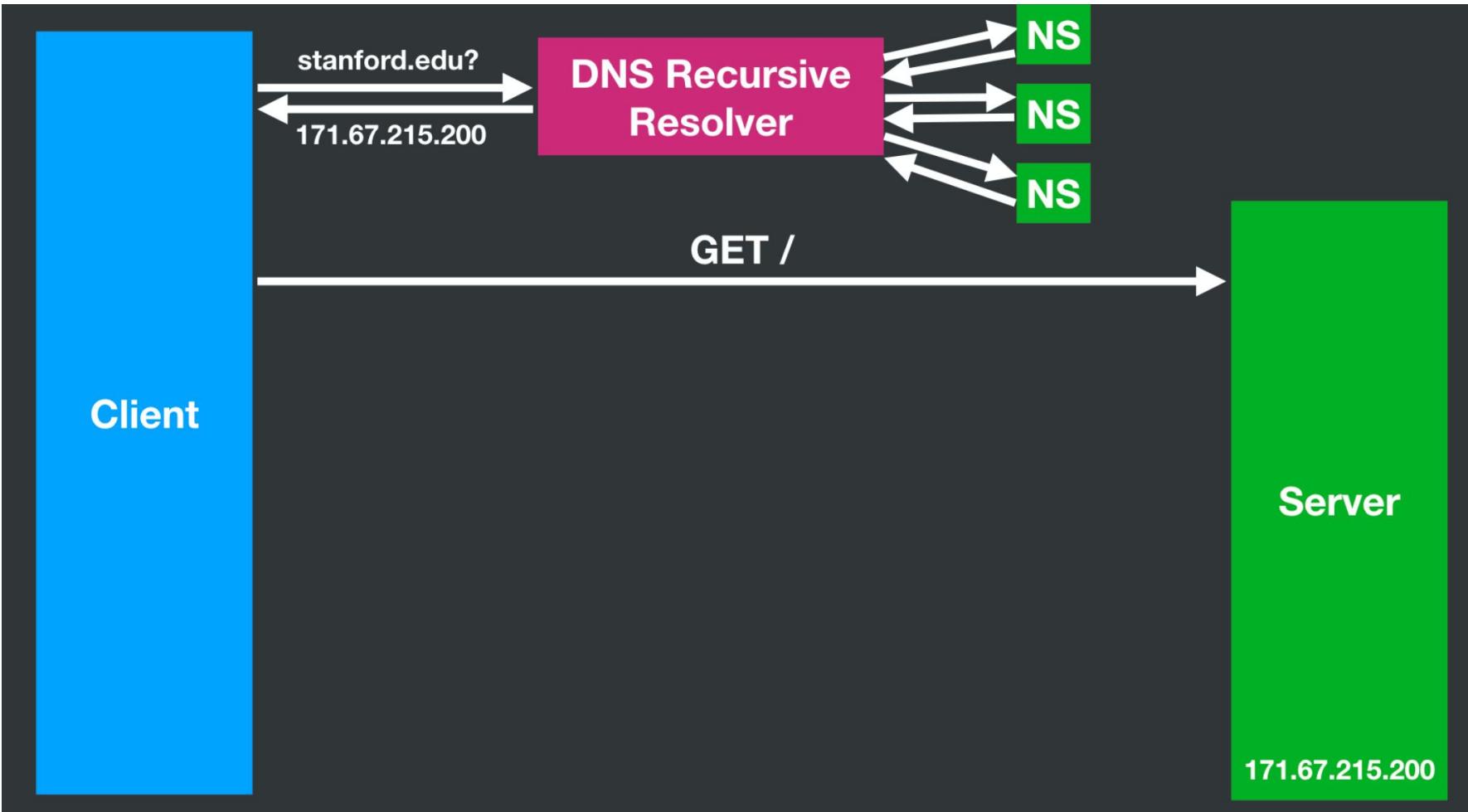


85 Feross Aboukhadijeh

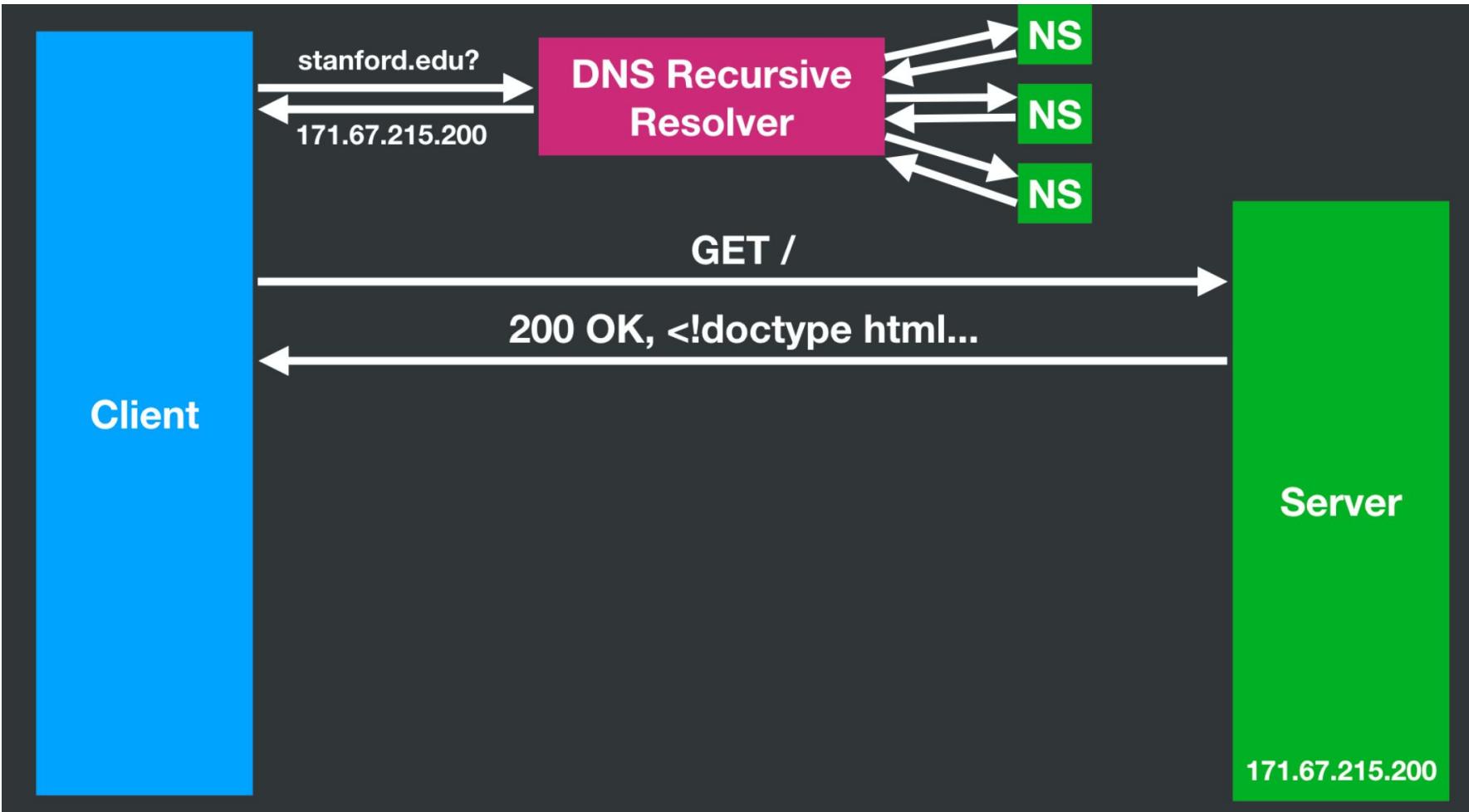
Putting it together



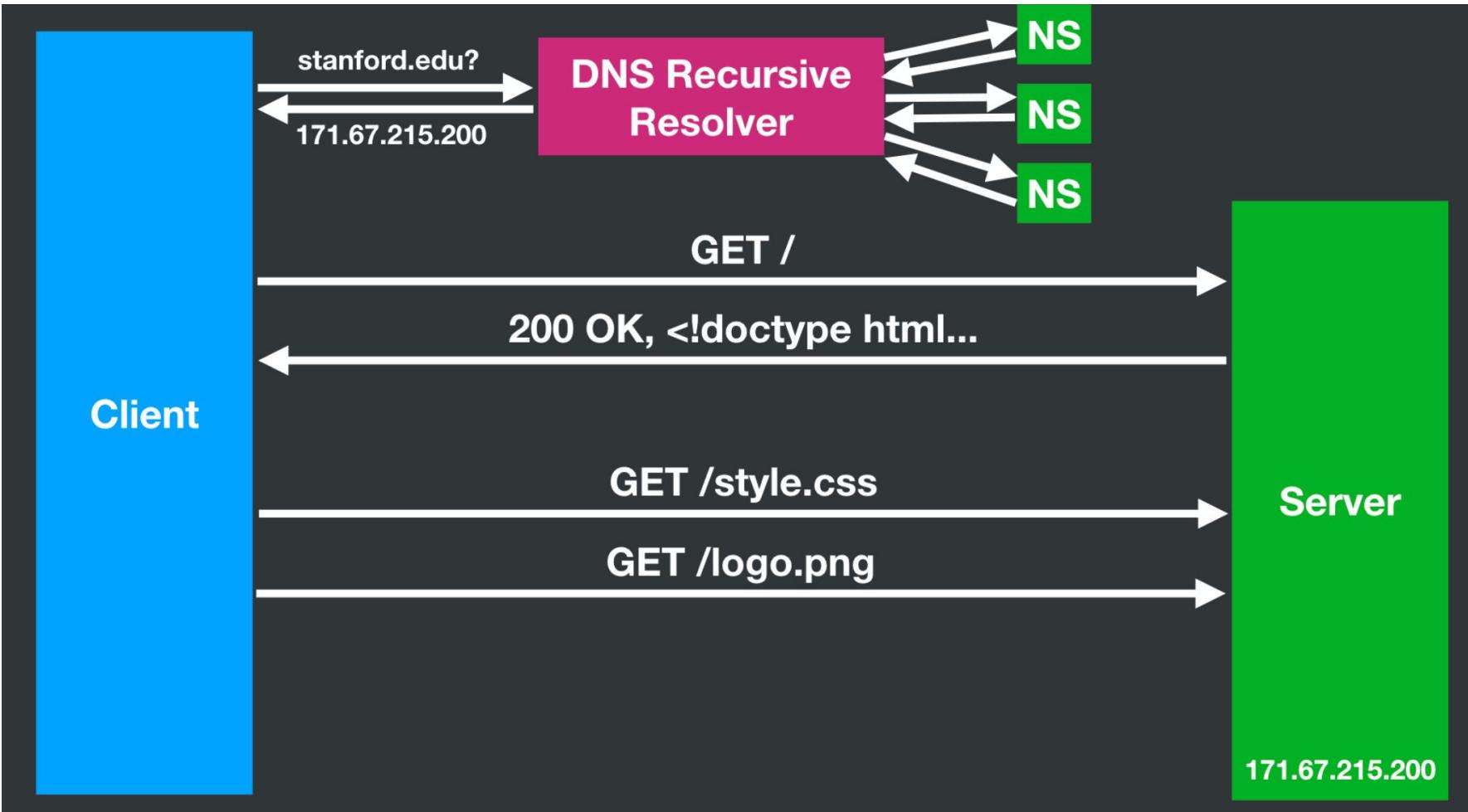
Putting it together



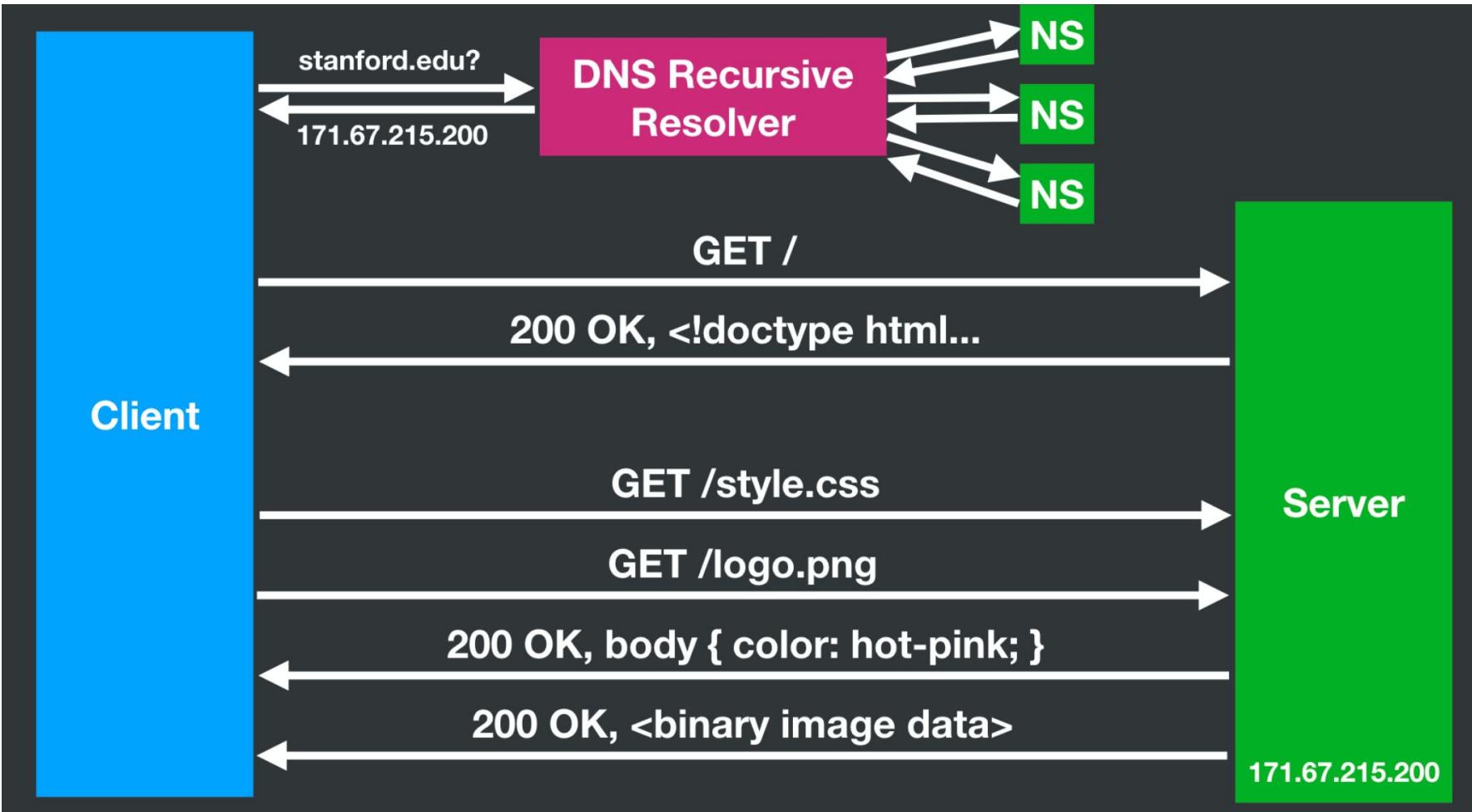
Putting it together



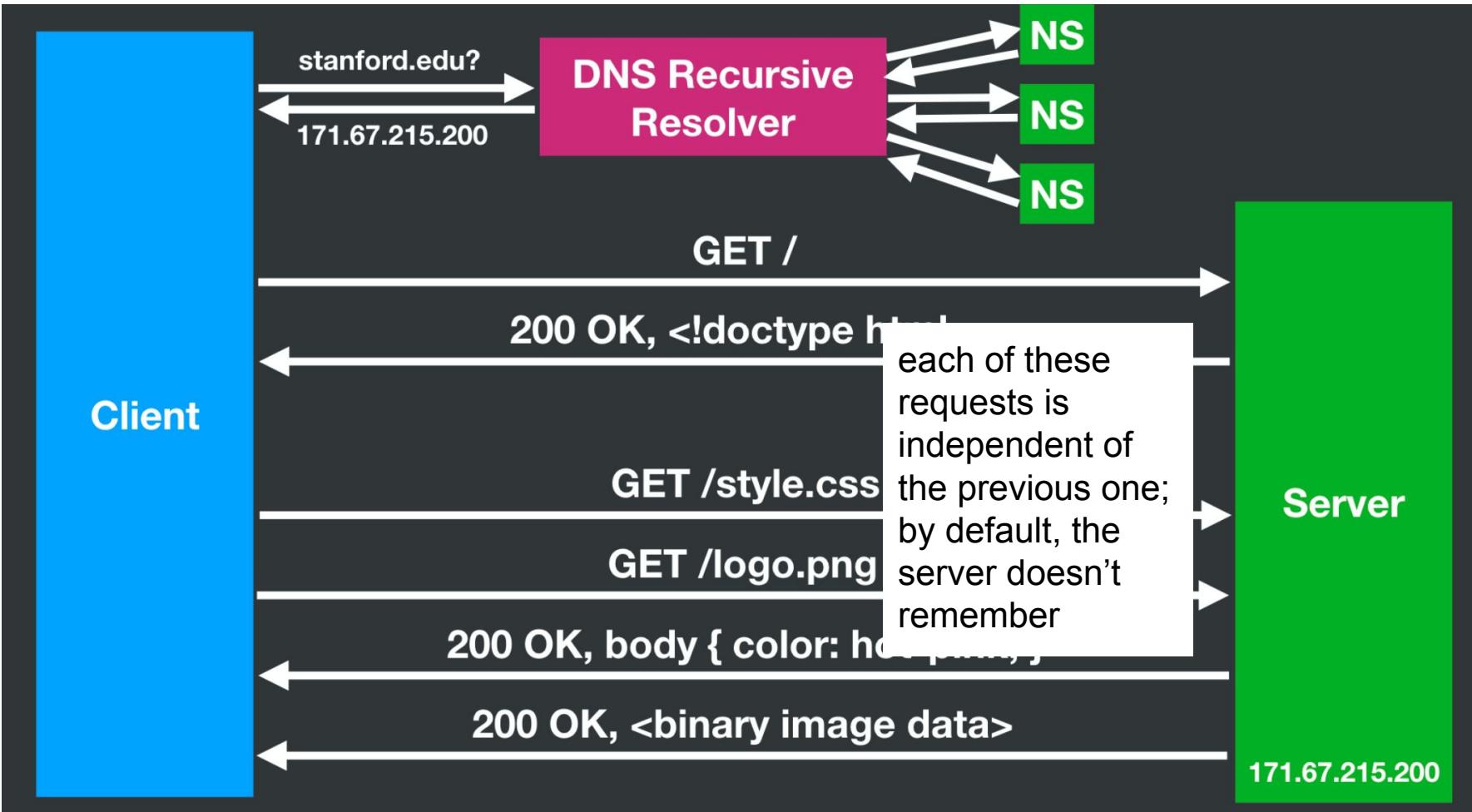
Putting it together



Putting it together

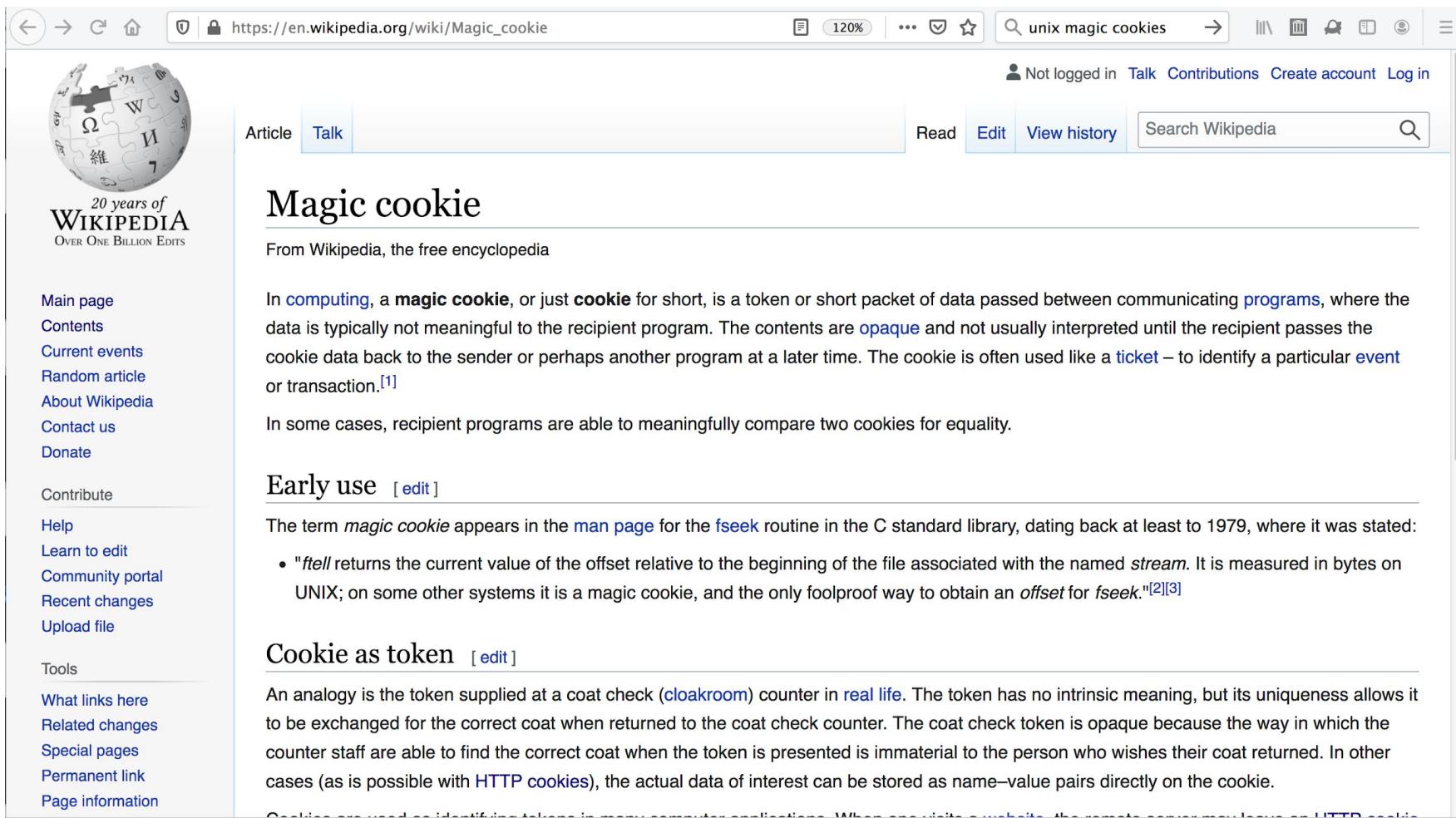


HTTP is *stateless*



Cookies

Why “cookies”?



The screenshot shows a web browser displaying the Wikipedia page for "Magic cookie". The URL in the address bar is https://en.wikipedia.org/wiki/Magic_cookie. The page content is as follows:

Magic cookie

From Wikipedia, the free encyclopedia

In computing, a **magic cookie**, or just **cookie** for short, is a token or short packet of data passed between communicating [programs](#), where the data is typically not meaningful to the recipient program. The contents are [opaque](#) and not usually interpreted until the recipient passes the cookie data back to the sender or perhaps another program at a later time. The cookie is often used like a [ticket](#) – to identify a particular [event](#) or transaction.^[1]

In some cases, recipient programs are able to meaningfully compare two cookies for equality.

Early use [edit]

The term *magic cookie* appears in the [man page](#) for the `fseek` routine in the C standard library, dating back at least to 1979, where it was stated:

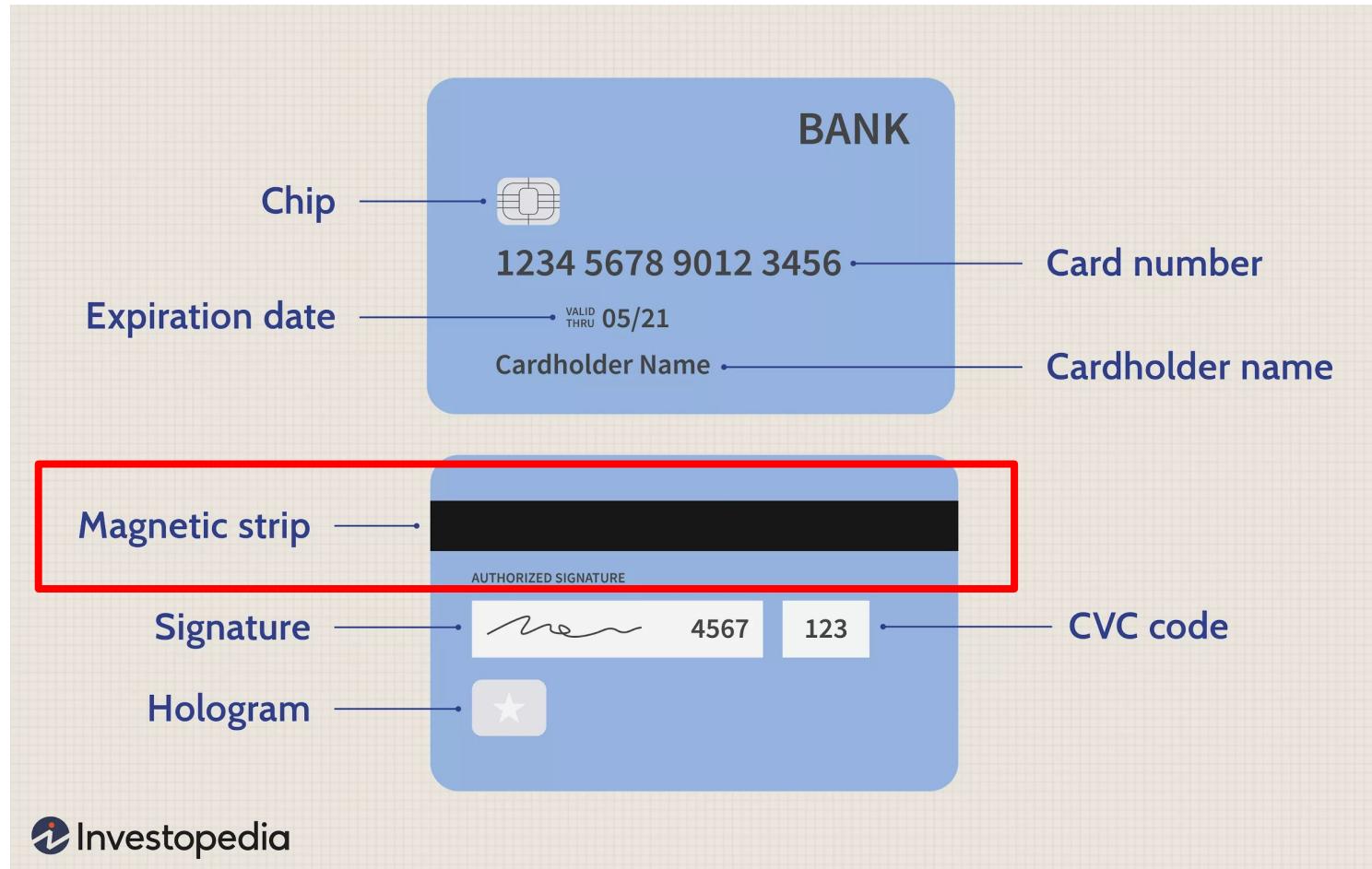
- "`ftell` returns the current value of the offset relative to the beginning of the file associated with the named *stream*. It is measured in bytes on UNIX; on some other systems it is a magic cookie, and the only foolproof way to obtain an *offset* for `fseek`."^{[2][3]}

Cookie as token [edit]

An analogy is the token supplied at a coat check ([cloakroom](#)) counter in [real life](#). The token has no intrinsic meaning, but its uniqueness allows it to be exchanged for the correct coat when returned to the coat check counter. The coat check token is opaque because the way in which the counter staff are able to find the correct coat when the token is presented is immaterial to the person who wishes their coat returned. In other cases (as is possible with [HTTP cookies](#)), the actual data of interest can be stored as name–value pairs directly on the cookie.

https://en.wikipedia.org/wiki/Magic_cookie

Real world opaque data structures



<https://www.investopedia.com/terms/c/creditcard.asp>

Cookies in the wild

```
% curl -I https://www.odu.edu/
HTTP/1.1 200 OK
Date: Thu, 28 Jan 2021 04:30:58 GMT
Server: Apache/2.4.6 (Red Hat Enterprise Linux)
Vary: Host
Accept-Ranges: bytes
Connection: close
Content-Type: text/html; charset=UTF-8
Set-Cookie: BIGipServerWEB_HTTPS_PROD.app~WEB_HTTPS_PROD_pool_int=rd741o0000000000000000ffff8052619eo80;
path=/; Httponly; Secure
% curl -I https://www.google.com/
HTTP/2 200
content-type: text/html; charset=ISO-8859-1
p3p: CP="This is not a P3P policy! See g.co/p3phelp for more info."
date: Thu, 28 Jan 2021 04:34:47 GMT
server: gws
x-xss-protection: 0
x-frame-options: SAMEORIGIN
expires: Thu, 28 Jan 2021 04:34:47 GMT
cache-control: private
set-cookie: 1P_JAR=2021-01-28-04; expires=Sat, 27-Feb-2021 04:34:47 GMT; path=/; domain=.google.com; Secure
set-cookie:
NID=208=cOFUezC7N6Pxds_6eh3UEYUw17ON0B78ja0GYq8ZIGvg8dq1fuS1NZS5kCJK0QEtcisRWceu0ZFPHS7WMd-2xcKsF_3BWgdOUcKBG
NW9mcwH6_1WWnaoTY4k-ugIG_JzvtQCSMr5Naonq-gkTf6KCmYuy_AwJenAiEN89KNC-TY; expires=Fri, 30-Jul-2021 04:34:47
GMT; path=/; domain=.google.com; HttpOnly
alt-svc: h3-29=:443"; ma=2592000,h3-T051=:443"; ma=2592000,h3-Q050=:443"; ma=2592000,h3-Q046=:443";
ma=2592000,h3-Q043=:443"; ma=2592000,quic=:443"; ma=2592000; v="46,43"
```

Cookies are opaque: don't parse them, just send them back

```
% curl -I https://www.google.com/
HTTP/2 200
content-type: text/html; charset=ISO-8859-1
p3p: CP="This is not a P3P policy! See g.co/p3phelp for more info."
date: Thu, 28 Jan 2021 04:48:50 GMT
server: gws
x-xss-protection: 0
x-frame-options: SAMEORIGIN
expires: Thu, 28 Jan 2021 04:48:50 GMT
cache-control: private
set-cookie: 1P_JAR=2021-01-28-04; expires=Sat, 27-Feb-2021 04:48:50 GMT; path=/; domain=.google.com; Secure
set-cookie:
NID=208=FZPAg05u4KrLWrvHY6rT-g47xszzMnSuF1IUZBMQkKGGzcXcNKB5cWk6DrnMlijEqlwXCx5-D8K486Db6hwf_mRTpR2Za5ho2MyGQWut_40v
Qxgnh3eKN28a5Yw6_LwysvSIaaTz9s6lyhzK9J3b3Vmhu1VkaopJo57XBcccXFS8; expires=Fri, 30-Jul-2021 04:48:50 GMT; path=/;
domain=.google.com; HttpOnly
alt-svc: h3-29=":443"; ma=2592000,h3-T051=":443"; ma=2592000,h3-Q050=":443"; ma=2592000,h3-Q046=":443";
ma=2592000,h3-Q043=":443"; ma=2592000,quic=":443"; ma=2592000; v="46,43"
mln2@Michaels-MacBook-Air ~ % curl -I -H "Cookie: 1P_JAR=2021-01-28-04" -H "Cookie:
NID=208=FZPAg05u4KrLWrvHY6rT-g47xszzMnSuF1IUZBMQkKGGzcXcNKB5cWk6DrnMlijEqlwXCx5-D8K486Db6hwf_mRTpR2Za5ho2MyGQWut_40v
Qxgnh3eKN28a5Yw6_LwysvSIaaTz9s6lyhzK9J3b3Vmhu1VkaopJo57XBcccXFS8" https://www.google.com/
HTTP/2 200
content-type: text/html; charset=ISO-8859-1
date: Thu, 28 Jan 2021 04:49:29 GMT
server: gws
x-xss-protection: 0
x-frame-options: SAMEORIGIN
expires: Thu, 28 Jan 2021 04:49:29 GMT
cache-control: private
set-cookie: 1P_JAR=2021-01-28-04; expires=Sat, 27-Feb-2021 04:49:29 GMT; path=/; domain=.google.com; Secure
alt-svc: h3-29=":443"; ma=2592000,h3-T051=":443"; ma=2592000,h3-Q050=":443"; ma=2592000,h3-Q046=":443";
ma=2592000,h3-Q043=":443"; ma=2592000,quic=":443"; ma=2592000; v="46,43"
```

see also: “--cookie” and “--cookie-jar”

Finding your cookies

The screenshot shows a web browser window for www.odu.edu. The page content includes the Old Dominion University logo and a sidebar with sections like 'About ODU', 'APP', and 'News'. Overlaid on the page is the 'Web Inspector' tool, specifically the 'Document' tab. A red box highlights the 'Request' section, which shows the following HTTP request headers:

```
GET /HTTP/1.1
Cookie: _ga=GA1.2.1649686035.1611808964; _gcl_au=1.712943533.1611808962; _gid=GA1.2.583076118.1611808964; BIGipServerWEB_HTTPS_PROD.app~WEB_HTTPS_PROD_pool_int=r74100000000000000000000ffff8052619e080; CASTGC-PROD_SS_O=61819F0147DA7529628E9B04293739D1A51BD48F0DF83100029CB52539AFCCEBC8835DB898026CB1A0D461DCDED1EE4AE74D33295B208659FFDF7A4A1018DDE4; experimentation_subject_id=ljgyYzQyOTdmLWI5MTItNDEwNy1iM2ExLtk1N2FmN2QyYzRiOCI%3D--849900f804f9d88337a08a5572afdf274fec72400
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Host: www.odu.edu
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/605.1.15 (KHTML, like Gecko) Version/14.0 Safari/605.1.1
Accept-Language: en-us
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
```

The 'Response' section shows the following headers:

```
HTTP/1.1 200 OK
Content-Type: text/html; charset=UTF-8
Connection: close
Accept-Ranges: bytes
Date: Thu, 28 Jan 2021 04:43:56 GMT
```

At the bottom of the browser window, there are navigation buttons for 'Screenshot' and other tabs, along with a status bar showing 'Auto — www.odu.edu'.

Cookies can implement *sessions*

Cookies allow state to be passed between the user-agent and server

- Cookies are used by the server to implement sessions
- Goal: Server keeps a set of data related to a user's current "browsing session"
- Examples
 - Logins
 - Shopping carts
 - User tracking

Session skeleton

First HTTP request:

```
POST /login HTTP/1.1
```

```
Host: example.com
```

```
username=alice&password=password
```

HTTP response:

```
HTTP/1.1 200 OK
```

```
Set-Cookie: username=alice
```

```
Date: Tue, 24 Sep 2019 20:30:00 GMT
```

```
<!DOCTYPE html ...
```

note: in this example and in class demos, the cookies are readable. in real life, they're *opaque* and only make sense to the server.

All future HTTP requests:

```
GET /page.html HTTP/1.1
```

```
Host: example.com
```

```
Cookie: username=alice;
```

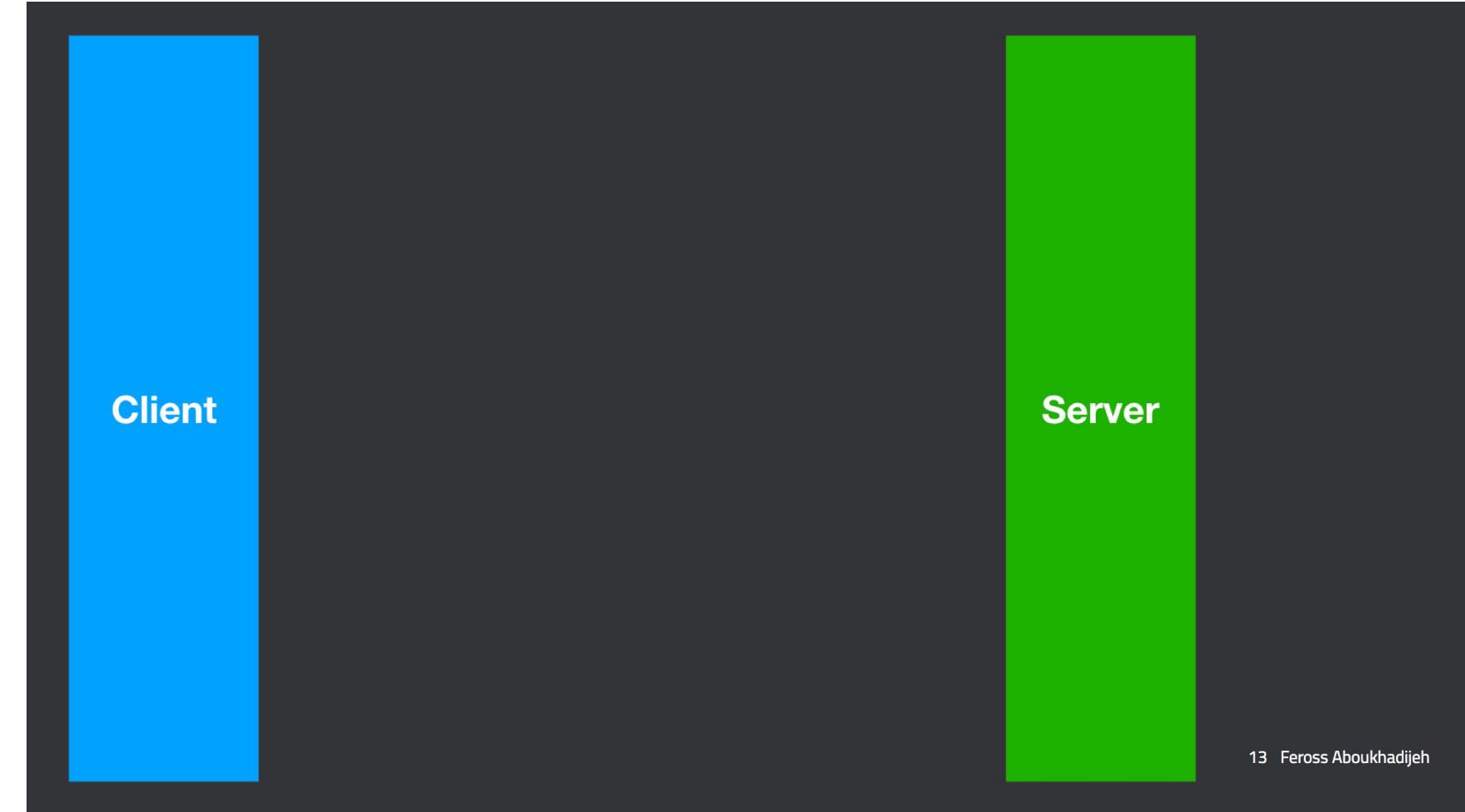
Ambient Authority

- Access control - Regulate who can view resources or take actions
- Ambient authority - Access control based on a global and persistent property of the requester
- The alternative is explicit authorization valid only for a specific action
- There are four types of ambient authority on the web
 - Cookies - most common, most versatile method
 - IP checking - library checks if you're on odu.edu network
 - Built-in HTTP authentication - rarely used
 - Client certificates - rarely used

Quick primer: signature schemes

- Triple of algorithms (G , S , V)
 - $G() \rightarrow (pk, sk)$ - generator returns public key and secret key
 - $S(sk, x) \rightarrow t$ - signing returns a tag t for input x
 - $V(pk, x, t) \rightarrow \text{accept|reject}$ - checks validity of tag t for given input x
- Correctness property
 - $V(pk, x, S(sk, x)) = \text{accept}$ should always be true
- Security property
 - $V(pk, x, t) = \text{accept}$ should almost never be true when x and t are chosen by the attacker

Login and session

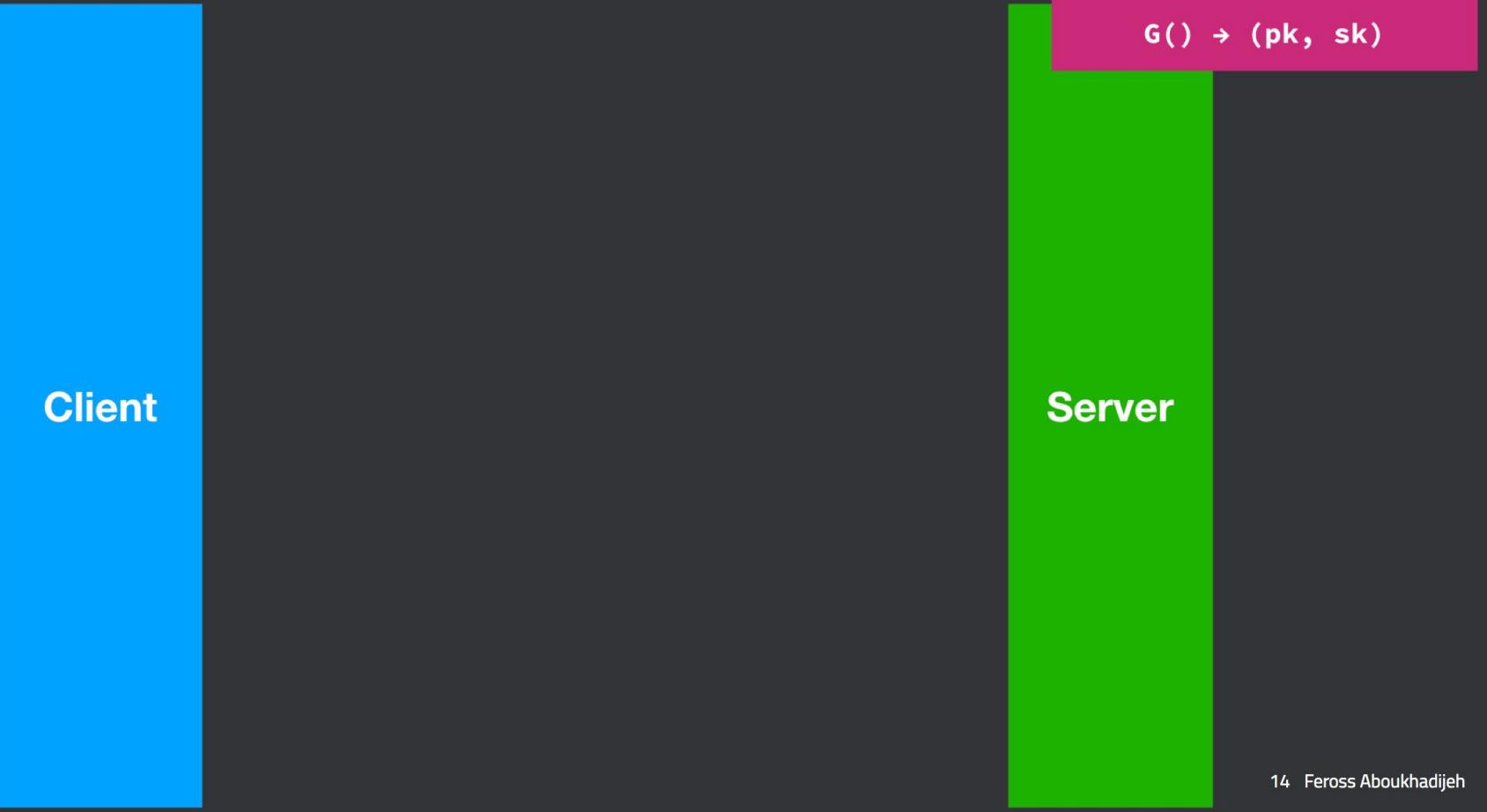


Client

Server

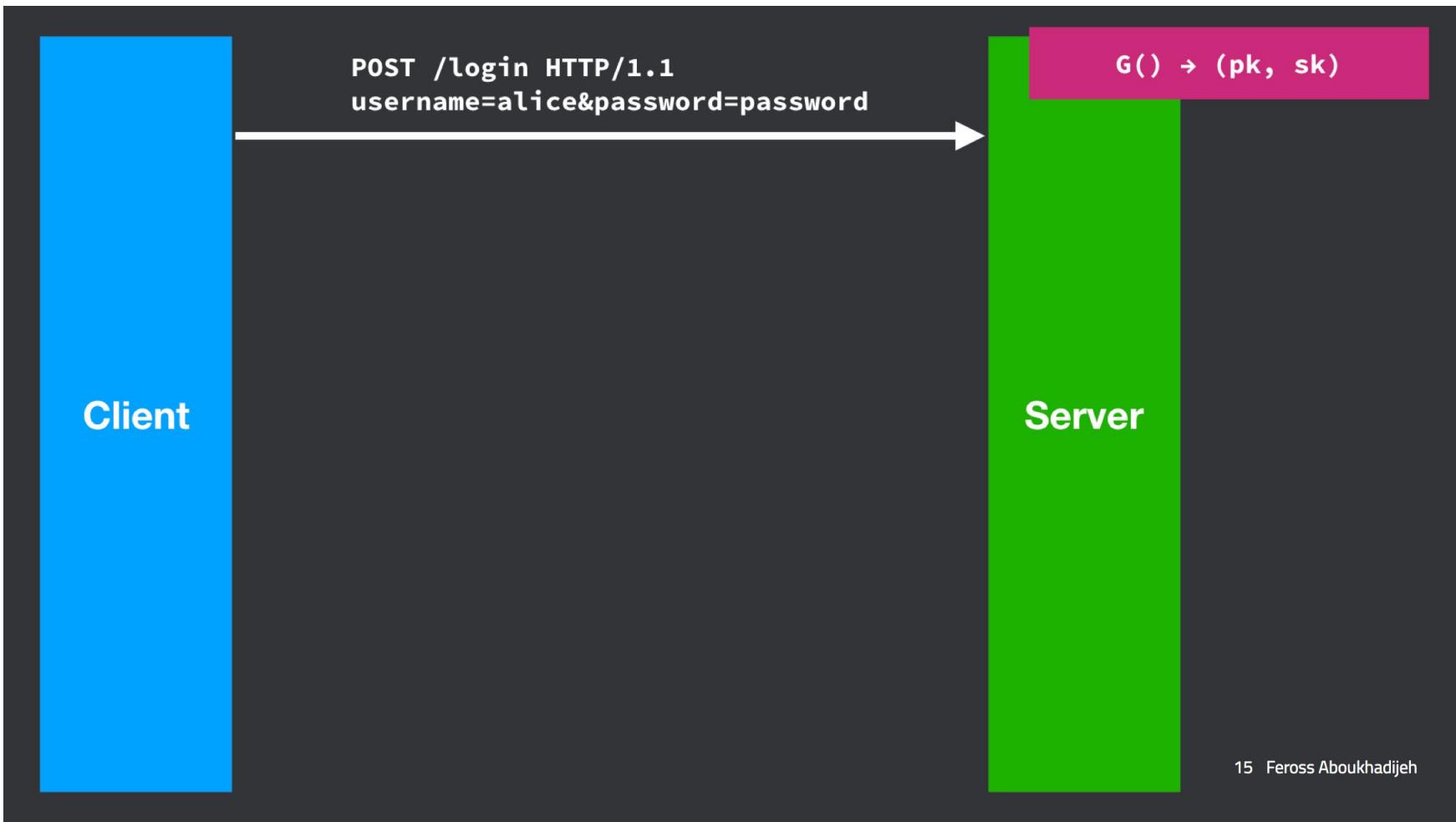
13 Feross Aboukhadijeh

Login and session



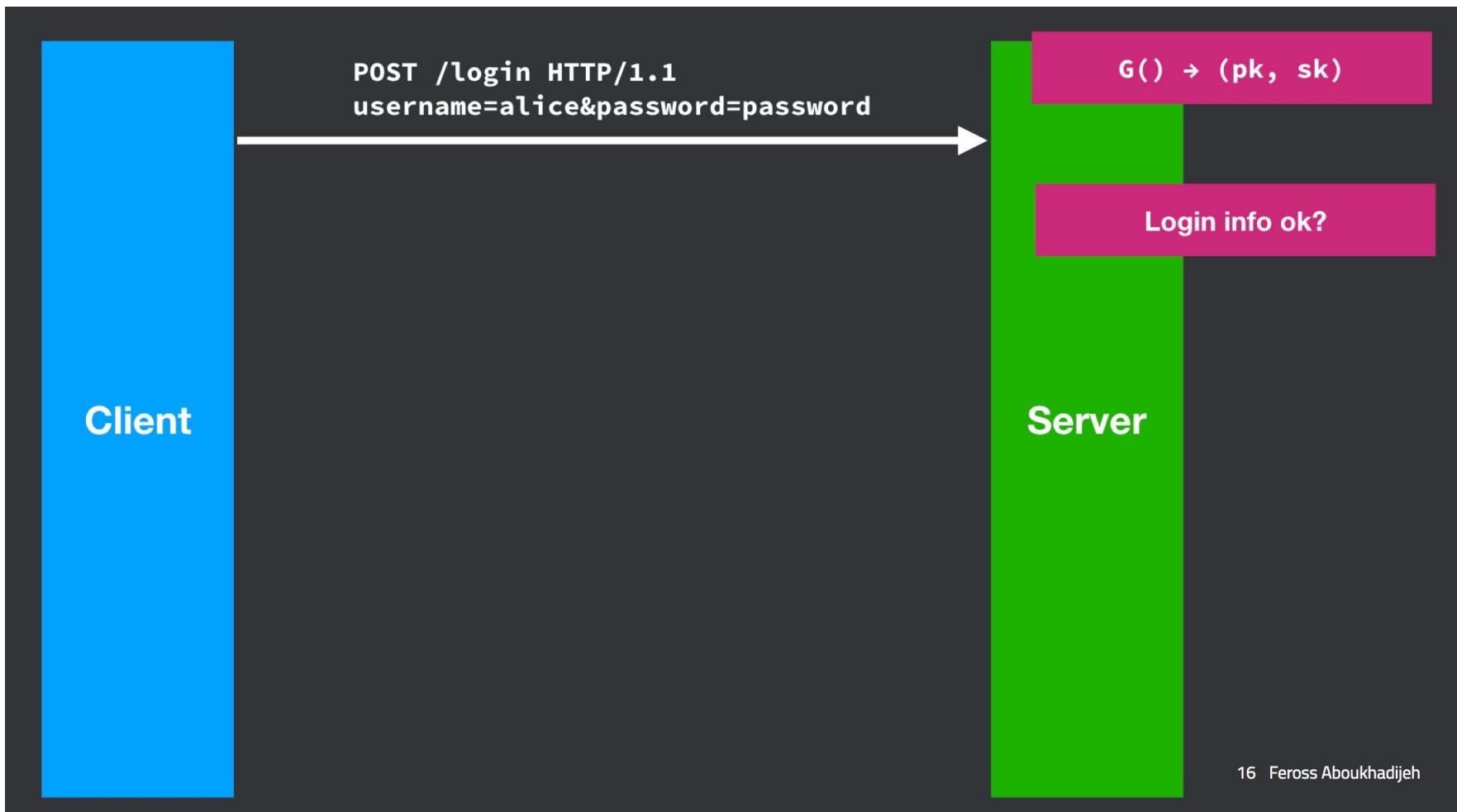
14 Feross Aboukhadijeh

Login and session



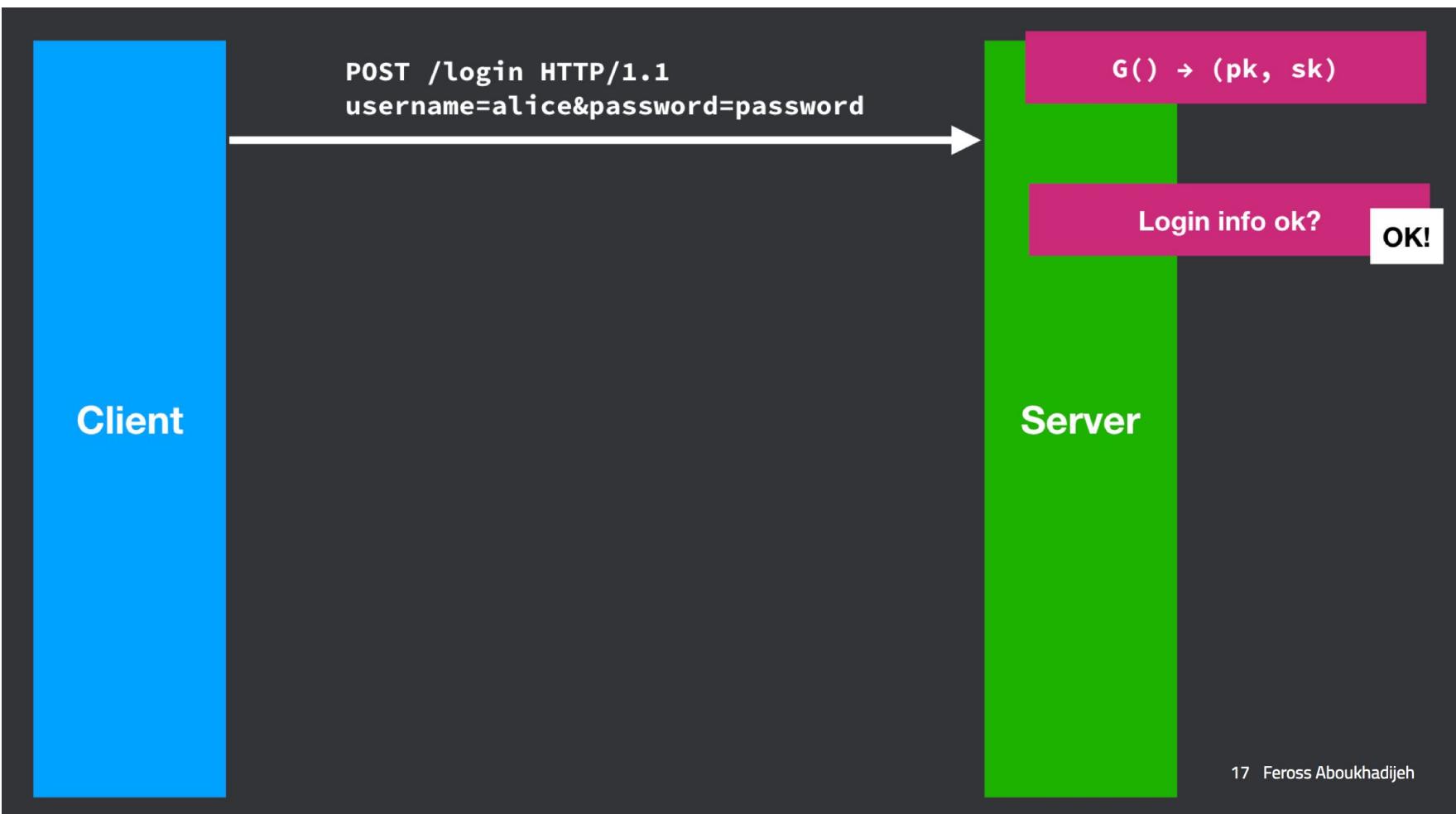
15 Feross Aboukhadijeh

Login and session



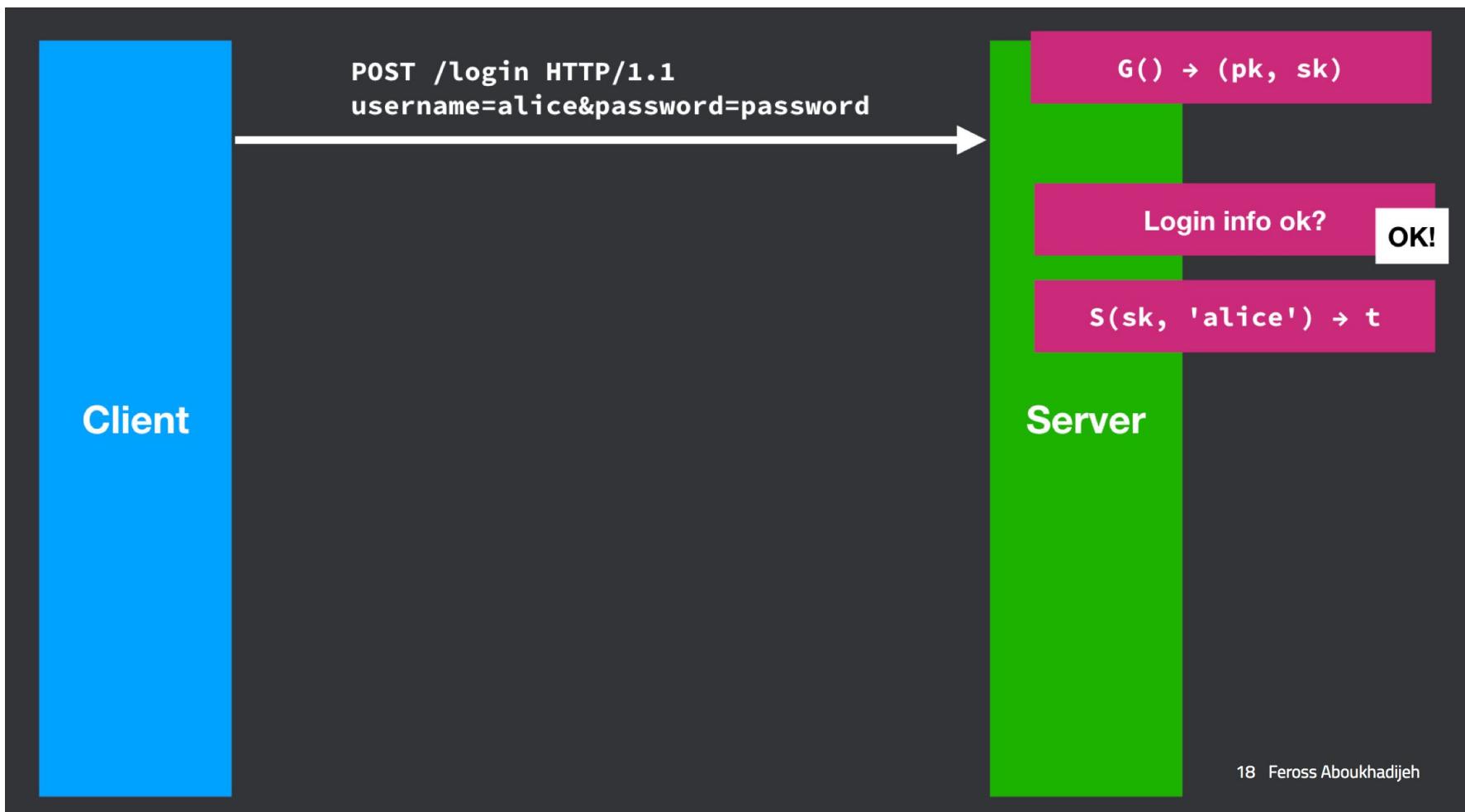
16 Feross Aboukhadijeh

Login and session

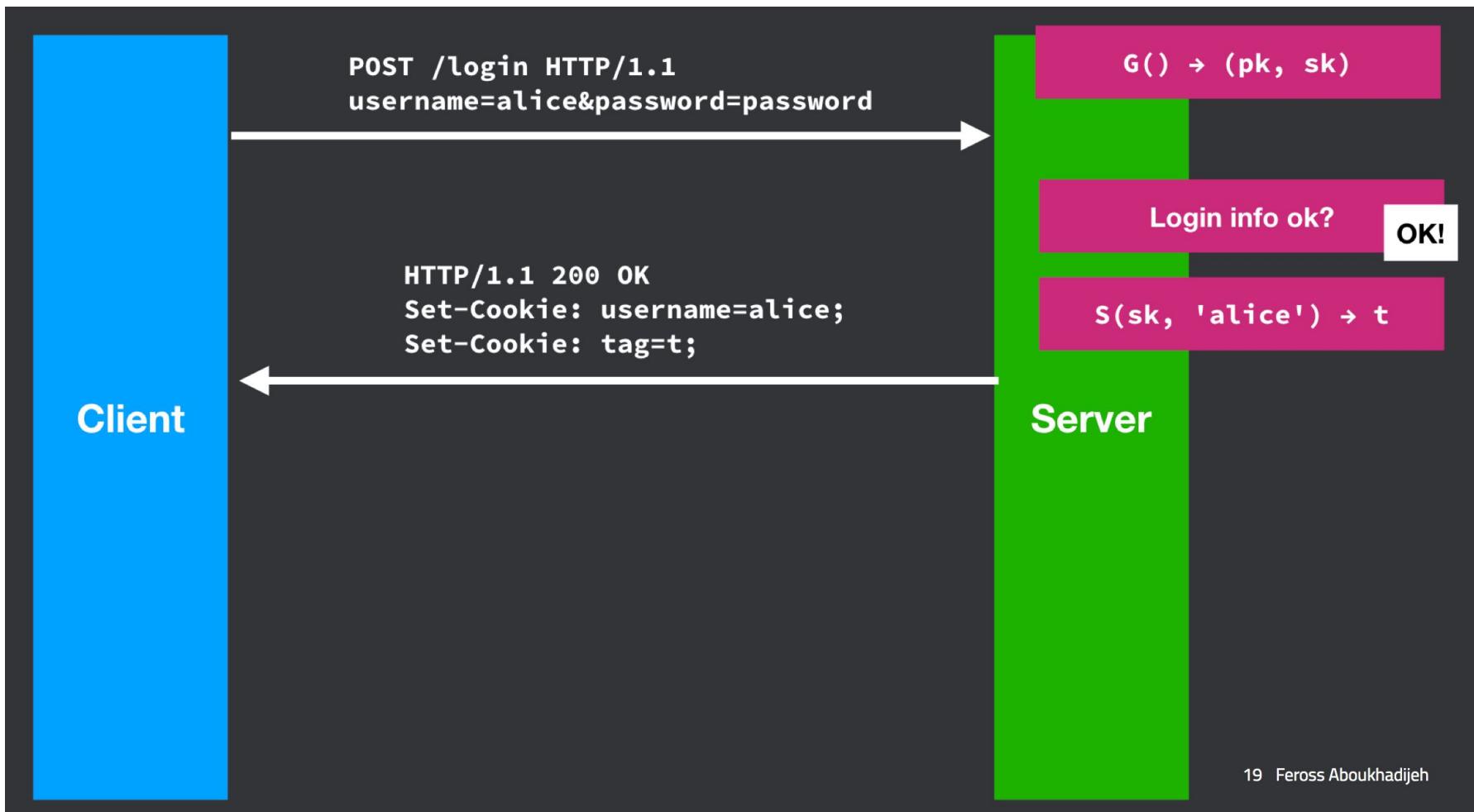


17 Feross Aboukhadijeh

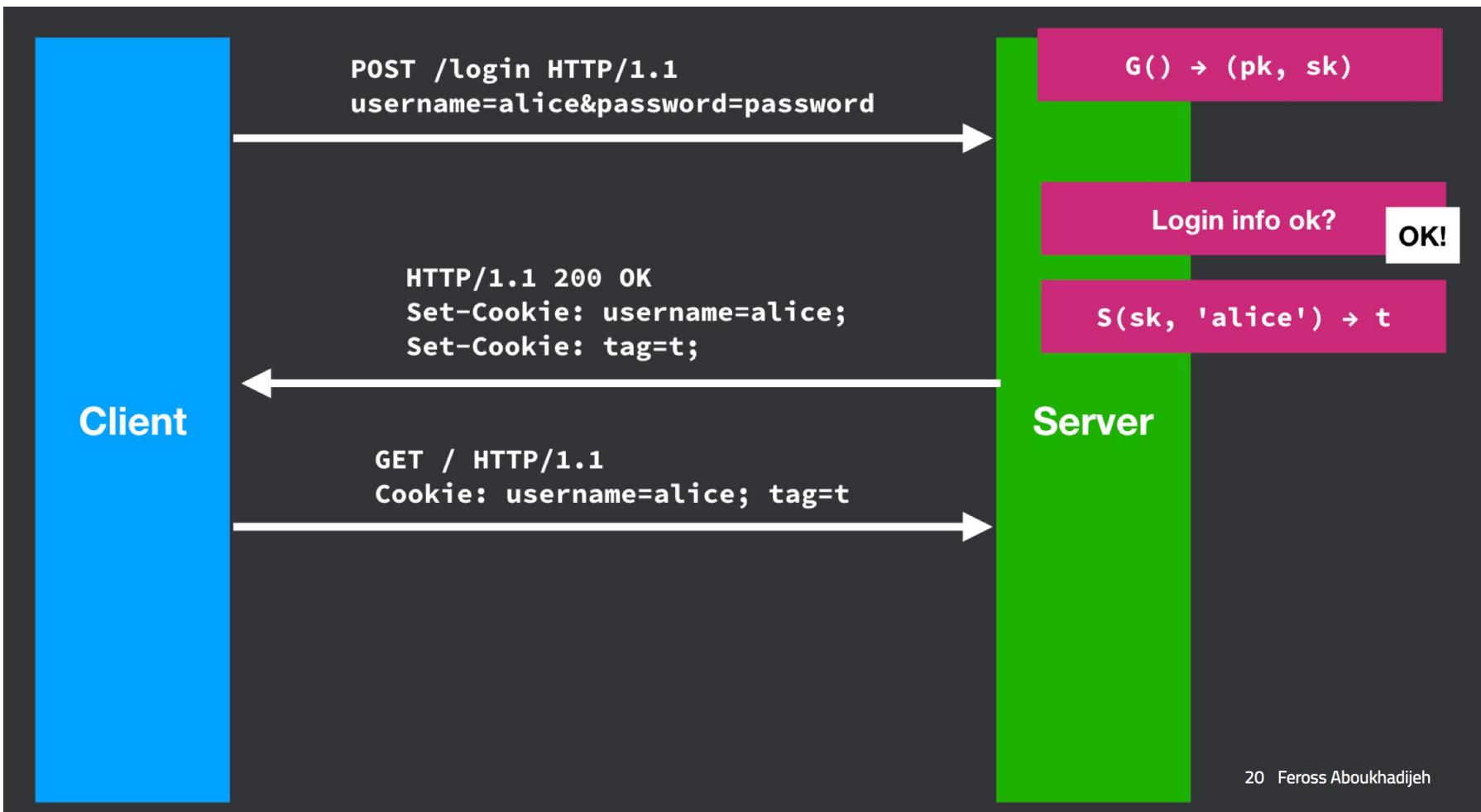
Login and session



Login and session

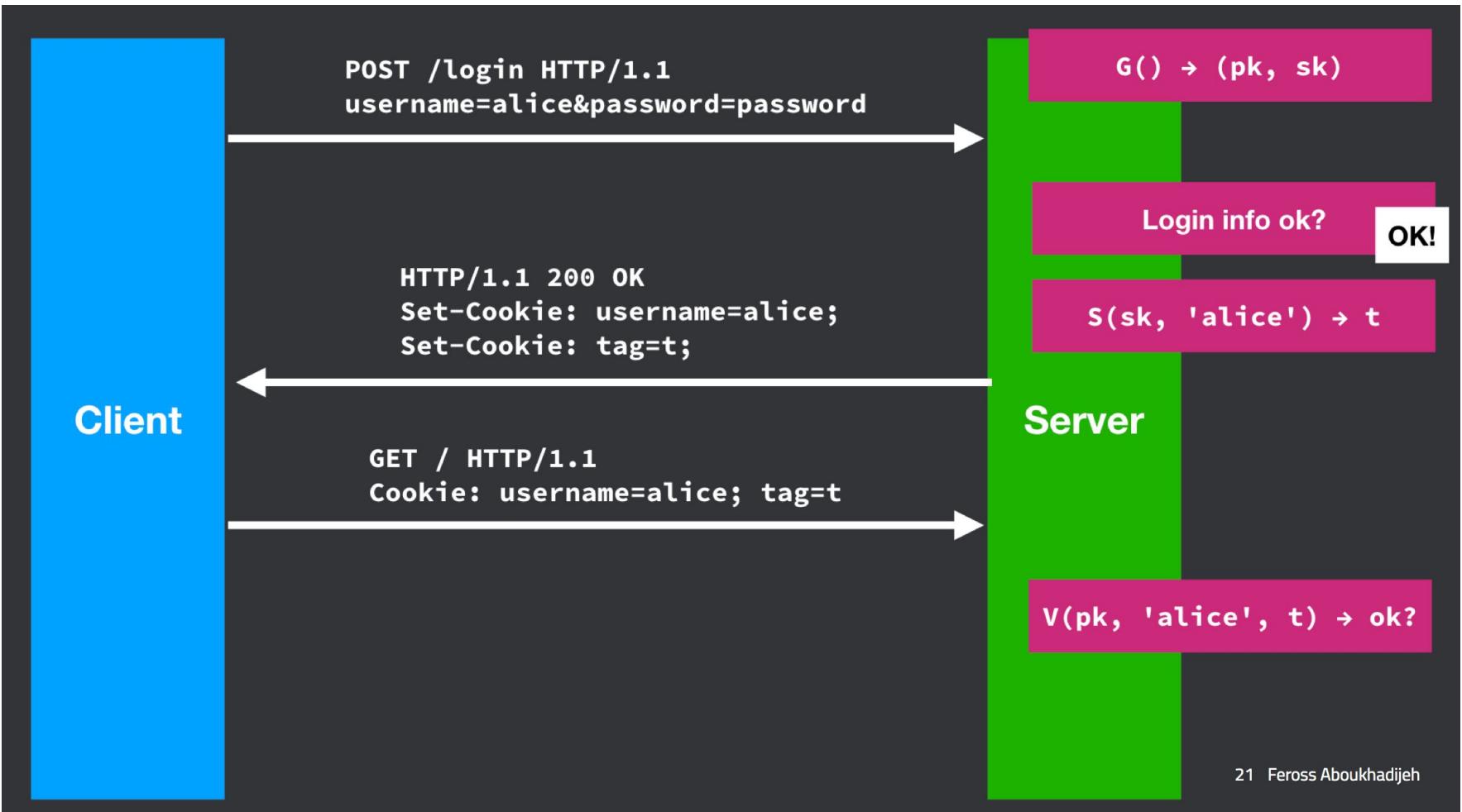


Login and session



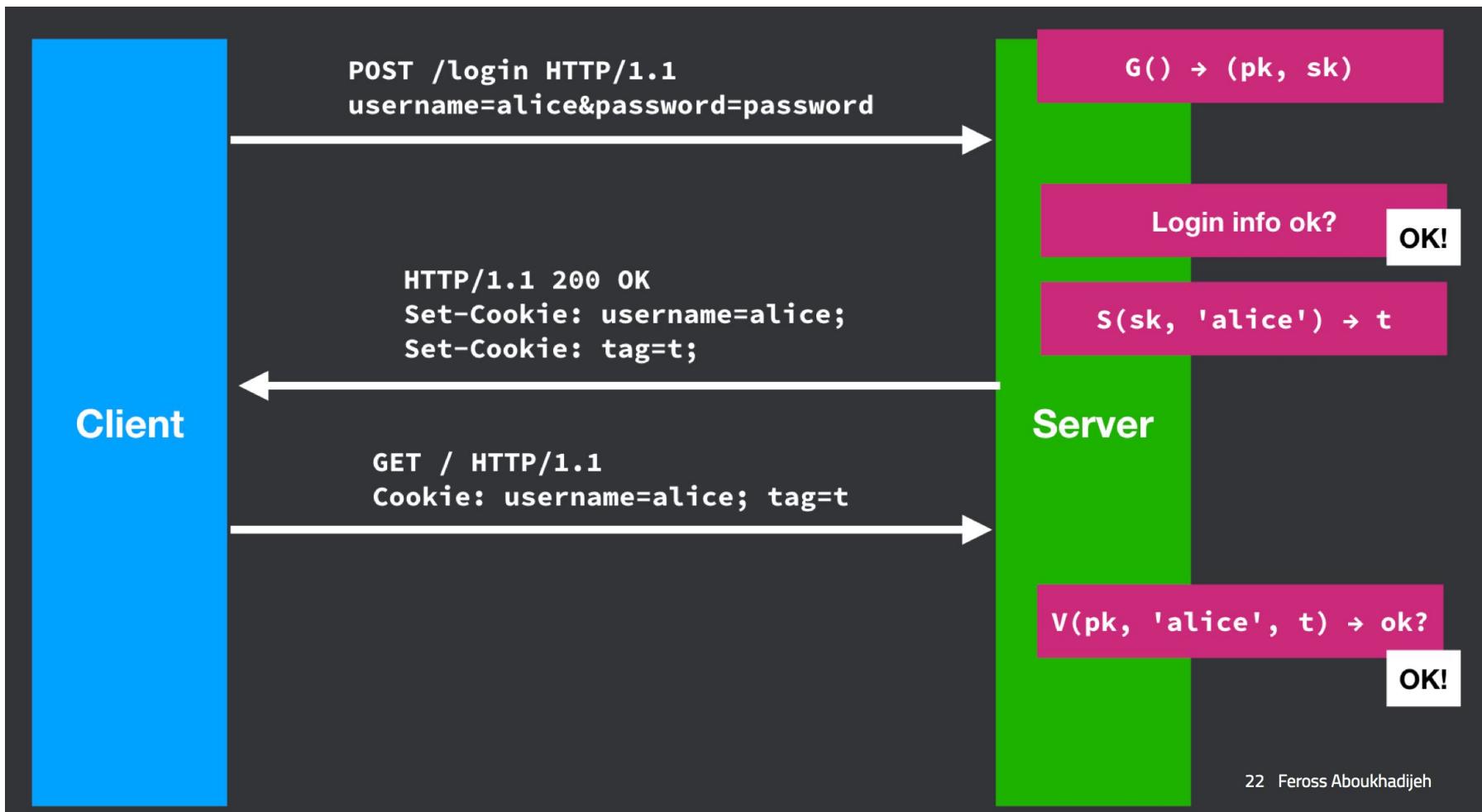
20 Feross Aboukhadijeh

Login and session

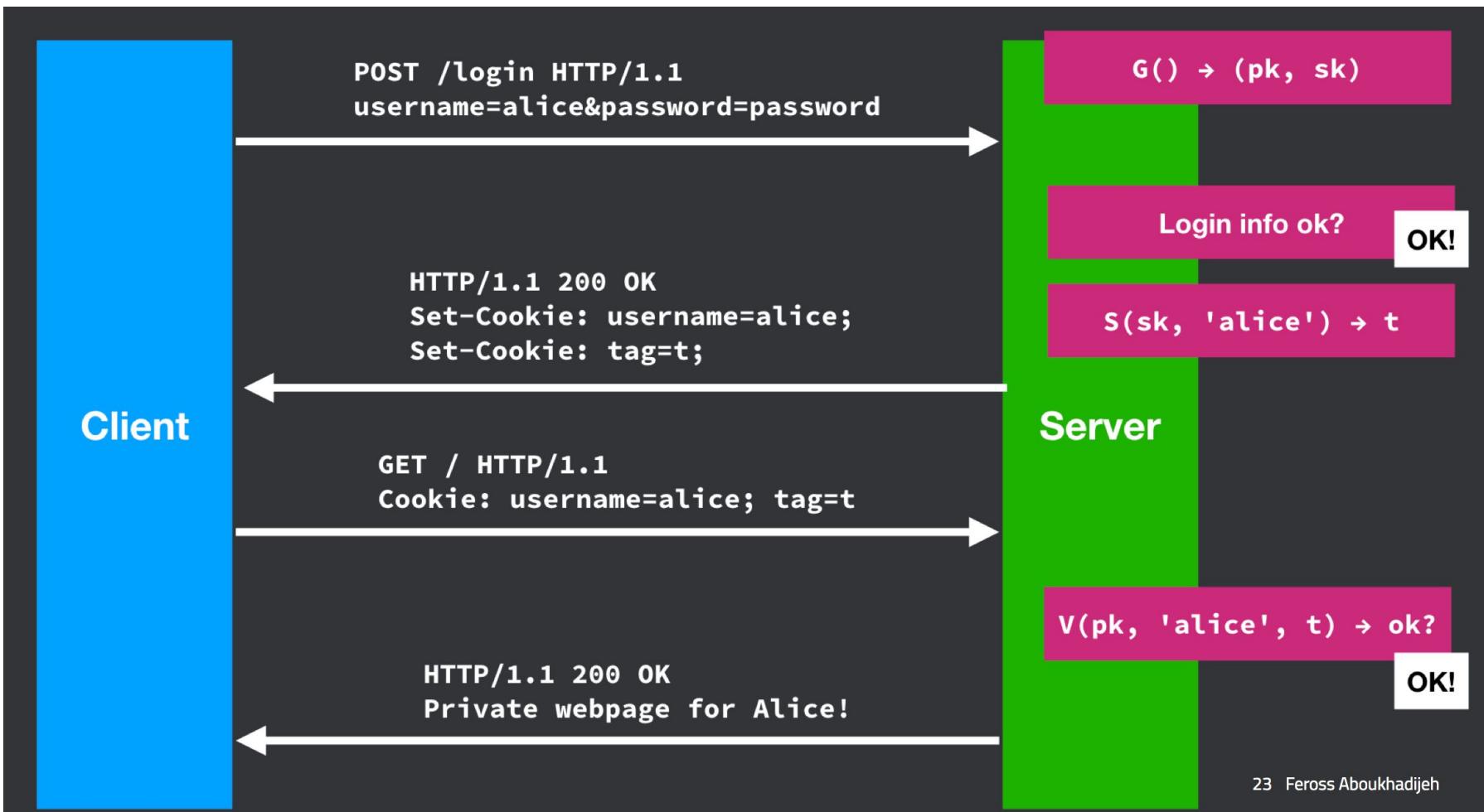


21 Feross Aboukhadijeh

Login and session

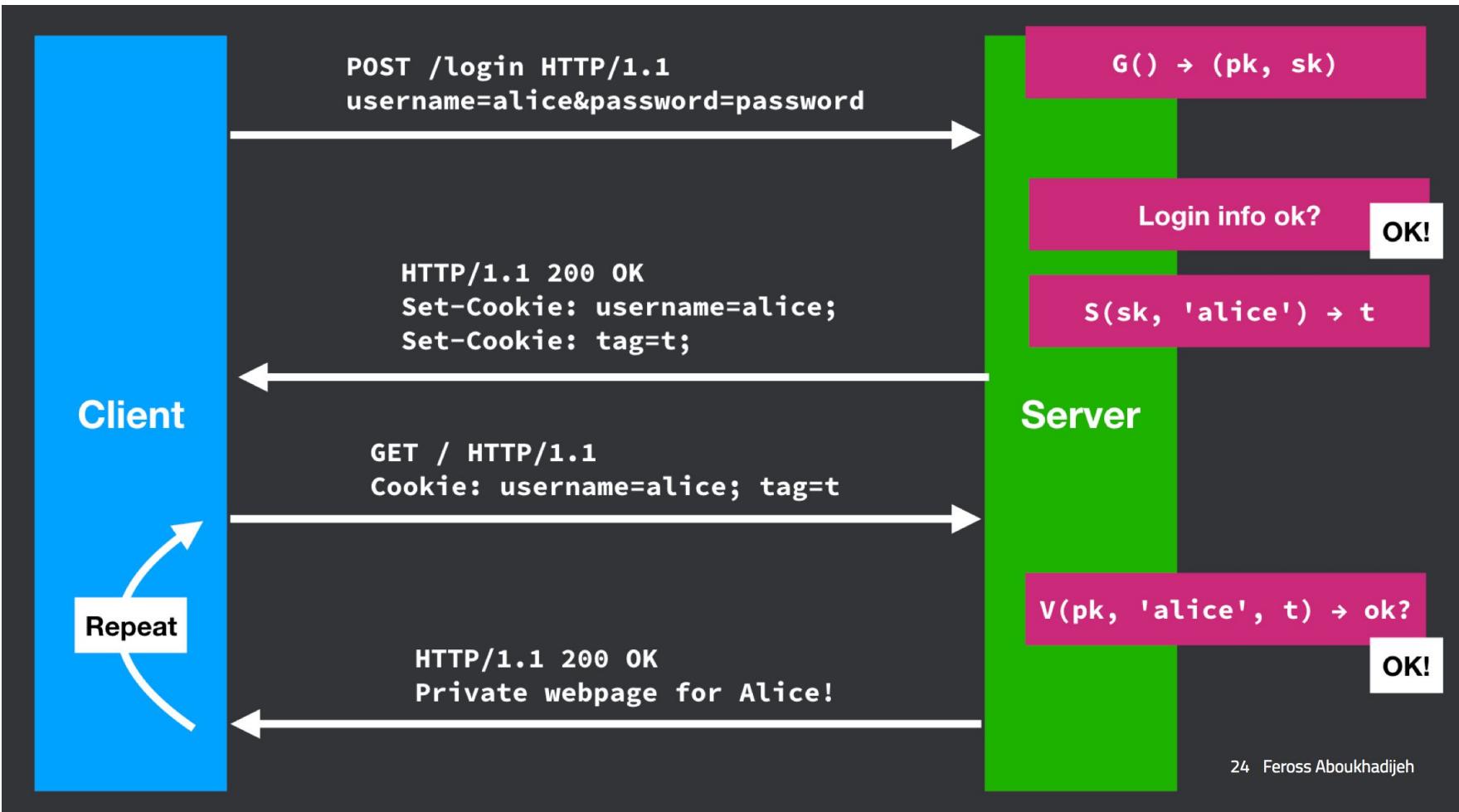


Login and session



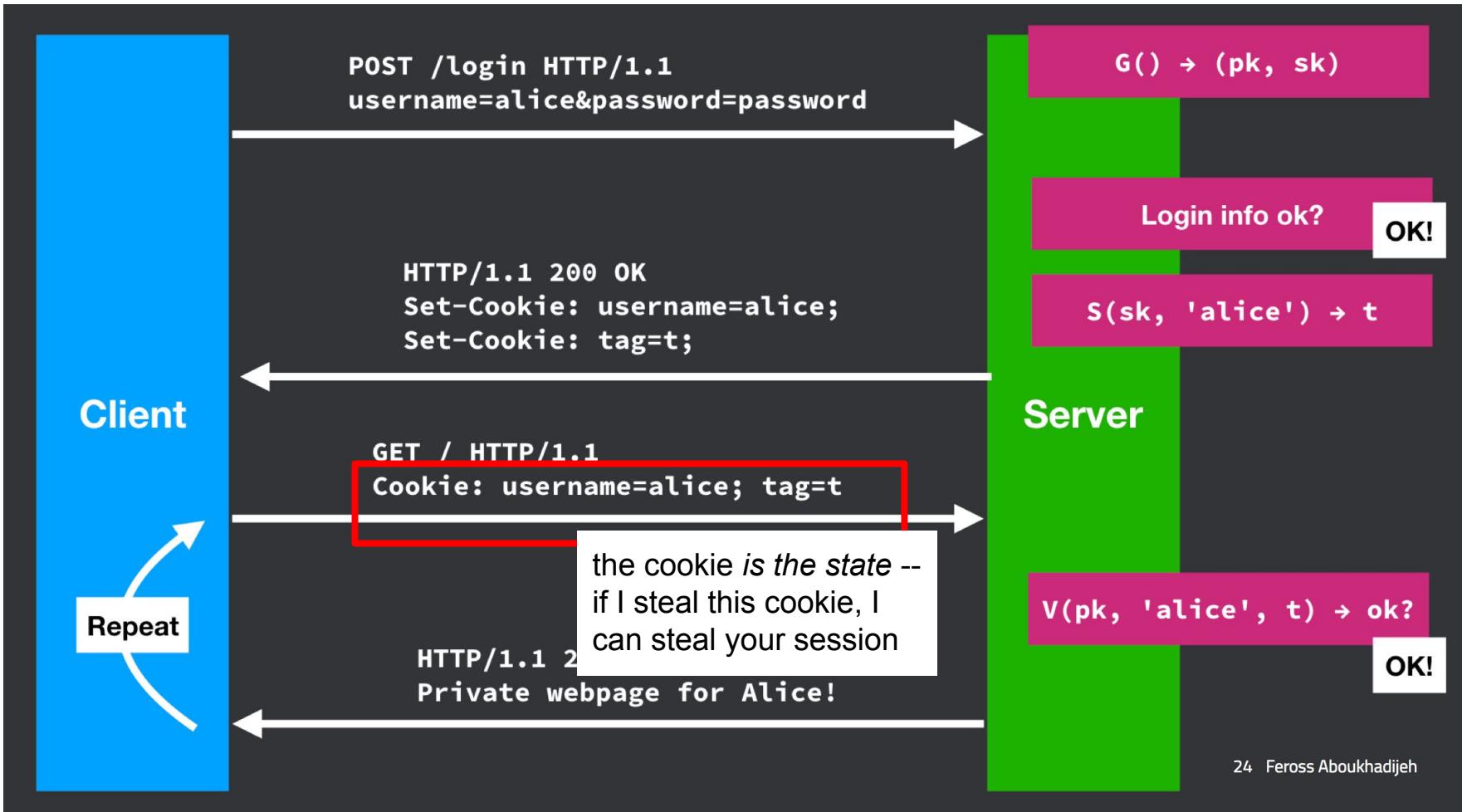
23 Feross Aboukhadijeh

Login and session



24 Feross Aboukhadijeh

If I know the cookie, I can be Alice



24 Feross Aboukhadijeh

History of Cookies

- Implemented in 1994 in Netscape and described in [4-page draft](#)
- No spec for 17 years
 - Attempt made in 1997, but made incompatible changes ([RFC 2109](#))
 - Another attempt in 2000 ("Cookie2", [RFC 2965](#)), same problem
 - Around 2011, another effort succeeded ([RFC 6265](#))
- Ad-hoc design has led to interesting issues

Cookie attributes

```
canis:/home/mln % curl -I --silent https://www.google.com/ | grep -i "Cookie:"  
set-cookie: 1P_JAR=2021-01-28-17; expires=Sat, 27-Feb-2021 17:08:10 GMT; path=/; domain=.google.com; Secure  
set-cookie:  
NID=208=QRqrD9wX6KJLXX5RH-pr8Q0UhqskKCjSAmnutkMRHT-QeI8E6pq3M5MVjIttWZBB4hXvoDUEubm0sv-n2ukDb43qt1Q8sk5qijlm9597z0A5X.  
8BooOLE7Dewr5Z21s97V5gR1ovSWCKbHJm7npE_X1sKkkVHTCEuOwEMLoK1M; expires=Fri, 30-Jul-2021 17:08:10 GMT; path=/;  
domain=.google.com; HttpOnly
```

- Expires - Specifies expiration date. If no date, then lasts for session
- Path - Scope the "Cookie" header to a particular request path prefix
 - e.g. Path=/docs will match /docs and /docs/Web/
 - Optimization only; do not use for security!
- Domain - Allows the cookie to be scoped to a domain broader than the domain that returned the Set-Cookie header
 - e.g. login.stanford.edu could set a cookie for stanford.edu

How long do cookies last?

- Sites can set Expires to a very far-future date and the cookie will last until the user clears it.
 - 2007: "The Google Blog announced that Google will be shortening the expiration date of its cookies from the year 2038 to a two-year life cycle." – [Search Engine Land](#)
- When Expires not specified, lasts for current browser session
 - Caveat: Browsers do session restoring, so can last way longer

Server can reset (delete) the cookie

- Set cookie with same name and an expiration date in the past
 - cf. [Unix epoch](#)
- Cookie value can be omitted; from previous google.com example:

```
set-cookie: NID=; expires=Thu, 01 Jan 1970 00:00:00 GMT;
```

Use dev tools to access, set, delete cookies in your browser

The screenshot shows the Chrome DevTools interface with the "Storage" tab selected. Under the "Cookies" section for the domain <https://www.google.com>, a specific cookie named "1P_JAR" is highlighted. The table lists various cookies with their details:

Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnly
1P_JAR	2021-01-29-...	.google.com	/	Sun, 28 Feb 2021... 19	19	false
_Secure...	r86R4-zlD9t...	.google.com	/	Sat, 28 Jan 2023 ... 51	51	false
_Secure...	Aji4QfFeXZ9j...	.google.com	/	Sat, 29 Jan 2022 ... 91	91	true
_Secure...	6AehLhzMH...	.google.com	/	Sat, 28 Jan 2023 ... 85	85	true
ANID	AHWqTUUmQb...	.google.com	/	Sat, 28 Jan 2023 ... 68	68	true
APSID	rKUymixXFeW...	.google.com	/	Sat, 28 Jan 2023 ... 40	40	false
CGIC	Cg1maXJlZm...	.google.com	/compl...	Tue, 27 Jul 2021 ... 126	126	true
CGIC	Cg1maXJlZm...	.google.com	/search	Tue, 27 Jul 2021 ... 126	126	true
DV	s_MjMohzSloe...	www.googl...	/	Fri, 29 Jan 2021 ... 33	33	false
HSID	ARIW8nZkMhr...	.google.com	/	Sat, 28 Jan 2023 ... 21	21	true
NID	208=Oh4uCB...	.google.com	/	Sat, 31 Jul 2021 ... 259	259	true
SAPISID	r86R4-zlD9t...	.google.com	/	Sat, 28 Jan 2023 ... 41	41	false
SIDCC	Aji4QfHWEOP...	.google.com	/	Sat, 29 Jan 2022 ... 80	80	false
SID	6AehLhzMH...	.google.com	/	Sat, 28 Jan 2023 ... 74	74	false
SNID	APx-OP294TY...	.google.com	/verify	Sat, 31 Jul 2021 ... 90	90	true
SSID	Aa6aVCExncb...	.google.com	/	Sat, 28 Jan 2023 ... 21	21	true

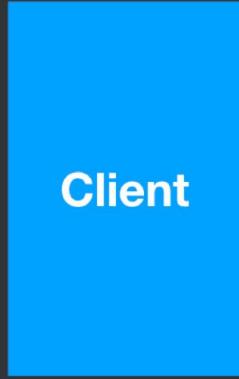
The right panel displays detailed information for the selected cookie "1P_JAR":

- Created: "Wed, 04 Aug 2032 09:55:16 GMT"
- Domain: ".google.com"
- Expires / Max-Age: "Sun, 28 Feb 2021 14:40:54 GMT"
- HostOnly: false
- HttpOnly: false
- Last Accessed: "Fri, 29 Jan 2021 14:40:54 GMT"
- Path: "/"
- SameSite: "None"
- Secure: true
- Size: 19

Session hijacking

- Sending cookies over unencrypted HTTP is a *very bad idea*
 - If anyone sees the cookie, they can use it to hijack the user's session
 - Attacker sends victim's cookie as if it was their own
 - Server will be fooled -- it only “knows” about you *via the cookie*

A nice session



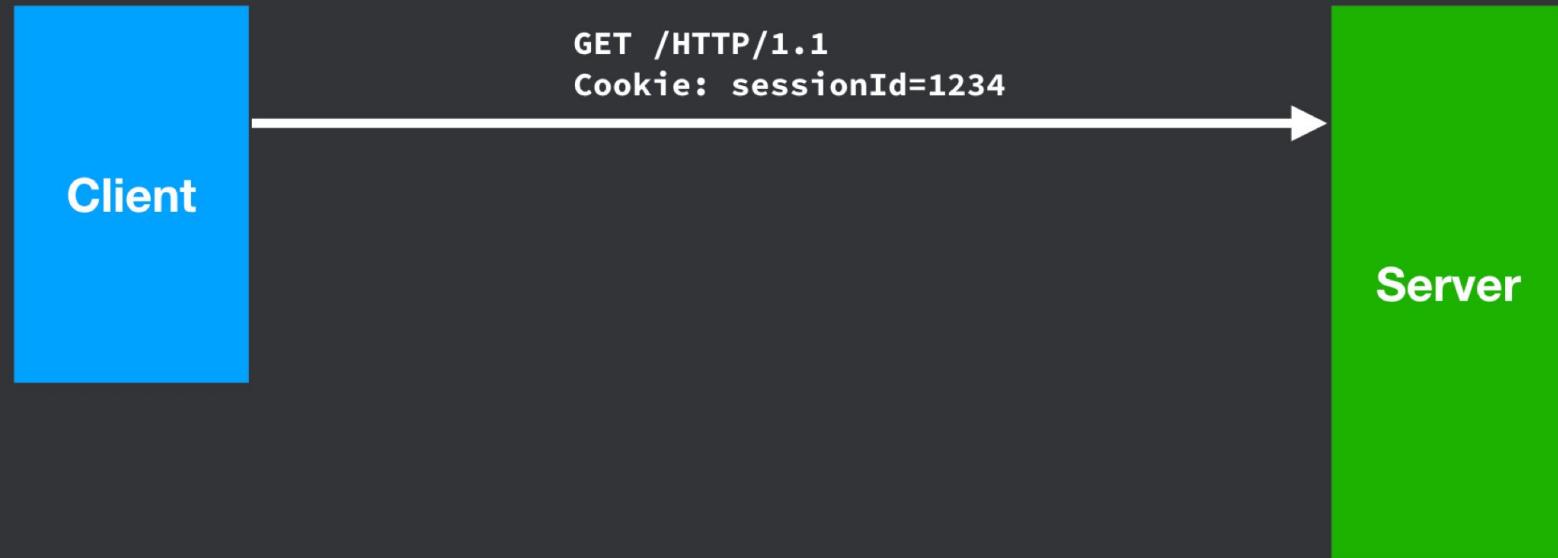
Client



Server

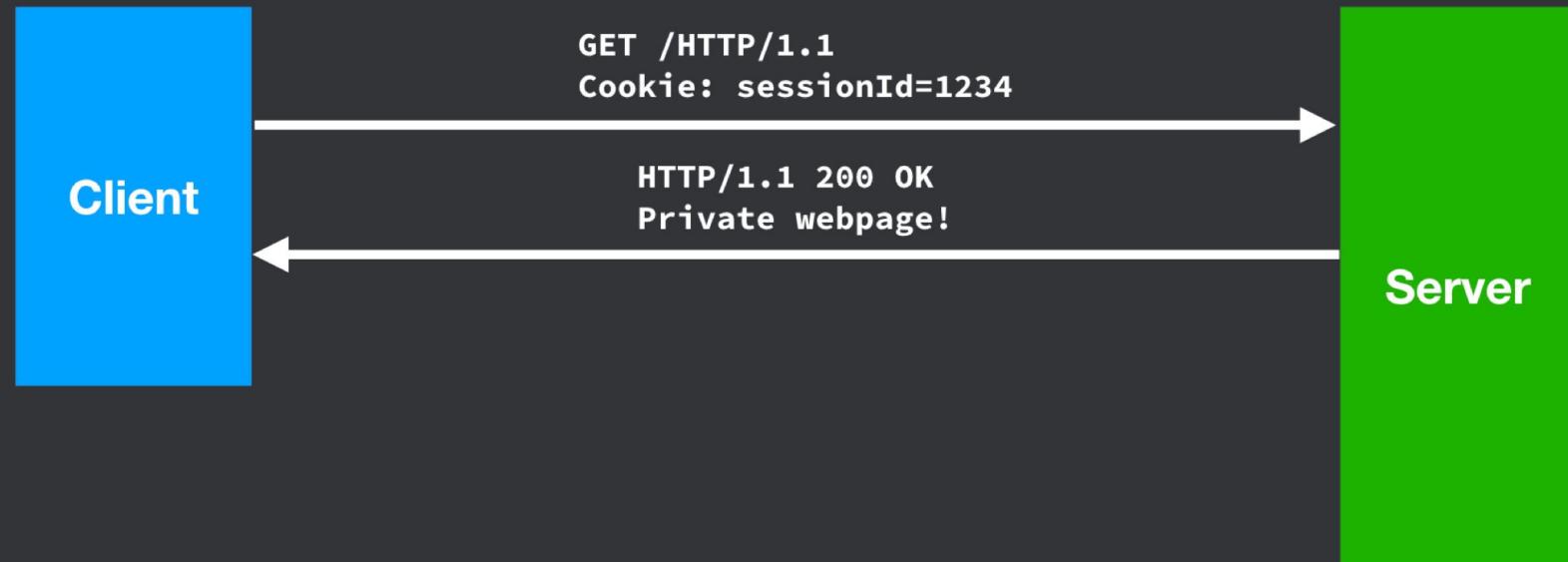
35 Feross Aboukhadijeh

A nice session



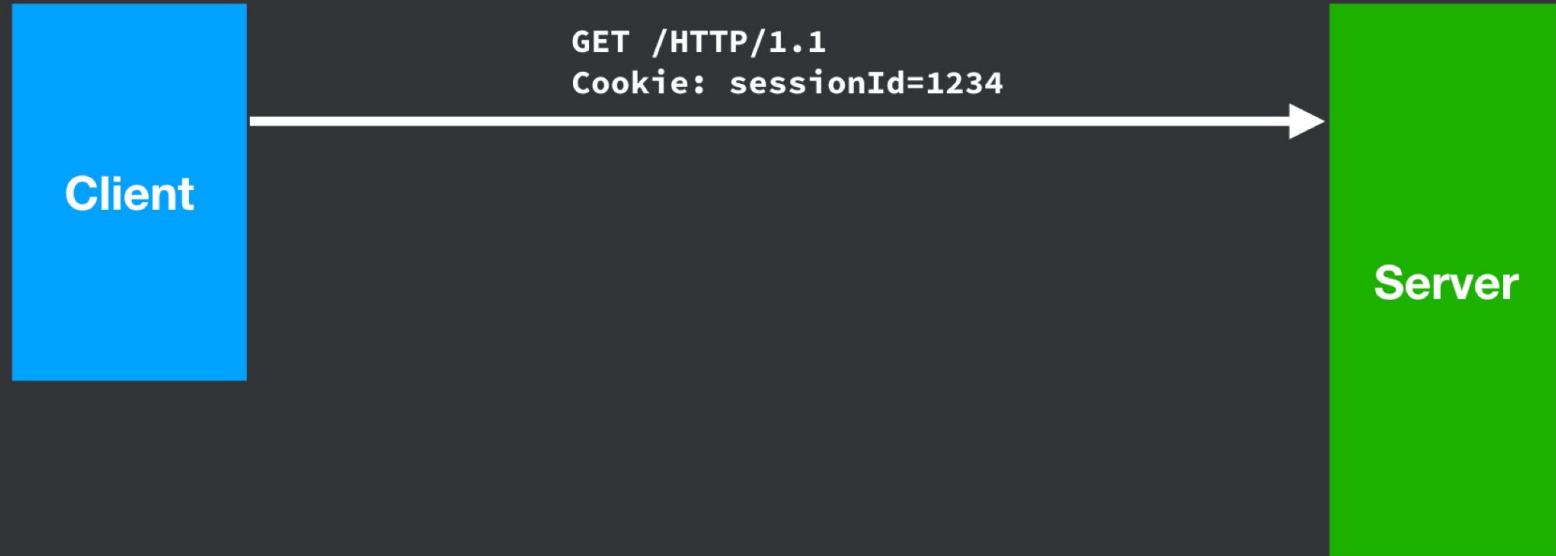
36 Feross Aboukhadijeh

A nice session



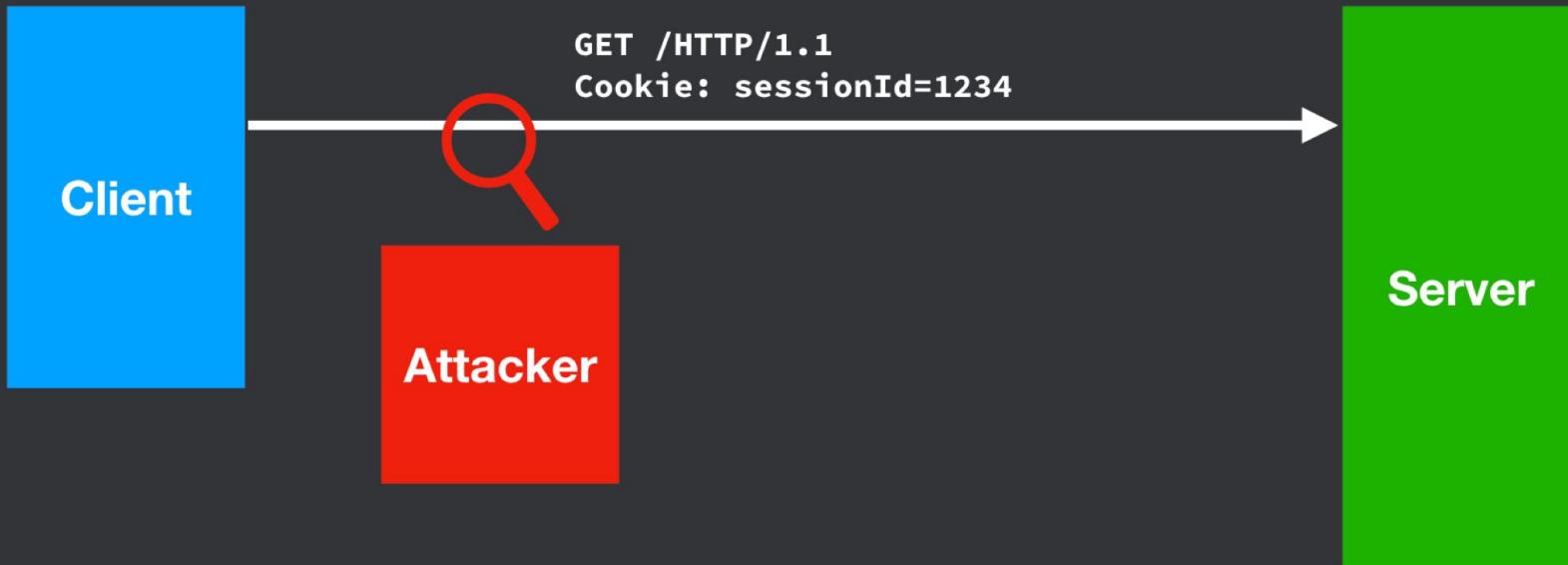
37 Feross Aboukhadijeh

Steal your cookie



36 Feross Aboukhadijeh

Steal your cookie



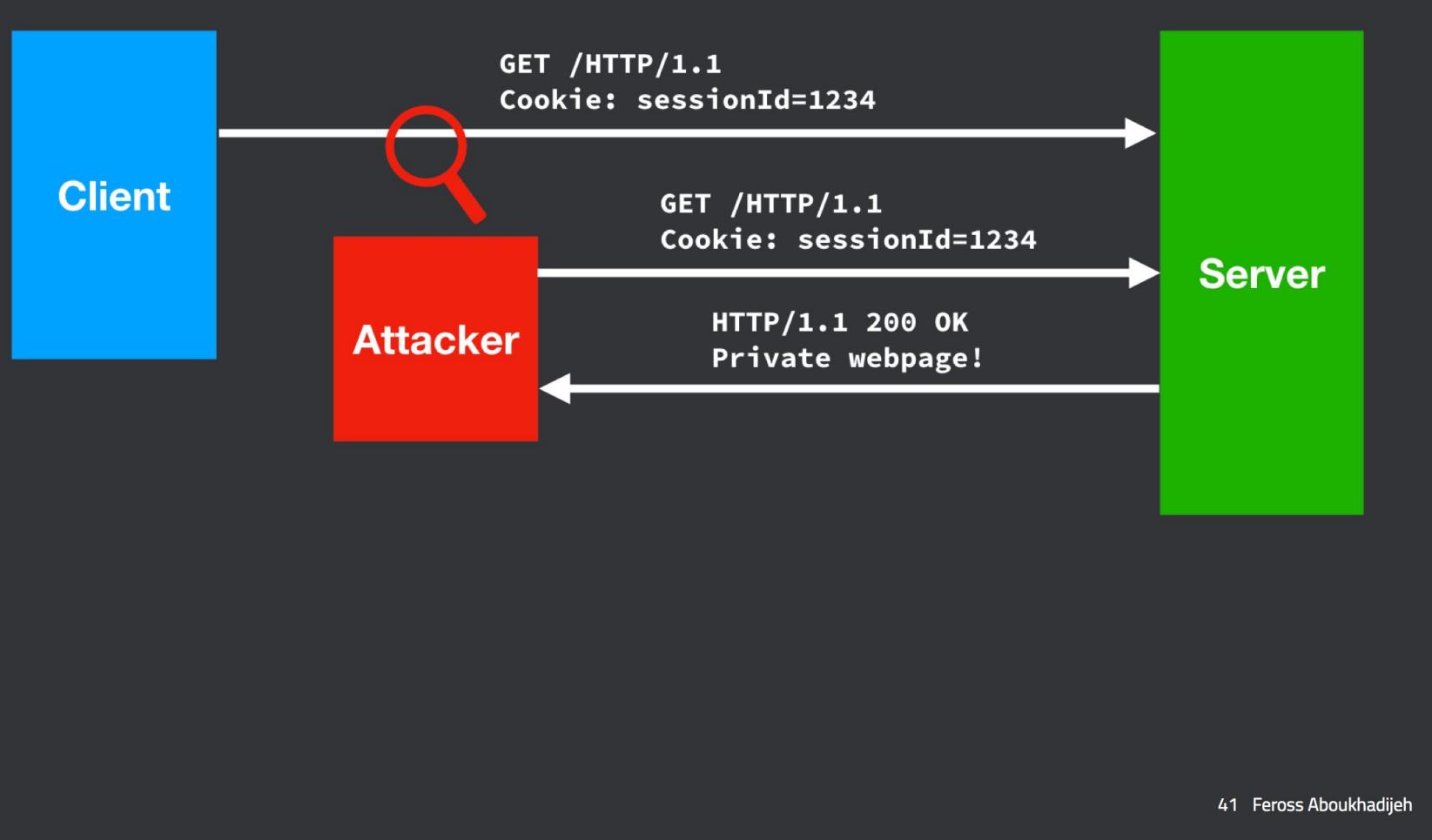
39 Feross Aboukhadijeh

Steal your cookie



40 Feross Aboukhadijeh

Steal your cookie



41 Feross Aboukhadijeh

Session hijacking mitigation

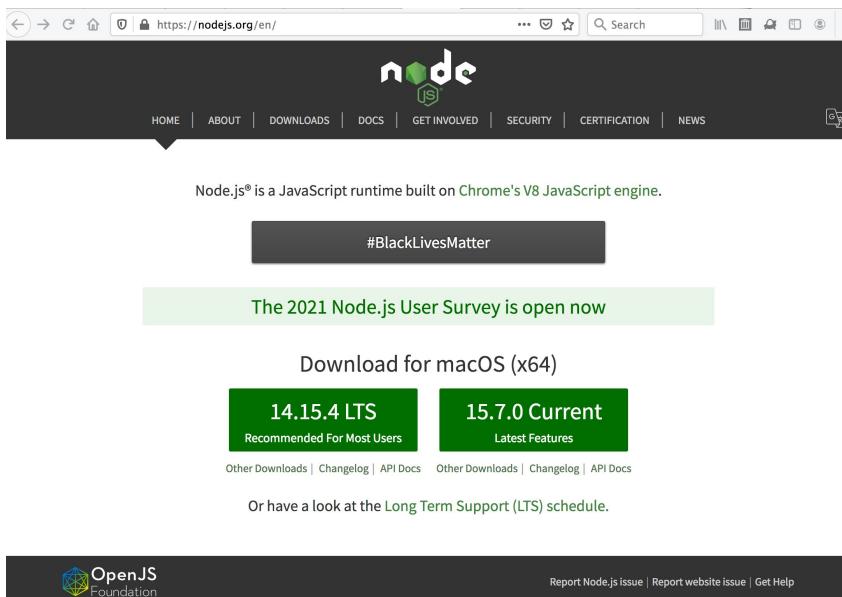
- Use **Secure** cookie attribute to prevent cookie from being sent over unencrypted HTTP connections
 - Set-Cookie:
key=value; Secure
- Even better: Use HTTPS for entire website

```
canis:/home/mln % curl -I --silent \  
http://www.google.com/ | grep -i "Secure"  
Set-Cookie: 1P_JAR=2021-01-28-17;  
Expires=Sat, 27-Feb-2021 17:52:32 GMT;  
Path=/; Domain=.google.com; Secure
```

```
canis:/home/mln % curl -I http://vt.edu  
HTTP/1.1 301 Moved Permanently  
Date: Thu, 28 Jan 2021 17:36:58 GMT  
Server: Apache  
Location: https://vt.edu/  
Cache-Control: max-age=600  
Expires: Thu, 28 Jan 2021 17:46:58 GMT  
Content-Type: text/html; charset=iso-8859-1
```

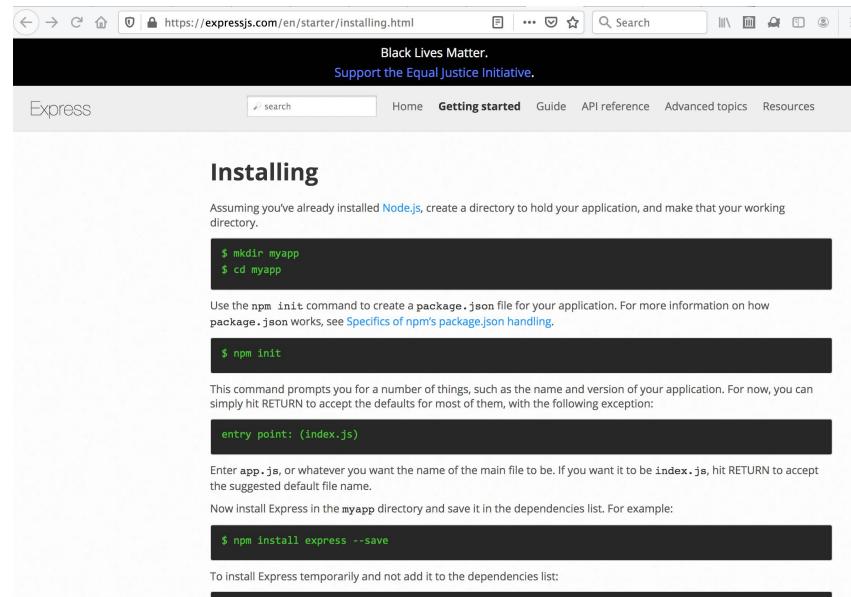
```
canis:/home/mln % curl -I www.odu.edu  
HTTP/1.0 301 Moved Permanently  
Location: https://www.odu.edu/  
Server: BigIP  
Connection: Keep-Alive  
Content-Length: 0
```

Setting up the world's worst bank with node.js



The screenshot shows the official Node.js website at <https://nodejs.org/en/>. The page features the Node.js logo at the top left. A navigation bar below it includes links for HOME, ABOUT, DOWNLOADS, DOCS, GET INVOLVED, SECURITY, CERTIFICATION, and NEWS. A search bar is located at the top right. The main content area contains a message about Node.js being built on Chrome's V8 JavaScript engine, a "#BlackLivesMatter" button, a green banner announcing the 2021 User Survey, and download links for Node.js 14.15.4 LTS (Recommended For Most Users) and 15.7.0 Current (Latest Features). At the bottom, there are links for Other Downloads, Changelog, API Docs, and a footer with the OpenJS Foundation logo and links for reporting issues.

<https://nodejs.org/en/>



The screenshot shows the Express.js documentation page at <https://expressjs.com/en/starter/installing.html>. The page has a "Black Lives Matter" header with a call to support the Equal Justice Initiative. It features a search bar and navigation links for Home, Getting started, Guide, API reference, Advanced topics, and Resources. The main content is titled "Installing" and provides instructions for creating a directory and using the npm init command. It includes code snippets for creating a directory and changing into it, as well as examples for creating a package.json file and installing Express. The page also mentions the entry point for the application.

<https://expressjs.com/en/starter/installing.html>

use npm, included in the node.js distribution, to install packages you don't have

```
$ node bank-02.js
internal/modules/cjs/loader.js:883
    throw err;
    ^

Error: Cannot find module 'cookie-parser'
Require stack:
- /Users/mln/Desktop/cs595-s21/cs595-s21/slides/code/bank-02.js
    at Function.Module._resolveFilename
[lots of error messages deleted]
    code: 'MODULE_NOT_FOUND',
    requireStack: [ '/Users/mln/Desktop/cs595-s21/cs595-s21/slides/code/bank-02.js' ]
}
$ npm install cookie-parser
npm WARN code@1.0.0 No description
npm WARN code@1.0.0 No repository field.

+ cookie-parser@1.4.5
added 1 package from 2 contributors and audited 51 packages in 2.992s
found 0 vulnerabilities
$ node bank-02.js
```

Who are Alice and Bob?!

The most common characters are Alice and Bob. Eve, Mallory, and Trent are also common names, and have fairly well-established "personalities" (or functions). The names often use rhyming mnemonics (for example, Eve, "eavesdropper"; Mallory, "malicious") where different players have different motives. Other names are much less common, and flexible in use. Sometimes the genders are alternated: Alice, Bob, Carol, Dave, Eve ...^[13]

Alice and Bob	The original, generic characters. Generally, Alice and Bob want to exchange a message or cryptographic key.
Carol, Carlos or Charlie	A generic third participant.
Chuck	A third participant, usually of malicious intent. ^[14]
Craig	A <i>password cracker</i> , often encountered in situations with stored passwords.
Dan, Dave or David	A generic fourth participant.
Erin	A generic fifth participant, but rarely used, as "E" is usually reserved for Eve.
Eve	An <i>eavesdropper</i> , who is usually a passive attacker. While they can listen in on messages between Alice and Bob, they cannot modify them. In <i>quantum cryptography</i> , Eve may also represent

The diagram shows two separate processes for Alice and Bob. For Alice, her public key (green) and private key (red) are combined to produce a shared secret (751A696C 24D97009). Similarly, for Bob, his public key (green) and private key (red) are combined to produce the same shared secret. This illustrates how two parties can independently generate a shared key using their own keys and a 'Combine keys' function.

This diagram shows a simplified version of the protocol. Alice (green) sends a message 'hello Alice' to Bob (purple). Both parties use their private keys to encrypt the message. Alice uses Bob's public key, and Bob uses Alice's public key. The resulting ciphertexts are 'x0Ak3 o\$2Rj' for Alice and 'x0Ak3 o\$2Rj' for Bob. This illustrates how public-key cryptography allows communication between two parties without a shared密钥.

https://en.wikipedia.org/wiki/Alice_and_Bob