HTTP

INTERNET ENGINEERING

Fall 2022

@1995parham

- Introduction
- Cookie
- Proxy & Cache
- Authentication

- Introduction
- Cookie
- Proxy & Cache
- Authentication

INTRODUCTION

- HTTP is the transfer protocol for web applications
  - H T T P: Hyper Text Transfer Protocol
  - HTTP 1.0 (RFC 1945), HTTP 1.1 (RFC 2068), HTTP 2 (RFC 7540)
  - In fact, it can be used to transfer everything (not only hyper text)
    - Text Documents e.g. HTML, XML, JSON, etc.
    - Multimedia e.g. JGP, GIF, MP4, MKV, etc.
    - Application e.g. PDF, ZIP, etc.

INTRODUCTION (CONT.)

- HTTP uses the client/server paradigm
  - HTTP server provides resource
  - HTTP client (usually web browser) gets resource
- But not pure client/server communication
  - Proxies
  - Caches
  - ...

INTRODUCTION (CONT.)

- HTTP is an application layer protocol
- HTTP assumes reliable communication
    - over TCP
    - default (server) port: 80
    - client port is chosen randomly per connection
- HTTP is Stateless
    - Server does not keep history/state of client
    - High performance & Low Complexity
    - Problematic in some applications (sessions)
        - Cookies
        - JSON Web Tokens

DATA RESOURCES TO TRANSFER

- HTTP is the protocol to transfer data between server and client (usually from server to client)
- Which data?
  - It can be anything
  - In web, usually, it is a resource/object on server
- Each resource must be identified/located uniquely
  - URI (Uniform Resource Identifier)
  - URL (Uniform Resource Locator)
- URIs identify and URLs locate; however, locators are also identifiers, so

> *every URL is also a URI, but there are URIs which are not URLs.*

URI IN ACTION

- A Uniform Resource Name (URN) is a URI that identifies a resource by name in a particular namespace.
- International Standard Book Number (ISBN) system

URL

<protocol(scheme)> :// <user> : <pass> @ <host> : < port> / <path> ? <query> # <frag>

- https://sbu.ac.ir/SitePages/Home.aspx
- https://www.bing.com/search?q=Hello+World&form=QBLH&sp=-1&pq=&sc=0-0&qs=n&sk=&cvid=7F2B496642F94D5F95989756B4FF60EE
- file:///home/parham/Documents/Git/parham/dotfiles
- http://libgen.is
- ftp://speedtest.tele2.net

URL (CONT.)

- Scheme: The application layer protocol
- HTTP: The web protocol
- HTTPS: Secure HTTP
- FTP: File Transfer Protocol
- File: Access to a local file
- javascript: Run javascript code
- mailto: Send mail to given address
- etc.

URL (CONT.)

- Path: The path of the object on host filesystem
- E.g. web server root directory is `/var/www/`
    - `http://www.example.com/1.html` → `/var/www/1.html`
    - `http://www.example.com/1/2/3.jpg` → `/var/www/1/2/3.jpg`
    - `http://www.example.com/1/2/../3.jpg` → `/var/www/1/3.jpg` 🤨

URL (CONT.)

- Query: A mechanism to pass information from client to active pages or forms
    - Fill information in a university registration form
    - Ask Bing! to search a phrase
- Starts with "?"
- `name=value` format
- "&" is the border between multiple parameters
- Try this out, by clicking me

HTTPBIN

A simple HTTP Request & Response Service.

URL (CONT.)

- Frag: A name for a part of resource
    - A section in a document
- Handled by browser
    - Browser gets whole resource (doc) from server
    - In display time, it jumps to the specific part
- Try this out, by clicking me

URL (CONT.)

- Domain names are case insensitive according to RFC 4343.
- The rest of URL is sent to the server via the GET method and etc. *“This may be case sensitive or not.”*

URL (CONT.)

- URL is encoded by client before transmission
- How: Each byte is divided into two 4-bit group, hexadecimal of the 4-bits are prefixed by %
    - ~ → 126 → %7E
- What & Why?
    - Non-ASCII (e.g., Persian Characters, Emoji)
    - Reserved character when are not used for special role
    - Unsafe character, e.g. space, %, ...

> *https://ganj.irandoc.ac.ir/api/v1/search/main?keywords=hellow%20world*

URL IN ACTION

- User asks the browser to retrieve a resource
- Browser finds the ip address of <host> (DNS lookup)

- Browser creates a TCP connection to the IP address and the <port>
- Browser sends http requests through the connection
- Browser gets the response and processes it

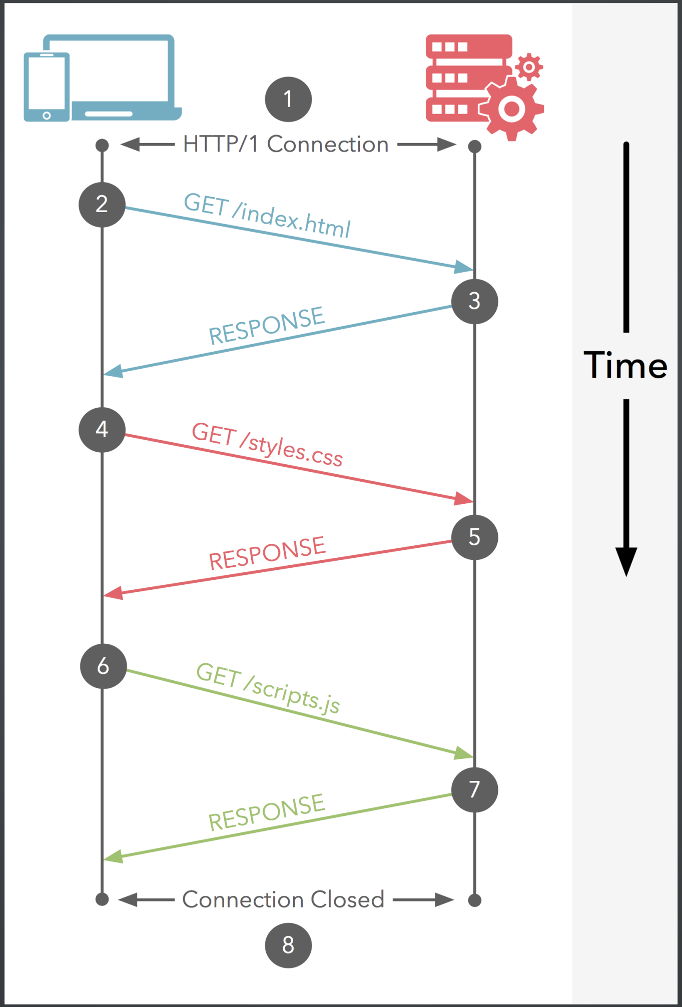
URL IN ACTION (CONT.)

```
2950  113.658623720 192.168.73.191  3.220.112.94  TCP 74  59890 → 80 [SYN] Seq=0
2956  113.848038152 3.220.112.94  192.168.73.191  TCP 74  80 → 59890 [SYN, ACK] S
2957  113.848146169 192.168.73.191  3.220.112.94  TCP 66  59890 → 80 [ACK] Seq=1
2958  113.848626070 192.168.73.191  3.220.112.94  HTTP  503 GET /bytes/1378 HTTP/
2962  114.079894429 3.220.112.94  192.168.73.191  TCP 66  80 → 59890 [ACK] Seq=1
2963  114.080912334 3.220.112.94  192.168.73.191  HTTP  1683  HTTP/1.1 200 OK
2964  114.080950090 192.168.73.191  3.220.112.94  TCP 66  59890 → 80 [ACK] Seq=43
```

HOW DOES HTTP WORK? TRANSACTIONS

- HTTP data transfer is a collection of transactions
- Each transaction is composed of 2 HTTP messages
- Requests are identified by methods
    - Method: The action that client asks from server
- Responses are identified by status codes
    - Status: The result of the requested action

HTTP TRANSACTIONS



HTTP TRANSACTIONS IN WEB

- (Typically) each web page contains multiple resources
    - The main skeleton HTML page
    - Some linked materials: figures, videos, JS, CSS, etc.
- Displaying a web page by a browser
    - Get the HTML page (first transaction)
    - Try to display the page (rendering)
    - Other resources are linked to the page
    - Get the resources (subsequent transactions)

HTTP TRANSACTIONS IN WEB (CONT.)

- HTTP Transactions & TCP Connections
  1. Non-persistent
     - A new TCP connection per object
     - Network overhead + Connection establish delay + Resource intensive
     - Parallel connections speed up browsing
  2. Persistent
     - Get multiple objects using a single TCP connection
     - No extra processing & networking overhead
     - Poor performance if implemented in serial manner
     - Pipeline requests speed up browsing (HTTP/1.1)

HTTP/2 FOR A FASTER WEB

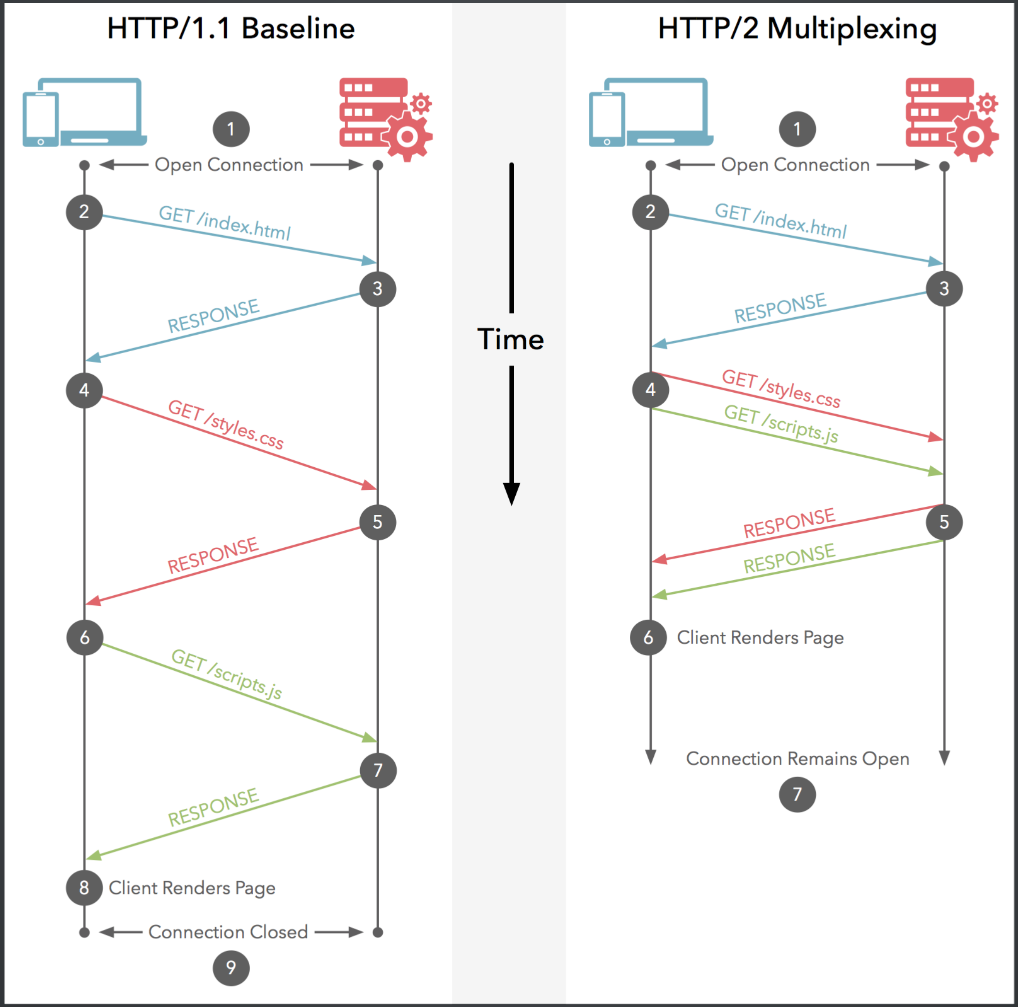
HTTP PIPELINING

- HTTP Pipelining is not difficult to deploy, it is impossible.
- it still allowed a single large or slow response to block all others that followed.

HTTP/2 MULTIPLEXING

- Multiplexing allows multiple request-response messages to be in flight over a single HTTP/2 connection, at the same time.

HTTP/2 FOR A FASTER WEB (CONT.)

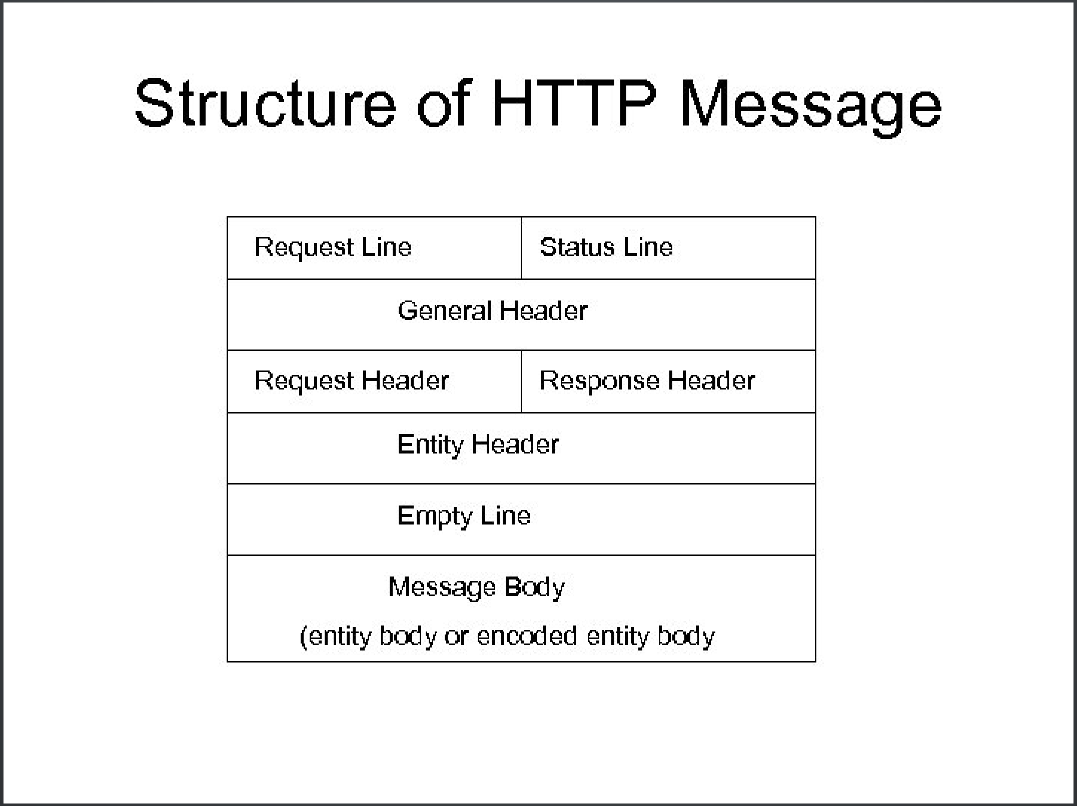


HTTP TRANSACTIONS IN WEB: HANDS ON

- Get a HTML page from a server
- Capture the packets
- Investigate the transactions

HTTP MESSAGES

- HTTP is text-based protocol
  - Human readable headers
  - The header is composed of some lines



Structure of HTTP Message

Request Line	Status Line
General Header	
Request Header	Response Header
Entity Header	
Empty Line	
Message Body (entity body or encoded entity body	

HTTP MESSAGES (CONT.)

- E.g. HTTP request message

```
GET /index.html HTTP/1.1
Host: www.aut.ac.ir
User-Agent: Mozilla/36.0
Accept-Language: en-us
Connection: keep-alive
```

```
Method<sp>Path<sp>Version<CRLF>
Header-Field:Header-Value<CRLF>
...
Header-Field:Header-Value<CRLF>
<CRLF>
Entity-Body
```

- E.g. HTTP response message

```
HTTP/1.1 200 OK
Date: Sun, 02 Oct 2018 20:30:40
Server: Apache/2.2.2
Last-Modified: Mon, 03 May 2017 10:20:22
Connection: keep-alive
Content-Length: 3000

data data data ...
```

```
Version<sp>Code<sp>Reason<CRLF>
Header-Field:Header-Value<CRLF>
...
Header-Field:Header-Value<CRLF>
<CRLF>
Entity-Body
```

HTTP METHODS

- Methods are actions that client asks from server to do on the specified resource (given by the path parameter)
- Which actions?
    - Basic data communication operations
        - Safe operations
            - Get a resource from server
            - Send data to server
        - Unsafe operations
            - Delete a resource on server
            - Create/Replace a resource on server

- Debugging and troubleshooting
    - Get information about a resource
    - Check what the server has got from a client
    - Get the list of operations which can be applied on a resource

HTTP METHODS (CONT.)

- GET: Retrieve resource from server
- HEAD: Similar to GET but the resource itself is not retrieved, just the HTTP response header
    - Useful for debugging or some other applications
- POST: Submit data to be processed by the specified resource
    - Data itself is enveloped in message body

HTTP METHODS (CONT.)

- DELETE: Remove the resource
    - Not popular in web, can be used in other applications
- PUT: Add message body as the specified resource to server
- PATCH: A PATCH request is considered a set of instructions on how to modify a resource. Contrast this with PUT; which is a complete representation of a resource.
- TRACE: Server echoes back the received message
    - For troubleshooting & debugging
- OPTIONS: Request the list of supported methods by server on the resource

HTTP RESPONSES

- The message for the result/response of the requested action
- Which responses?
    - Basic responses
        - Success
        - Failure
            - Bad client request
            - Server problem
            - ...
    - Others
        - E.g., Redirection to other resources

HTTP RESPONSES (CONT.)

- 2xx
    - Successful responses
    - 200: OK
    - 201: Created
    - 204: No Content
- 4xx
    - Client errors
    - 400: Bad Request
    - 401: Unauthorized (Authorization required)
    - 403: Forbidden
    - 404: Not Found
    - 405: Method Not Allowed

HTTP RESPONSES (CONT.)

- 5xx
  - Server errors
  - 500: Internal Server Error
  - 501: Not Implemented
  - 503: Service Unavailable
- 3xx
  - Redirects
  - 301: Moved Permanently
  - 302: Found
    - redirect status response code indicates that the resource requested has been temporarily moved to the URL given by the Location header
  - 307: Moved Temporarily
    - Resource has been moved, Redirection
    - Location header contains the new location of resource
  - 304: Not Modified

HTTP RESPONSES (CONT.)

- 1xx
    - Informational responses
    - 101: Switching Protocol
        - This code is sent in response to an Upgrade request header from the client, and indicates the protocol the server is switching to.

HTTP MESSAGES HANDS ON

- Connect to a web server
    - Telnet can create TCP socket
- Play with the server by sending HTTP methods and checking the responses

HTTP HEADERS

- Headers are additional information that is sent by client to server and vice versa
    - Most (almost all) are optional
- Which headers?
    - Information about client
    - Information about server
    - Information about the requested resource
    - Information about the response
    - Security/Authentication
    - ...

HTTP HEADERS

- General headers
    - Appear both on request & response messages
- Request headers
    - Information about request
- Response headers
    - Information about response
- Entity headers
    - Information about body (size, ...)
- Extension headers
    - New headers (not standard)

GENERAL HEADERS

- Date: Date & Time that message is created
- Connection: Close or Keep-Alive
    - Close: Non-persistent connection
    - Keep-Alive: Persistent connection
- Via: Information about the intermediate nodes between two sides
    - Proxy servers

REQUEST HEADERS

- Host: The name of the server (required, why?)
- Referer: URL that contains requested URL
- Information about the client
    - User-Agent: The client program
    - Accept: The acceptable media types
    - Accept-Encoding: The acceptable encoding
    - Accept-Language: The acceptable language

REQUEST HEADERS (CONT.)

- Range: Specific range (in byte) of resource
- Authorization: Response to the authenticate
  - Will be discussed later
- Cookie: To return back the cookies
  - Will be discussed later
- If-Modified-Since: Request is processed if the objected is modified since the specified time.
  - Used in Web Caching
  - Will be discussed later

RESPONSE HEADER

- Server: Information about server
- WWW-Authenticate: Used to specify authentication parameters by server
- Proxy-Authenticate: Used to specify authentication parameters by proxy
- Set-Cookie: To send a cookie to client
- Location: The location of entity to redirect client

RESPONSE HEADER (CONT.)

- Last-Modified: The date and time of last modification of entity
- Content-Range: Range of this entity in the entire resource
- Expires: The date and time at which the entity will expire

ENTITY HEADERS

- Content-Length: The length of body (in byte)
- Content-Type: The type of entity
    - MIME types: text/xml, image/gif
- Allow: The allowed request methods can be performed on the entity
    - This is in response of OPTIONS method

from “*https://avatars1.githubusercontent.com/u/8181240?v=4*”

```
{
  "cache-control": "max-age=300",
  "content-length": "23504",
  "content-type": "image/jpeg",
  "expires": "Fri, 23 Dec 2022 17:19:59 GMT",
  "last-modified": "Fri, 09 Sep 2022 15:58:35 GMT"
}
```

EXTENSION HEADERS

- Custom proprietary headers have historically been used with an X- prefix, but this convention was deprecated in June 2012
- implementation-specific and private-use parameters could at least incorporate the organization's name
    - ExampleInc-foo
    - VND.ExampleInc.foo (vnd stands for vendor)
- or primary domain name
    - com.example.foo
    - http://example.com/foo

HTTP TOOLS

- cURL
- Postman
- Hoppscotch
- HTTPie

- Introduction
- Cookie
- Proxy & Cache
- Authentication

STATELESS PROBLEM

- HTTP is a stateless protocol
    - Server does not remember it's client
- How to personalize pages (personal portal)?
    - Use http header: X-Forwarded-For
        - Is not usually sent by browsers
    - Find client IP address from TCP connection
        - The problem is NAT
    - Clients move but IP address does not

SOLUTION OF STATELESS PROBLEM: COOKIE [RFC 6265]

- Types
    - Session cookies: To identify a session
    - Persistent cookies: To identify a client (browser)

COOKIES (CONT.)

```
GET /cookies/set?name=parham&family=alvani HTTP/1.1
Host: httpbin.org
```

```
HTTP/1.1 302 FOUND
Date: Mon, 07 Sep 2020 05:19:50 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 223
Connection: keep-alive
Server: gunicorn/19.9.0
Location: /cookies
Set-Cookie: name=parham; Path=/
Set-Cookie: family=alvani; Path=/
```

```
GET /cookies HTTP/1.1
Host: httpbin.org
Cookie: name=parham; family=alvani
```

```
HTTP/1.1 200 OK
Date: Mon, 07 Sep 2020 05:23:53 GMT
Content-Type: application/json
Content-Length: 58
Connection: keep-alive
Server: gunicorn/19.9.0
Access-Control-Allow-Origin: *
Access-Control-Allow-Credentials: true

{
  "cookies": {
    "family": "alvani",
    "name": "parham"
  }
}
```

COOKIES (CONT.)

- Limitation
    - Cannot be used to store large data
    - At least 300 cookies
    - At least 4096 bytes per cookie
    - At least 20 cookies per unique host or domain name
- Cookies are text files
    - No virus spread
- There is not any request from server to read cookies
    - By default cookies are sent by browser
    - Browser checks URL and finds appropriate cookies

COOKIES (CONT.)

- Client can control cookies
    - Disable cookies: no cookie is saved & used
    - View & Delete cookies
- Server can control cookies by its attributes
    - Expiration time
    - Domain
    - Path
    - Security
    - ...

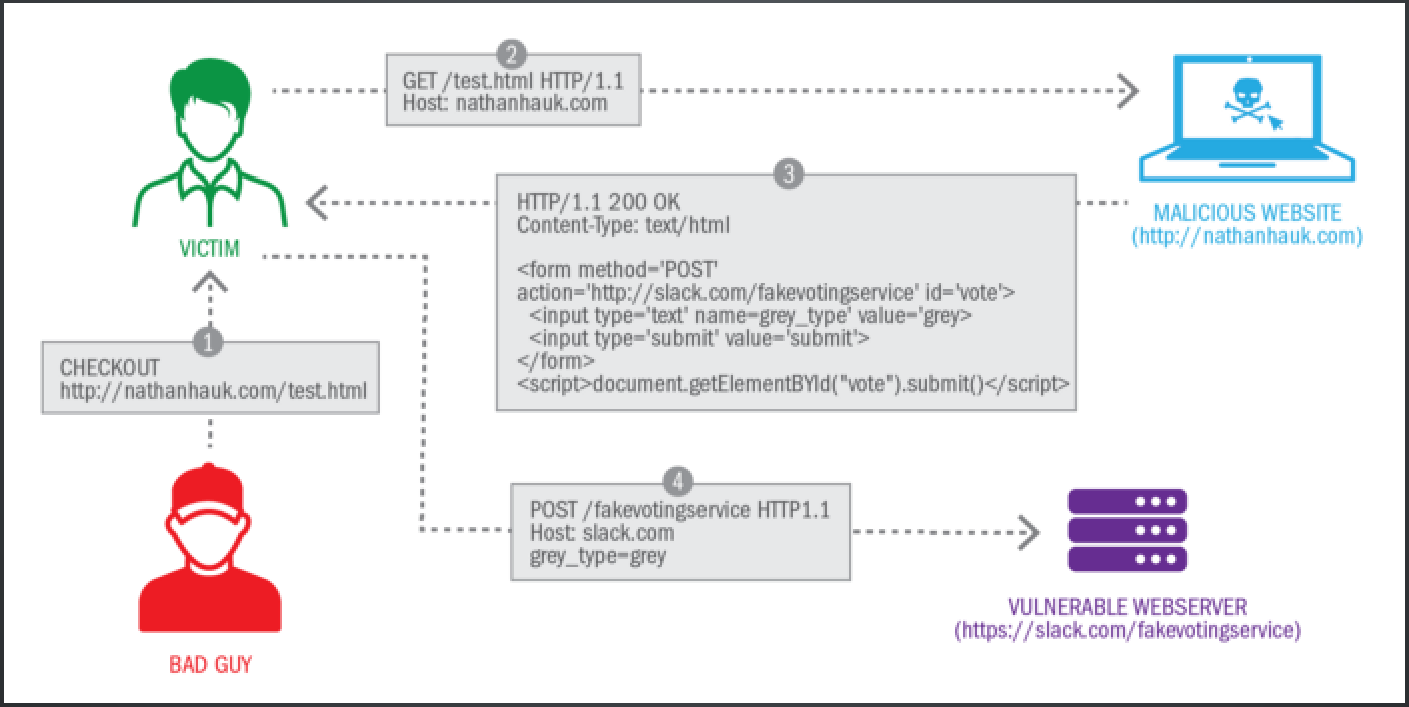
COOKIES ATTRIBUTES

- Expire & Max-Age: The life time of the cookie
    - Expire: An absolute time to delete cookie
    - Max-Age: The maximum life time (sec) of cookie
    - If exists, shows a permanent cookie
    - Send a past time (or negative) to delete a cookie
- Secure: Cookie is sent only if channel is secure
    - Specially useful for login sessions cookies
    - A cookie with the Secure attribute is sent to the server only with an encrypted request over the HTTPS protocol, never with unsecured HTTP (except on localhost), and therefore can't easily be accessed by a man-in-the-middle attacker.

- HttpOnly: Cookie is sent only if HTTP is used
    - JavaScript cannot access to the cookies
- SameSite allows you to declare if your cookie should be restricted to a first-party or same-site context
    - Lax: Cookies are not sent on normal cross-site subrequests (for example to load images or frames into a third party site), but are sent when a user is navigating to the origin site (i.e. when following a link).
    - Strict: Cookies will only be sent in a first-party context and not be sent along with requests initiated by third party websites.
    - None: Cookies will be sent in all contexts, i.e in responses to both first-party and cross-origin requests. If `SameSite=None` is set, the cookie Secure attribute must also be set (or the cookie will be blocked).

CSRF

- Cross-site request forgery (CSRF) attacks rely on the fact that cookies are attached to any request to a given origin, no matter who initiates the request.



1
```
CHECKOUT
http://nathanhauk.com/test.html
```

2
```
GET /test.html HTTP/1.1
Host: nathanhauk.com
```

3
```
HTTP/1.1 200 OK
Content-Type: text/html

<form method='POST'
action='http://slack.com/fakevotingservice' id='vote'>
  <input type='text' name=grey_type' value='grey>
  <input type='submit' value='submit'>
</form>
<script>document.getElementBYId("vote").submit()</script>
```

4
```
POST /fakevotingservice HTTP1.1
Host: slack.com
grey_type=grey
```

WARNING!

Neither Strict nor Lax are a complete solution for your site's security. Cookies are sent as part of the user's request and you should treat them the same as any other user input. That means sanitizing and validating the input. Never use a cookie to store data you consider a server-side secret.

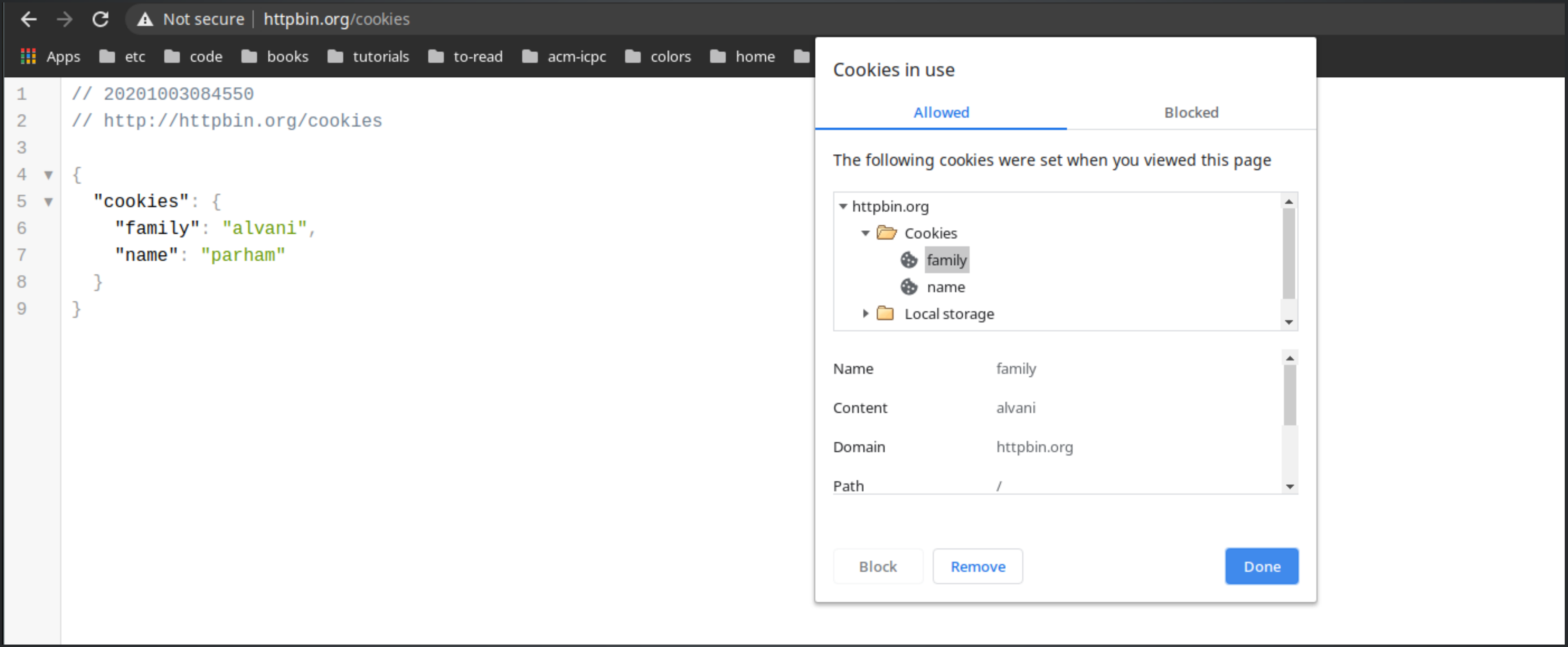
COOKIES ATTRIBUTES: DOMAIN & PATH

- Domain & Path determine the scope of the cookie
    - For which path and domain, the cookie is saved & returned back by browser
- If Domain is omitted, defaults to the host of the current document URL.
    - Browser returns back the cookie for the domain and not sub-domains
- If Path is omitted, defaults to the path of the current document URL.
    - Browser returns back the cookie for the path and also for all sub-paths

- If present then browser checks validity
    - If they are valid then Browser returns back the cookie for that domain & that path and also for all sub-domains and sub-paths

COOKIES ATTRIBUTES: DOMAIN & PATH

- Validity check by major browsers
    - Domain names must start with dot
        - Some browsers accept names without dot as domain
        - Contrary to earlier specifications, leading dots in domain names (.example.com) are ignored.
    - Don’t accept for other domains than the base domain
    - Don’t accept cookies for sub-domains
    - Accept cookies for higher domains
        - Except the top level domains, e.g., .com, .ac.ir
    - Accept cookies for other (sub or higher) paths
        - A path that must exist in the requested URL, or the browser won't send the Cookie header.

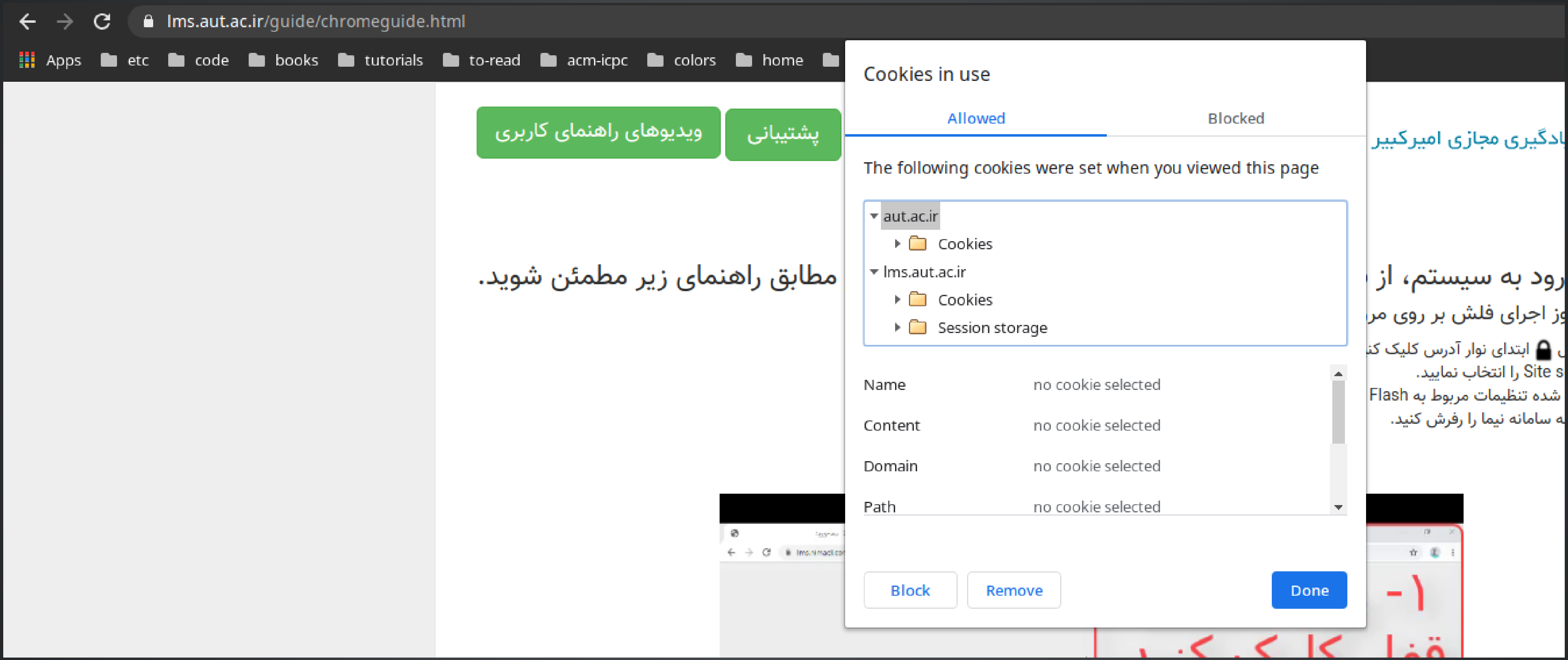


```
// 20201003084550
// http://httpbin.org/cookies

{
  "cookies": {
    "family": "alvani",
    "name": "parham"
  }
}
```

Cookies in use

The following cookies were set when you viewed this page

Name	family
Content	alvani
Domain	httpbin.org
Path	/



lms.aut.ac.ir/guide/chromeguide.html

Apps etc code books tutorials to-read acm-icpc colors home

ویدیوهای راهنمای کاربری
پشتیبانی
یادگیری مجازی امیرکبیر

ورود به سیستم، از مطابق راهنمای زیر مطمئن شوید.

مجوز اجرای فلش بر روی مر

ابتدای نوار آدرس کلیک کن

Site s را انتخاب نمایید.

شده تنظیمات مربوط به Flash

به سامانه نیما را رفرش کنید.

Cookies in use

Allowed Blocked

The following cookies were set when you viewed this page

- aut.ac.ir
  - Cookies
- lms.aut.ac.ir
  - Cookies
  - Session storage

Name	no cookie selected
Content	no cookie selected
Domain	no cookie selected
Path	no cookie selected

Block Remove Done

COOKIES ATTRIBUTES: DOMAIN & PATH: HANDS ON

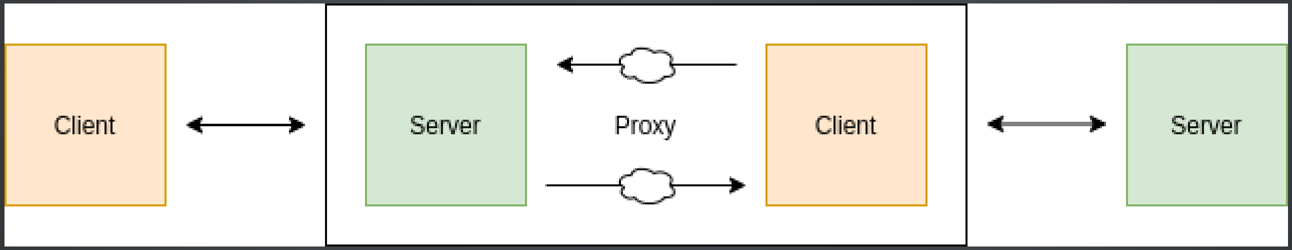
SINGLE SIGN-ON (SSO)

- e.g. CAS Enterprise Single Sign-On

- Introduction
- Cookie
- Proxy & Cache
- Authentication

PROXY

- Proxies sit between client and server
- Act as server for client
- Act as client for server



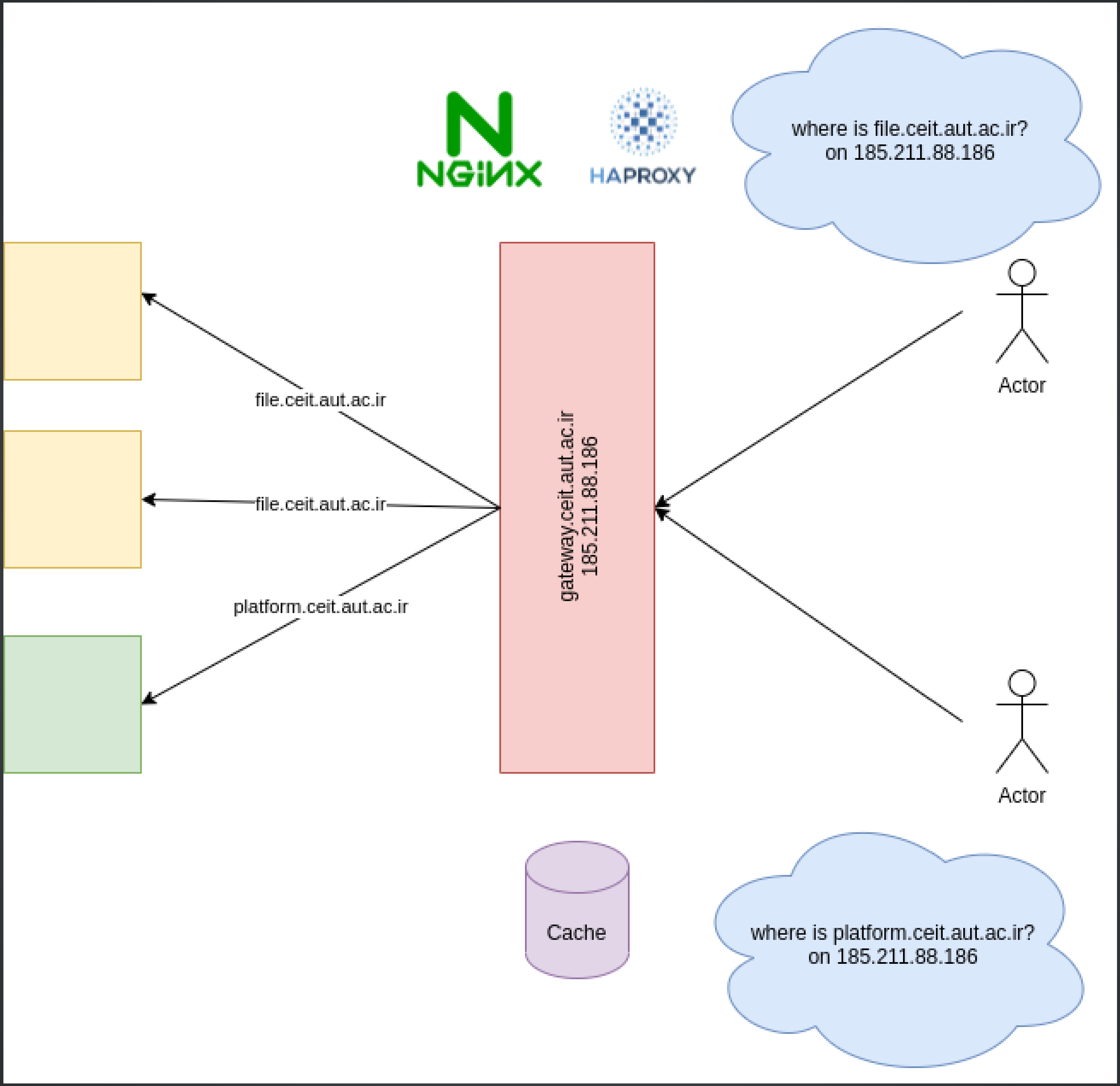
FORWARD PROXIES

- A forward proxy, or gateway, or just "proxy" provides proxy services to a client or a group of clients.

REVERSE PROXIES

- As the name implies, a reverse proxy does the opposite of what a forward proxy does:
- A forward proxy acts in behalf of clients (or requesting hosts), a reverse proxy acts in behalf of servers.
- Forward proxies can hide the identities of clients whereas reverse proxies can hide the identities of servers.

- Reverse proxies have several use cases, a few are:
  - Load balancing: distribute the load to several web servers,
  - Cache static content: offload the web servers by caching static content like pictures,
  - Compression: compress and optimize content to speed up load time.



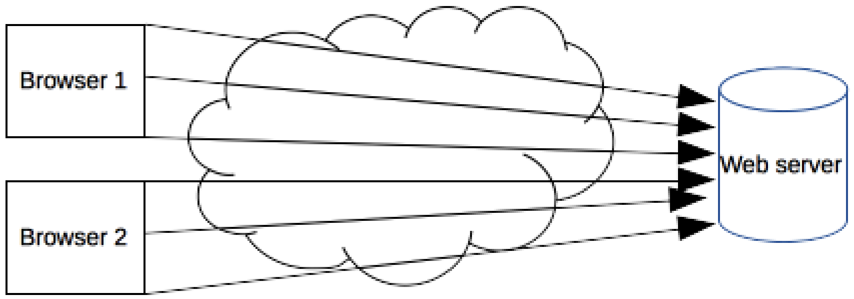
where is file.ceit.aut.ac.ir?
on 185.211.88.186
gateway.ceit.aut.ac.ir
185.211.88.186
file.ceit.aut.ac.ir
file.ceit.aut.ac.ir
platform.ceit.aut.ac.ir
Actor
Actor
Cache
where is platform.ceit.aut.ac.ir?
on 185.211.88.186

HTTP PROXY APPLICATIONS

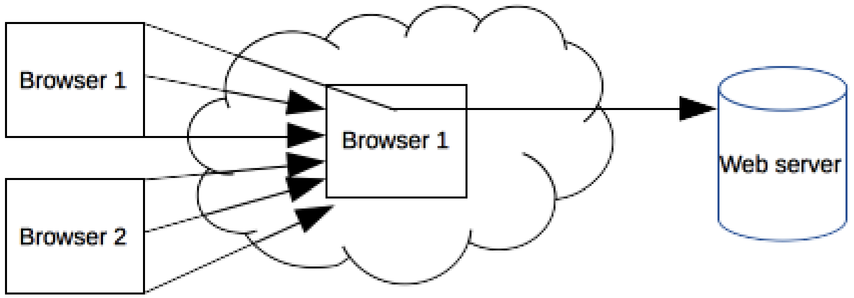
- Authentication
    - Client side: Authenticate clients before they access web
    - Server side: Authenticate clients before access the server
- Accounting: Log client activities
- Security: Analyze request before sending it to server
    - Integrated in modern firewalls
- Filtering: Limit access to specified contents
- Anonymizer: Anonymous web browsing
- Caching (more details in the following slide)

CACHING

- Caching: save a copy of a resource and use it instead of requesting server
- Browser has its own local caches
- Cache server is special proxy for caching
- Benefits
    - Reduce redundant data transfer
    - Reduce network bottleneck
    - Reduce load on server
    - Reduce delay

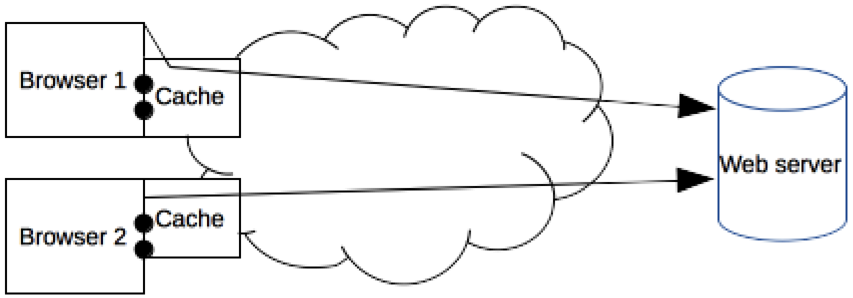


All identical requests are going through to the server.



The first request is going through.

Subsequent identical requests are served by the shared cache.
(more efficient)



The first request of each client is going through.

Subsequent identical requests are not even sent, but served by the local cach.
(most efficient, except for first requests)

CACHING ALGORITHM

- If the object is not cached, it is got from server, saved in cache, and sent to client
- Else, if object is in cache
    - Cache server must return only fresh objects
    - Freshness check
- Objects life-time specified by server
- The `Cache-Control` HTTP/1.1 general-header field is used to specify directives for caching mechanisms in both requests and responses.

EXPIRATION

the maximum amount of time a resource will be considered fresh.

```
Cache-Control: max-age=<seconds>
```

NO CACHING

The cache should not store anything about the client request or server response.

```
Cache-Control: no-store
```

CACHE BUT REVALIDATE

A cache will send the request to the origin server for validation before releasing a cached copy.

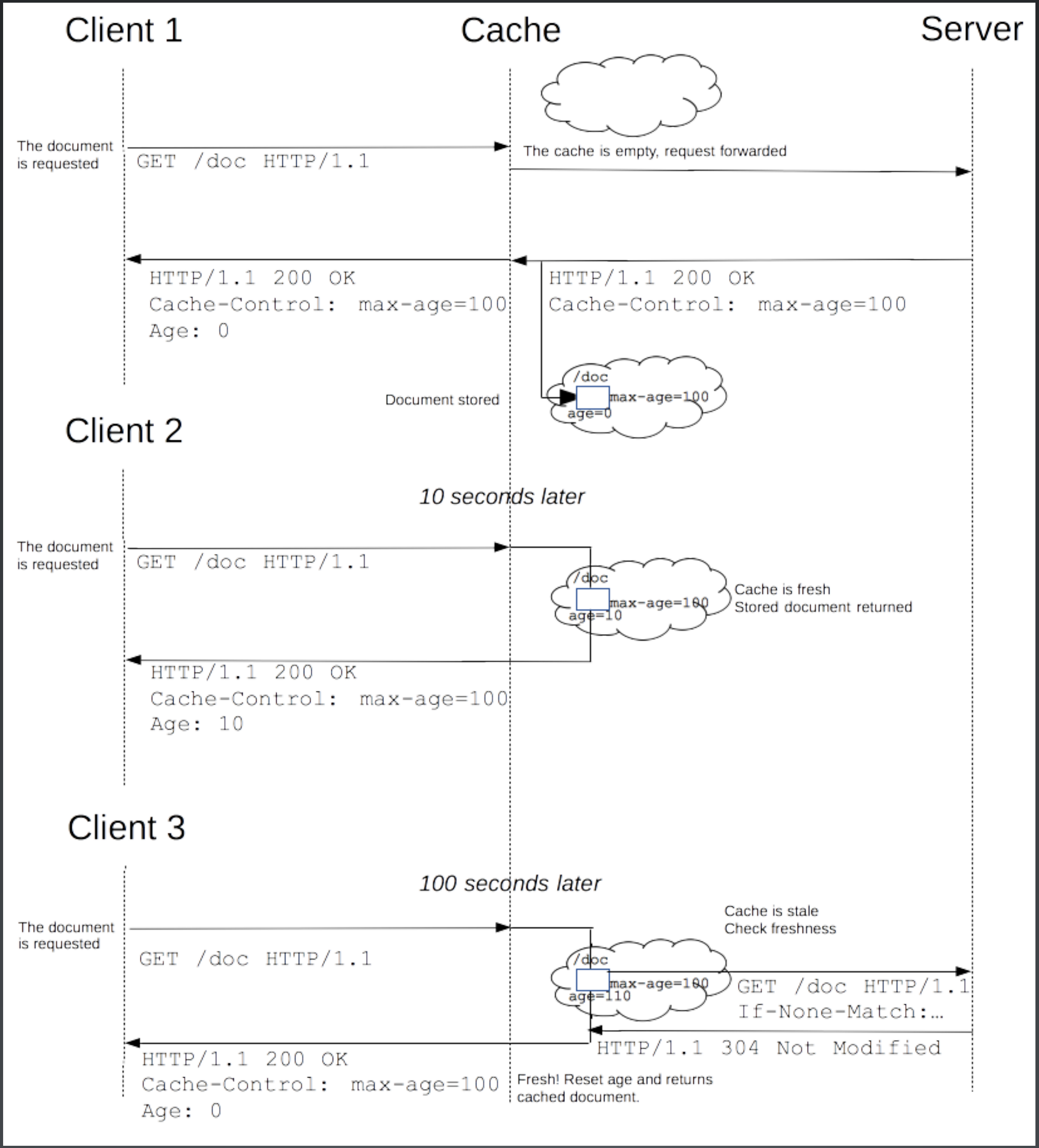
```
Cache-Control: no-cache
```

- If requested object is not expired
    - Cache server gives it to client

CACHING ALGORITHM (CONT.)

- If requested object is expired
    - Its freshness must be checked
- Freshness is checked by conditional request
    - `If-Modified-Since`: current last-modified time
    - `If-None-Match`: the server will send back the requested resource, with a 200 status, only if it doesn't have an ETag matching the given ones.
        - The `ETag` HTTP response header is an identifier for a specific version of a resource.

- Server responses
    - 304 Not modified response + new expire time
        - Cached copy is valid until the specified time
    - 200 OK
        - Server provides a new version of the object
        - Cache server updates cached copy


Client 1
Cache
Server
The document is requested
GET /doc HTTP/1.1
The cache is empty, request forwarded
HTTP/1.1 200 OK
Cache-Control: max-age=100
Age: 0
HTTP/1.1 200 OK
Cache-Control: max-age=100
/doc
max-age=100
age=0
Document stored
Client 2
10 seconds later
The document is requested
GET /doc HTTP/1.1
/doc
max-age=100
age=10
Cache is fresh
Stored document returned
HTTP/1.1 200 OK
Cache-Control: max-age=100
Age: 10
Client 3
100 seconds later
Cache is stale
Check freshness
The document is requested
GET /doc HTTP/1.1
/doc
max-age=100
age=110
GET /doc HTTP/1.1
If-None-Match:…
HTTP/1.1 304 Not Modified
HTTP/1.1 200 OK
Cache-Control: max-age=100
Age: 0
Fresh! Reset age and returns cached document.

- Introduction
- Cookie
- Proxy & Cache
- Authentication

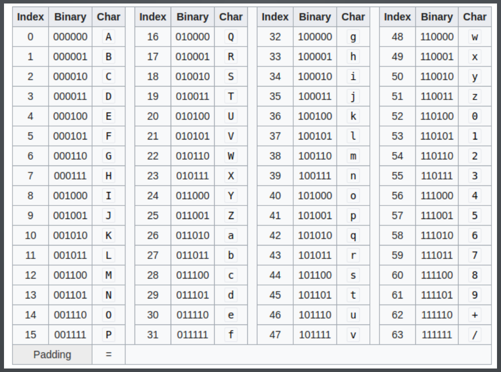
AUTHENTICATION VS AUTHORIZATION

- Authentication is the verification of the credentials of the connection attempt.
- This process consists of sending the credentials from the remote access client to the remote access server in an either plaintext or encrypted form by using an authentication protocol.
- Authentication is the verification that the connection attempt is allowed.
- Authorization occurs after successful authentication.

HTTP AUTHENTICATION

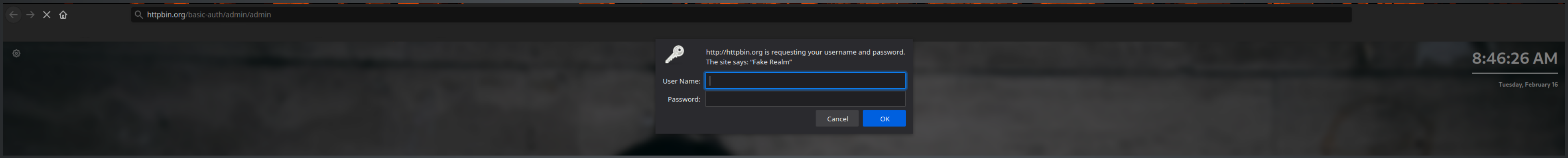
BASE64

- Base64 is a group of binary-to-text encoding schemes that represent binary data in an *ASCII string* format by translating it into a radix-64 representation.



Index	Binary	Char	Index	Binary	Char	Index	Binary	Char	Index	Binary	Char
0	000000	A	16	010000	Q	32	100000	g	48	110000	w
1	000001	B	17	010001	R	33	100001	h	49	110001	x
2	000010	C	18	010010	S	34	100010	i	50	110010	y
3	000011	D	19	010011	T	35	100011	j	51	110011	z
4	000100	E	20	010100	U	36	100100	k	52	110100	0
5	000101	F	21	010101	V	37	100101	l	53	110101	1
6	000110	G	22	010110	W	38	100110	m	54	110110	2
7	000111	H	23	010111	X	39	100111	n	55	110111	3
8	001000	I	24	011000	Y	40	101000	o	56	111000	4
9	001001	J	25	011001	Z	41	101001	p	57	111001	5
10	001010	K	26	011010	a	42	101010	q	58	111010	6
11	001011	L	27	011011	b	43	101011	r	59	111011	7
12	001100	M	28	011100	c	44	101100	s	60	111100	8
13	001101	N	29	011101	d	45	101101	t	61	111101	9
14	001110	O	30	011110	e	46	101110	u	62	111110	+
15	001111	P	31	011111	f	47	101111	v	63	111111	/
Padding		=									

HTTP AUTHENTICATION (CONT.)


httpbin.org/basic-auth/admin/admin

http://httpbin.org is requesting your username and password. The site says: "Fake Realm"

User Name:

Password:

HTTP AUTHENTICATION (CONT.)

```
GET /basic-auth/admin/admin HTTP/1.1
Host: httpbin.org
Authorization: Basic YWRtaW46YWRtaW4=
```

```
HTTP/1.1 200 OK
Date: Mon, 07 Sep 2020 14:14:25 GMT
Content-Type: application/json
Content-Length: 48
Connection: keep-alive
Server: gunicorn/19.9.0
Access-Control-Allow-Origin: *
Access-Control-Allow-Credentials: true

{
  "authenticated": true,
  "user": "admin"
}
```

DIGEST AUTHENTICATION

- Basic authentication is insecure
  - Password is sent in base64 encoding
  - Attacker can easily find it
- Digest authentication: Don’t send password
  - Send its **digest** (hash)
- Digest/hash function
  - One way function, irreversible
- Attacker cannot find password 😌
- But! Reply attack 😞
  - Attacker resends the same digest then he will be authenticated
  - Use Nonce
- Digest authentication uses nonce and digest together

DIGEST AUTHENTICATION (CONT.)

- Client requests a private resource
- Server creates a nonce (the server only issues a new nonce for each 401 response)

```
WWW-Authenticate: Digest realm="testrealm@host.com",
                        qop="auth,auth-int",
                        nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",
                        opaque="5ccc069c403ebaf9f0171e9517f40e41"
```

- Client computes digest of password & nonce

```
HA1 = MD5(username:realm:password)
HA2 = MD5(method:digestURI)
response = MD5(HA1:nonce:HA2)
```

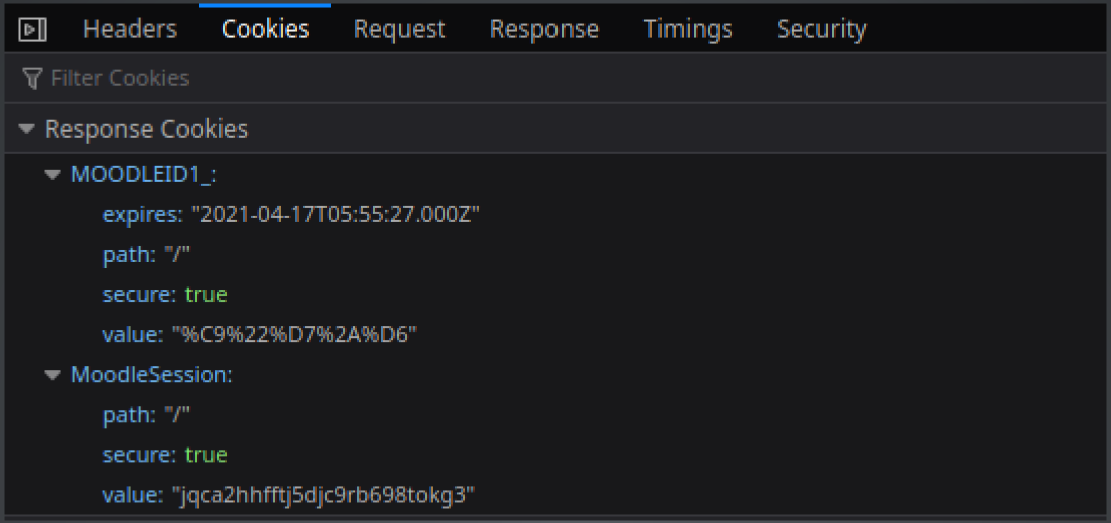
```
Authorization: Digest username="Mufasa",
                     realm="testrealm@host.com",
                     nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",
                     uri="/dir/index.html",
                     qop=auth,
                     nc=00000001,
                     cnonce="0a4f113b",
                     response="6629fae49393a05397450978507c4ef1",
                     opaque="5ccc069c403ebaf9f0171e9517f40e41"
```

- Server looks up the password of username and computes `hash(pass, nonce)`
- More details in RFC7616

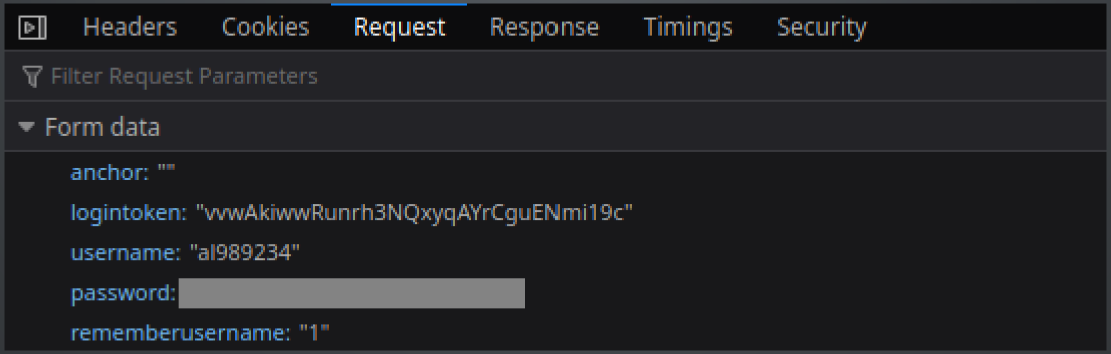
IN THE WILD

- There is no need for basic/digest authentication in our new world we can send **plaintext** passwords over **secure** HTTP.
- In both ways, we need a solution for stateful connection to not repeat the authentication procedure for every request.

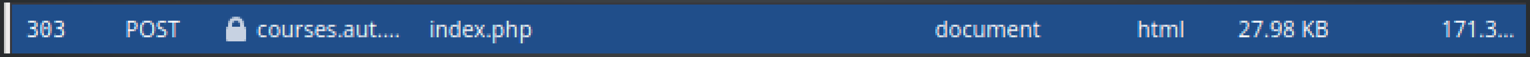
COURSES.AUT.AC.IR



```
Response Cookies
  MOODLEID1_:
    expires: "2021-04-17T05:55:27.000Z"
    path: "/"
    secure: true
    value: "%C9%22%D7%2A%D6"
  MoodleSession:
    path: "/"
    secure: true
    value: "jqca2hhfftj5djc9rb698tokg3"
```



```
Form data
  anchor: ""
  logintoken: "vvwAkiwwRunrh3NQxyqAYrCguENmi19c"
  username: "al989234"
  password: [redacted]
  rememberusername: "1"
```



```
303   POST   courses.aut....   index.php        document   html   27.98 KB      171.3...
```

BEARER AUTHENTICATION

- Bearer authentication (also called token authentication) is an HTTP authentication scheme that involves security tokens called bearer tokens
- The name Bearer authentication can be understood as *“give access to the bearer of this token.”*

JWT

- JSON Web Tokens are an open, industry standard RFC 7519 method for representing claims securely between two parties.

SAMPLE TOKEN

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IlBhcmhhbSBBbHZhbmkiLCJpYXQiOjE1MTYyMzkwMjIsIn
gWWHu5Ps_F6lbqJRBXkNjEk_-0QdLhN9l2MNjWOcj90
```

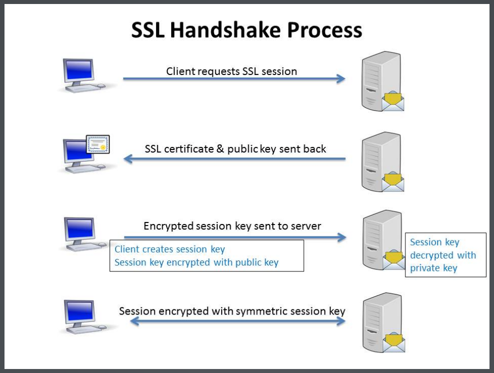
```json
{
  "alg": "HS256",
  "typ": "JWT"
}

{
  "sub": "1234567890",
  "name": "Parham Alvani",
  "iat": 1516239022,
  "project": "an awesome project"
}
```

SNAPP! TOKEN

```
{
  "alg": "RS512",
  "kid": "z8a4l4oOFEqgehRYDBZP+fprPnLDLmabkslOxVVpLNE",
  "typ": "JWT"
}
{
  "aud": [
    "passenger"
  ],
  "email": "parhamalvani@gmail.com",
  "exp": 1646469738,
  "iat": 1645260138,
  "iss": 1,
  "jti": "2NFKm5FfEey65wIArBQAz289hDgf/E0gjnyXrNCM0v4",
  "sid": "25JzmlUBAwtMfQvT7qmOalw5M7p",
  "sub": "KpQxO5glyv04Ad1"
}
```

SECURITY

- Digest authentication protect password only
- Data is completely insecure
- No mechanism in HTTP to protect data
- HTTP over SSL/TLS is the popular solution
    - An encrypted tunnel between client & server
    - Send HTTP traffic through the tunnel


SSL Handshake Process
Client requests SSL session
SSL certificate & public key sent back
Encrypted session key sent to server
Client creates session key
Session key encrypted with public key
Session key decrypted with private key
Session encrypted with symmetric session key

REFERENCES 📚

- Prof. Bahador Bakhshi's Internet Eng. Course's Slides
- HTTP: Learn your browser's language!
- What is the difference between a URI, a URL and a URN?
- HTTP/2 for a Faster Web
- REST: Good Practices for API Design
- What is the maximum size of a cookie, and how many can be stored in a browser for each web site?
- Base64
- Digest access authentication