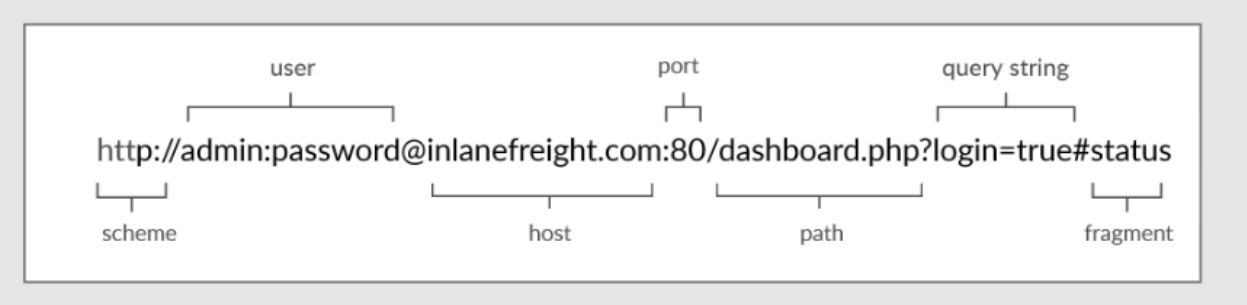
HTTP Fundamentals

HTTP əlaqəsi müştəri (client) və serverdən ibarətdir. Müştəri serverdən hər hansı resursu tələb edir. Server isə bu tələbi emal edir və istənilən resursu geri qaytarır. HTTP ünsiyyəti üçün standart port 80-dir, amma bu, veb serverin konfiqurasiyasına görə başqa porta da dəyişdirilə bilər. İnternetdən müxtəlif saytlara daxil olduğumuz zaman da eyni sorğulardan istifadə olunur. İstədiyimiz sayta çatmaq üçün tam domen adını (FQDN) URL şəklində daxil edirik, məsələn: **www.example.com**.

URL (Uniform Resource Locator) — internetdəki resursların ünvanıdır. Sadə dillə desək, bu, brauzerdə yazdığımız veb ünvanıdır. HTTP üzərindən resurslara **URL** vasitəsilə daxil oluruq. URL sadəcə ziyarət etmək istədiyimiz saytı göstərməkdən daha çox məlumat təqdim edir. Bir URL-in quruluşu:



scheme - Müştərinin istifadə etdiyi **protokolu** müəyyən etmək üçün istifadə olunur və **iki slash (://)** ilə bitir.

user info - Bu, **istəyə bağlı** bir hissədir və daxilində **etibarlılıq məlumatlarını (credential)** saxlayır (**istifadəçi adı:şifrə** formatında). **@ işarəsi** ilə **host** hissəsindən ayrılır.

host - **Resursun yerləşdiyi ünvanı** göstərir. Bu, **host adı** və ya **IP ünvanı** ola bilər.

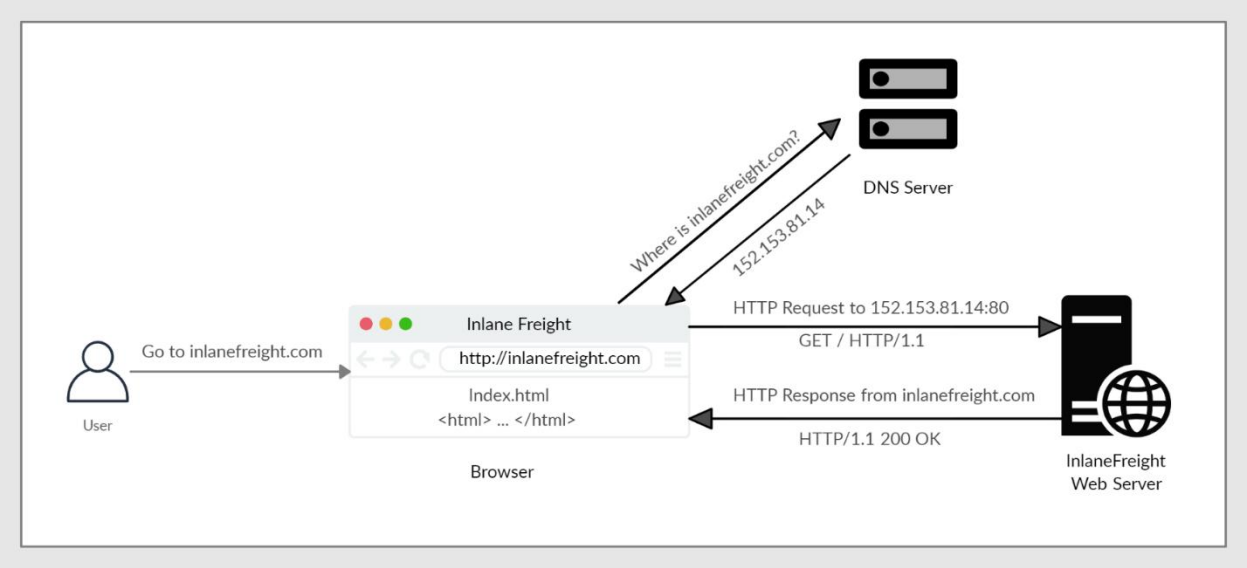
port - **Host** hissəsindən **iki nöqtə (:)** ilə ayrılır. Əgər **port** göstərilməyibsə, **http** üçün standart port **80**, **https** üçün isə **443** olur.

path - **Əldə ediləcək resursa** gedən yolu göstərir. Bu, **fayl** və ya **qovluq** ola bilər. Əgər yol göstərilməyibsə, server **default index** (məsələn, **index.html**) qaytarır.

query string - **Sual işarəsi (?)** ilə başlayır və **parametr** (məsələn, **login**) və **dəyər** (məsələn, **true**) cütündən ibarətdir. Birdən çox parametr **ampersand (&)** ilə ayrılır.

Fragments - **Brauzer** tərəfindən **müştəri tərəfində** işlənir və əsas resursun içində müəyyən hissəni (məsələn, **başlıq** və ya **bölmə**) göstərir.

HTTP Flow



Yuxarıdakı diaqram **HTTP sorğusunun quruluşunu** təqdim edir. **İstifadəçi** ilk dəfə **URL** (**inlanefreight.com**) brauzerə daxil etdikdə, brauzer **DNS (Domain Name System)** serverinə sorğu göndərir ki, domeni **həll etsin** və ona uyğun **IP ünvanını** əldə etsin. **DNS server** isə **inlanefreight.com** üçün **IP ünvanını tapır** və geri qaytarır. Bütün **domen adları** bu şəkildə həll olunmalıdır, çünki server **IP ünvanı olmadan** əlaqə qura bilmir.

Qeyd: Brauzerlər əvvəlcə lokal '/etc/hosts' faylını yoxlayır. Əgər tələb olunan domen orada yoxdursa, onda digər DNS serverləri ilə əlaqə qurulur. Biz '/etc/hosts' faylını istifadə edərək DNS resolation əllə həyata keçirə bilərik, yəni əvvəl IP ünvanı, sonra isə domen adı yazmaqla.

Brauzer **tələb olunan domenə bağlı IP ünvanını** əldə etdikdən sonra, **default HTTP portuna** (məsələn, **80**) **GET sorğusu** göndərir və **root yolunu (/)** tələb edir. Sonra **veb server** sorğunu qəbul edir və emal edir. **Standart olaraq**, serverlər **/** üçün sorğu alanda **index faylını** qaytaracaq şəkildə qurulur.

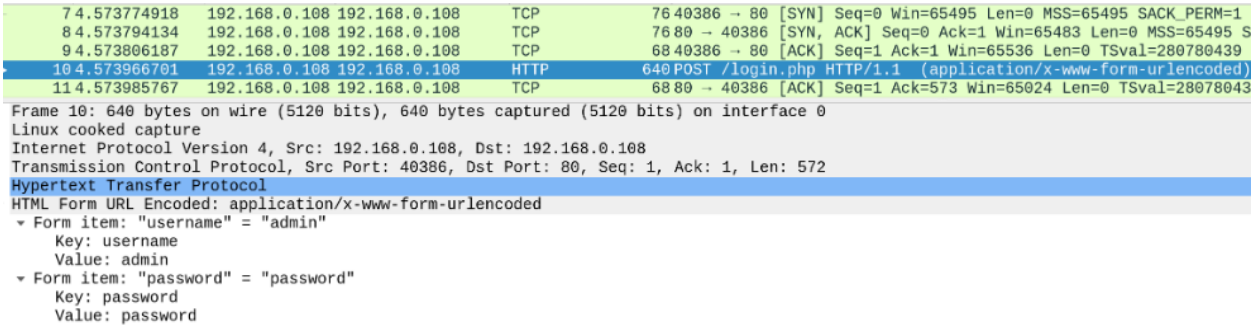
Bu nümunədə **index.html faylının məzmunu** oxunur və **veb server tərəfindən HTTP cavabı** kimi geri göndərilir. **Cavab** həmçinin **status kodunu** (məsələn, **200 OK**) da ehtiva edir ki, bu da sorğunun uğurla icra edildiyini göstərir. Daha sonra **veb brauzer index.html məzmununu** göstərir və istifadəçiyə təqdim edir.

HTTPS Overview

Yuxarıda **HTTP sorğularının necə göndərildiyini və emal edildiyini** müzakirə etdik. Lakin **HTTP-nin əsas çatışmazlıqlarından biri** budur ki, bütün **məlumatlar açıq mətn (clear-text)** şəklində ötürülür. Bu isə o deməkdir ki, **mənbə** ilə **təyinat nöqtəsi** arasında olan istənilən şəxs **Man-in-the-Middle (MiTM) hücumu** edərək ötürülən məlumatı görə bilər.

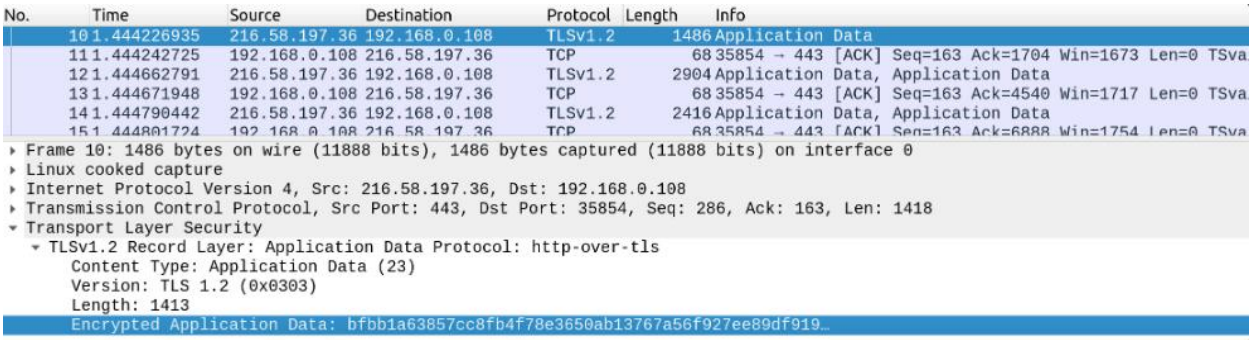
Bu problemi aradan qaldırmaq üçün **HTTPS (HTTP Secure)** protokolu yaradıldı. Bu protokolda bütün əlaqələr **şifrələnmiş formada** ötürülür. Buna görə də, üçüncü tərəf sorğunu ələ keçirsə belə, **məlumatı oxuya bilməyəcək**. Bu səbəbdən **HTTPS** internetdəki saytlar üçün əsas sxemə çevrilib, **HTTP isə tədricən aradan çıxarılır** və yaxın zamanda əksər **brauzerlər HTTP saytlarına girişə icazə verməyəcək**.

Əgər bir **HTTP sorğusunu** araşdırsaq, **veb brauzer** ilə **veb tətbiq** arasında **təhlükəsiz əlaqə təmin edilmədikdə** yaranan vəziyyəti görə bilərik. Məsələn, aşağıda bir **HTTP login sorğusunun** məzmunu göstərilib:

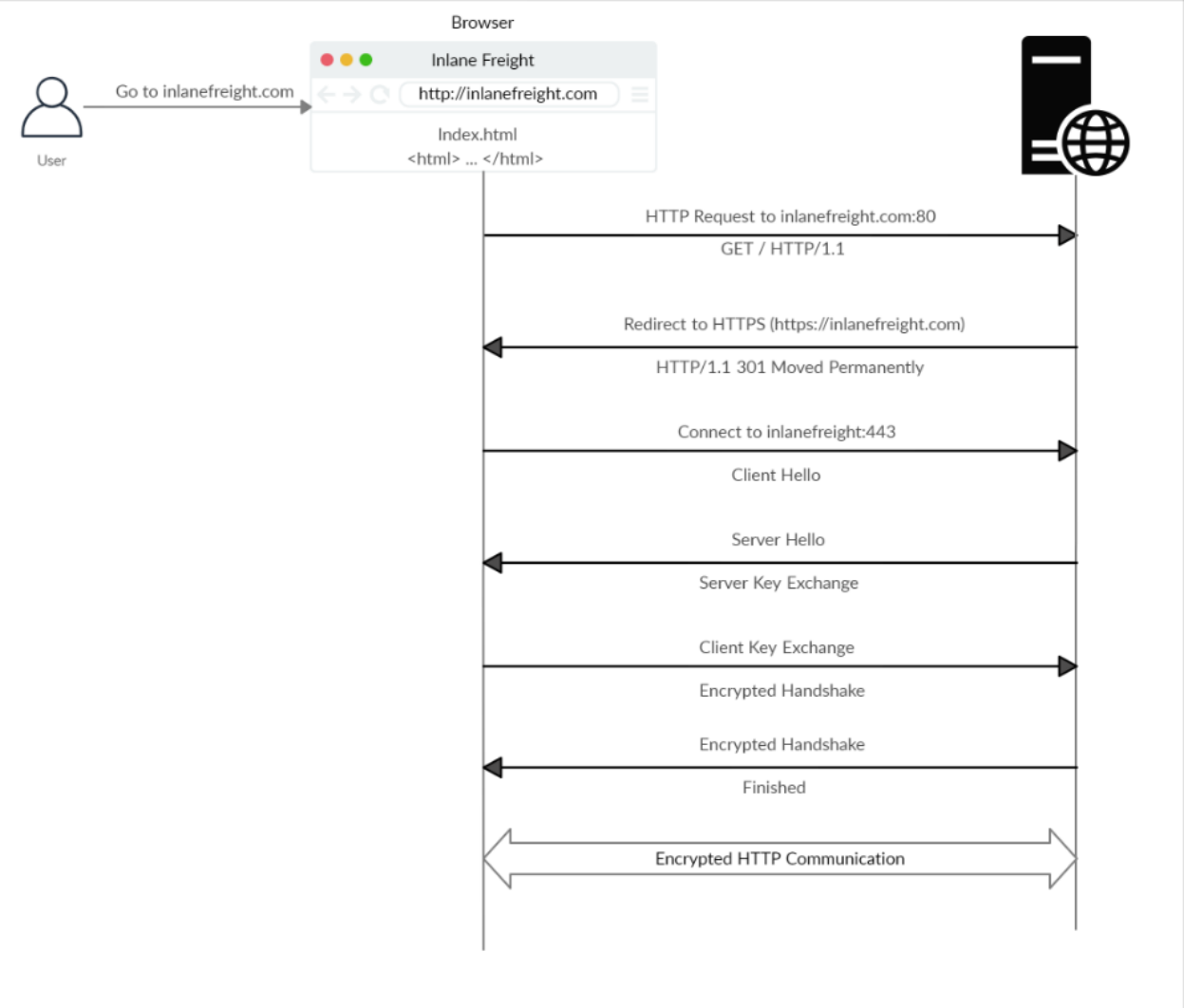


Biz görürük ki, **login məlumatları (istifadəçi adı və şifrə) açıq mətn** şəklində görünür. Bu isə eyni şəbəkədə olan (məsələn, **ictimai Wi-Fi şəbəkəsi**) birinin sorğunu ələ keçirib həmin **məlumatlardan pis niyyətlə istifadə etməsini** asanlaşdırır.

Buna qarşılıq olaraq, **kiminsə HTTPS sorğusunun trafikini ələ keçirib analiz etdiyini** fərz etsək, o zaman görünən məlumat təxminən belə olacaq:



HTTPS Flow



Əgər **https://** əvəzinə **http://** yazaraq **HTTPS-i məcburi tətbiq edən** bir sayta daxil olmağa çalışsaq, brauzer əvvəlcə **domeni həll etməyə çalışır** və istifadəçini həmin saytı saxlayan **veb serverə yönləndirir**. İlk sorğu **port 80-ə** (yəni **şifrələnməmiş HTTP protokolu**) göndərilir. Server bunu aşkar edir və müştərini **HTTPS-in təhlükəsiz portu olan 443-ə** yönləndirir. Bu yönləndirmə **301 Moved Permanently** cavab kodu ilə edilir. Daha sonra **müştəri (veb brauzer) "client hello"** adlı paket göndərir və öz məlumatlarını bildirir. Ardınca server **"server hello"** cavabı verir və **SSL sertifikatlarının mübadiləsi** üçün açar mübadiləsi (key exchange) həyata keçirilir. **Müştəri açarı/sertifikatı yoxlayır** və öz açarını göndərir. Bundan sonra **şifrələnmiş handshake** başlayır ki, bu da **şifrələmənin və ötürmənin düzgün işlədiyini təsdiqləyir**.

Handshake uğurla başa çatdıqdan sonra **normal HTTP ünsiyyəti davam edir**, lakin bundan sonra bütün məlumatlar **şifrələnmiş şəkildə ötürülür**.

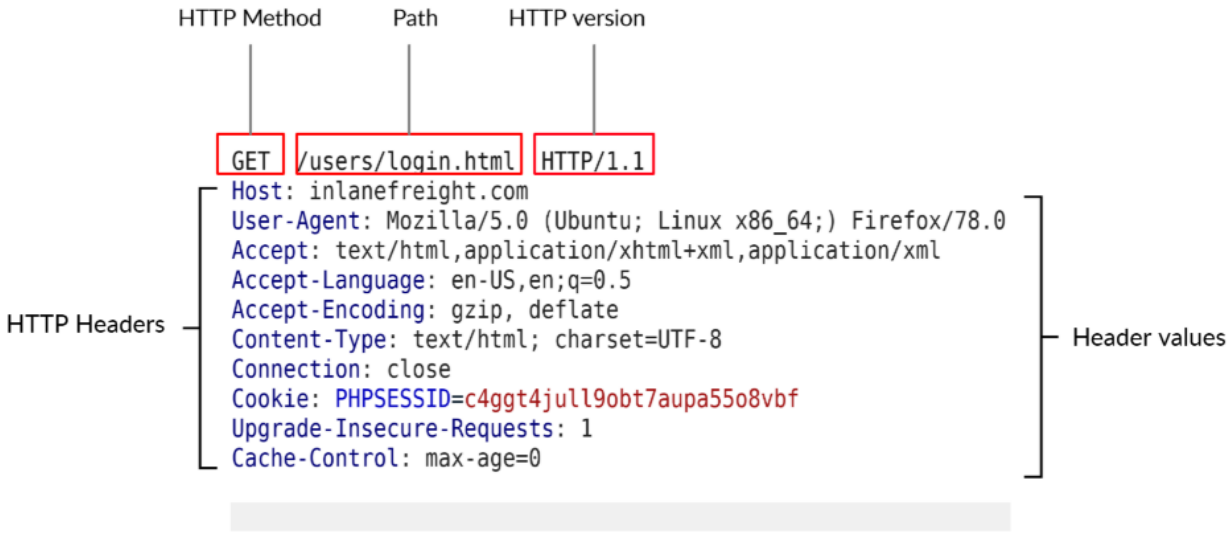
Qeyd: Müəyyən hallarda hücumçu HTTP downgrade attack həyata keçirə bilər. Bu hücum HTTPS ünsiyyətini HTTP-yə endirir, nəticədə məlumatlar yenidən açıq mətn şəklində ötürülür. Bu, istifadəçidən xəbərsiz MITM (Man-in-the-Middle) proxy qurmaqla, bütün trafiki hücumçunun hostu vasitəsilə keçirməklə edilir. Lakin əksər müasir brauzerlər, serverlər və veb tətbiqləri bu hücumdan qorunma mexanizmlərinə malikdir.

HTTP Request and Response

HTTP connection əsasən **HTTP sorğusundan** və **HTTP cavabından** ibarətdir. **HTTP sorğusu** müştəri (**məsələn, cURL və ya brauzer**) tərəfindən göndərilir və **server** (**məsələn, veb server**) tərəfindən emal olunur. Sorğular **serverdən tələb etdiyimiz bütün detalları** ehtiva edir, məsələn: **resurs (URL, yol, parametrlər)**, hər hansı **sorğu məlumatı**, **başlıqlar (headers)** və ya **seçimlər**.

Server HTTP sorğusunu aldıqdan sonra onu **emal edir** və **HTTP cavabı** göndərir. Bu cavab **status kodunu** və əgər sorğunu göndərən şəxs resursa çıxış icazəsinə malikdirsə, **resurs məlumatını** ehtiva edə bilər.

HTTP Request



Yuxarıdakı şəkildə **HTTP GET sorğusu** göstərilir və bu sorğu aşağıdakı URL-ə göndərilir:

http://inlanefreight.com/users/login.html

Hər bir HTTP sorğusunun ilk sətri boşluqlarla ayrılmış üç əsas sahədən ibarətdir:

Method (GET) - **HTTP metodu** və ya **verb**, hansı əməliyyatın icra ediləcəyini göstərir.

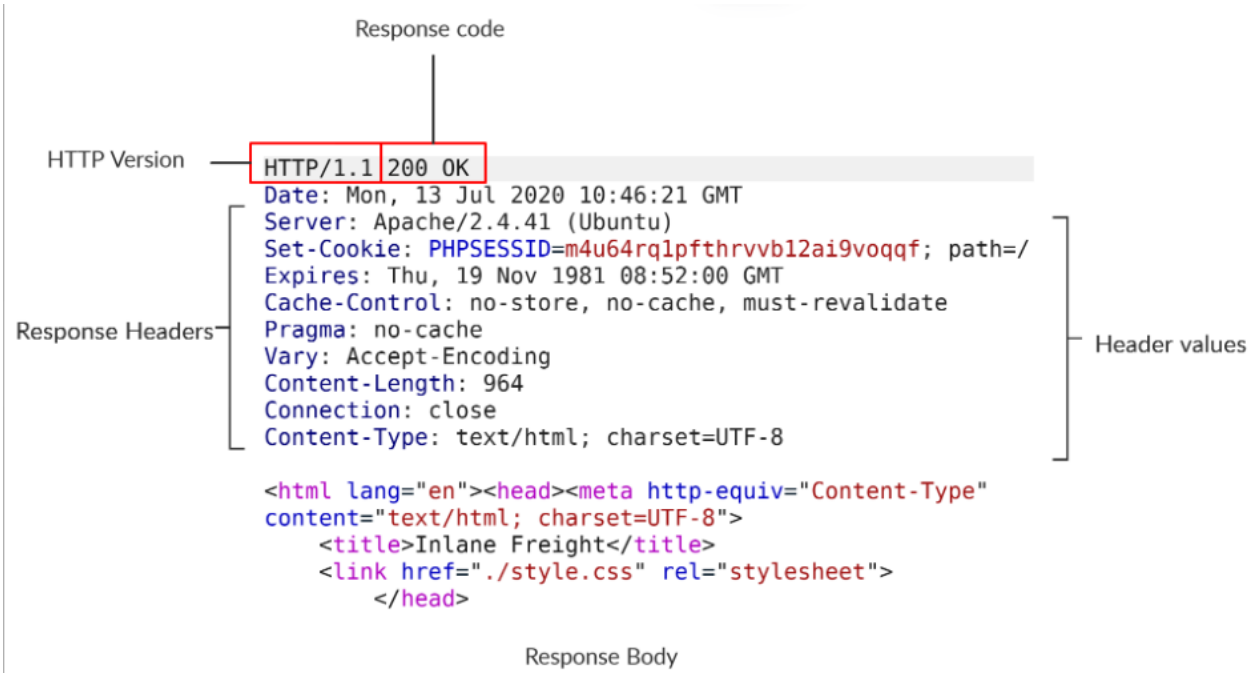
Path (/users/login.html) - **Əldə ediləcək resursa** gedən yol. Bu sahəyə əlavə olaraq **query string** də əlavə edilə bilər (məsələn: **?username=user**).

Version (HTTP/1.1) - Üçüncü və son sahədir, **HTTP versiyasını** göstərir.

Bundan sonrakı sətirlərdə **HTTP başlıqları (headers)** yerləşir, məsələn: **Host**, **User-Agent**, **Cookie** və digər mümkün başlıqlar. Bu başlıqlar sorğunun müxtəlif atributlarını göstərmək üçün istifadə olunur. **Başlıqların sonu yeni sətirlə (newline) tamamlanır**, çünki server sorğunu təsdiqləmək üçün buna ehtiyac duyur. Son olaraq sorğu, **sorğu gövdəsi (request body)** və **məlumatla** bitə bilər.

HTTP Response

Server sorğumuzu aldıqdan sonra cavabını göndərir. Aşağıda misal bir HTTP response göstərilmişdir:



HTTP cavabının ilk sətri boşluqla ayrılmış iki sahədən ibarətdir:

- **Birinci sahə**: **HTTP versiyası** (məsələn, **HTTP/1.1**).
- **İkinci sahə**: **HTTP cavab kodu** (məsələn, **200 OK**).

Cavab kodları sorğunun vəziyyətini müəyyən etmək üçün istifadə olunur. **İlk sətrin ardından** cavab öz **başlıqlarını (headers)** göstərir, bu da **HTTP sorğusundakı başlıqlara** bənzəyir.

Nəhayət, cavab **başlıqlardan sonra yeni sətr (newline)** ilə ayrılmış **cavab gövdəsi (response body)** ilə bitə bilər. **Cavab gövdəsi** adətən **HTML kodu** olur. Lakin cavab həm də digər formatlarda ola bilər, məsələn:

- **JSON** məlumatı,
- **veb resursları** (şəkillər, stil faylları, skriptlər),
- hətta **sənədlər** (məsələn, **PDF** faylları) ki, bunlar **veb serverdə yerləşdirilib**.

HTTP Methods and Codes

HTTP bir resursa daxil olmaq üçün bir neçə **metodu** dəstəkləyir. **HTTP protokolunda** müxtəlif sorğu metodları mövcuddur ki, onlar **brauzerin serverə məlumat, formalar və ya fayllar göndərməsinə** imkan verir. Bu metodlar, digər şeylərlə yanaşı, **serverə sorğunu necə emal etməyi** və **necə cavab verməyi** göstərmək üçün istifadə olunur. Bunlar aşağıdakılardır:

GET - Müəyyən bir **resursu tələb edir**. Əlavə məlumat **URL-də query string vasitəsilə** serverə ötürülə bilər (məsələn: **?param=value**).

POST - **Məlumatı serverə göndərir**. Bu metod **mətn, PDF və digər ikili fayllar (binary data)** kimi müxtəlif giriş növlərini emal edə bilir. Bu məlumat **başlıqlardan (headers) sonra sorğu gövdəsinə əlavə olunur**. **POST metodu** adətən **informasiya göndərmək (formalar/loginlər)** və ya **sayta fayl (şəkillər, sənədlər) yükləmək** üçün istifadə olunur.

HEAD - Serverə **GET sorğusu göndəriləndə qaytarılacaq başlıqları** tələb edir. **Sorğu gövdəsini qaytarmır** və adətən **resurs yüklənmədən əvvəl cavabın uzunluğunu yoxlamaq üçün** istifadə olunur.

PUT - Serverdə **yeni resurslar yaradır**. Bu metodun **nəzarətsiz icazəsi zərərli məlumatların yüklənməsinə** səbəb ola bilər.

DELETE - Serverdə mövcud olan bir **resursu silir**. Əgər düzgün qorunmayıbsa, bu metod **kritik məlumatların silinməsi** ilə nəticələnə bilər.

OPTİONS - Server haqqında məlumat qaytarır, məsələn, **hansı metodların qəbul edildiyini** göstərir.

PATCH - **Resursa qismən dəyişikliklər tətbiq edir** (tam yeniləmə deyil).

Bu siyahı yalnız ən çox istifadə olunan HTTP metodlarının bir neçəsini vurğulayır. Müəyyən bir metodun mövcudluğu həm **server konfiqurasiyasından**, həm də **tətbiqin loqikasından** asılıdır. HTTP metodlarının tam siyahısı və onların spesifikasiyaları üçün bu linkə baxa bilərsiniz:

https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Methods

Status Codes

HTTP status kodları müştəriyə sorğunun vəziyyətini bildirmək üçün istifadə olunur. HTTP serveri beş əsas status kod sinfi qaytara bilər:

Sinif | İzah
1xx – Məlumat verir və sorğunun emalına təsir etmir.
2xx – Sorğu uğurla yerinə yetirildikdə qaytarılır.
3xx – Server müştərini yönləndirdikdə qaytarılır.
4xx – Müştərinin düzgün olmayan sorğusunu göstərir. Məsələn, mövcud olmayan resursu tələb etmək və ya düzgün olmayan format göndərmək.
5xx – HTTP serverin özündə problem olduqda qaytarılır.

Aşağıda bu HTTP status kod siniflərindən tez-tez görülən nümunələr verilmişdir:

200 OK – Sorğu uğurla yerinə yetirildikdə qaytarılır və cavab bədənində adətən tələb olunan resurs olur.
302 Found – Müştərini başqa bir URL-ə yönləndirir. Məsələn, uğurlu girişdən sonra istifadəçini dashboard-a yönləndirmək.
400 Bad Request – Yanlış tərtib edilmiş sorğularda qaytarılır, məsələn, xətt sonlandırıcılarının çatışmazlığı.
403 Forbidden – Müştərinin resursa uyğun giriş icazəsinin olmadığını bildirir. Server zərərli giriş aşkar etdikdə də qaytarıla bilər.
404 Not Found – Müştəri serverdə mövcud olmayan bir resurs tələb etdikdə qaytarılır.
500 Internal Server Error – Server sorğunu emal edə bilmədikdə qaytarılır.

Tam standart HTTP status kodları siyahısını görmək üçün bu linkə baxa bilərsiniz. Standart kodlardan əlavə, Cloudflare və ya AWS kimi müxtəlif serverlər və provayderlər öz status kodlarını tətbiq edirlər.