

Web Programlama I

Ders 09 - Kimlik Doğrulama ve Yetkilendirme

07.12.2021

Erciyes Üniversitesi

Bilgisayar Mühendisliği Bölümü

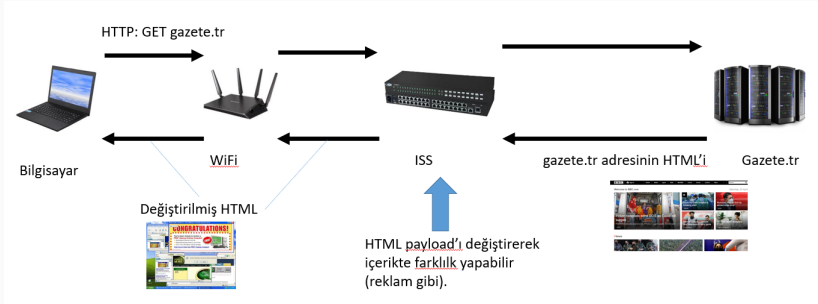
Eğitmen: Ömür ŞAHİN

- Cookie ile session tabanlı kimlik doğrulama
- NodeJS/React uygulamalarına kimlik doğrulama ekleme
 - HTML/CSS vs. JSON

1-HTTPS

- HTTP mesajları şifrelenmediği için güvenli değildir
- HTTPS, HTTP'nin bütün mesajlarının şifrelenmiş halidir
- Kritik öneme sahip olmayanlar da dahil olmak üzere bütün iletişim şifrelenir.
- Örnek: İnternet Servis Sağlayıcısı web sayfalarına reklam ekleyebilmektedir.

- Bir gazete web sayfasının şifrelenmesinin mantığı nedir?
- Eğer bu web sayfası şifrelenmezse istemci ile hedef sunucu arasında mesaj değiştirilebilir.



- HTTPS ile iletişim kurulduğunda ilk olarak dijital bir sertifika indirilir.
- Dijital Sertifika:
 - Domain adı
 - Güvenilir sertifika yetkilisi
 - Sunucuya ait public anahtar (RS algoritması gibi)
- Sertifikaların son kullanma tarihi bulunmaktadır.
- Bir sertifika taklit edilemez çünkü imzalamak için özel anahtara ihtiyaç vardır.

Sertifika

Genel Ayrıntılar Sertifika Yolu

 **Sertifika Bilgisi**

Bu sertifika, aşağıdaki amaçlarla verilmektedir:

- Uzak bir bilgisayarın kimliğini elde eder
- 2.23.140.1.2.2

Verilen: *.google.com

Veren: GTS CA 101

Geçerlilik: 23.03.2021 - 15.06.2021

Verenin Açıklaması

Tamam

Google


Google Search

I'm Feeling Lucky

Google offered in: Türkçe


Sertifika geçersiz ise?

Gizlilik hatası

← → ↻  Güvenli değil | expired.badssl.com

Sertifika

Genel Ayarlar Sertifika Yolu

 **Sertifika Bilgisi**

Bu sertifikanın süresi sona ermiş veya daha geçerli değil.


Verilen: *.badssl.com

Veren: COMODO RSA Domain Validation Secure Server CA

Geçerlilik: 9.04.2015 - 13.04.2015

Verenin Açıklaması


Tamam



Bağlantınız gizli değil

Saldırganlar **expired.badssl.com** üzerinden bilgilerinizi çalmaya çalışıyor olabilir (örneğin, şifreler, mesajlar veya kredi kartları). [Daha fazla bilgi](#)

NET::ERR_CERT_DATE_INVALID

 Chrome'un sağladığı en yüksek güvenlik düzeyinden faydalanmak için [gelişmiş korumayı açın](#)

Gelişmiş

Güvenliğe geri dön

Eğer sertifika geçersiz ise

- 2 temel sebebi bulunabilir.
- 1) Sertifikanın süresi geçmiş olabilir ve geliştirici henüz yenilememiştir.
- Man-in-the-middle saldırısı
 - Bağlanmaya çalıştığınız web uygulaması gibi davranarak giriş bilgilerinizi çalmaya çalışıyor olabilir (Örnek: orijinal web sayfası ile aynı ancak sahte bir login sayfasını göstermeye çalışıyor olabilir).
 - Bu türde bir saldırı kolay mıdır?
 - Bir router kullanarak gerçekleştirilebilir. Bir kafe yakınlarında "Free WiFi" isimli WiFi ağ açıp bu türde bir yönlendirmeyi yapabilirsiniz. Eğer kullanıcı güvenilmeyen sertifikaya devam et derse istediğiniz sayfaya yönlendirebilirsiniz.
 - Güvenmediğiniz ağlarda ASLA devam et butonuna tıklamayın.

- Bu ders kapsamında lokalde çalışırken HTTP kullanacağız.
 - HTTPS eklerken pek çok ayarlama yapmak gerekmektedir.
- "Cloud" server üzerine deploy ederken HTTPS kullanabiliriz.

2-Login

- **Kimliklendirme (Authentication)**

- X kullancısının kim olduğunu biliyor muyum?
- X ve Y kullanıcılarını birbirinden nasıl ayırabilirim.

- **Yetkilendirme (Authorization)**

- X kullancısının kim olduğunu bildikten sonra, hangi eylemlere yapmasına izin verilir?
 - Verileri silebilir mi?
 - Başkalarına ait verileri görebilir mi?
 - vs.
- Bu eylemler yalnızca şifreleme ile mantıklıdır. Böylelikle kimse mesajları çözemez ve değiştiremez.

- Kimliklendirme gerçekleşmez ise sunucu:
 - Login sayfasına 3xx durum kodu ile yönlendirme yapılabilir
 - 401 Unauthorized hatası ile Hata sayfası gösterilir
- Kimliklendirme yapılır ancak yetkilendirme sağlanmaz ise sunucu:
 - Örnek: X kullanıcısı Y kullanıcısına ait veriye erişmeye çalışırsa
 - 3xx yönlendirme
 - 403 Forbidden durum kodu

- Yetkilendirme sunucuda gerçekleştirilir ve dile bağımlıdır.
 - Spring, PHP, .Net, NodeJS vs.
 - Kullanıcı 3xx veya 403 kodlarından birini alabilir.
- Blacklist: Varsayılan olarak her şeye izin verilir. Kullanıcı/grup'un hangi eylemleri yapmasına izin verilmez ise belirtilir.
 - Genellikle çok iyi bir fikir değildir. Oldukça kritik eylemlerin blacklist'e eklenmeden unutulma ihtimali yüksektir.
- Whitelist: Varsayılan olarak hiçbir şeye izin verilmez. Neye izin verildiği belirtilir.
 - Bir şeyi yapmaya izin vermeyi unutmak (fonksiyonelliği azaltır) bir şeyi yasaklamayı unutmaktan (güvenlik problemi) daha iyidir.

- Sunucu kullanıcının kim olduğunu bilemez.
- Sunucu yalnızca gelen HTTP/s mesajlarını bilebilir.
 - Bir tarayıcıya ihtiyaç yoktur. TCP bağlantısı scriptler aracılığı ile de kurulabilir.
- HTTP/s durumsuzdur.
- HTTP/s çağrılarının aynı kullanıcıdan geldiğini belirtmek gerekir.
- Kullanıcı kim olduğuna ait bilgileri her bir çağrıda göndermelidir.
- Ancak kullanıcılar (örnek: hackerlar) yanlış bilgi gönderebilir.

- Kullanıcı tekil bir id ile kayıt olur.
- Ayrıca girişte kullanacağı bir şifre belirtmelidir.
- HTTP/S başka kullanıcıların hesaplarına giriş yapma girişimlerini engellemez.

- Her HTTP isteğinde kullanıcı ID'si ve şifre'nin gönderilmesi gerekir.
- HTTP başlığı içerisinde Authorization key'i ile gönderilebilir.
- ID/pwd'nin nasıl kodlandığını belirten farklı formatlar olabilir.
- Basic (RFC-7617): string "id:pwd" Base64 kodlama
- Örnek: id=test ve şifre=123456 olsun bu durumda
Authorization: Basic dGVzdDoxMjM0NTY=

- Base64 bir şifreleme değildir.
- id/pwd gönderilirken HTTPS kullanılması zorunludur.
 - Aksi takdirde ağdaki herhangi biri okuyabilir.

- id/pwd ile giriş işlemi yalnızca bir sefer yapılır.
- Sunucu user id ile ilişkili bir token döner.
- Bundan sonraki bütün isteklerde id/pwd göndermek yerine bu token gönderilir.
- Bu token belli bir zaman boyunca geçerlidir. Eğer süre dolarsa yeni bir token almak gerekir.
- Peki faydaları nedir?

- Eğer bir token çalınırsa, hacker süresi dolana kadar kısıtlı bir süre boyunca erişim sağlar.
- Kullanıcı çıkış yaparsa token geçersiz hale gelir ve sunucu bu token ile gelen istekleri reddeder.
 - Hacker token'ı çalsa bile artık kullanılamaz olacaktır.
- Şifre değiştirme veya para transferi gibi kritik işlemlerde yeni id/pwd kullanılarak giriş yapma gerekir.
 - Böylelikle hacker çalmış olduğu token'ı kullanamaz.

- Sunucuya "Authorization: Basic ..." gibi bir istek geldiğinde token üretmesi üzerine tasarlanabilir.
- Herhangi yeni bir endpointte gerçekleştirilir.

id/pwd token oluşturmak için nasıl gönderilir?

- Güvenlikten ve bunun sunucuya nasıl uygulanacağından söz ederken tarayıcıdan gelmesi gerekmeyen istekler ele alınmalıdır.
- id/pwd verilerek token alınan bir endpoint olabilir.
 - Böyle bir endpoint aşağıdaki parametreleri alabilir.
- GET /login?userId=x&passowrd=y
 - userId/password URL parametresi olarak /login endpointine gönderilebilir.
 - HTTP cevabında body içerisinde token z gelebilir.
- GET /yapilmakIstenenSey?token=z
 - "token=z" parametresi bütün HTTP/S isteklerinde gönderilebilir.

Ancak bu oldukça KÖTÜ bir çözümdür

- Bu çözüm işe yarar ancak,
- `"/login?userId=x&password=y"` tarayıcı geçmişinde kaydedilebilir.
- Bütün a taglarına `"?token=z"` eklemek oldukça zahmetlidir.
- Tarayıcı kısayollarına eklemek oldukça problemlidir.

- Kullanıcı id ve şifreleri asla GET ile gönderilmemelidir.
 - GET özellikleri istek içerisinde Body göndermeye izin vermez.
- POST metodu ile HTTP body içerisinde gönderilmelidir.
- POST /login "userId":id, "password":pwd

- Tarayıcılar authentication token'ı bir yerde saklamaya ihtiyaç duyarlar.
- Token'lar her bir HTTP isteğine eklenmelidir.
- Bu tokenları saklamanın en iyi yolu HttpOnly olarak işaretlenmiş HTTP Cookie'lerdir.
 - HTTP requestlerin içine otomatik olarak eklenirler.
 - JavaScript tarafından okunamazlar.
- HttpOnly cookie olarak saklamazsanız XSS ataklarına daha fazla açık hale gelirsiniz.
 - Bu atak türünü sonraki ders göreceğiz.

- Kimliklendirme token'ları URL içerisinde değil HTTP header içerisinde gönderilmelidir.
- **Cookie**: Kullanıcı bilgilerini tanımlayan özel HTTP header'larıdır.
- Kullanıcı cookie bilgisini seçemez. Bu bilgi server tarafından gelir.
- Kullanıcı kendi HTTP mesajını oluşturabilir bu yüzden sunucunun cookie'nin geçerli olup olmadığını kontrol etmesi gerekir.

Cookie ile giriş

- Tarayıcı: POST /login
 - HTTP body içerisinde, kullanıcı adı X ve şifre
- Sunucu: Eğer giriş başarılı ise POST'a "Set-Cookie" başlığı ile çeşitli cookie'ler barındıran Y cookie değerini yanıt olarak döndürülür.
 - Sunucunun Y cookie'sinin X kullanıcısına ait olduğunu hatırlaması gerekir.
 - Set-Cookie: <cookie-adı>=<cookie-değeri>
- Tarayıcı: Bundan sonra her bir HTTP isteğinde "Cookie:Y" değeri header içerisinde gönderilir.
- Çıkış yap: Cookie Y ile kullanıcı X arasındaki ilişki sunucu tarafında kaldırılır.
- Sunucu: HTTP request içerisinde ya cookie bulunmaz ya da 401 hatası veren geçersiz/tarihi geçmiş bir cookie bulunur.



HTML sayfası
isteği

GET /index.html

AJAX ile
bilgilerin
gönderilmesi

POST /api/login
userId=foo&password=bar

Sonraki HTTP
requestlere cookie
başlığının
eklenmesi

HTTP/1.1 204
Set-cookie: 123456

GET /api/someData
Cookie: 123456



Log in

Don't have an account? [Create one.](#)

Username:

Password:

☐ Remember me (up to 30 days)

Bilgilerin
onaylanması. Eğer
doğru ise bir cookie
id tarafından
tanımlanan
session'in
oluşturulması.

- Sunucular genellikle login işleminden bağımsız olarak "Set-Cookie" headerını gönderirler.
 - Kimliklendirmeden bağımsız olarak isteklerin aynı kullanıcıdan mı geldiği bilinmek istenebilir.
 - Cookie'ler session tanımlamak için kullanılabilir.
- Giriş yaptıktan sonra session oluşturulabileceği gibi (eski cookie geçersiz kılınır ve yenisi oluşturulur) var olan session cookie'si (örnek: ilk GET ile çekilen login sayfasıyla birlikte gelen cookie bilgisi) de kullanılabilir.
- Session cookie bilgisi tekrar kullanılıyor ise bütün sayfaların HTTPS olarak kullanılması gerekir.
 - Login sayfası da dahil olmak üzere hepsinde HTTPS kullanın.
 - Önce HTTP ile giriş yapıp daha sonra HTTPS ile değiştirmeyin.

- Tarayıcılar yerel olarak cookie'leri saklayabilir.
- Her bir HTTP/S isteğinde bu cookie HTTP header içerisinde gönderilir.
- Cookie'ler yalnızca onları ayarlamasını isteyen sunucuya gönderilir.
 - Örnek: foo.com tarafından ayarlanmış bir cookie bar.org adresine gönderilmez.
- Eğer HttpOnly olarak işaretlenmez ise JavaScript cookie değerlerini okuyabilir.
- Peki neden problem yaratır?
 - JS ile bütün cookie'leri okuyan bir web sayfası oluşturabilir ve kullanıcının Google/Facebook/Banka hesabı gibi bilgilere erişim sağlayabilirsiniz.

- Set-Cookie: <name>=<value>;
Expires=<date>;Secure;HttpOnly
- Expires: Cookie ne kadar süreliğine saklanacak.
- Secure: Cookie yalnızca HTTPS üzerinden gönderilir.
 - Bazı saldırı türlerinde aynı sunucuya HTTPS yerine HTTP istek yapılması vardır. Böylelikle cookie bilgileri plain text olarak okunabilir.
- HttpOnly: JS'in cookie değerlerini okumasına izin vermez.
 - XSS ataklarından kaçınmak için oldukça önemlidir.

- Sunucu tarafından session/login cookieleri hariç başka cookieler de ayarlanabilir.
- Çerez kullanımı ile ilgili çeşitli yasalar bulunmaktadır.
- Peki neden? Çünkü takip edilme ve gizlilik endişeleri sebebiyle.



@ICSandwichGuy

icecreamsandwichcomics.com

Tracking (Takip)

- Pek çok site diğer sitelerdeki kaynaklara ihtiyaç duymaktadır.
 - Resim, JavaScript, CSS dosyaları, vs.
 - Facebook'taki Like butonu.
- Y adresindeki kaynakları kullanan X adresindeki HTML sayfası indirildiğinde HTTP Y adresine, Y'ye ait önceki cookieleri de içeren de GET isteği yapar.
- Facebook'tan çıkış yapsan bile hangi sayfaları gezdiğini bilebilir.
- Daha da kötüsü hiç FB kullanmasanız bile FB tarayıcınızı takip edebilir.
- Bu yalnızca X sayfasını açmanızla birlikte gerçekleşecektir. Herhangi bir şeye tıklamanıza gerek yoktur!!!
- referrer HTTP header: Y'den gelmeyen ancak Y'ye yapılan isteklerin alan adı orijini
 - Örnek: Referer:X X'den Y'ye istek atıldığında eklenir.

Name	X	Headers	Preview	Response	Initiator	Timing	Cookies
track		General					
100?userGroups=7090-7467-7510-6020-f		Request URL: https://www.facebook.com/tr/?id=[REDACTED]µdata&d1=https%3A%2F%2Fwww.hepsiburada.com%2F&r1=&if=false&t=[REDACTED]&[DataLayer]=%5B%5D&c[Meta]=%7B%22title%22%3A%22%27%3B%Crkiye%27nin%20En%20B%C3%BByC%3BCk%20Online%20Al%4B%1C%59fveri%5C%59f%20Sitesi%20Hepsiburada.com%22%7D&c[OpenGraph]=%7B%7D&c[Schema.org]=%5B%5D&c[JSON-LD]=%5B%5D&sw=1920&sh=1080&v=2.9.39&r=stable&ec=1&o=30&fbp=fb.1.1619296653045,144861492&it=1619296652670&coo=false&es=automatic&tm=3&rqm=GET					
100?userGroups=7090-7467-7510-6020-f		Request Method: GET					
integrator.js?domain=www.hepsiburada.c		Status Code: 200					
?random=1619296652813&cv=9&fst=16'		Remote Address: 185.60.218.35:443					
container.html		Referrer Policy: no-referrer-when-downgrade					
?random=1619296652813&cv=9&fst=16'		Response Headers (12)					
client-standart.js		Request Headers					
AddToCart-standart.js		:authority: www.facebook.com					
format:webp		:method: GET					
format:webp		:path: /tr/?id=[REDACTED]µdata&d1=https%3A%2F%2Fwww.hepsiburada.com%2F&r1=&if=false&t=[REDACTED]&[DataLayer]=%5B%5D&c[Meta]=%7B%22title%22%3A%22%27%3B%Crkiye%27nin%20En%20B%C3%BByC%3BCk%20Online%20Al%4B%1C%59fveri%5C%59f%20Sitesi%20Hepsiburada.com%22%7D&c[OpenGraph]=%7B%7D&c[Schema.org]=%5B%5D&c[JSON-LD]=%5B%5D&sw=1920&sh=1080&v=2.9.39&r=stable&ec=1&o=30&fbp=fb.1.1619296653045,144861492&it=1619296652670&coo=false&es=automatic&tm=3&rqm=GET					
osd.js?cb=%2Fr20100101		:scheme: https					
format:webp		accept: image/avif,image/webp,image/apng,image/svg+xml,image/*,*/*;q=0.8					
format:webp		accept-encoding: gzip, deflate, br					
8390192428486553754		accept-language: tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.7					
ext.js		cache-control: no-cache					
track		cookie: datr=tyRHXXRP_Fs4FEoGxpBhIL90; sb=Pw2BX5_0_k79XV8h-0fhZzSY; fr=0d1l8sfPUcd5o561..Bc3qo0.UA.Gbp.0.0.Bggt3Q.					
format:webp		pragma: no-cache					
r_cldjar.js?cache=r20110914		referrer: https://www.hepsiburada.com/					
Tid=B91502007900202&ev=Microdata&d		sec-ch-ua: " Not A;Brand";v="99", "Chromium";v="98", "Google Chrome";v="98"					
view?xai=AKAOjsvwVRhmxdGWooAUkr		sec-ch-ua-mobile: ?0					
container.html		sec-fetch-dest: image					
view?xai=AKAOjsvhUriVHE_FpVMctGedm		sec-fetch-mode: no-cors					
data:image/png;base...		sec-fetch-site: cross-site					
webtrekl.js							
sodar?sv=200&tid=gpt&tv=20210420016							
pixel?id=KAE							
track							
appboy.min.js							
sodar2.js							
7f85a56ba4.css							
data/							
sync							
runner.html							
aframe							
font-awesome-css.min.css							

Kendini nasıl koruyabilirsin?

- uBlock origin gibi tarayıcı eklentileri ile.
 - Bilinen harici tracker'ları engeller.
- Firefox gibi bazı tarayıcılar varsayılan olarak başlatmaktadır.
 - Chrome bir reklamcılık firması tarafından geliştirildiği için bunu uygulanma ihtimali oldukça düşüktür. :)



Name	Status	Type	Initiator	Size	Time	Waterfall
jquery.min.js	200	script	adRecover.js:1	33.5 kB	196...	
?random=1619296104285...	200	gif	?random=1...	65 B	121...	
?random=1619296104285...	200	gif	?random=1...	64 B	131...	
analytics.js	200	script	VM401:1	19.6 kB	184...	
xgemius.js	200	script	VM403:2	10.6 kB	158...	
beacon.js	200	script	VM404:1	1.8 kB	232...	
adpushup.js	200	script	VM406:1	94.5 kB	57 ...	
lsget.html	200	docu...		2.8 kB	88 ...	
info?referrer=https%3A//e...	200	gif	VM405:1	603 B	295...	
collect?t=dc&aip=1&r=3&...	200	xhr	analytics.js:39	29 B	142...	
block.jpg?t=1619296108267	200	jpeg	adRecover.js:1	904 B	28 ...	
sync	200	ping	adpushup.js:1	178 B	218...	
collect?v=1&v=j90&a=279...	200	gif	analytics.js:26	55 B	65 ...	
gpt.js	200	script	adpushup.js:1	21.1 kB	123...	
feedback?data=eyJcmVhd...	200	png	adpushup.js:1	271 B	92 ...	
rexdot.js?l=100&id=bJub8...	200	script	xgemius.js:2...	501 B	204...	
ga-audiences?t=sr&aip=1&...	200	gif	analytics.js:26	64 B	124...	
ga-audiences?t=sr&aip=1&...	200	gif	analytics.js:26	63 B	122...	
b?c1=2&c2=19592957&uns...	204	text/p...	beacon.js:1	333 B	86 ...	
container.html	200	text/h...	pubads imp...	3.5 kB	398...	
container.html	200	text/h...	pubads imp...	25 B	67 ...	
sodar?sv=200&tid=gpt&tv...	200	xhr	pubads imp...	7.0 kB	205...	
manifest.json	200	manif...	Other	566 B	3.6...	
favicon-32x32.png	200	png	Other	1.1 kB	86 ...	
sodar2.js	200	script	pubads imp...	6.5 kB	127...	
runner.html	200	docu...	sodar2.js:26	5.0 kB	67 ...	
aTo5JV-AjziuZNTa8Sp43jLV...	200	script	tpc.google...	5.7 kB	67 ...	
gen_204?id=sodar2&v=222...	204	gif	sodar2.js:31	20 B	123...	
activeview?xai=AKAOjvBU...	200	gif	amp-analyti...	64 B	124...	
110 requests 1.2 MB transferred 3.0 MB resources						

SİNEMA

9,1b

Kimilerince 1998'in En İyi Filmi Olan Dark City Hakkında Az Bilinir...

SAĞLIK

14,8b

Vücutlarındaki Bazı Uzunları Kesip Atmak İsteyenlerin Muzdarip Olduğu...

UZAY

3,6b

Astronotların Uzayda Kullandığı, Su Altında Bile Yazabilen Uzay Kalem

sorunsallar

hiç soru sorulmamış...

soru sor

başlığın kanalları

#programlama #teknoloji

kanal öner...

Name	Status	Type	Initiator	Size	Time	Waterfall
adRecover.js	(block...	script	ublock-origi...	0 B	1.8...	
ns.html?id=GTM-WXV2Z47	(block...	docu...	ublock-origi...	0 B	7 ms	
6xK3dSBYKcSV-LCoeQqfX1...	200	font	css?family=...	16.1 kB	74 ...	
6xKydsBYKcSV-LCoeQqfX1R...	200	font	css?family=...	15.8 kB	69 ...	
4755744?style=gray	200	docu...	jquery-com...	1.7 kB	30 ...	
topicmattersummary?slug=...	200	xhr	jquery-com...	564 B	6.0...	
6xKydsBYKcSV-LCoeQqfX1R...	200	font	css?family=...	16.0 kB	69 ...	
data:image/png;base...	200	png	chrome-erro...	(mem...	0 ms	
data:image/png;base...	200	png	chrome-erro...	(mem...	0 ms	
embedable.min.css?5782b0...	200	styles...	4755744?sty...	3.7 kB	194...	
iylojCaE40Ta7FnP-63754608...	200	jpeg	4755744?sty...	14.5 kB	90 ...	
eksi.woff194997930	200	font	ust469736-1...	6.4 kB	6.1...	
8iOtxY8zykpEmmAT-63754...	200	jpeg	4755744?sty...	9.1 kB	88 ...	
S8IBQ4GhYR12UVOa-63754...	200	jpeg	4755744?sty...	13.6 kB	89 ...	
css?family=Martel:300,400...	200	styles...	embedable...	500 B	101...	
css?family=Open+Sans:400...	200	styles...	embedable...	878 B	105...	
eksico.svg	200	svg+x...	4755744?sty...	3.4 kB	185...	
6xK3dSBYKcSV-LCoeQqfX1...	200	font	css?family=...	15.0 kB	69 ...	
6xKydsBYKcSV-LCoeQqfX1R...	200	font	css?family=...	15.0 kB	68 ...	
eksisozluk_logo.svg	200	svg+x...	ust469736-1...	1.9 kB	6.0...	
eksiseyler-text-logo@2x.png	200	png	ust469736-1...	3.8 kB	6.4...	
pena-text-logo@2x.png	200	png	ust469736-1...	2.4 kB	6.4...	
mem8YaGs126MiZpBA-UFV...	200	font	css?family=...	14.4 kB	114...	
mem5YaGs126MiZpBA-UN...	200	font	css?family=...	15.1 kB	94 ...	
mem8YaGs126MiZpBA-UF...	200	font	css?family=...	11.3 kB	72 ...	
mem5YaGs126MiZpBA-UN...	200	font	css?family=...	11.7 kB	128...	
googletagmanager_gtm.js?...	200	script	gtm.js	1.5 kB	4 ms	
manifest.json	(pend...	manif...	Other	0 B	Pen...	
favicon-32x32.png	200	png	Other	1.1 kB	85 ...	
46 requests	382 kB transferred	837 kB resources	Finish: 48.12 s	DOMContentLoaded: 39.92 s		

- Daha az HTTP çağrısı, daha iyi kullanıcı deneyimi
 - Daha az bandwidth tüketimi
 - Sayfalar daha hızlı render edilir
 - Mobil aboneliklerde veri transfer sınırı
- Takip için kullanılan JS scriptleri CPU'nuzu kullanır

3-Şifreler

- Kullanıcının doğrulanmasına ihtiyaç vardır.
- Çok kısa ve basit olmamalıdır. Aksi takdirde brute-force ile rahatlıkla kırılabilir.
- Güvenlik vs Kullanılabilirlik: İyi bir dengenin yakalanması zordur.
 - Örnek: İdealde her web sayfası için farklı bir şifre olmalıdır ve bunlar da sıklıkla değiştirilmelidir. Ancak bunu kim, neden yapar ki? :)



- Kullanıcı oluşturulduğunda şifreyi saklamak için bir alana ihtiyaç bulunmaktadır (genellikle veritabanı).
- ASLA ŞİFRELERİ PLAIN TEXT OLARAK SAKLAMAYIN.
- Şifreler hashlenmelidir.
- Hacker veritabanına tam erişimde bulunsa bile şifreyi elde edemeyecektir.
 - SQL Injection ataklarındaki tipik örnektir.
 - Ancak pek çok durumda karşımıza çıkabilir örneğin mutsuz bir çalışan veya çöpe atılmış bir hard disk :)
- Bir hacker aynı şifreyi farklı sitelere girişte de kullanmaya çalışabilir.
- **UYARI:** Bu derste şifreleri hashleme ile veya doğru şekilde saklama ile uğraşmayacağız.

4-React ve NodeJS ile Güvenlik

- Express/NodeJS'te kimliklendirme işlemleri için **Passport** isimli kütüphaneyi kullanacağız.
- Cookie'ler ile session-based kimliklendirme kullanacağız.
- Üye ol, giriş ve çıkış işlemlerine ait REST API oluşturacağız.
- Eğer geçerli bir session cookie bilgisi gelirse Passport otomatik olarak giriş yapan kullanıcıya ait user objesini oluşturacaktır.

- React'ın kimliklendirme için bir yapısı bulunmamaktadır.
- Componentlerin giriş yapıp yapılmamasına göre render edilmesi gerekmektedir.
- Oturum açıp açmama durumu geçerli bir session cookie'sine sahip olup olmamakla belirlenmektedir.
- Ancak React HttpOnly cookie'lere erişim sağlayamamaktadır.
 - Erişim sağlase bile geçerli bir cookie olup olmadığını React bilememektedir.

- Kullanıcının giriş yapması durumunda bir değişken ile tutulabilir.
 - Componentler render edilirken bu değişken kullanılır.
- Yetkilendirme gerektiren her HTTP çağrısında, gerekirse bu değişken güncellenir.
 - Eğer çıkış işlemi gerçekleşirse 401 hatası alınacaktır.
- React, mevcut oturum açma/oturum kapatma durumunu gösteremez, bunun yerine son HTTP etkileşimindeki durumu gösterir.
 - Doğrudan çıkış yapılmasa bile cookie'nin süresi dolabilir.

Frontend Güvenliği

- Frontend tarafında oturum açma durumunun güvenlik üzerine hiçbir etkisi yoktur. Yalnızca kullanılabilirliği etkileyen hangi componentlerin görüntüleneceğini belirler.
- Güvenlik kısmı backend tarafında ele alınmak zorundadır. Bir arayüz olmadan da TCP bağlantısı kurulup istek yapılabilir.
- Frontend'den bağımsız olarak her bir REST endpoint'i kimliklendirmeye göre korunmalıdır.

Giriş Yapmadıysa,
Kaynak üretecek
isteğin yapılacağı
AJAX butonu render
edilmez.



Giriş yaptıysa
AJAX istek yapacak
buton üretilir.



onClick = POST /api/protected



Sunucu



Sunucu