ağ güvenliği- web security 2

Bir siteyi frame/iframe olarak yüklemediysek direkt tek bir yapı olarak yüklendiyse tüm içeriğin domain aynı olmalı.

attacker.com banka.com'a link verebilir, img kendi sitesinde src="bank.com/img" gibi dahil edebilir. Ama dosyanın içini görüp düzenleyemez.

SOP- Origins

JS, XMLHttpRequst veya JQuery ile bir şeylerin yüklenmesi için istek yollayabilir.

Malicious XMLHttpRequests

```
// running on attacker.com
$.ajax({url: "https://bank.com/account",
  success: function(result){
      $("#div1").html(result);
  }
});

// Will this request run?
// Should attacker.com be able to see Bank Balance?
```

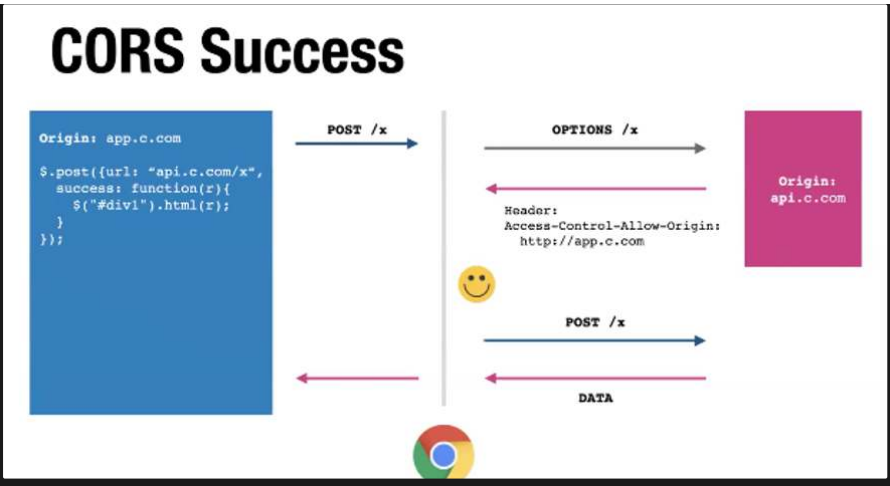
You can only read data from **GET** responses if they're from the same origin (or you're given permission by the destination origin to read their data)

You cannot make **POST/PUT** requests to a different origin… unless you are granted permission by the destination origin (*usually*, caveats to come later)

XMLHttpRequests requests (both sending and receiving side) are policed by **Cross-Origin Resource Sharing (CORS)**

ACAO(Access-Control-Allow-Origin)

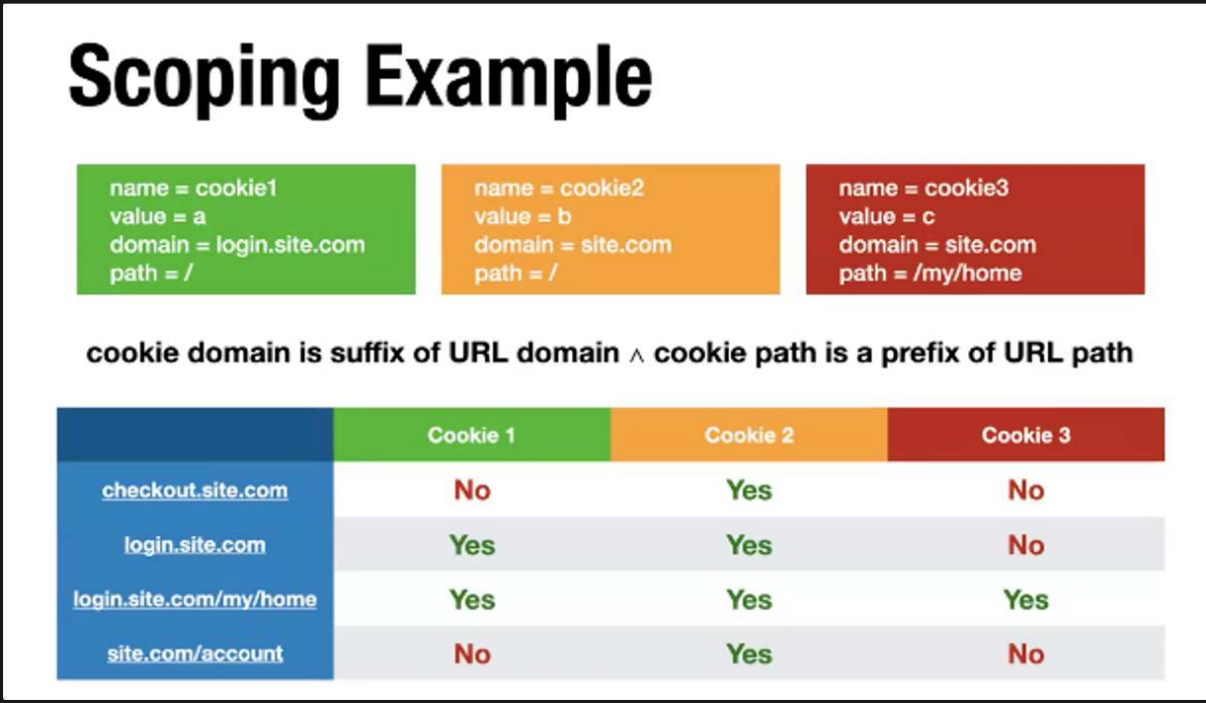
Farklı originden gelen JS isteği için(GET) okuma izni



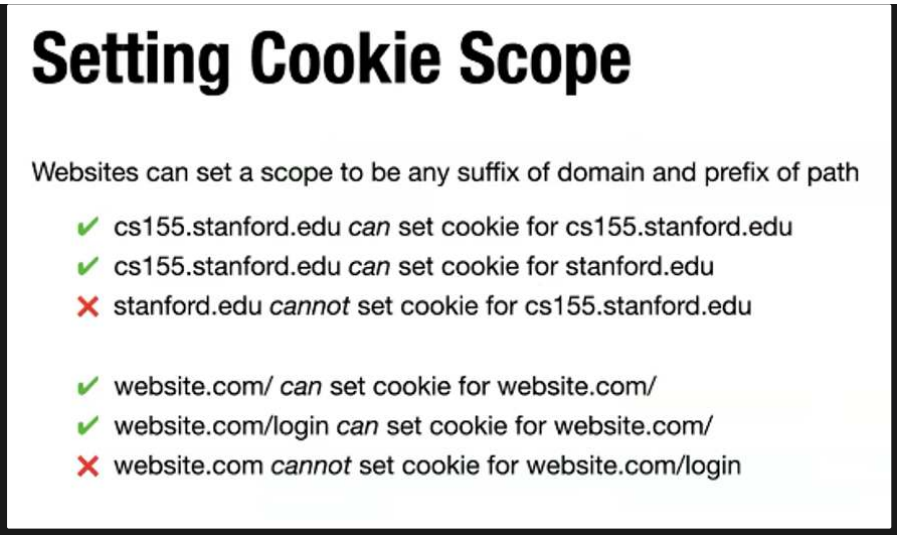
Wildcard Origins: Her GET isteğine açık. Public bir şey yapıyorsan okey.

SOP - Cookies

domain, path kontrolü yapılır.

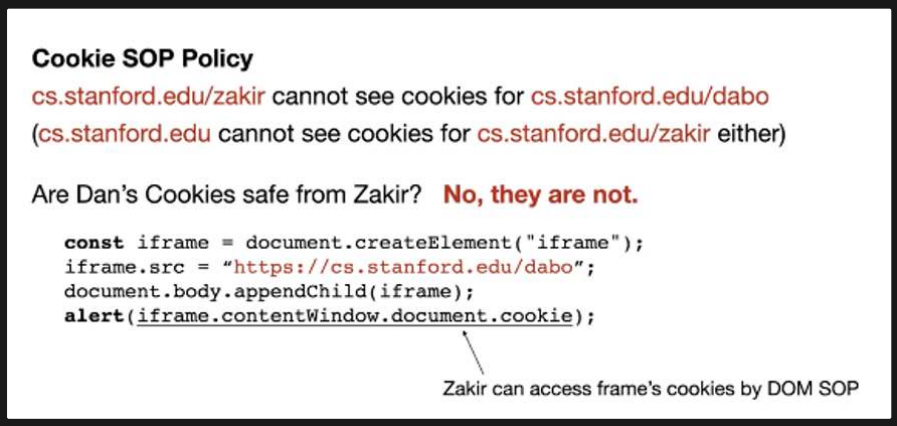


	Cookie 1	Cookie 2	Cookie 3
checkout.site.com	No	Yes	No
login.site.com	Yes	Yes	No
login.site.com/my/home	Yes	Yes	Yes
site.com/account	No	Yes	No



No Domain Cookies

Domain set edilmez. Cookie POST'u yollayan hostname haricinde kimseye gitmez.



Çakışma oldu.

HttpOnly Cookies(Secure Cookies)

Bu çakışmayı ortadan kaldırmak için HttpOnly Cookies kullanırız. DOM üzerinden erişim kapatılıyor. Buradaki problem de network attackerlar. Bunu çözmek için de sadece ve sadece HTTPS kullanıldığında HttpOnly Cookies yolluyoruz.

Cookies & Web Attacks

Cross-Site Request Forgery(CSRF)

CSRF bir saldırı yöntemi. Farklı sayfaya gerçekleştirmek istemeyeceği isteklerde bulunuluyor.

Cookie based authenticationlar değişikliğe sebep olan istekleri(POST vs.) önlemede yeterli olmuyor. Yaygınca kullanılan dört teknik var.

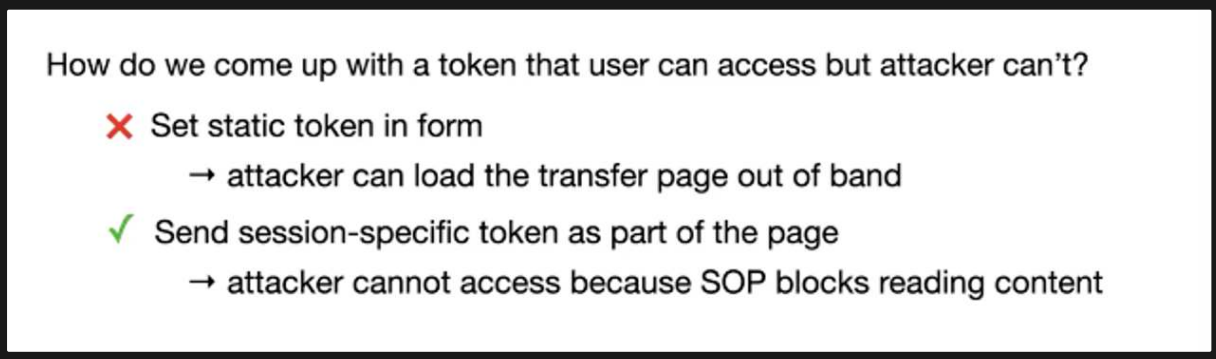
Referer Validation

Adrese göderen bir önceki sayfanın kaydı tutuluyor.

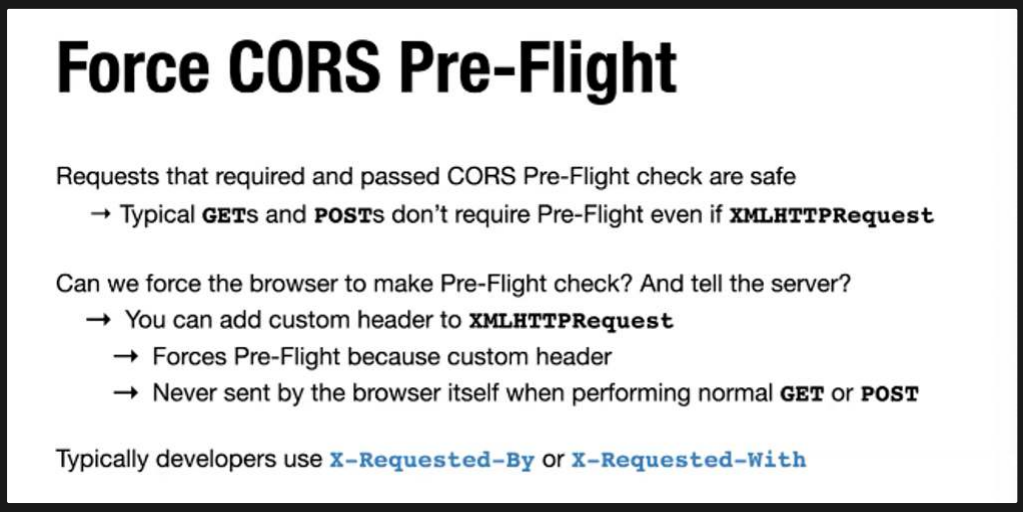
Geldiği alan boşsa ne yapacağız? Bunu için secret validation token var.

Secret Validation Token

Formdaki gizli alanlarda bir değer gönderiliyor bu değer kontrol edilerek request gerçekleştiriliyor. Bu değerin statik olmaması lazım yoksa saldırgan kopyalar.



Custom HTTP Header



sameSite Cookies

Browser cookie'yi scope içerisinde mutlaka gönderiyor. Bir req direct site değil cross site req ise gönderilmemesini sağlayacak bir option.

a-) Strict Mode

Never send, bağlantılı link varsa bile.

b-)Lax Mode

Bağlantılı link varsa GET'e izin veriyor ama POST veya PUT'a izin vermiyor.

CSRF ile authentication dışında internet trafiğine saldırılar da olabilir. Girdiğimiz bir link DNS ayarlarımızı değiştirerek trafiğimize hijacking yapabilir. Bir adrese gittiğimizi sanarken aslında saldırganın gitmemizi istediği adrese gidiyoruz.

SQL Injection

En kritik saldırı.

Bir input ile bir komutun çalıştırılması hedefleniyor. SQL sorgularında da bunu yapabiliyoruz.

```
$u = $_POST['login']; // zakir'--
$pp = $_POST['password']; // 123

$sql = "SELECT id FROM users WHERE uid = '$u' AND pwd = '$p'";
//      "SELECT id FROM users WHERE uid = 'zakir'-- AND pwd…"
$rs = $db->executeQuery($sql);
//    (No Error)
if $rs.count > 0 {
   // Success!
}
```

'- - ile kalan SQL kodu comment satırı oldu böylece şifre ne olursa olsun kullanıcı giriş yapabildi

Tablo silindi

Escaping Database Server

```
$u = $_POST['login']; // '; exec xp_cmdshell 'net user add usr pwd'--
$pp = $_POST['password']; // 123

$sql = "SELECT id FROM users WHERE uid = '$u' AND pwd = '$p'";
//      "SELECT id FROM users WHERE uid = '';
         exec xp_cmdshell 'net user add usr pwd123'-- "

$rs = $db->executeQuery($sql);
// No Error…(and with a resulting local system account)
```

Sistem kullanıcısı tanımlandı

> Never trust user input

Parametre işlenmeden önce komut vb. varsa diye temizlenmesi gerekiyor.

Cross Site Scripting(XSS)

Uygulama güvenilmeyen veriyi alıp browser'a yolladığında gerçekleşir

Embedded Script

https://google.com/search?q=<script>alert("hello")</script>

```
<html>
  <title>Search Results</title>
  <body>
    <h1>Results for <?php echo $_GET["q"] ?></h1>
  </body>
</html>
```

Sent to Browser

```
<html>
  <title>Search Results</title>
  <body>
    <h1>Results for <script>alert("hello")</script></h1>
  </body>
</html>
```

Input script olabilir.

Embedded Script ile cookie çalınabilir. Sizin sunucunuz cookie gönderme isteğinde bulunduğu için güvenlik önlemlerini aşarak saldırgan siteye cookie yollanabilir.

XSS'i engellemek için yapılacak yegane şey:

Content Security Policy