Web Security 101

**Öncelikle** modern web nasıl çalışır **onu anlamamız gerekiyor**

1. **Bölüm ARP nedir nasıl çalışır**

****

**Burası bir yerel ağ.**

**Routerın varsayılan ağ geçidi (gateway) 10.0.0.1’dir.**

* **PC1’in IP’si: 10.0.0.5**
* **PC2’nin IP’si: 10.0.0.17**

**PC1 ve PC2 ilk kez haberleştiğinde, PC1 PC2’nin IP adresini öğrenir ve ona bir ARP sorgusu göndererek MAC adresini alır zaten bi kere tanıştılar şimdi ve bidaha tanışmasına gerek kalmaz . Aynı şekilde PC2 de PC1’in MAC adresini öğrenir. Bu bilgiler, her iki cihazın ARP tablolarına kaydedilir iki insanın tanışınca ismini hafızada tutmak gibi. Böylece tekrar tekrar sorgu yapmalarına gerek kalmaz ve doğrudan haberleşebilirler.**

**Router ise bu iletişime karışmaz, çünkü aynı ağda olan cihazlar birbirleriyle doğrudan iletişim kurar. Ancak, PC1 veya PC2 internete çıkmak ya da farklı bir ağa erişmek isterse, router devreye girer ve bu cihazların MAC adreslerini kendi ARP tablosuna ekler.**

**Bilgisayarlar router ile iletişime geçtiğinde, router’ın MAC adresi de cihazların ARP tablolarında görünür.**

**İss in admin paneline girdiğimizde bu mac adresinde bu ip adresi var diyip yanına apple , linux , Android logosunu gösterdiği tablo işte bu arp tablosu. Hangi cihaz olduğunu da mac adresi başlangıcından üreticiyi tanıyarak gösteriyor. Yerel ağda işeyiş bu şekilde**

**ARP POİSONİNG :**

**Ben ayağa kalktım, "İrem, gel, seninle konuşmak istiyorum" dedim. İrem geldi kapıdan. Evde İrem değil de, 200 kişiden başka biri gelse, "Evet geldim, ben İrem’im" dediğinde, ben onun İrem dışında başka biri olup olmadığını öğrenemiyorum. ARP Poisoning dediğimiz saldırı burada başlıyor.**

**PC2 derse ki, "10.0.0.5’in (PC1’in) MAC’i X değil, artık Y" derse, modemin 10.0.0.17 ile konuşmak istediği trafik iptal olur ve bunların hepsi bu adama gider. Böylece 1 MAC ve 2 IP olmuş olur; ortadaki adam olmuş olunur. (Man in the Middle)**

**HTTP ile x.com’a gitmek istersen, saldırgan bu trafiğin tamamını görebilir.**

**İnternete çıkarken x.com'un bir IP’sini bulacaksın ve sunucuya bağlanmak için trafikten çıkacağız. Bu trafiğe çıkmadan önce bazı eksik bilgiler var. Bu adam hangi DNS ile konuşacak?**

**Bilgisayarı açtığında DNS’i otomatik girmiyoruz, biri otomatik olarak söylüyor. İşte onu söyleyen de DHCP’dir. Ondan dolayı 8.8.8.8 bilgisi ve Gateway yani 10.0.0.1 vardır.**

**Yerel ağ dışında biriyle konuşmak istediğimizde geçeceğimiz bir geçiş kapısı var. Abdullah ile konuşmak için Abdullah diye bağırdık, Abdullah sınıfta yoksa sınıfın kapısından çıkmamız gerekiyor. O kapı da gateway anlamına geliyor.**

**Konuşmak istediğimiz adamın bizimle aynı ağda olup olmadığını anlamamız için de ihtiyacımız olan bilgi Subnet 255.255.255.0’dır.**

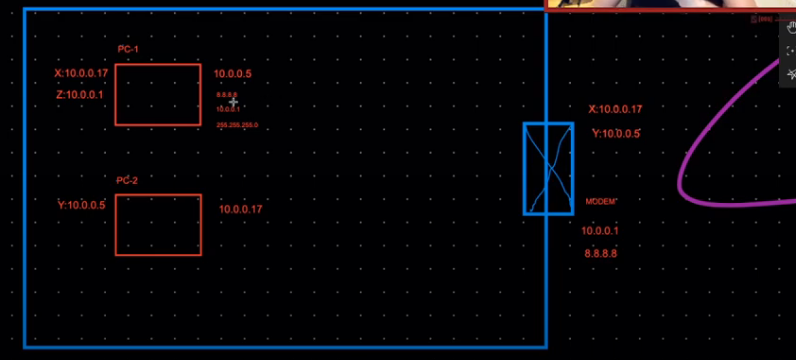
**Günün sonunda PC1'in IP’si 1234 olan ve internetteki bir makineyle aynı ağın içinde olup olmadığını Subnet Mask ile anlıyor.**

**Subnet Mask'ın işi, "Konuşmak istediğim adamla aynı ağda mıyım, değil miyim?" sorusuna cevap vermek.**

**Subnet Mask (Alt Ağ Maskesi):  
Subnet maskesi, bir cihazın hangi IP adreslerinin aynı ağda olduğunu anlamasına yardımcı olur. Örneğin, 255.255.255.0 gibi bir subnet maskesi, cihazın 10.0.0.1 ile 10.0.0.255 arasındaki IP adresleriyle aynı ağda olduğunu gösterir. Bu sayede cihazlar, aynı ağda olup olmadıklarını kontrol edebilir ve yerel ağda iletişim kurabilir.**

**Gateway (Ağ Geçidi):  
Gateway, cihazların yerel ağ dışındaki ağlarla iletişim kurmasını sağlayan bir "geçiş noktası"dır. Eğer bir cihaz başka bir ağdaki bir cihaza ulaşmak istiyorsa, bu trafiği yönlendirecek bir Gateway’e ihtiyacı vardır. Genellikle router, Gateway olarak kullanılır ve ağ dışına çıkılacaksa, tüm trafik önce bu Gateway'e yönlendirilir.**

**DHCP (Dynamic Host Configuration Protocol):  
DHCP, ağda bulunan cihazlara otomatik olarak IP adresi, Subnet Mask, Gateway ve DNS gibi ağ bilgilerini atayan bir protokoldür. Bu, cihazların elle IP ayarları yapmasına gerek kalmadan ağ bağlantısını kolaylaştırır. Cihazlar, ağda bağlandığında DHCP sunucusundan bu bilgileri alır ve internet erişimi sağlar.**

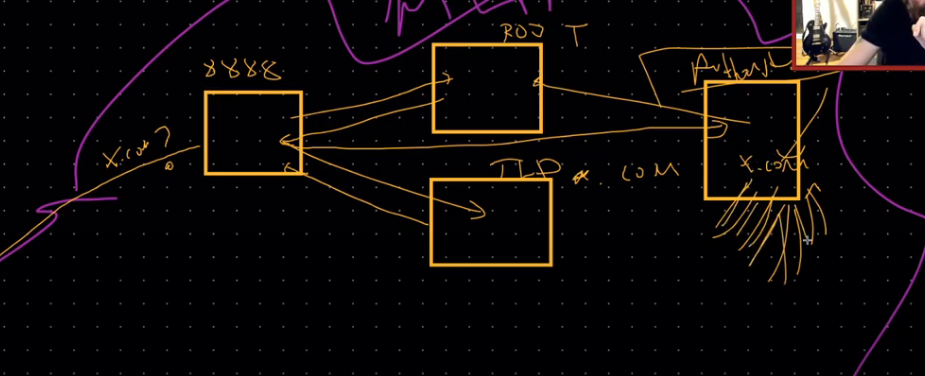
****

**Yerel ağ kısmı bu kadardı şimdi x.com yazdık burdan sonra neler olmaya başlıyor**

**Dış ağ:**

**x.com'a gitmeye çalıştığımızda, işletim sistemi önce hosts dosyamıza bakar ve x.com burada var mı diye kontrol eder. Eğer varsa, oradan açar; yoksa DNS devreye girer.**

**x.com yazdığımızda işletim sistemimizdeki DNS'e UDP 53 protokolü ile sorduk (örneğin, 8.8.8.8). "x.com'un IP adresi nedir?" dedik. Eğer DNS, bu soruya cevap veremiyorsa, sorgu önce root DNS'e, o da bilmiyorsa TLD'ye yönlendirdi. TLD, x.com'u çözmek için "Bu bilgiyi bilmiyorum, ama x.com'un kayıtlarını tutana git" dedi. Bu durumda, autotitafit DNS devreye girer ve x.com’un DNS sunucusuna ulaşılır.**

****

**Örneğin, dig hürriyet.com yazdığımızda, hürriyetin com.tr ile ilgili tüm DNS kayıtlarının bilindiği DNS sunucusu gösterilir. Autotitafit yani son gittiğimiz yer sadece x.com'un DNS sorgularına cevap vermelidir. Bu akışta, DNS'i öğrenip, bu bilgiyi kendi PC'mize kadar getirmiş olduk.**

**x.com’un IP adresi 1.3.3.7 oldu ve x.com'un IP'sini öğrendik.**

**PC1 yerine bu sefer ağımızdaki PC2, x.com’u sorduğunda bu süreç bir daha yaşanmaz; çünkü 8.8.8.8 doğrudan cevabı verecektir.**

**DNS'teki Zafiyetler:**

**Eğer 8.8.8.8’i yanıltırsanız, x.com’a yapılacak HTTP taleplerini kendi istediğiniz şekilde yönlendirebilirsiniz. Ayrıca, x.com’un DNS kayıtlarını ele geçirirseniz:**

* **MX kayıtlarını değiştirirseniz, tüm e-postaları üzerinize alabilirsiniz.**
* **Ağ kayıtlarını değiştirirseniz, tüm kayıtları üzerinize alabilirsiniz.**
* **TXT kayıtlarına istediğiniz bilgiyi girebilirsiniz.**

**Örneğin, dig mx hürriyet.com yazarsak, bu komut Outlook'un sunucusunda tüm e-postaların tutulduğunu gösterir.**

**TLD’lere Yönelik Siber Saldırılar:**

**Eğer TLD sunucuları hacklenirse, dünyadaki tüm \*\*\*\*.com adreslerinin cevabını sahte şekilde döndürebilirler.**

**Geçmişte, bir Türk hacker .tr domainlerinin sunucusuna erişim sağladı ve tüm .tr uzantılı alanları istediği gibi çözdürebiliyordu.**

**Özet:**

* **DNS sorgusu, hosts dosyasına bakılarak başlar.**
* **Eğer bulunamazsa, DNS devreye girer ve root, TLD sunucularına yönlendirilir.**
* **Sonuçta, x.com’un IP'si 1.3.3.7 olarak öğrenilir.**
* **DNS zafiyetleri, DNS sunucularını yanıltarak HTTP trafiğini kontrol etme, e-posta ve ağ kayıtlarını ele geçirme gibi tehlikeleri barındırır.**
* **TLD sunucularına yapılan saldırılar, tüm internet üzerindeki alan adı çözümleme işlemlerini etkileyebilir.**

**Bu adımlardan geçerek, internetin 1.3.3.7 IP adresine kadar ulaşılır.**

**TCP Paketi ve NAT Süreci**

**Bu olaylar tcp protokolü tarafından bu şekilde gerçekleşir :**

1. **Bağlantı Kurulması:  
   PC1 (10.0.0.5), internetteki x.com adresine bağlanmak istiyor. Ancak, internetteki cihazlar 10.0.0.x gibi yerel IP’leri tanımaz. Bu yüzden, bu bağlantının router üzerinden yönlendirilmesi gerekiyor.**
2. **Paketin Hazırlanması:  
   TCP paketi oluşturulur. İçinde olanlar:**
   * **Kaynak IP: 10.0.0.5**
   * **Kaynak Port: 45000 (örnek)**
   * **Hedef IP: x.com’un IP’si (örneğin, 1.3.3.7)**
   * **Hedef Port: 80 (HTTP için)**
3. **Router’a Ulaşması:**
   * **PC1 paketi hazırlar ve ağ geçidi (gateway) olan router’a yollar (10.0.0.1).**
   * **Router paketi alır ama 10.0.0.5’in IP’si internet için geçerli olmadığı için bunu değiştirmesi gerekir.**
4. **NAT Devreye Girer:**
   * **Router, PC1’in 10.0.0.5 olan IP adresini, kendi dış IP’si (örneğin 85.120.45.30) ile değiştirir.**
   * **Aynı zamanda 45000 olan kaynak portunu bir NAT tablosuna kaydeder.**

**Yeni Paket:**

* + **Kaynak IP: 85.120.45.30**
  + **Kaynak Port: 45000**
  + **Hedef IP: 1.3.3.7**
  + **Hedef Port: 80**

1. **Paket İnternete Çıkar:**
   * **Artık dış dünyada, bu paketin 85.120.45.30 IP’li cihazdan geldiği görülür.**
2. **Yanıt Gelmesi:**
   * **x.com’un sunucusu, yanıtı 85.120.45.30:45000 adresine yollar.**
   * **Router, NAT tablosuna bakar ve 45000 portunun 10.0.0.5’e ait olduğunu bilir.**
3. **Paket Yerel Ağa Geri Dönüş Yapar:**
   * **Router, gelen paketin IP’sini tekrar 10.0.0.5 olarak değiştirir ve PC1’e yollar.**

**Önemli Noktalar:**

**✅ Tüm cihazlar (PC1, PC2 vb.) internete çıkarken aynı dış IP’yi (85.120.45.30) kullanır.  
✅ Router, NAT tablosu sayesinde hangi cihazın hangi paketi gönderdiğini takip eder.  
✅ Bu sistem, aynı anda birçok cihazın internete çıkmasını ve dış dünyadan gizlenmesini sağlar.**

**Tcp paketi yerel ağ içinde:**

**--------------------------------------------------------------**

**| Kaynak IP: 10.0.0.5 | Hedef IP: 1.3.3.7 |**

**| Kaynak Port: 44321 | Hedef Port: 80 |**

**--------------------------------------------------------------**

**| Veri: x.com’a bağlanmak istiyorum |**

**--------------------------------------------------------------**

**Kaynak IP: 10.0.0.5 (Senin bilgisayarın yerel IP’si)**

**Hedef IP: 1.3.3.7 (İnternetteki x.com’un IP’si)**

**Kaynak Port: 44321 (Bilgisayarının rastgele atadığı çıkış portu)**

**Hedef Port: 80 (Web sunucusunun HTTP portu)**

**Tcp paketi dış ağa çıkarken:**

**-----------------------------------------------------------------**

**| Kaynak IP: 85.120.45.30 | Hedef IP: 1.3.3.7 |**

**| Kaynak Port: 55890 | Hedef Port: 80 |**

**-----------------------------------------------------------------**

**| Veri: x.com’a bağlanmak istiyorum |**

**-----------------------------------------------------------------**

**Kaynak IP: 85.120.45.30 (Modemin internet sağlayıcısından aldığı genel IP)**

**Hedef IP: 1.3.3.7 (x.com’un IP’si değişmez) Kaynak Port: 55890 (modemin atadığı yeni port)**

**hedef Port: 80 (Web sunucusuna gideceği için değişmez)**

**TCP Üçlü El Sıkışma (3-Way Handshake) Basitçe**

**İnternette bir siteye bağlanırken, önce bağlantının güvenli şekilde kurulması gerekir. Bunu sağlamak için SYN - SYN/ACK - ACK adımlarıyla bir iletişim başlatılır.**

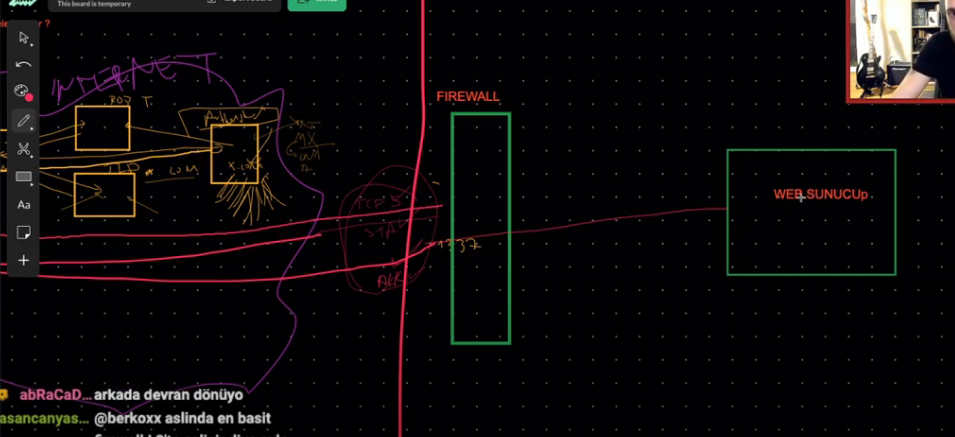
**1. SYN (İlk İstek Gönderme)  
Senin bilgisayarın, x.com ile bağlantı kurmak için bir istek yollar.  
 "Merhaba, ben seninle konuşmak istiyorum."**

**2. SYN-ACK (Sunucunun Cevabı)  
x.com’un sunucusu isteği alır ve seni onayladığını belirten bir yanıt yollar.  
 "Tamam, isteğini aldım. Sen de beni gördün mü?"**

**3. ACK (Bağlantıyı Onaylama)  
Senin bilgisayarın, sunucunun isteğini onaylar ve bağlantı tamamlanır.  
 "Evet, bağlantıyı kurduk. Artık veri gönderebiliriz."**

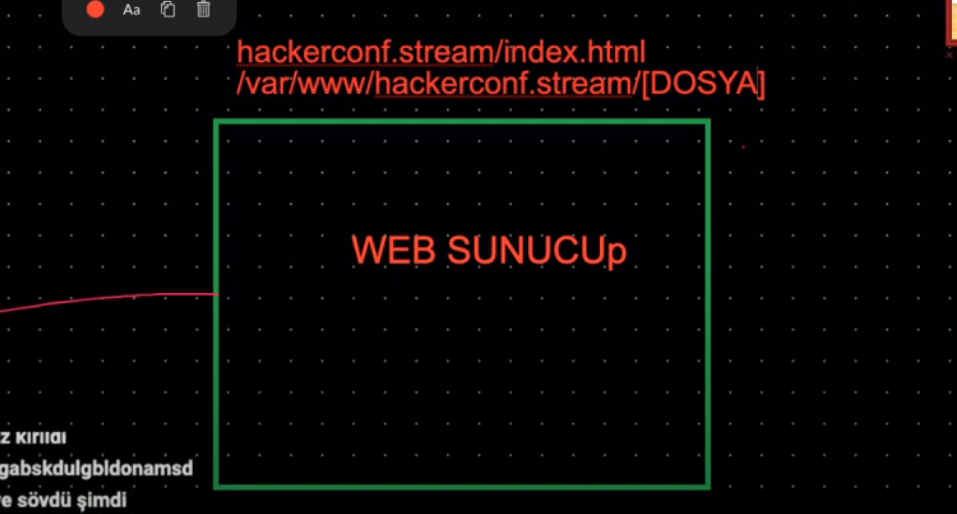
**Sonuç:  
Bağlantı kurulduktan sonra x.com’dan veri almaya başlarsın ve sayfa tarayıcında açılır.**

**Bu aşamada sayfayayı açma aşamasına geldik.**

****

**Kuruma sürekli sürekli tcp syn gibi istekler gönderildiğinde önce firewall a gider ve saldırgan thrreway handshake işlemini firewall ile konuşmuş olur**

**Ancak, saldırgan SYN-ACK yanıtlarına cevap vermezse veya sahte IP adresleri kullanarak istekleri gönderirse, bağlantılar tamamlanmaz ve firewall bu durumun anormal olduğunu fark edebilir. Eğer bu tür SYN istekleri aşırı artarsa, sfirewall bunu SYN Flood saldırısı olarak algılar ve belirli IP’leri engelleyebilir. Veya rate limiting kuralı ile Belli bir süre içinde çok fazla SYN isteği gelirse, yanıt vermeyi geciktirir veya durdurur. Ya da farklı yöntemler ile engeller.**

****

**Hackerconf.stream ‘i çağırdığımızda sunucunun içindeki /var/www/hackerconf.strem[Dosya] olarak gelecek**

**Bir e-ticaret sitesi düşünelim: hackerconf.stream. Başta tek bir sunucuda çalışıyordu ve gelen istekler /var/www/hackerconf.stream/ dizininden yanıtlanıyordu.**

**Ama satışlar patladı, siteye çok fazla insan girmeye başladı ve tek bir sunucu yetmemeye başladı. Ne yaptık?**

**Yeni web sunucuları ekledik → Aynı kaynak kodları 3 farklı sunucuya yükledik.  
Veritabanını ayrı bir sunucuya taşıdık → Web sunucularının hepsi aynı merkezi database ile konuşsun dedik.  
Reverse Proxy ekledik → Kullanıcılar siteye girince gelen istek direkt web sunucularına gitmesin, önce bir yük dengeleyici (reverse proxy) üzerinden geçsin dedik.**

**Reverse Proxy Nasıl Karar Veriyor?**

**Gelen istekleri şu mantıkla yönlendiriyor:**

* **Kullanıcının oturumu (session) hangi sunucunun diskinde kayıtlıysa, o sunucuya yönlendiriyor.**
* **Eğer yeni bir kullanıcı gelirse, sistem yük durumuna bakarak en az yoğun olan sunucuya gönderiyor.**

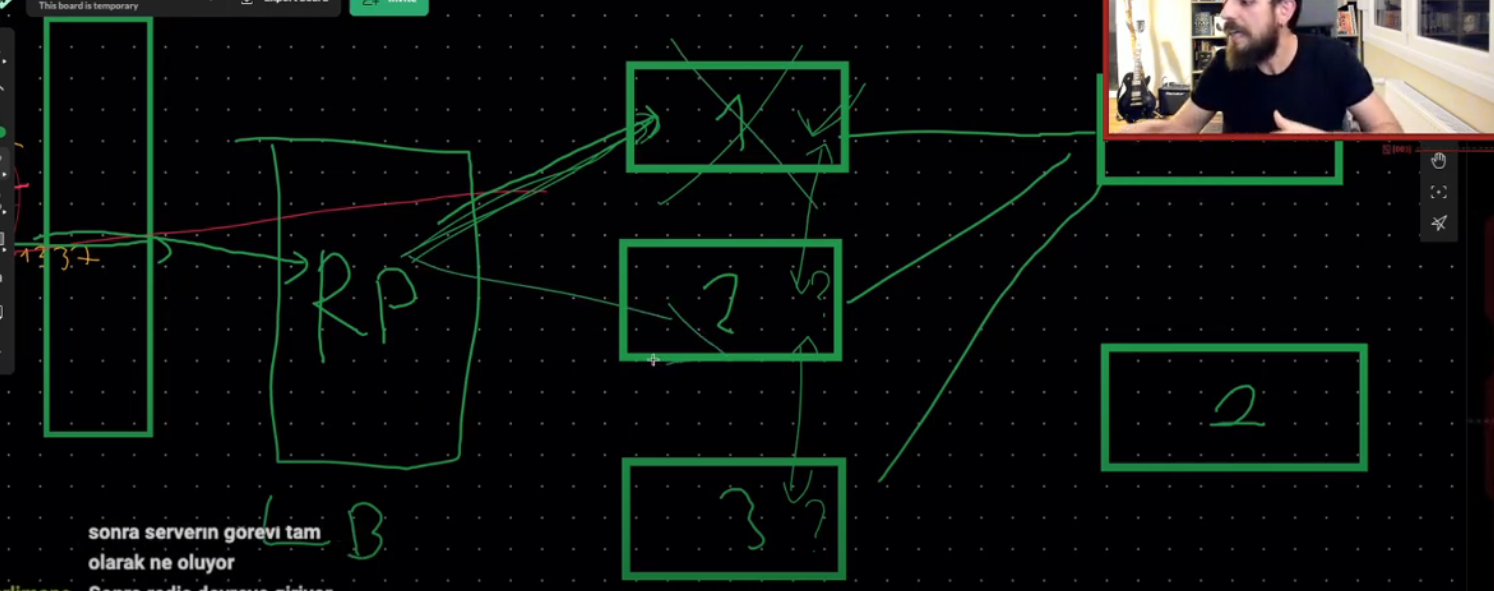
**Bu sistem sayesinde yük dengelenmiş oluyor, bir sunucu çökse bile diğerleri çalışmaya devam edebiliyor.**

**Daha anlaşılır olması açısından oyun sunucusundan örnek verelim**

**Oyun Sunucularında Nasıl Çalışır?**

**Oyuncu oyuna girerken önce ana giriş sunucusuna bağlanır.  
Reverse Proxy burada devreye girer ve oyuncunun hangi oyun sunucusuna bağlanacağını belirler.  
 Eğer bir oyuncu İstanbul’daysa, onu Türkiye’deki en yakın oyun sunucusuna yönlendirir. Eğer o sunucu doluysa, en az yoğun olan başka bir sunucuya atar.  
Oyun sırasında hangi sunucuda başladıysan, oyunun bitene kadar o sunucuda kalırsın (tıpkı web sitelerindeki session mantığı gibi).**

**Bu sistem sayesinde oyun daha hızlı çalışır, ping düşer, herkes aynı anda bağlansa bile yük eşit dağılmış olur. Eğer bir oyun sunucusu çökerse, oyuncular diğer sunuculara yönlendirilir ve oyun devam eder.**

****

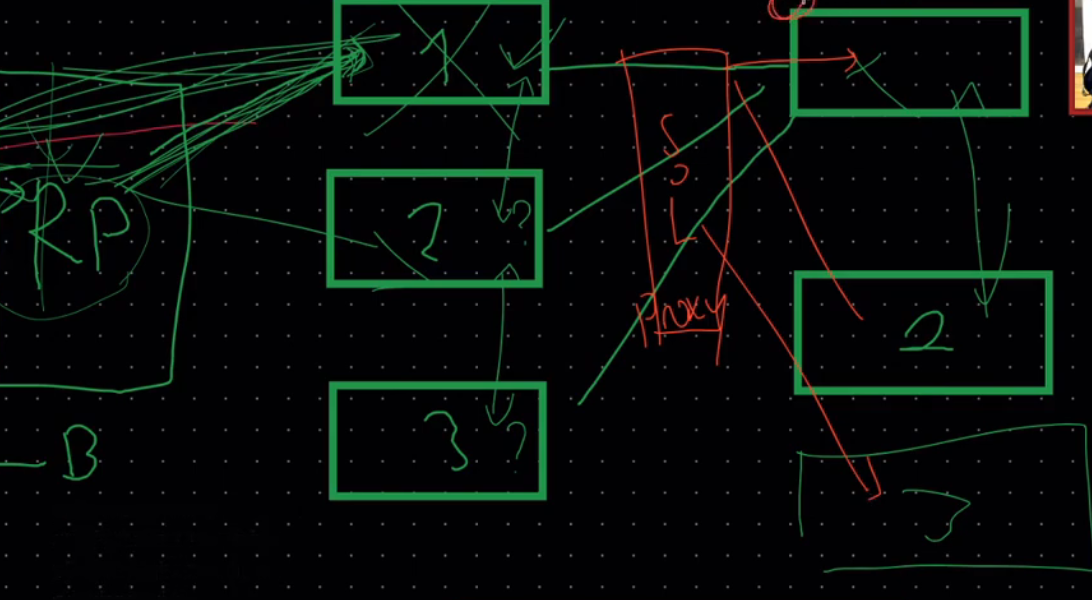
**Requestte cloudflare’in verdiği bu cookie nedir ?**

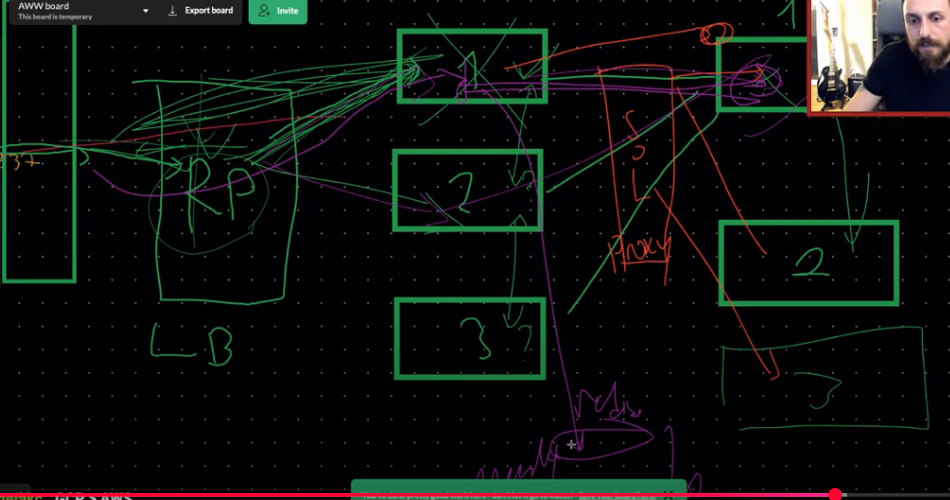
**Cloudflare’in verdiği cookie, seni önceden bağlandığın sunucuya yönlendirmek için kullanılır. Reverse proxy bu cookie’yi okur ve “Bu kullanıcı daha önce 1. sunucuya gitmişti, yine oraya göndereyim.” der. Ama bu cookie sadece client ile reverse proxy arasında çalışır, backend sunuculara geçmez.**

**Peki, tek bir veritabanı yetmediğinde ne olur?  
Diyelim ki 3 tane database var, sistemin yükü çok fazla ve tek bir database bu sorguları karşılamaya yetmiyor. İşte burada SQL Proxy devreye girer. SQL sorguları önce bu proxy’ye gider, o da hangi database müsaitse ona yönlendirir. Böylece yük dağıtılır, sistem hızlanır ve çökmez.**

**Terim açıklaması:**

**Client (İstemci): Sunucuya bağlanıp ondan hizmet alan cihaz veya yazılımdır. Örneğin, tarayıcıyla bir siteye girdiğinde tarayıcın bir client’tir.**

****

****

**Database çökerse ne olacak:**

**Diyelim ki elimizde 3 tane database var ama tek biri yetmiyor. Bu durumda SQL sorgularını yönlendirecek bir SQL Query Proxy servisine ihtiyacımız var. Tüm SQL sorgularını önce bu servise soruyoruz, o da hangi database uygunsa ona yönlendiriyor.**

**Peki session'ı ne yapacağız?  
Başta diskte tuttuk, ama diskte problem yaşanıyor. Sonra database'e taşıdık. Neden? Çünkü kullanıcı hangi sunucuya gelirse gelsin, session’ı oradan çekebilsin. Kullanıcının çerezi (cookie) ile session’a erişebiliriz.**

**Ama burada bir sorun var:  
Session’ları database’de tutarsak, yazma (write) işlemleri artar ve her işlem sonunda I/O (input/output) yükü oluşur. Yani sürekli diske yazma/silme yapmak performans kaybı yaratır.**

**Bu yüzden geçici verileri daha hızlı saklamak için session servisleri geliştirildi: Redis, Memcached gibi.**

**Özetle:**

* **Kullanıcı kullanıcı adı/parola girdi.**
* **SQL Proxy, sorguyu uygun database’e yönlendirdi.**
* **Database kullanıcıyı doğruladı.**
* **Session oluşturuldu ve session servisine (Redis/Memcached) yazıldı.**

**Böylece session işlemleri hızlı ve verimli bir şekilde yönetilmiş oldu.**

**Redis gibi session servisleri : RAM’i güçlü olan ve anlık verileri çok hızlı işleyebilen özel birer veri saklama sunucusudurlar. Memory de tutar veriyi**

**Database neden çöker?**

1. **Aşırı yük → Çok fazla sorgu gelirse database yetişemez, çöker.**
2. **Donanım arızası → Disk bozulabilir, RAM yetmeyebilir.**
3. **Yazılım hataları → Kötü optimize edilmiş sorgular sistemi yavaşlatır.**
4. **Bağlantı sorunları → Network kesintisi veya saldırılar erişimi engeller.**

**Çözüm:**

* **SQL Proxy → Gelen sorguları uygun database’e yönlendirir, yükü dengeler.**
* **Session Servisleri (Redis/Memcached) → Session’ları disk yerine RAM’de saklar, hızlı erişim sağlar, database yükünü azaltır.**

**Sırası ile işlemler şu şekilde :**

**Kullanıcı istek yaptı → Kullanıcı adı/parola girdi.**

**Reverse Proxy → İsteği uygun sunucuya yönlendirdi.**

**SQL Proxy → Database yükünü dengeleyerek uygun veritabanına sorguyu iletti.**

**Database → Kullanıcıyı doğruladı.**

**Session Servisi (Redis/Memcached) → Session oluşturuldu ve RAM’de saklandı.**

**Yanıt Kullanıcıya Döndü → Kullanıcı giriş yaptı, session bilgisi çerez (cookie) ile eşleştirildi.**

**Bir web sitesi düşünelim, sisteme dosya yüklüyoruz. Ama dosya sadece tek bir sunucuda kalırsa sorun çıkıyor. Mesela ben logout oldum, sonra tekrar giriş yaptım ama sunucu çöktü. O zaman bu dosyaya erişemem. İşte burada CDN (Content Delivery Network) devreye giriyor.**

**Sistem Nasıl Çalışıyor?**

1. **Kullanıcı dosya yüklüyor.**
   * **Normalde dosya direkt web sunucusunun diskine kaydedilirdi. Ama bu sorun yaratır çünkü web sunucusu çökerse dosya gider.**
   * **Bu yüzden modern sistemlerde dosya direkt CDN sunucularına gönderiliyor.**
2. **Dosyanın adı database’e kaydediliyor.**
   * **Uygulama sunucusu, database’e sadece dosyanın adını veya bağlantısını yazıyor.**
   * **Dosyanın kendisi uygulama sunucusunda tutulmuyor, CDN sunucularında saklanıyor.**
3. **Kullanıcı dosyaya erişmek istediğinde ne oluyor?**
   * **Web uygulaması, dosyanın kaydını database’den alıyor.**
   * **HTML sayfasında, dosyanın bulunduğu link CDN üzerinden veriliyor.**
   * **Kullanıcının tarayıcısı (browser), doğrudan CDN’den dosyayı çağırıyor.**
   * **Bu sırada ana sunucuya hiç yük binmiyor çünkü tüm trafik CDN üzerinden geçiyor.**

**CDN’nin Avantajları**

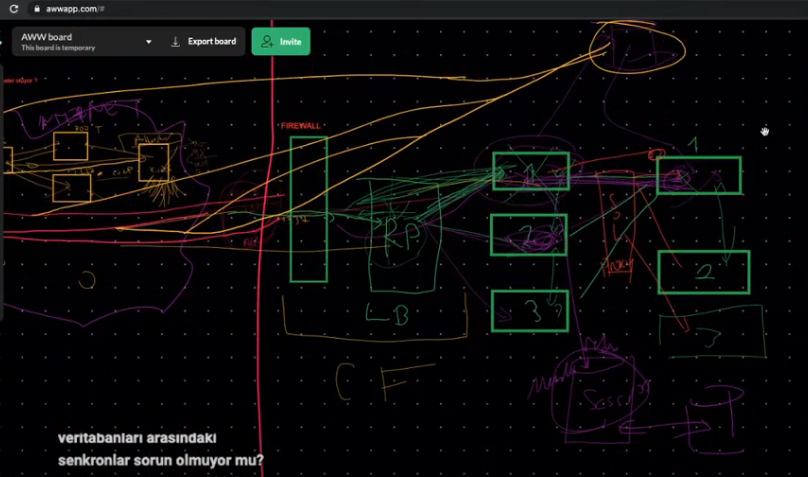
* **Yük dağıtılıyor: Kullanıcılar dünyanın her yerinden bağlanıyor ama tek bir sunucuya yüklenmiyor.**
* **Performans artıyor: CDN sunucuları dosyayı coğrafi olarak en yakın noktadan veriyor.**
* **Cloudflare gibi servisler cache (önbellek) kullanıyor:**
  + **Örneğin bir kullanıcı Çin’den geldi, dosyayı bir kere ana sunucudan aldı.**
  + **Sonraki kullanıcılar geldiğinde Cloudflare kendi önbelleğinden (cache) direk cevap dönüyor.**
  + **Böylece ana sunucudan tekrar dosya çekmeye gerek kalmıyor, hız kazanılıyor.**

**Güvenlik ve Reverse Proxy Bağlantısı**

* **CDN, reverse proxy gibi çalışıyor.**
* **Firewall ile korunuyor, saldırıları süzüyor.**
* **Kullanıcılar farkında olmadan aslında ana sunucuya değil, CDN’e bağlanıyor.**

**Sonuç:**

**CDN kullanarak statik dosyalar yönetilir, hız artar ve sunucu üzerindeki yük azalır. Daha önce konuştuğumuz ARP, TCP 3'lü el sıkışma gibi mekanizmalar burada da aynen devam eder.**

****

**CDN (Content Delivery Network) Nedir?**

**CDN, internet üzerindeki içerikleri daha hızlı ve verimli bir şekilde kullanıcılara iletmek için kullanılan bir ağdır. İçerikler, dünya genelindeki farklı sunucularda depolanır ve kullanıcıya en yakın sunucudan iletilir.**

**CDN Nasıl Çalışır?**

* **İçerikler, farklı lokasyonlardaki sunucularda depolanır.**
* **Kullanıcı, içerik talep ettiğinde, en yakın sunucu üzerinden içerik sağlanır.**

**Avantajlar:**

* **Daha hızlı yükleme: İçerikler daha yakın sunuculardan gelir, hız artar.**
* **Yük dengelemesi: Sunucular arasında trafik dengelenir.**
* **Daha iyi güvenlik: DDoS saldırılarına karşı dayanıklıdır.**
* **Kesintisiz erişim: Sorun yaşanan sunuculara rağmen diğer sunucular devreye girer.**

**Kullanım Alanları:**

* **Web siteleri, video yayınları, e-ticaret ve mobil uygulamalarda kullanılır.**

**Örnek Sağlayıcılar:**

* **Cloudflare, Akamai, Amazon CloudFront, Google Cloud CDN.**

**Özet:**

**CDN, içeriği hızlı ve güvenli bir şekilde sunmak için dağıtık sunucu ağı kullanır, bu da hız ve performans kazandırır.**

****

**ElasticSearch sunucuları :**

**Maliyet nedeniyle optimize edilmiş veri depolama ve sorgulama sistemleridir. Bu sunucular, gelen tüm HTTP isteklerini (örneğin, login işlemleri gibi) kaydeder ve analiz eder.**

**DRC (Disaster Recovery Center) :**

**DRC adı verilen yedek sistemler bulunur. DRC, ana sunucuların birebir kopyasıdır ve içindeki tüm veriler eşzamanlı olarak güncellenir. Eğer ana veri merkezi çökerse, DRC devreye girerek kesinti yaşanmadan hizmetin devam etmesini sağlar.**

**Bu yapı, yüksek erişilebilirlik (HA) ve veri güvenliği açısından kritik bir çözümdür.**

**ElasticSearch nasıl daha ucuz:**

**ElasticSearch'in daha ucuz olmasının sebebi, veriyi saklama ve işleme yönteminin daha verimli olmasıdır.**

* **Klasik veritabanları (MySQL, MSSQL gibi) çok fazla işlem yaparak çalışır, bu yüzden güçlü sunuculara ihtiyaç duyar, yani pahalıdır.**
* **ElasticSearch, veriyi daha kolay bir şekilde işler ve daha az donanımla çalışabilir.**

**Neden Daha Ucuz?**

1. **Daha az güçlü sunucuya ihtiyaç duyar, bu yüzden donanım masrafı düşer.**
2. **Lisans ücreti yoktur, yani kullandıkça ödeme yapmazsın.**
3. **Daha hızlı çalıştığı için büyük verileri işlemek için fazladan sunucu eklemene gerek kalmaz.**
4. **Depolama alanını daha iyi kullanır, bu yüzden fazladan disk satın almaya gerek kalmaz.**

**Özetle, daha az sunucu, daha az disk ve daha az güç harcayarak aynı işi yapar, bu yüzden daha az maliyetlidir.**

**ElasticSearch Hizmeti Örnekleri**

1. **Amazon OpenSearch (AWS) – Yönetilen ElasticSearch, otomatik çalışır.**
2. **Elastic Cloud – Elastic’in resmi bulut hizmeti.**
3. **Google Cloud ElasticSearch – Google altyapısıyla entegre çalışan ElasticSearch.**
4. **Azure Managed ElasticSearch – Microsoft Azure’un sunduğu yönetilen ElasticSearch.**

**Ayrıca, kendi sunucuna kurarak ücretsiz de kullanabilirsin.**

**SON**

Ekstra:

mülakat soruları ve terimler

**ARP (Address Resolution Protocol) Nedir?**

**ARP, bir IP adresini, o ağa bağlı cihazın MAC adresine çevirmek için kullanılan protokoldür. Yerel ağ içinde cihazların haberleşmesini sağlar.**

**Saldırı Türü: ARP Spoofing – Sahte MAC adresi ile ağa sızarak veri trafiğini yönlendirme saldırısıdır.**

**CSRF (Cross-Site Request Forgery) Nedir?**

**CSRF, kullanıcının tarayıcısındaki oturumunu kötüye kullanarak onun adına istek gönderen bir saldırıdır.**

**Örnek: Kullanıcı bankacılık sitesinde oturum açmışken kötü niyetli bir site, onun adına para transferi talebi gönderebilir.**

**Korunma Yöntemleri:**

* **CSRF token kullanımı**
* **Referer header kontrolü**
* **Doğrulama mekanizmaları (CAPTCHA, şifre onayı gibi)**

**3. TCP ve UDP Nedir? Farkları Nelerdir?**

**Bunlar, cihazların internet üzerinden iletişim kurmasını sağlayan taşıma katmanı protokolleridir.**

| **Özellik** | **TCP (Transmission Control Protocol)** | **UDP (User Datagram Protocol)** |
| --- | --- | --- |
| **Bağlantı** | **Bağlantı tabanlı (Connection-Oriented)** | **Bağlantısız (Connectionless)** |
| **Güvenilirlik** | **Veriyi sıralı ve eksiksiz yollar** | **Veri kaybı olabilir, sıralama yok** |
| **Hız** | **Daha yavaş (Doğrulama mekanizmaları var)** | **Daha hızlı (Doğrulama yok)** |
| **Kullanım Alanları** | **Web (HTTP, HTTPS), E-posta, FTP** | **Canlı yayın, VoIP, DNS, Oyunlar** |

**XSS (Cross-Site Scripting) Nedir?**

**XSS, kötü niyetli kodun (genellikle JavaScript) bir web sitesine enjekte edilerek kullanıcıların tarayıcısında çalıştırılmasıdır.**

**XSS Türleri:**

* **Stored XSS (Kalıcı XSS): Zararlı kod veritabanına kaydedilir, tüm kullanıcıları etkiler.**
* **Reflected XSS (Yansıyan XSS): Zararlı kod, URL veya form üzerinden çalıştırılır, kullanıcıyı kandırarak çalıştırması sağlanır.**
* **DOM-Based XSS: Kullanıcının tarayıcısında doğrudan çalışır, sunucuya istek gitmeden oluşur.**

**Korunma Yöntemleri:**

* **Input validation (Giriş doğrulama) ve output encoding (çıkış kodlaması)**
* **CSP (Content Security Policy) kullanımı**
* **HTML özel karakter kaçırma işlemi (escaping)**

ZAFİYETLER

**Dosya Yolu Geçiş Zafiyeti (File Path Traversal )**

Dosya yolu geçiş zafiyeti, saldırganın uygulamanın normalde erişemeyeceği dosyalara erişmesini sağlar. Genellikle dizin değiştirme karakterleri (../) kullanılarak üst dizinlere çıkılır ve hassas dosyalara ulaşılır.

Örneğin, bir web sitesi belirli bir dosyayı göstermek için URL'de bir parametre kullanıyorsa:

https://example.com/view?file=report.pdf

Bir saldırgan bu parametreyi değiştirerek şu şekilde sistem dosyalarına erişebilir:

https://example.com/view?file=../../../../etc/passwd

Bu sayede sistemdeki kullanıcı bilgilerini içeren dosya görüntülenebilir.

**Korunmasız Yönetici İşlevselliği (Unprotected Admin Functionality)**

Bir web uygulamasında yönetici (admin) işlevselliklerine yeterli erişim kontrolü uygulanmadığında, yetkisiz kişiler yönetim paneline veya kritik sistem ayarlarına erişebilir.

Örneğin, bir web sitesi yönetici girişini doğrulamadan doğrudan bir panel sunuyorsa:

https://example.com/admin

Bir saldırgan, herhangi bir kimlik doğrulaması yapmadan bu sayfaya erişebilir ve yönetici yetkilerine sahip olabilir. Eğer doğrudan belirli işlemleri gerçekleştiren URL’ler varsa, bunları da kullanabilir:

https://example.com/admin/deleteUser?user=1234

Bu durumda saldırgan, URL’yi kullanarak yetkisiz bir şekilde kullanıcıları silebilir veya sistemde değişiklik yapabilir.

**Öngörülemeyen URL ile Korumasız Yönetici İşlevselliği (Unprotected Admin Functionality via Unpredictable URL)**

Eğer bir web uygulamasındaki yönetici paneli veya kritik işlemler, özel bir kimlik doğrulama yerine sadece tahmin edilmesi zor bir URL’ye dayanıyorsa, saldırganlar bu URL’yi keşfederek yetkisiz erişim sağlayabilir.

Örneğin, yönetici paneline giriş yapmak için herhangi bir doğrulama olmadan sadece belirli bir URL’ye erişim yeterliyse:

https://example.com/secret-admin-panel-98765

Bir saldırgan bu URL’yi tahmin ederek ya da dizin tarama araçlarıyla (dirb, gobuster gibi) keşfederek giriş yapabilir ve yönetici yetkilerine sahip olabilir. Aynı şekilde, belirli işlemleri gerçekleştiren özel URL’ler (örneğin kullanıcı silme veya ayar değiştirme) doğrudan erişime açıksa, saldırganlar bunları kullanabilir. Javascript ve Html kodlarından BurpSuite Üzerinden Bulabilir.

**Parametre Tabanlı Erişim Kontrol Yöntemleri (Parameter-Based Access Control)**

Bazı web uygulamaları, kullanıcı yetkilendirmesini sadece URL parametreleri veya istemci tarafından kontrol edilebilen diğer veriler üzerinden yaptığı için saldırganlar bu değerleri değiştirerek yetkisiz erişim sağlayabilir.

Örneğin, bir kullanıcı profil sayfası şu şekilde çalışıyorsa:

https://example.com/profile?user=123

Bir saldırgan, user parametresini değiştirerek başkasının hesabına erişebilir:

https://example.com/profile?user=456

Eğer sistem sadece bu parametreye dayanarak yetkilendirme yapıyorsa, saldırgan başkasının bilgilerini görebilir veya değiştirebilir.

**Güvensiz Erişim Kontrol Mekanizmaları**

Bazı uygulamalar kullanıcının erişim haklarını veya rolünü oturum açma sırasında belirler ve ardından bu bilgileri kullanıcı tarafından kontrol edilebilen bir konumda depolar. Bu şunlar olabilir:

* **Gizli bir alan (Hidden Field)**
* **Bir çerez (Cookie)**
* **Önceden ayarlanmış bir sorgu parametresi (Query String Parameter)**

Uygulama, gönderilen değere göre erişim kontrol kararları verir. Örneğin:

<https://insecure-website.com/login/home.jsp?admin=true>  
<https://insecure-website.com/login/home.jsp?role=1>

Bu yaklaşım güvenli değildir çünkü bir kullanıcı değeri değiştirerek yönetici yetkileri kazanabilir ve yetkisiz erişim sağlayabilir.

**Kullanımı:**

Burp Suite Proxy ile isteği yakala.

Intercept is on durumundayken, GET veya POST parametrelerini değiştir (örneğin role=user → role=admin). (administrator de olabilir)

Forward diyerek isteği gönder ve yanıtı kontrol et.

HTTP History sekmesinde değişikliklerin etkisini incele.

**Yatay Ayrıcalık Yükseltme (Horizontal Privilege Escalation)**

Kullanıcı, kendi hesabının dışında, aynı seviyede başka bir kullanıcının bilgilerine erişim sağlar. Burada kullanıcı kimliği tahmin edilmez, ancak kullanıcının yetkileri sınırlandırılmamıştır ve saldırgan, başka bir kullanıcının hesabına (aynı rol seviyesinde) erişebilir.

Bazı uygulamalarda, kullanıcı kimlikleri URL veya parametrelerde öngörülebilir şekilde gönderilir. Örneğin, kullanıcı ID'si 123 olan bir profil şu şekilde olabilir:

https://example.com/profile?id=123

Bir saldırgan, ID'yi değiştirerek başka kullanıcıların profillerine erişebilir:

https://example.com/profile?id=124

Bu tür öngörülebilir kimlikler, güvenlik açığı oluşturur çünkü saldırgan kimlik numaralarını tahmin ederek yetkisiz erişim sağlayabilir.

**Kullanımı:**

Blogda gezerken Carlosun profiline gir

<https://example.com/blogs?userId=32b42fdb-8c01-4888-aeb0-ad577643fe6c>

bu şekilde user id’i gösteriyor şimdi kendi profilimize girelim

<https://example.com/my-account?id=4407ca72-ced5-4ba8-a2b3-3cbcf69b969b>

buradaki id değerini Carlosun id değeri ile değiştiriyoruz. Bu sayede Carlos hesabına girmiş olduk.

**CSRF (Çapraz Site İstek Sahteciliği)**

**kötü niyetli bir web sitesinin, sen farkında olmadan senin tarayıcını kullanarak başka bir siteye senin adına yetkisiz işlemler yaptırmasıdır.**

**CSRF, saldırganın seni kandırarak istemediğin bir işlemi yapmanı sağladığı bir saldırı türüdür.**

1. **Sen bankana giriş yaptın. (**[**https://ziraat.com**](https://ziraat.com)**)**
2. **Tarayıcın oturum çerezlerini kaydetti. (Banka, seni tanıyor)**
3. **Saldırganın kötü niyetli sitesine girdin. (**[**https://sahteziraat.com**](https://sahteziraat.com)**)**
4. **Saldırganın sitesi, senin tarayıcın üzerinden bankaya para transferi isteği gönderiyor.**
5. **Bankanın sistemi, senin tarayıcından geldiği için isteği gerçek sanıyor ve işlemi onaylıyor.**

**Şifre neden istenmez?**

**Sen zaten giriş yapmışsın. Banka seni tanımış, tarayıcında çerezler var.Tarayıcın otomatik olarak çerezleri ekliyor.  
Banka, çerezleri gördüğü için işlemi şifre sormadan onaylıyor.**

**CSRF koruması olan bankalar şunları yapar:**

**CSRF Token Kullanır → Her isteğe özel bir kod ekler.  
Ekstra Kimlik Doğrulama → Önemli işlemlerde tekrar şifre sorar.  
Referer Header Kontrolü → İstek hangi siteden geldi, kontrol eder.**

**Kullanımı: (portswiggerda denemedim)**

**GET Request ile Sömürme**

**Eğer banka sistemi GET request ile para transferi yapıyorsa, daha kolay bir saldırı senaryosu yazabiliriz.**

**Örneğin, banka aşağıdaki gibi GET isteği ile işlem yapıyorsa:**

[**https://bankam.com/transfer?amount=5000&to=TR1234567890**](https://bankam.com/transfer?amount=5000&to=TR1234567890)

**html ile saldırı: gizli resim yöntemi**

**<img src="https://bankam.com/transfer?amount=5000&to=TR1234567890">**

**Kurban bu sayfaya girdiğinde, resim yükleniyormuş gibi görünür ama aslında banka hesabından para gider. Veya sayfaya girer girmez javascript ile işlem arka planda gerçekleşir**

**SSRF (Server-Side Request Forgery - Sunucu Taraflı İstek Sahteciliği)**

**saldırganın bir sunucuyu kandırarak başka bir sunucuya istekte bulunmasını sağladığı bir güvenlik açığıdır.**

**Örnek:**

**Bir web sitesi, kullanıcıdan bir URL alarak içeriğini çekip gösteriyor:**

**fetch("https://example.com/get-page?url=http://hedef-site.com")**

**Burada normalde kullanıcı, herhangi bir web sayfasını görüntüleyebilir. Ama saldırgan, iç ağa veya yetkili API’lere istek yaptırabilir**

**https://example.com/get-page?url=http://localhost/admin**

**Bu sayede saldırgan, normalde erişemeyeceği iç sistemleri görebilir veya hassas verilere ulaşabilir.**

**Bu açık, genellikle kötü filtrelenmiş URL yönlendirmelerinden veya dışardan alınan girdilerin sunucu tarafında işlenmesi nedeniyle oluşur.**

**Kullanımı:**

**E ticaret sitesinde stok kontrolü yapan bir parametre olduğunu farkettik**

**Herhangi bir ürüne girip stok kontrolü yaptık ve bunu burp ile yakalyıp repeatera gönderdik ardından api= kısmını** [**http://localhost**](http://localhost) **olarak değiştirdik ve iç ağa yetkili apiye istek yaptık**

**Render dediğimizde admin panelini gördük Response da incelediğimizde admin yazısının görüp** [**http://localhost/admin**](http://localhost/admin) **olarak değiştirdik ve Responsda**

**<a href="/admin/delete?username=wiener"> bunu farkedip**

**stockApi=http://localhost/admin/delete?username=Carlos olarak değiştirerek carlosu sildik**

**aynı senaryo localhost yerine** [**http://192.168.0.1:8080/admin**](http://192.168.0.1:8080/admin) **gibi de olabilir ipnin son hanesini 1den 255e kadar bruteforce atarak geçerli ip i bulduk**

**Parola Ifşasıyla İstek Parametresi Tarafından Kontrol Edilen Kullanıcı Kimliği**

Bu güvenlik açığı, uygulamanın kullanıcı kimliğini ve şifreyi kontrol etme şekliyle ilgilidir. Eğer uygulama, kullanıcı kimliği ve şifreyi URL parametreleri veya istek body’sinde gönderiyorsa, bu bilgiler kolayca ele geçirilebilir ve manipüle edilebilir.

**Zafiyetin Açıklaması:**

**Parola ifşası**: Uygulama, şifreyi açıkça URL parametrelerinde veya HTTP isteklerinde gönderiyorsa, bu şifre ifşa olur. Bu, saldırganın bu bilgiyi yakalayarak başka bir kullanıcı veya yönetici hesabına erişim sağlamasına neden olabilir.

**İstek Parametresi ile Kullanıcı Kimliği Kontrolü**: Kullanıcı kimliği, genellikle **id** parametresi gibi URL parametrelerinde kontrol edilir. Eğer **id** parametresi değiştirilerek yönetici kimliği girilirse, yönetici yetkilerine sahip kullanıcı bilgilerine ulaşılabilir.

<https://example.com/user_account?id=carlos>

Eğer id parametresi, başka bir kullanıcının kimliğiyle değiştirilirse (örneğin, id=administrator), yönetici bilgilerine ulaşılabilir

**Kullanımı:**

Kendi hesabımıza login olup o isteği burp ile repeater’a gönderiyoruz

my-account?id=wiener yazan yeri my-account?id=administrator olarak değiştiriyoruz

Response kısmında html koldarı içinden admin kullanıcı adı ve şifreyi buluyoruz ve giriş yapıyoruz

**Kimlik Doğrulama Açıkları (Authentication Vulnerabilities)  
Kimlik doğrulama açıkları**

saldırganların hassas verilere ve işlevlere erişmesini sağlayan açıklardır. Bu zafiyetler, kimlik doğrulama süreçlerindeki zayıflıklardan kaynaklanabilir ve genellikle şu şekilde istismar edilir:

* Zayıf şifreler
* Brute-force (kaba kuvvet) saldırıları
* Kullanıcı adı numaralandırma
* Kimlik doğrulama token'larının zayıf yönetimi  
  Bu tür açıklar, yetkisiz erişim sağlanmasına ve daha fazla güvenlik açığının ortaya çıkmasına yol açabilir.

**Kimlik Doğrulama ve Yetkilendirme Farkı (Authentication vs. Authorization)**  
Kimlik doğrulama, bir kullanıcının gerçekten iddia ettiği kişi olup olmadığını doğrulama işlemidir. Yetkilendirme ise bir kullanıcının bir işlemi yapmaya yetkili olup olmadığını kontrol eder.

**Brute-force (Kaba Kuvvet) Saldırıları (Brute-force Attacks)**  
Brute-force saldırısı, bir saldırganın kullanıcı kimlik bilgilerini tahmin etmek için deneme-yanılma yöntemini kullanmasıdır. Bu saldırılar genellikle kullanıcı adı ve parola listeleri ile otomatikleştirilir. Saldırganlar, bu süreci daha verimli hale getirmek için mantık kullanabilir veya halka açık bilgilere dayanarak tahminlerde bulunabilir.  
Bu tür saldırılar, şifre korumasız veya zayıf şifreler içeren sistemlerde ciddi güvenlik tehditlerine yol açar.

**Kullanıcı Adı Numaralandırma (Username Enumeration)**  
Kullanıcı adı numaralandırma, bir saldırganın, web sitesinin davranışındaki değişiklikleri gözlemleyerek geçerli bir kullanıcı adını belirlemesidir. Bu genellikle giriş sayfasında, doğru bir kullanıcı adı girildiğinde ancak yanlış parola girildiğinde meydana gelir.  
Ayrıca kayıt sayfasında, zaten alınmış bir kullanıcı adı girildiğinde de benzer bir durum yaşanabilir. Bu, saldırganların geçerli kullanıcı adlarını hızlıca listelemelerini sağlar, böylece brute-force saldırılarını daha verimli hale getirir.

**Kullanımı:**

**Burp ile bruteforce saldırsı :**

Hedef uygulamanın login sayfasına git ve geçersiz bir kullanıcı adı ve şifre girerek bir istek gönder. (testkullanıcı testşifre)

2. Burp Intruder’a Geçiş

Burp Suite'te Proxy > HTTP History sekmesine git. Gönderdiğin giriş isteğini bul ve bu isteği Burp Intruder'a gönder.

3. Username Parametresini Hazırlayın

Burp Intruder’da username parametresinin etrafındaki § sembollerini göreceksin (örneğin: username=§invalid-username§). Bu parametreyi seç payload pozisyonu olarak ayarla. Şifre geçici olarak testşifre kalsın

4. Payload Türünü Ayarla

Intruder'da Sniper attack seçeneğinin aktif olduğundan emin ol. Payloads paneline git ve Simple list payload türünü seç.

5. Kullanıcı Adı Listesini Yapıştır

Kullanıcı adı adaylarını içeren listeyi payload bölümüne yapıştır.

6. Saldırıya Başla

Start attack butonuna basarak saldırıyı başlat.

7. Sonuçları İncele

Saldırı tamamlandıktan sonra, Length sütununu kontrol et ve bir girişin diğerlerinden daha uzun olduğunu fark et. Bu, geçerli bir kullanıcı adı olduğunu gösterebilir. Invalid username mesajıyla karşılaşılan diğer cevaplardan farklı olarak, Incorrect password mesajını içeren yanıtı bul. Geçerli kullanıcı adını not al.

8. Şifreyi Brute-Force Et

Saldırıyı sonlandır ve username parametresini doğru kullanıcı adıyla değiştir. Ardından password parametresi için yeni bir payload pozisyonu ekle. Şimdi username=identified-user&password=§invalid-password§ şeklinde olmalı.

9. Şifre Listesini Yapıştır

Şifre adayları listesini payload bölümüne yapıştır ve tekrar Start attack butonuna bas.

10. Şifreyi Doğrula

Saldırı bitince, Status sütununu kontrol et. 200 durum kodu alan istekler başarısız girişleri, 302 durum kodu alan ise başarılı girişi gösterir. 302 yanıtını alan şifreyi not al. Giriş yap.

**2FA simple bypass (basit 2fa atlatma)**

Zayıf 2FA Uygulaması: E-posta veya SMS doğrulama kodlarını ele geçirme. Şifre sıfırlama sayfasında, hedefin e-posta adresini girip doğrulama kodunu almak.

TOTP Zaman Hatası: Zaman tabanlı 2FA (TOTP) kullanılıyorsa, zaman senkronizasyonu hatasından faydalanarak doğrulama kodunu tahmin etme.

Sosyal Mühendislik: Kullanıcıyı manipüle ederek, 2FA kodunu almak. Örneğin, telefon numarasını veya e-posta adresini ele geçirip, kullanıcıyı kandırarak kodu almak.

Çerez Ele Geçirme: XSS saldırısı ile kullanıcı oturum çerezlerini ele geçirip, oturum açmak.

SIM Swap: Hedefin telefon numarasını ele geçirerek, SIM kartını takas ettirip, SMS doğrulama kodlarını almak**.**

**Port Swigger çözümü:**

Kendi hesabına eriş linki not et /my-account

Hedef hesaba giriş yap doğrulama gelince url’e /my-account yaz giriş yap

Bu zafiyet, sistemin 2FA doğrulama mekanizmasının, kullanıcı kimlik doğrulama aşamasından sonra URL manipülasyonu ile bypass edilmesine olanak tanımasından kaynaklanır. Yani, sistem sadece şifreyi kontrol etmekte, ancak 2FA doğrulamasını geçerli kullanıcının hesabıyla bağdaştırmada eksik kalmaktadır.

**Bettercap ile arp poisoning (ekstra)**

**Wifi kartını monitör moda alma**

**sudo ip link set wlan0 down**

**sudo iw dev wlan0 set type monitör**

**sudo sysctl -w net.ipv4.ip\_forward=1 ip forwardı 1 e ayarladık**

**bettercap kullanımı**

**bettercap -iface eth0 diyerek ağ arayüzümü seçerek betercapı başlattık**

**net.probe on**

**net.show bağlı ip adreslerini gösteriyor**

**set arp.spoof.fullduplex true**

**set arp.spoof.targets 192.168.10.1 hedefimi tüm ağ olarak aldım**

**arp.spoof on**

**net.sniff on dedik ve başladık**