Hazırlayan : Serhat Yıldız

Bölüm A: Teori ve Mimari (Research & Logic)

1. Web Anatomisi: HTTP Protokolü

- Tarayıcınız bir web sitesine girerken sunucuyla konuşur. Sunucunun verdiği cevaplardaki HTTP Durum Kodlarından (Status Codes) 200, 302, 401, 403 ve 500 kodlarının bir pentester (sızma testi uzmanı) için anlamı nedir? (Örn: 403 gördüğümüzde ne anlarız, pes mi ederiz?)

200 OK (istek Başarılı) kodu içeride olduğumuzu gösterir. Dizini veya dosyayı bulduk demektir. Ayrıca fuzzing yaparken bu kodu almak hedefe ulaştığımızı gösterir.

302 Found (Geçici Yönlendirme) : Genelde login sonrası yönlendirmelerde çıkar. Login Bypass veya Open Redirect açıkları için harika bir ipucudur.

401 Unauthorized (Yetkisiz Giriş) : Evet önümüzde bir kapı vardır ama elimizde anahtarımız olmadığı durumlarda karşımıza çıkan koddur. Brute force (kaba kuvvet) saldırısı için hedef noktasıdır.

403 Forbidden (Yasaklandı) : Giriş yasak olduğunu söyleyen koddur. Bir pentester için "Burada çok değerli bir hazine var ve birileri burayı korumaya çalışıyor" anlamına gelir ve genellikle Bypass tekniklerini deneriz.

500 Internal Server Error (Sunucu Hatası) : Sunucunun devreler gitti anlamına gelen bir kod da denebilir . Çünkü gönderdiğimiz bir veri sunucuyu çökertmiş olabilir. SQL Injection veya Code Execution için en büyük belirtidir.

2. Kimlik ve Yetki: "Sen Kimsin?"

- Cookie, Session ve Token: Bu üç kavramın farkı nedir? Bir web sitesi, sayfayı her yenilediğimizde bizim "aynı kişi" olduğumuzu nasıl anlar?

Cookie(Çerez) : Sunucu bizim tarayıcımıza bıraktığı küçük bir veri dosyası sayesine bizi tanır. O siteye attığımız her istekte biz o veriyi sunucuya biz olduğumuzu tanıması için göndeririz.

Session(Oturum) : Sunucu bizim verilerimizin hepsini içinde saklar ve bize ise bir ID atar. Biz her geldiğimizde bizi o ID ile tanır.

Token(Jeton) : Genellikle tarayıcmızın Local Storage alanında veya Session Storage alanında saklanır. Sunucu bize dijital imzalı bir paket verir ve biz her istek attığımızda paketin doğru olup olmadığını kontrol edip bizi içeri alır . Burada dikkat etmemiz gereken bir şey var : Paketimizdeki imzayı not almıyor sadece sunucu imzanın gerçek olup olmadığına bakıyor.

Kısaca şöyle ayırt edebiliriz : Cookie , bizim tarayıcımızdaki küçük notlar ; Session , sunucunun hafızasındaki klasör iken Token ise modern ve bağımsız kimlik kartı gibidir.

- **Authentication vs Authorization:** Biri "Kimlik Doğrulama", diğeri "Yetkilendirme"dir. Sisteme başarıyla giriş yapmak (Login) hangisidir? Girdiğiniz sistemde başkasının mesajlarını okuyabilmek (IDOR) hangisinin zafiyetidir?

Authentication bir kimlik doğrulama aşamasıdır. Sistem sana "Sen kimsin ?" sorusunu sorar. Sisteme başarıyla giriş yapmak işlemi tam olarak budur. Authorization ise yetkilendirme aşamasıdır (Örnek vereyim : Bir kullanıcı bir dosyayı sadece okuyabilir ya da hem okuyabilir hem silebilir demek bu aşamanın görevidir.). Sistem sana " Neler yapabilirsin ?" sorusunu sorar . Girdiğimiz sistemde başkasının mesajlarını okuyabilmek (IDOR - Insecure Direct Object Reference) kesinlikle bir Authorization (Yetkilendirme) zafiyetidir. Eğer hedefimiz admin paneline girmekse, yapacağımız saldırı türleri (Brute Force, SQLi Authentication Bypass) Authentication mekanizmasını kırmaya yöneliktir. Eğer sisteme zaten düşük yetkili bir kullanıcı olarak girdiysek ve adminin yapabileceği işlemleri yapmaya veya başkasının verilerini okumaya

çalışıyorsak (Privilege Escalation, IDOR), burada aradığımız açık Authorization eksikliğidir.

3. Zafiyet Mimarisi: IDOR

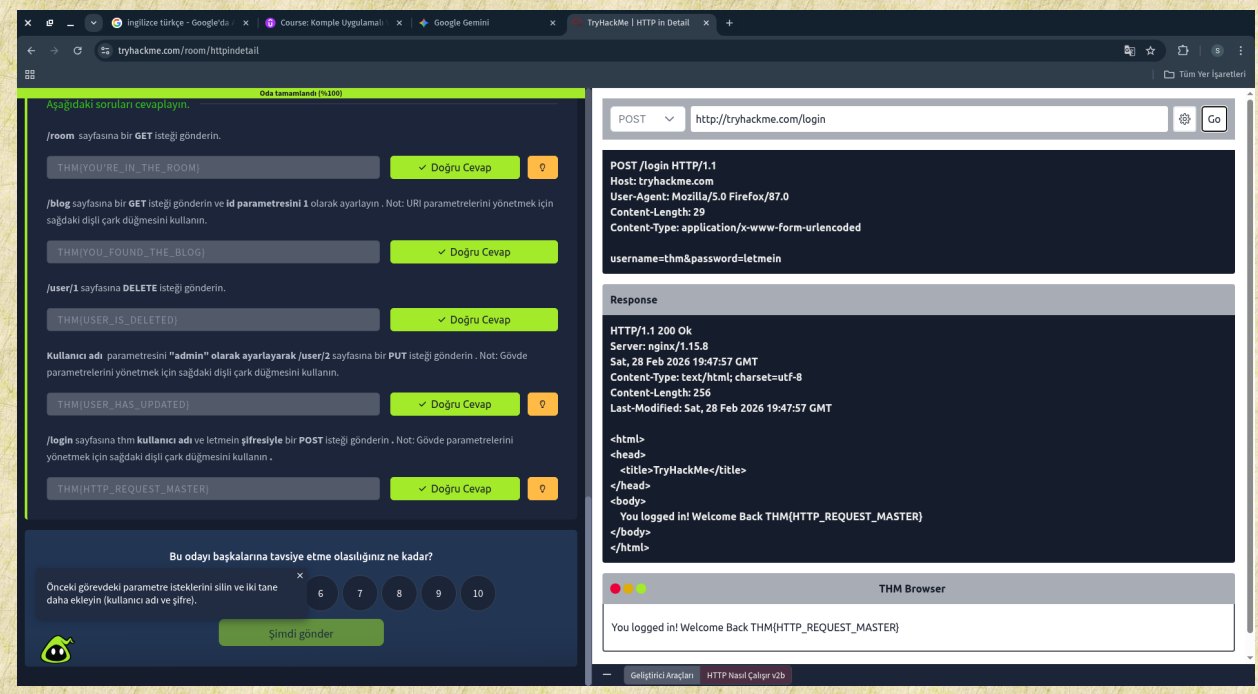
- **IDOR (Insecure Direct Object Reference)** nedir? Bir yazılımcı hangi güvenlik kontrolünü yapmayı unutursa uygulamasında IDOR zafiyeti ortaya çıkar?

Anlamı "Güvensiz Doğrudan Nesne Başvurusu" olan IDOR, bir web uygulamasında kullanıcının veritabanındaki bir nesneye (dosya, mesaj, fatura, profil vb.) dışarıdan müdahale edilebilir bir parametre (örneğin URL'deki bir ID numarası) üzerinden doğrudan erişebilmesidir. Diyelim ki adres çubuğunda ....?id=101 şeklinde ise biz id değerini 105 yaptığımız zaman bizi farklı bir kullanıcıya ait bir kullanıcıyı gösteriyorsa bu unutulan Object-Level Authorization kontrolünden kaynaklı bir IDOR zaafiyeti oluşmuştur.

Bölüm B: Saha Eğitimi (TryHackMe & Dorking)

Görev 1: İnternetin Röntgeni (TryHackMe)

- Oda: TryHackMe | HTTP in Detail (Veya alternatif olarak Burp Suite: The Basics odası).
- İstenen: Odayı tamamlayın. HTTP Request (İstek) başlıklarındaki User-Agent ve Host parametrelerinin ne işe yaradığını kendi cümlelerinizle açıklayın.



- Görev 2: Google Dorking (Açık Kaynak İstihbaratı) Google sadece bir arama motoru değil, dünyanın en büyük zafiyet tarayıcısıdır.
- Senaryo: Hedefiniz ankasec.co (veya seçtiğiniz başka bir Bug Bounty hedefi).
- Görev: Hedefle ilgili potansiyel olarak hassas dosyaları (Admin paneli, PDF belgeleri, Backup dosyaları vb.) bulmak için kullanacağınız 3 farklı Google Dork sorgusu yazın ve bu sorguların tam olarak ne aradığını açıklayın. (Örn: site:hedef.com ext:pdf).
- Kanıt: Yazdığınız Dork'lardan birini Google'da aratıp sonucunun ekran görüntüsünü ekleyin.

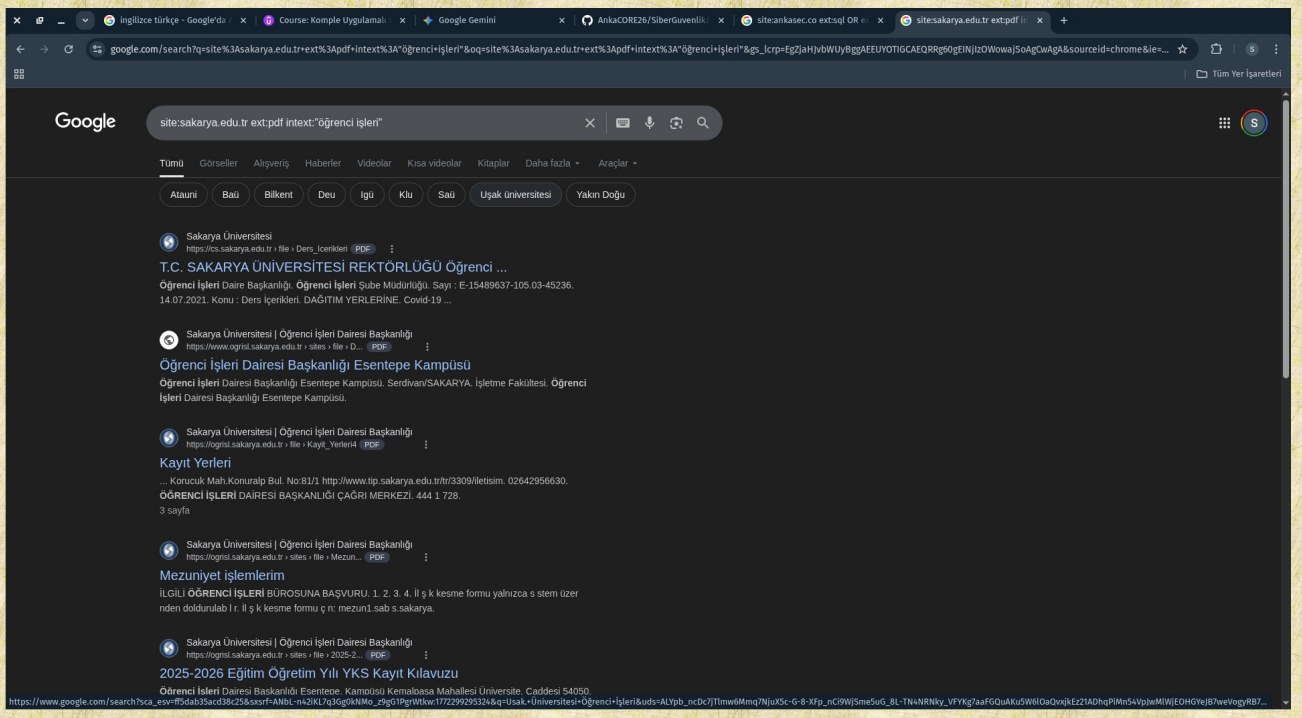
1. Veritabanı ve Config Sızıntısı (Backend Avcısı)

Sorgu: site:ankasec.co ext:sql OR ext:bak OR ext:env OR ext:properties

Bu sorgu , sunucuda unutulmuş veritabanı dump'larını (.sql, .bak) veya içinde veritabanı parolalarının, API key'lerin yazdığı yapılandırma dosyalarını (.env, application.properties) bulur.

2. Yönetim Paneli ve Gizli Giriş Kapıları

Sorgu: site:ankasec.co inurl:admin OR inurl:dashboard intitle:"login"
Bu sorgu , URL'sinde "admin" veya "dashboard" geçen, sekme başlığında ise "login" yazan sayfaları Google'ın hafızasından çekmemizi sağlar. kısacası bizim nereye Brute-force yapacağımızı veya SQL Injection deneyeceğimizi gösteren haritadır.
3. İndekslenmiş Hassas Belgeler ve Loglar
Sorgu: site:ankasec.co ext:pdf OR ext:log intext:"password" OR intext:"gizli"
Bu sorgu , burada çalışanlar tarfından yanlış yerde unutulmuş veya yüklenmiş şifre veya gizli bir dökümanın loglarını veya pdf halini çekmemizi sağlar. Sistemin çalışma mantığını ve iç yapısını çözmek için mükemmel bir istihbarattır.



BÖLÜM C: Operasyonel Görev (Boss Fight)

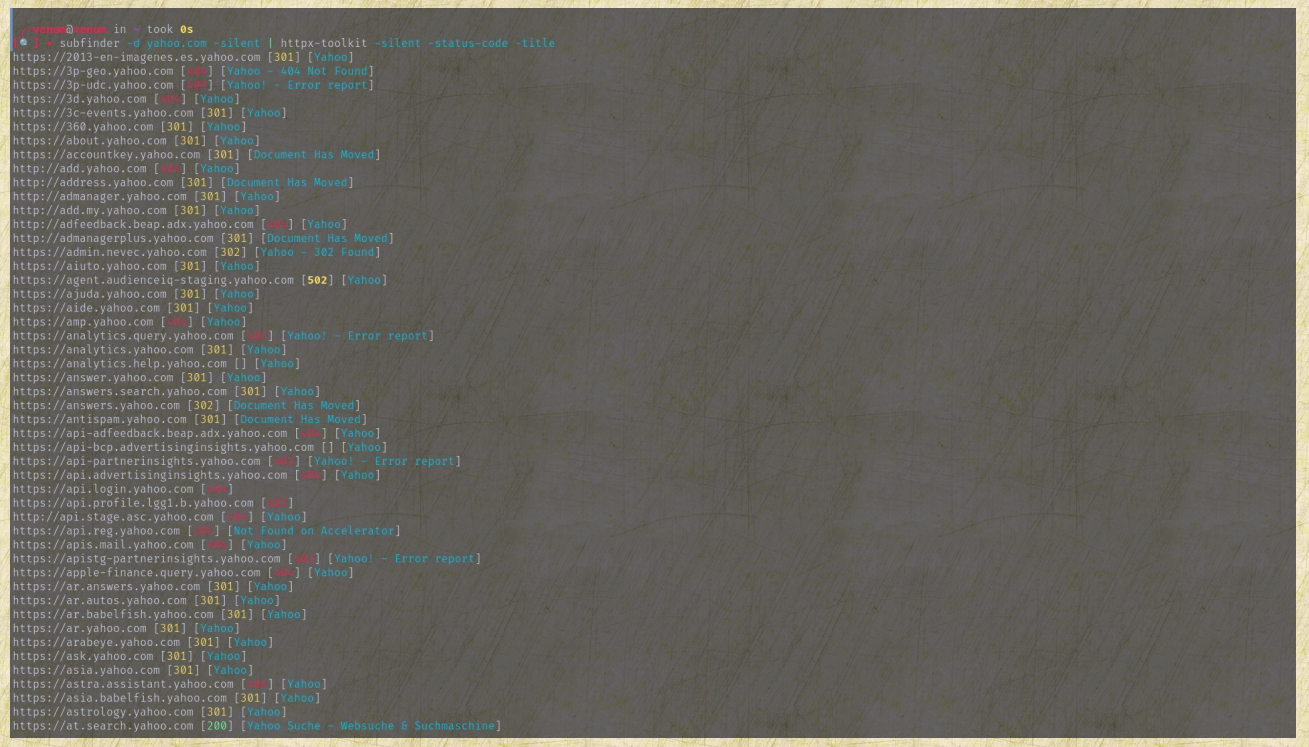
Cephe 1: Hayalet Avı (Recon Haritası)

- Hedef: HackerOne veya Bugcrowd üzerinde "Public" (Herkese açık) ve "Safe Harbor" (Test izni olan) bir program seçin. (Örn: Yahoo, Red Bull, Ford).
- Operasyon:

Terminalinizi açın. subfinder veya amass kullanarak hedef domainin alt alan adlarını (subdomain) tarayın.

Bulduğunuz sonuçları httpx aracına vererek hangilerinin "Canlı" (Live) olduğunu kontrol edin.

- Raporlama: Bulduğunuz en ilginç 3 subdomain'i ve httpx'ten dönen HTTP durum kodlarını raporlayın. (Neden bu 3'ü ilginizi çekti?). Terminal çıktısının ekran görüntüsünü ekleyin.



“ subfinder -d yahoo.com -silent | httpx-toolkit -silent -status-code -title “komut satırını kullanarak bu aramayı yaptım . Peki bu satırı nasıl oluşturdum ?

subfinder : İnternetteki açık kaynakları (OSINT), sertifika veri tabanlarını (crt.sh), GitHub'ı ve arama motorlarını tarayarak Yahoo'ya ait unutulmuş alt alan adlarını (subdomain) bulur.

-d yahoo.com : Hedef kısmımızdır (Domain).

-silent : Normalde üstteki araçlar çalışırken ekrana devasa logolar, taranan kaynaklar ve gereksiz loglar basar. Bu komut araca direkt şunu der : "Bana net cevabı (temiz adres) ver ara kısımları gösterme."

| işareti : Bir nevi köprü görevi görür, desek yeridir. birinci aracın bulduğu o temiz adres listesini havada yakalar ve hiç diske yazmadan doğrudan (Httpx) 'e gitmesini sağlar.

Httpx-toolkit : Arch Linux sisteminde isim çakışmasını önlemek için bu şekilde kullandık. Gelen her adrese ışık hızında HTTP(S) istekleri (ping) yollar.

-silent , yine aynı işlevde kalabalık yapmasını engeller . Sadece ayakta olan sunucuların gösterilmesini sağlar.

-status-code : Sunucunun bize cevabına göre verilen kodları bu kısımda görürüz.(200,403,401)

-title : Sayfanın HTML sekme başlığını (Title) çeker. URL'de test yazmasa bile sayfa başlığında [Admin Dashboard] veya [Staging Environment] yazıyorsa, siteyi tarayıcıda bile açmadan hedefin ne olduğunu saniyesinde anlarız.

1-) https://credstore.yahoo.com [401]

"Credstore" (Credential Store), genellikle sistemlerin kimlik bilgilerini, şifreleri veya API anahtarlarını tuttuğu mahzenlere verilen addır. Böyle bir yapının internete açık olması büyük bir risktir. [401] vermesi sunucunun ayakta olduğunu ve bu sunucuya girme yetkimizin olmadığının söyler . İçeriye girmek için Brute-force veya Auth-Bypass testlerini yapabiliriz.

2-) https://admin.nevec.yahoo.com [302]

İç ağ veya sistem yöneticileri için ayrılmış bir admin panelinin dış ağdan erişilebilir olması her zaman bir yapılandırma zafiyeti potansiyelidir. [302 Found] kodu , sunucunun bizi başka bir adrese yönlendirdiğinin kanıtıdır (Büyük ihtimalle bir SSO- Tekil Oturum Açma Sayfası). Yönlendirme mekanizmalarında Open Redirect veya SSRF açıkları aranarak testler yapılabilir.

3-) https://agent.audienceiq-staging.yahoo.com [502]

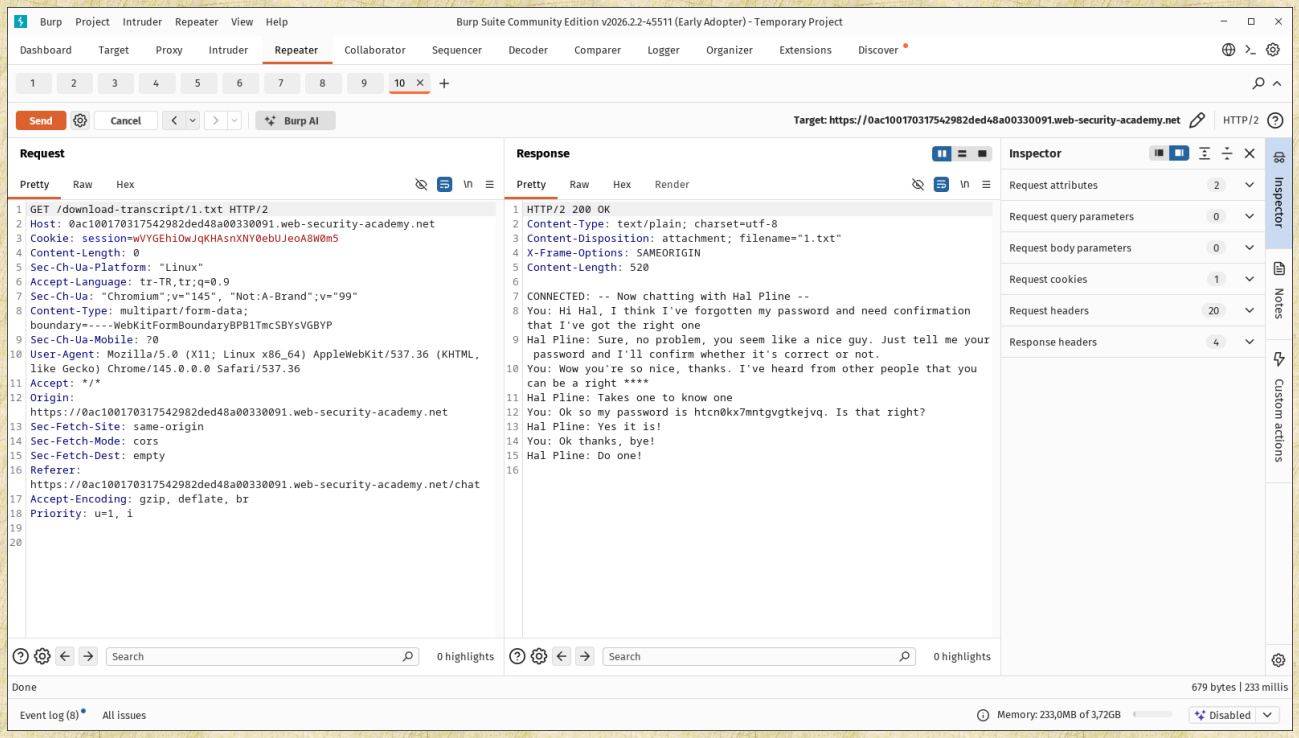
URL' deki " staging" kelimesi buranın yayından olmayan test veya geliştirme ortamı olduğunu gösteriyor. Test ortamları her zaman canlı sistemlerden daha az güvenlidir. [502 Bad Gateway] hatası ise arka plandaki uygulamanın çöktüğünü veya yanlış yapılandırıldığını gösterir. Bu durum, sunucunun hata mesajları (Stack Trace) sızdırmasına veya unutulmuş .env dosyalarını dışarı vermesine sebep olabilir. Bu yüzden bu URL de diğerleri gibi çok önem arz ediyor.

Cephe 2: Yetkiyi Delmek (PortSwigger Labs)

Web güvenliğinin kutsal mekanı PortSwigger'da gerçek IDOR senaryoları çözeceğiz.

- Platform: PortSwigger Web Security Academy (Kayıt olun, ücretsizdir).
- Görev: Aşağıdaki iki IDOR/BAC laboratuvarını çözün:
- Lab: Insecure direct object references
- Lab: User ID controlled by request parameter
- Raporlama Formatı: "Çözdüm" demek yok. Burp Suite kullanarak trafiği nasıl yakaladığınızı (Proxy), isteği Repeater'a nasıl attığınızı ve parametreyi (id veya user) değiştirerek başkasının verisine nasıl ulaştığınızı adım adım anlatın.
- Kanıt: Burp Suite Repeater ekranında başarılı response (Örn: Başkasının API anahtarını veya şifresini gördüğünüz an) açıkça görünmelidir.

- **Lab: Insecure direct object references**

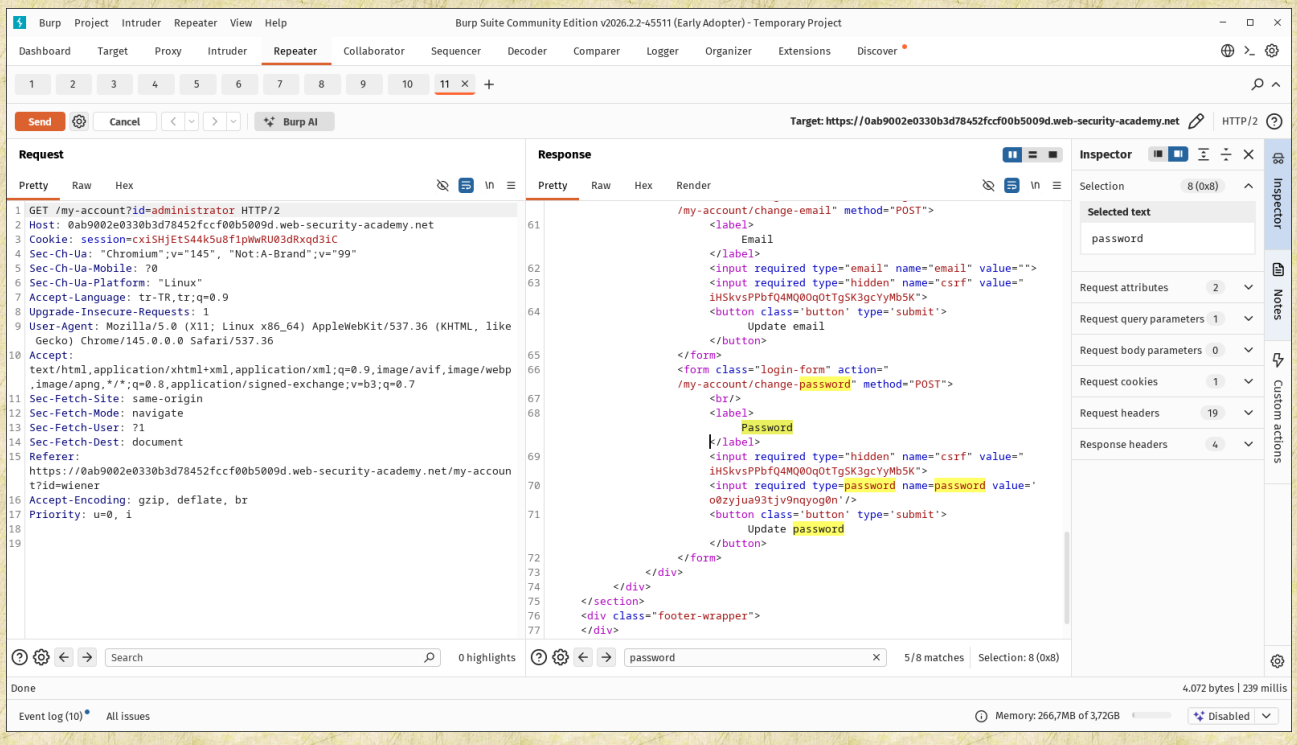


Burp Suite'in kendi tarayıcısından hedefe girdim ve "Live chat" sayfasını açtım. Amacım sohbeti indirme butonuna ("View transcript") basıp aradaki trafiği yakalamaktı.

- Burp üzerinden "Intercept is on" yaparak ağı gerdim ve butona bastım. Fakat burada ufak bir hata oldu; sohbeti indiren GET isteği yerine, tarayıcının mesajı gönderdiği o ilk POST isteğini havada yakaladım.
- Geri dönüp baştan başlamak yerine, yakaladığım paketi `Ctrl + R` ile Repeater'a attım ve isteği kendi ellerimle manipüle ettim. En üstteki satırı tamamen silip yerine `GET /download-transcript/1.txt HTTP/2` yazdım. Amacım sistemdeki 1 numaralı (ilk) kullanıcının sohbet dosyasına yetkisiz erişmekti. Alt kısımdaki gereksiz veri yüklerini (Body) de sildim.
- "Send" butonuna bastığımda, sistemde nesne seviyesinde yetki kontrolü (IDOR açığı) olmadığı için sunucu kim olduğumu sorgulamadı ve bana doğrudan `200 OK` yanıtını döndürdü.

- Sağ taraftaki yanıt (Response) ekranında Carlos adlı kullanıcının sohbet geçmişi ve şifresi (`htcn0kx7mntgvgtkejvq`) açıkça ifşa oldu. Bu şifreyle "My account" kısmından sisteme sızıp laboratuvarı çözdüm.

- **Lab: User ID controlled by request parameter**



Burp tarayıcısından labı açıp bize verilen `wiener` (şifre: `peter`) hesabı ile giriş yaptım ve "My account" sayfasına gittim. *(Neden: Amacım sistemin normal bir kullanıcıya profil bilgilerini getirirken arka planda nasıl bir yol izlediğini görmekti.)*

- Burp Suite'e geçip "Intercept is on" diyerek ağı gerdim ve tarayıcıdan "My account" sayfasına tekrar tıkladım. Giden isteği havada yakaladım. *(Neden: Tarayıcının benim bilgilerimi getirmek için sunucuya hangi parametreleri gönderdiğini araya girip görmek istiyordum.)*

- Yakaladığım paketin en üst satırında `GET /my-account?id=wiener` yazdığını gördüm. Hemen `Ctrl + R` ile bu isteği Repeater'a yollayıp Intercept'i (yakalamayı) kapattım. *(Neden: Sistemin kim olduğumu güvenli bir oturumdan (Session) değil, doğrudan URL'deki `id=wiener` yazısından anladığını fark ettim. Repeater'a attım ki bu parametreyi manipüle edip sistemi kandırabileyim.)*
- Repeater ekranında sol taraftaki URL'de yazan `id=wiener` kısmını silip `id=administrator` yazdım ve "Send" diyerek sunucuya yolladım. *(Neden: Sistemde "Sıfır Güven" (Zero Trust) veya yetki kontrolü olup olmadığını test ediyordum. Sunucuya "Ben wiener değilim, bana administrator'un profilini getir" emrini vererek IDOR zafiyetini tetikledim.)*
- Sağ tarafta `200 OK` yanıtı döndü. Dönen HTML kodlarının içinde arama yaparak `administrator` kullanıcısının şifresini buldum. Şifre, sayfadaki gizli bir `value="o0zyjua93tjv9nqyog0n"` etiketinin içine açıkça yazılmıştı. *(Neden: Yazılımcı, arka planda "Bu sayfayı isteyen kişi gerçekten admin mi?" diye sormadığı için sistem bana adminin tüm sayfasını ve şifresini ifşa etti.)*
- Bulduğum bu şifreyle normal giriş ekranından `administrator` hesabına girdim. Üstte açılan "Admin panel" linkine tıklayıp Carlos kullanıcısını sildim ve laboratuvarı tamamen çözdüm.

BÖLÜM D: Mühendislik Vizyonu (Reflection)

- 1. IDOR'un Gerçek Hayat Etkisi (Impact): Senaryo: Bir hastanenin e-randevu sisteminde test yapıyorsunuz. Kendi tahlil sonucunuza bakarken URL'nin hastane.com/tahlil?id=500 olduğunu gördünüz.
- Soru: id=501 yaptığınızda başkasının HIV veya Kanser test sonucunu görebiliyorsanız, bu zafiyetin KVKK (veya GDPR) açısından şirkete maliyeti ve marka itibarına vereceği zarar nedir? Bir mühendis olarak bu açığı şirkete raporlarken "Risk Seviyesini (Impact)" nasıl açıklarsınız?

1. IDOR'un Gerçek Hayat Etkisi (Impact): Senaryo: Bir hastanenin e-randevu sisteminde test yapıyorsunuz. Kendi tahlil sonucunuza bakarken URL'nin hastane.com/tahlil?id=500 olduğunu gördünüz.

Risk Seviyesi (Severity): KRİTİK (Critical - CVSS Puanı: 9.0+)

Saldırı Karmaşıklığı (Attack Complexity): Düşük (Low). Zafiyeti sömürmek için ileri düzey bir hack bilgisine gerek yoktur; sadece URL'deki bir rakamı değiştirmek yeterlidir.

Gizlilik İhlali (Confidentiality Impact): Yüksek (High). Veri gizliliği tamamen ortadan kalkmıştır.

Sistemde 'Nesne Seviyesinde Yetkilendirme (Object-Level Authorization)' kontrolleri tamamen eksiktir. Kötü niyetli bir saldırgan, Burp Suite Intruder gibi basit bir otomasyon aracı kullanarak id parametresini 1'den 100.000'e kadar saniyeler içinde tarayabilir (Enumerate). Bu durum, sistemdeki tüm hastaların 'Korumalı Sağlık Bilgilerinin (PHI)' dışarıya

toplu olarak sızdırılmasına (Data Breach) olanak tanımaktadır. Zafiyetin derhal kapatılması (Hotfix) ve veri tabanı erişim loglarının geçmişe dönük incelenmesi hayati önem taşımaktadır.

2. Güvenin Bedeli (Zero Trust):

- **Soru:** IDOR ve Broken Access Control zafiyetlerinin temelinde yatan yazılımcı hatası, "Kullanıcıdan gelen girdiye (Input) güvenmek"tir. Bir yazılımcı bu hatayı yapmamak için arka planda (Backend) nasıl bir kontrol (Check) mekanizması kurmalıdır?

Bir yazılımcı bu hataya düşmemesi için 4 aşamadan oluşan kontrol mekanizması kurmalıdır.

1.Aşama : İstemci (Client) Her Zaman Yalancıdır : Yani tarayıcıdan, URL'den veya API isteğinden (JSON Body) gelen id=501 değeri asla bir "emir" olarak kabul edilemez. O sadece bir "talep"tir. Backend, bu talebi körü körüne veritabanına sokup işlem yapmamalıdır. 2. Aşama : Kimliği Asla URL'den Alma (Session/Token Trust) : Yani Sisteme giren kişinin kim olduğunu asla dışarıdan gelen bir parametreye (örn: ?user_id=10) bakarak anlamamalı. Backend, o anki aktif kullanıcıyı sadece kendi güvendiği sunucu hafızasından (Session) veya dijital olarak imzalanmış, değiştirilemez bir kaynaktan (JWT - JSON Web Token) çekmelidir.

3. Aşama : Nesne Seviyesinde Yetki Kontrolü (BOLA/Object-Level Authorization) : Yani sorgu her zaman çift şartlı olmalıdır. SELECT * FROM tahliller WHERE tahlil_id = 501 AND hasta_id = (Token'dan_Gelen_Aktif_Kullanici_ID) gibi saldırgan tahlil_id değerini farklı yapsa da hasta_id değerini bulma olasılığı daha düşük olduğundan çift şartlı sorgu her zaman daha güvenlidir.

4. Aşama : Tahmin Edilemez Referanslar (Defense in Depth - UUID) : Yani Sıralı tam sayılarla işlemler yapmıycaz bunun yerine veri tabanında 550e8400-e29b-41d4-a716-446655440000 gibi karmaşık GUID/UUID

yapıları kullanmalıyız. Bu IDOR'u tek başına engellemez (yetki kontrolü yine şarttır) ama saldırganın diğer nesnelerin ID'sini tahmin edip otomatik tarama (Brute-Force) yapmasını imkansız hale getirir.