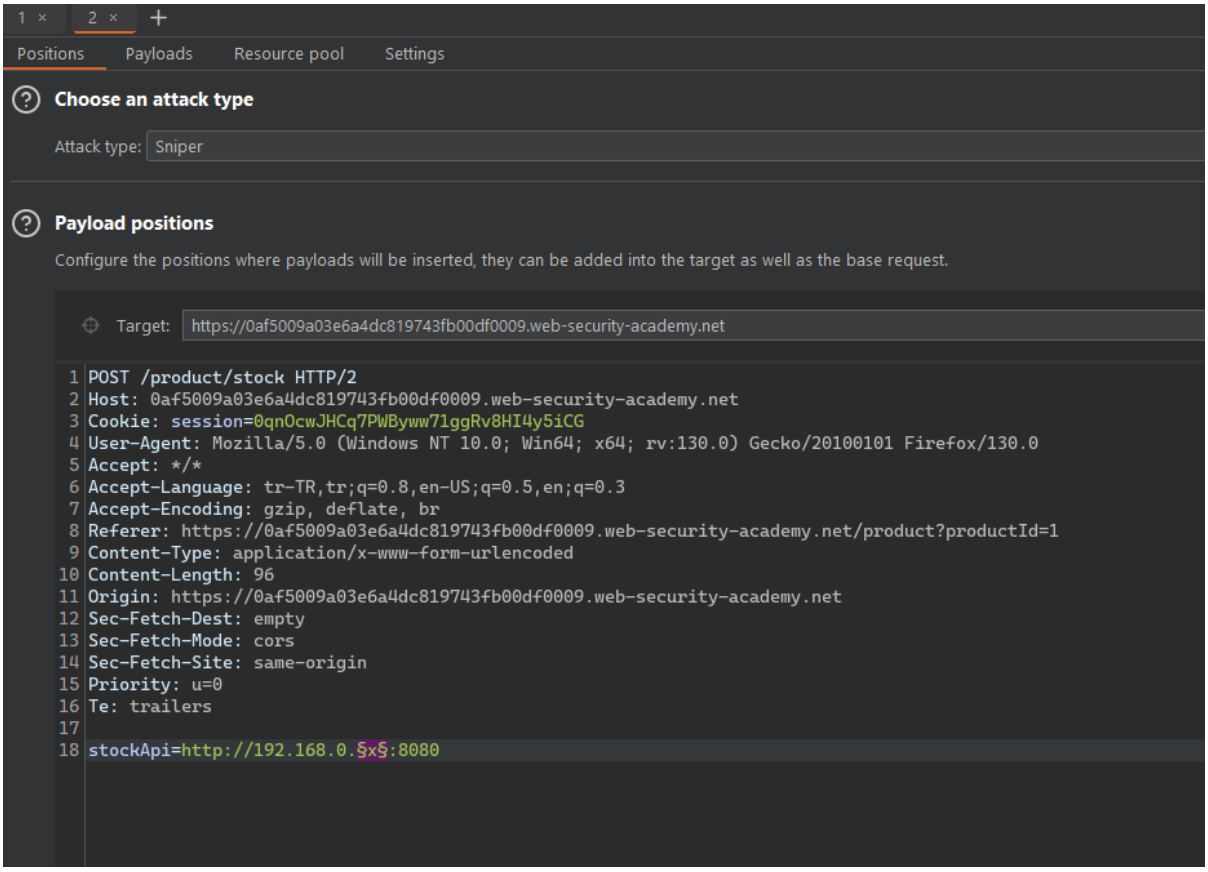
- Bu write-up'taki tüm lablar portswigger academy web sitesinden
- owasp10'den seçtiğim açılar -> **SSRF** && **Injection kategorisinden SSTI** && **Access Control Vulnerabilities**

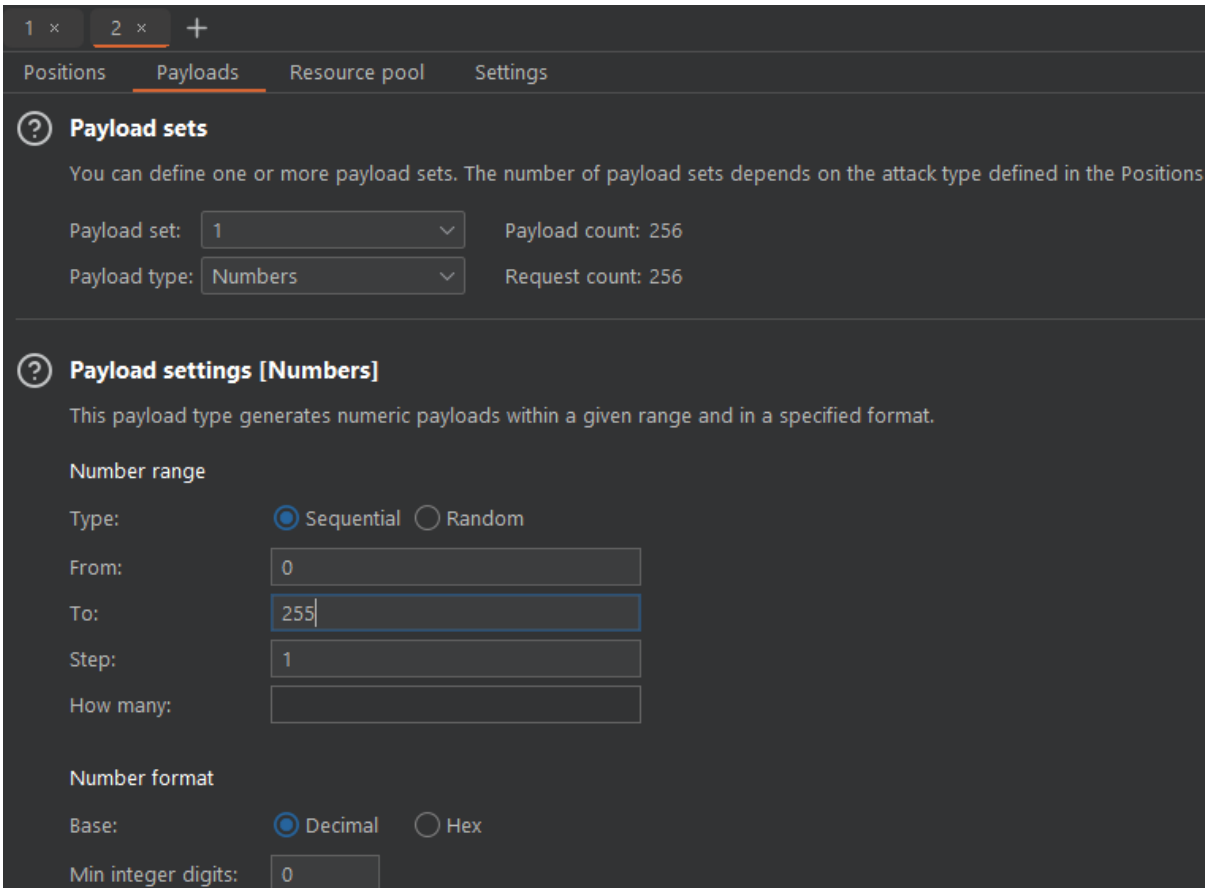
SSRF

First Lab Title -> Lab: Basic SSRF against another back-end system

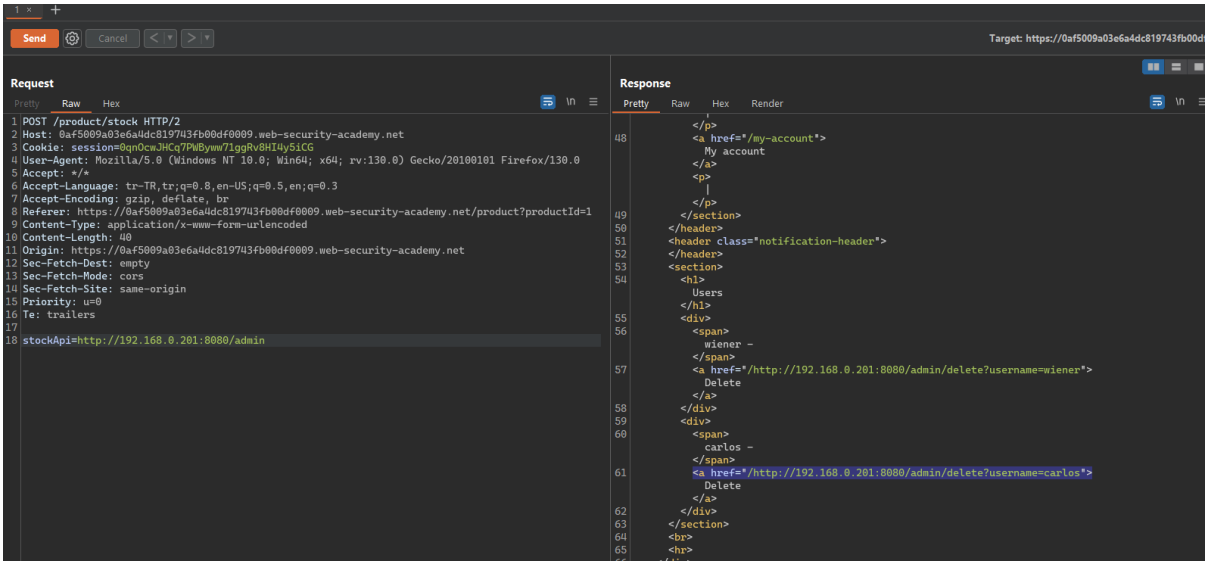
- **Lab Açıklaması:**
  - internal bir sistemden data çeken bir stock check fonksiyonu var. Lab'ı çözmek için stock check fonksiyonunu kullanarak 192.168.0.x internal ip range'ini tarayıp 8080 port'unda çalışan admin interface'ine istek yaparak, carlos adlı kullanıcıyı silmemiz gerekiyor
- **Lab Çözümü:**
  1. Home page'de herhangi bir ürünün altındaki 'View Details' butonuna tıklayıp, açılan sayfanın alt tarafında ki 'Check Stock' butonuna tıklayarak request'i intercept ediyoruz
  2. Request'i intruder'a gönderdikten sonra, request body'sindeki 'stockApi' parametresinin value'sunu http://192.168.0.x:8080 olarak değiştirip, x harfini 'Add' diyerek işaretliyoruz



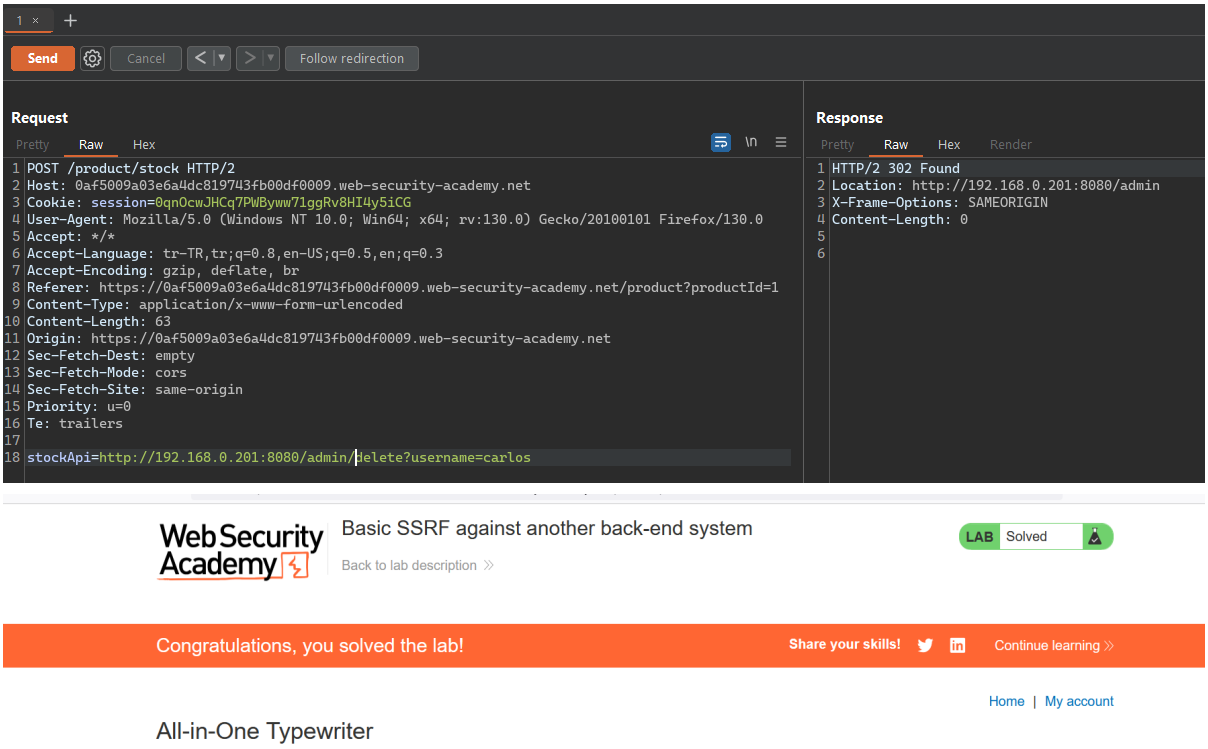
  3. Sonrasında, üstte bulunan 'Payloads' seçeneğini seçiyoruz ve 'Payload Type' kısmından 'Numbers' seçeneğini seçiyoruz. Allta açılan kısımda 'From' için 0 ve 'To' için 255 değerlerini giriyoruz.



4. Sonrasında 'Start attack' diyerek brute-force'u başlatıyoruz.
5. Sonucu incelediğimizde 1 ve 201 sayıları dışında 500 status code aldığımızı gözlemliyoruz. 201 numarasında gördüğümüz 404 bize diğerlerinden farklı olarak bu ip'de bir uygulama olabileceğini gösteriyor
6. Gerçek bir senaryoda muhtemelen direrctory brute-force yaparak olası tüm directory'leri tespit ederdik. Bu labın hedefi admin interface'sine ulaşıp, carlos'u silmek olduğu için, bu requesti repeater'a gönderip basitçe /admin directory'sine istek yapıyoruz ve dönen response'ta carlos user'ını silmemiz için request yapmamız gereken endpoint'i görüyoruz

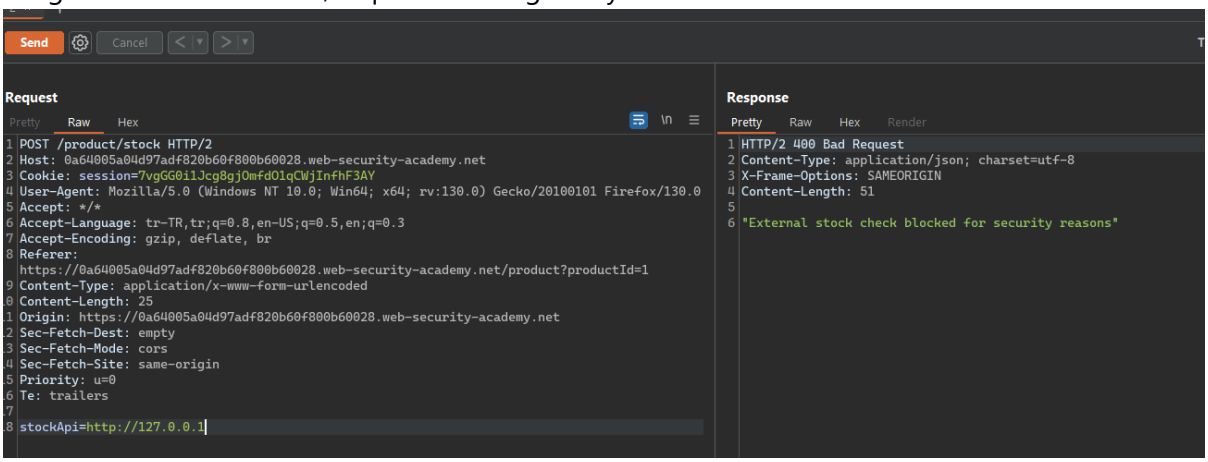


7. Son olarak bu endpointe istek yapıyoruz ve lab çözülmüş oluyor



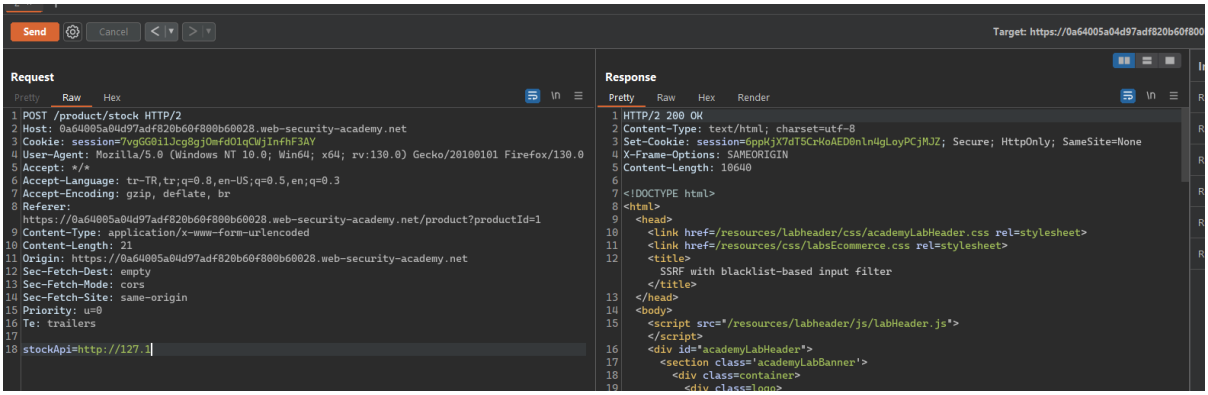
Second Lab Title -> Lab: SSRF with blacklist-based input filter

- **Lab Açıklaması:**
  - internal bir sistemden data çeken bir stock check fonksiyonu var. Lab'ı çözmek için 'http://localhost/admin' internal endpoint'ine ulaşıp sonrasında carlos adlı kullanıcıyı silmemiz gerekiyor. Developer 2 tane zayıf anti-SSRF defence konuşlandırmış.
- **Lab Çözümü:**
  1. Home page'de herhangi bir ürünün altındaki 'View Details' butonuna tıklayıp, açılan sayfanın alt tarafında ki 'Check Stock' butonuna tıklayarak request'i intercept ediyoruz
  2. Request'i repeater'a gönderip, request body'sindeki 'stockApi' parametresinin değerine 'http://127.0.0.1' yapıp, request'i gönderiyoruz. Gelen response'ı incelediğimizde, developer'ın aldığı önlemlemden ötürü, request'imiz engelleniyor

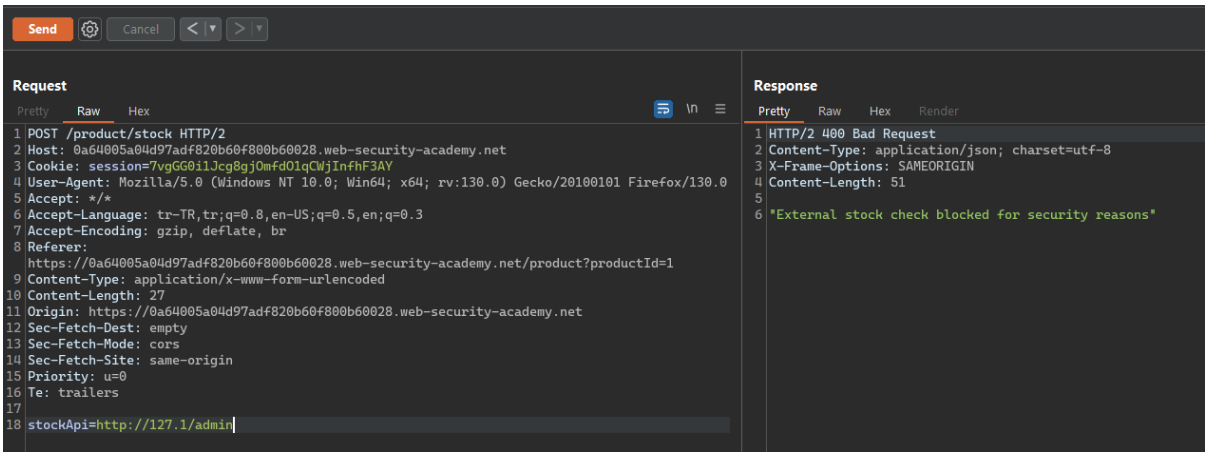


3. Bu request'i 'http://localhost' için yaptığımızda da aynı sonucu alıyoruz. Bunun ardından 127.0.0.1 ip'sini, aynı işlevi gören 127.1 ip'si ile değiştirip isteği gönderiyoruz ve developer'ın ilk önlemini

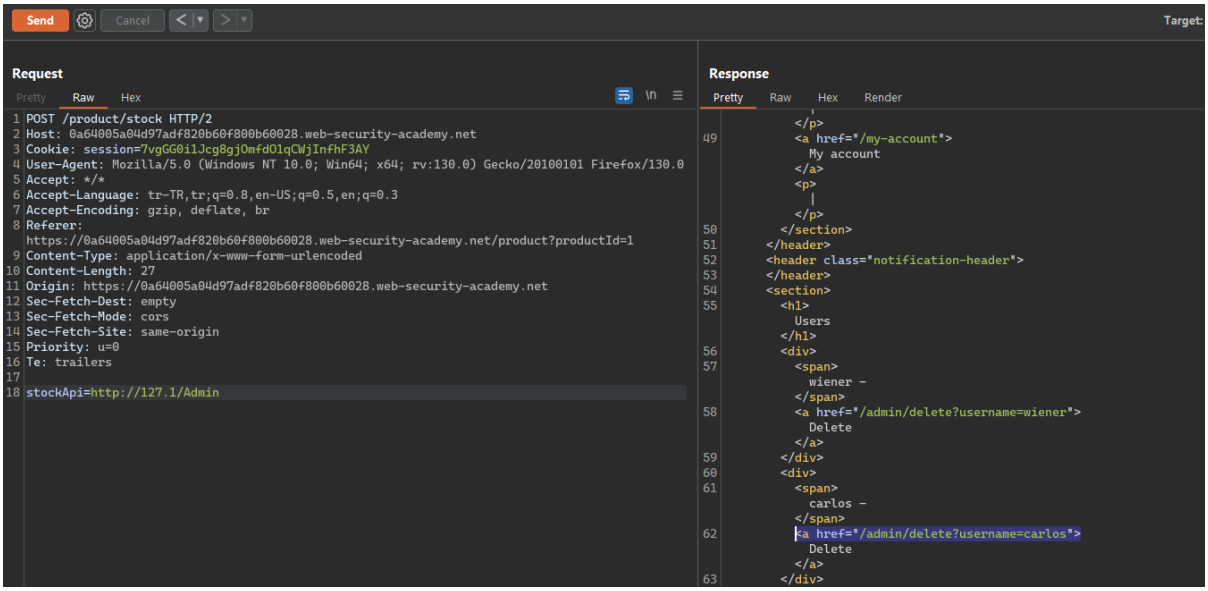
aşıyoruz



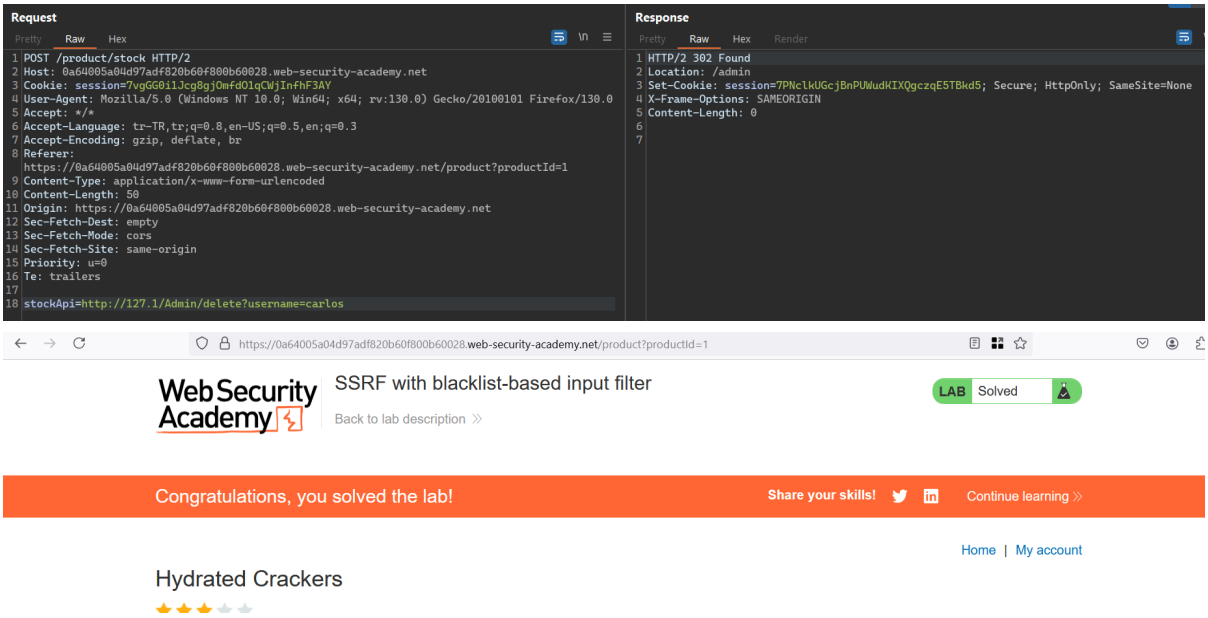
4. Sonrasında /admin directory'si için istek yaptığımızda ise, developer'ın ikinci önlemi olan admin word'ünün yasaklı olduğunu öğreniyoruz.



5. Bu önlemi aşmak için admin word'ünün harflerinden bir yada birden fazlasını büyük yazıyoruz ve developer gelen word'ün kontrolünü yapmadan önce küçük yada büyük harf'e çevirmediği için bu önlemide basitçe aşmış oluyoruz

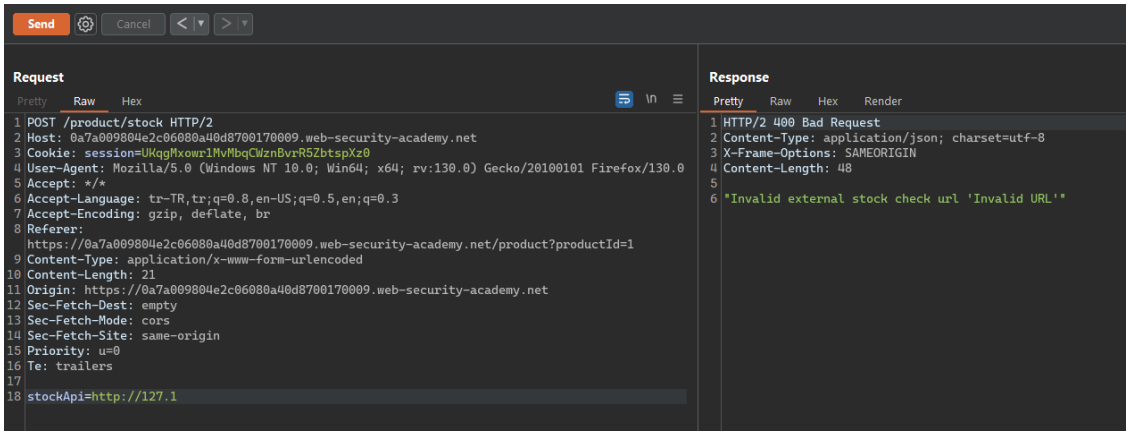


6. Son olarakta, carlos'u silmek için, gerekli endpoint'e istek yapıp lab'ı çözüyoruz.

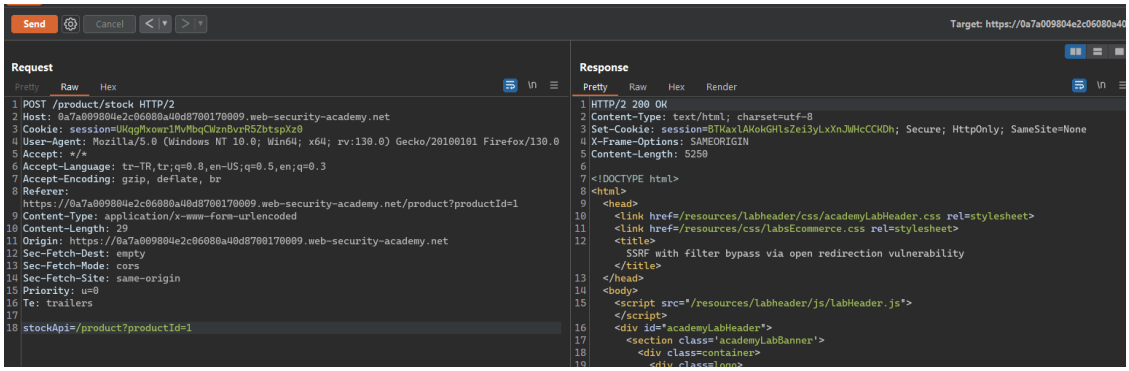


Third Lab Title -> SSRF with filter bypass via open redirection vulnerability

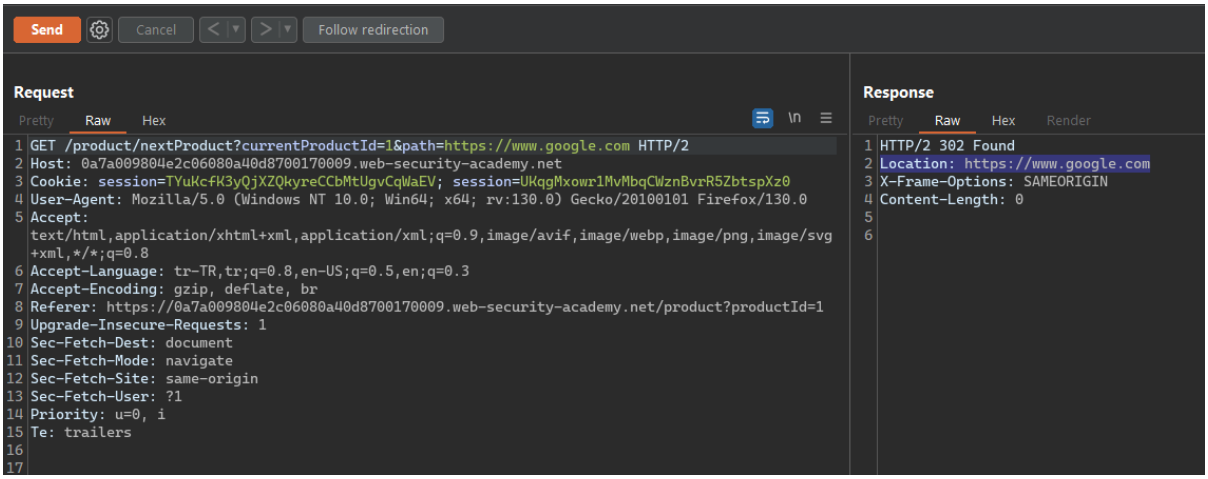
- **Lab Açıklaması:**
  - internal bir sistemden data çeken bir stock check fonksiyonu var. Lab'ı çözmek için 'http://192.168.0.12:8080/admin' url'ine istek yaparak admin interface'ine istek yapmamız ve sonrasında carlos user'ını silmemiz gerekiyor. Developer önlem olarak url'i kısıtlayıp sadece bulunduğumuz web uygulamasındaki endpoint'lere istek yapabilmemize sebep olmuş ve bu önlemi aşmak için bulunduğumuz web uygulamasında open redirect bulup bu önlemi aşmamız bekleniliyor.
- **Lab Çözümü:**
  1. Home page'de herhangi bir ürünün altındaki 'View Details' butonuna tıklayıp, açılan sayfanın alt tarafında ki 'Check Stock' butonuna tıklayarak request'i intercept ediyoruz
  2. Request'i repeater'a gönderip, request body'sindeki 'stockApi' parametresinin değerine herhangi bir şekilde http:// yada https:// gibi url scheme yazıp istek yaptığımızda 'Invalid Url' Hatası dönüyor ve sadece bulunduğumuz uygulamadaki endpoint'lere istek yapabildiğimizi görüyoruz.
     - Başarısız request (http:// yada https:// kullandığımızda)



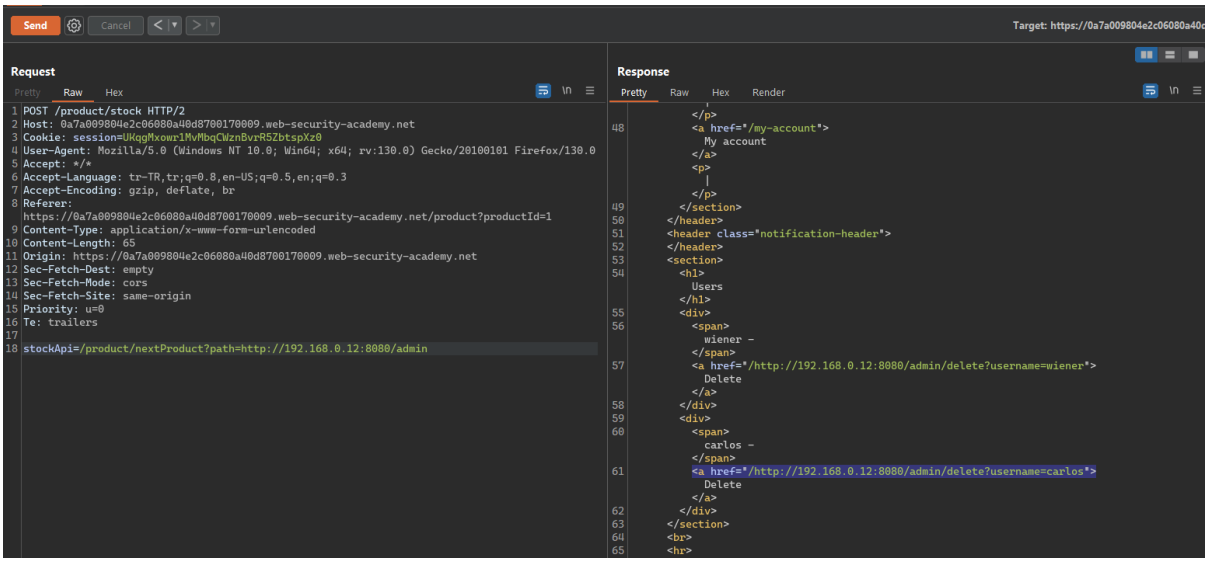
- Başarılı Request (bulunduğumuz uygulamadaki herhangi bir endpoint'e istek yaptığımızda)



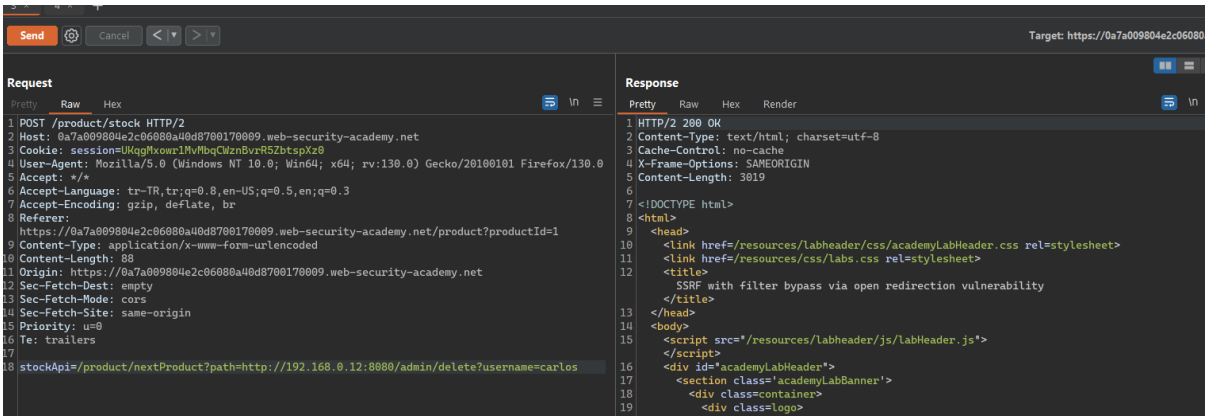
3. Bu sorunu aşmak için open redirect zafiyetine ihtiyacımız var ve uygulamayı biraz gezdikten sonra bu zafiyeti herhangi bir product'ı seçtiğimizde açılan product'ın sayfasının en altında bulunan 'Next Product' seçeneğine tıkladığımızda buluyoruz
4. Bu seçeneği seçtiğimizde yapılan request'i intercept edip repeater'a gönderiyoruz. Bu request'teki path parametresini herhangi bir url ile değiştirip istek yaptığımızda herhangi bir güvenlik önlemine takılmadan, yazdığımız url'e yönlendiriliyoruz.



5. Open redirect'i keşfettikten sonra yapmamız gereken şey; öncelikle path parametre değerini 'http://192.168.0.12:8080/admin' ile değiştirip daha sonrasında ssrf olan request bodysine, open redirect açığı olan bu endpoint'i yazmak. Beklediğimiz davranış şu: ssrf request'i öncelikle open redirect olan endpoint'e istek yapacak, sonrasında open redirect zaafiyetinden dolayı 'http://192.168.0.12:8080/admin' endpoint'ine yönlendirilecek ve günün sonunda başarılı bir şekilde admin interface'ine istek yaptırtmış olacağız.
6. Bunu için ssrf request body'sine şu endpoint'i yazıyoruz -> '/product/nextProduct?path=http://192.168.0.12:8080/admin' ve request'i gönderiyoruz



7. Başarılı şekilde admin interface'ine isteğimizi yaptık ve şimdi carlos adlı kullanıcıyı sileceğiz ve lab çözülmüş olacak

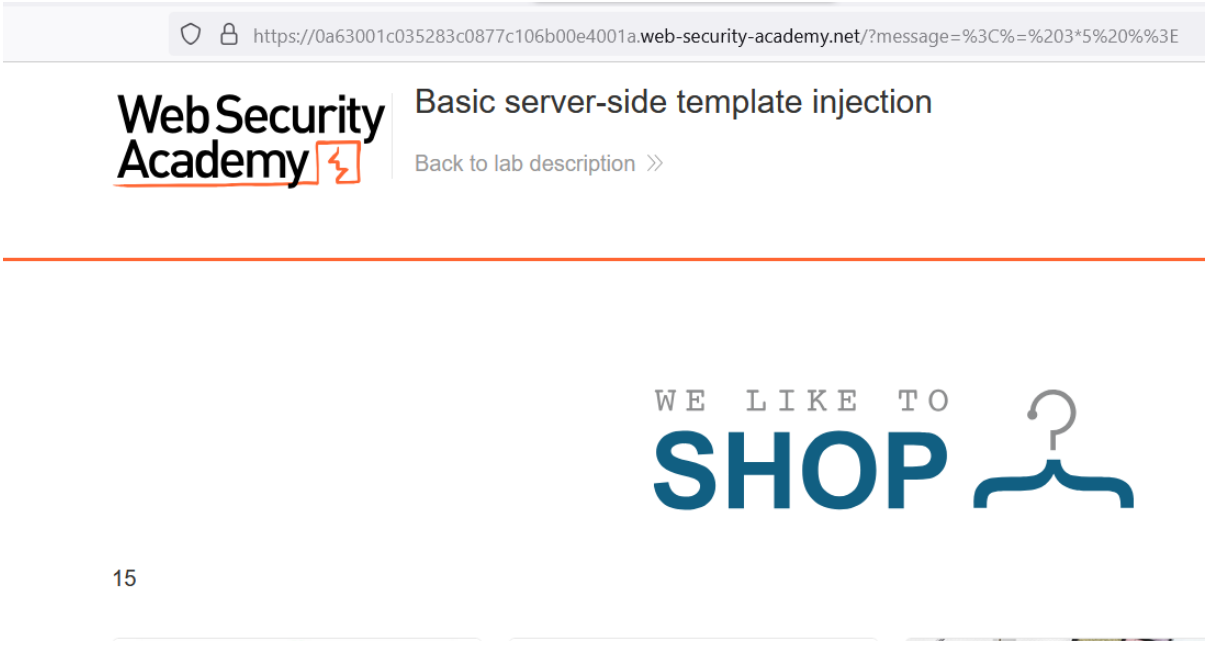


Injection > SSTI

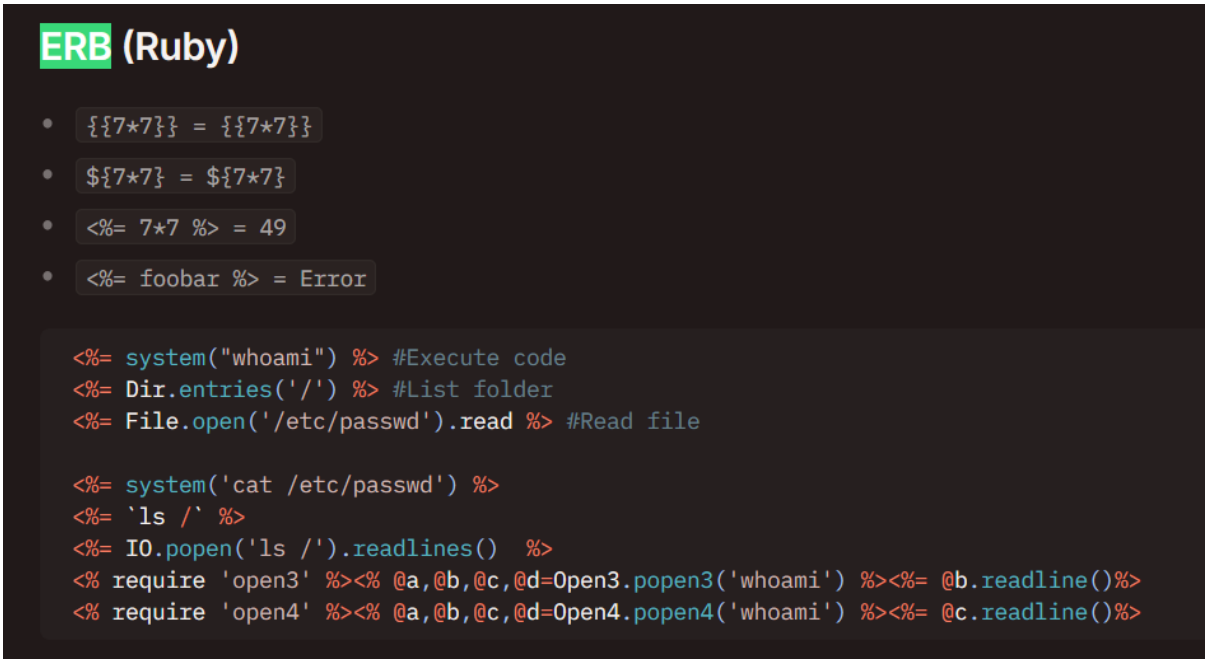
First Lab Title -> Lab: Basic server-side template injection

- **Lab Açıklaması:**
  - ERB template'inin unsafe yapılandırılmasından ötürü ssti açığı var. Lab'ı çözmek için ERB dokümentasyonunu inceleyip, morale.txt adlı file'ı RCE ile silmemiz gerekiyor.
- **Lab Çözümü:**
  1. Home page'deki ilk product'a tıkladığımızda url'de message adlı bir parametre beliriyor ve bu parametrenin değeri ekrana yazılıyor.
  2. Tespit aşamasında {{5 * 3}} {% 3 * 5 %} gibi farklı ssti tespit yöntemleri denedikten sonra, <%= 3 * 5 %> payload'ı işe yarıyor ve ssti'ı bu noktada tespit ediyoruz. (Ekranda 15 yazması ssti olduğunu

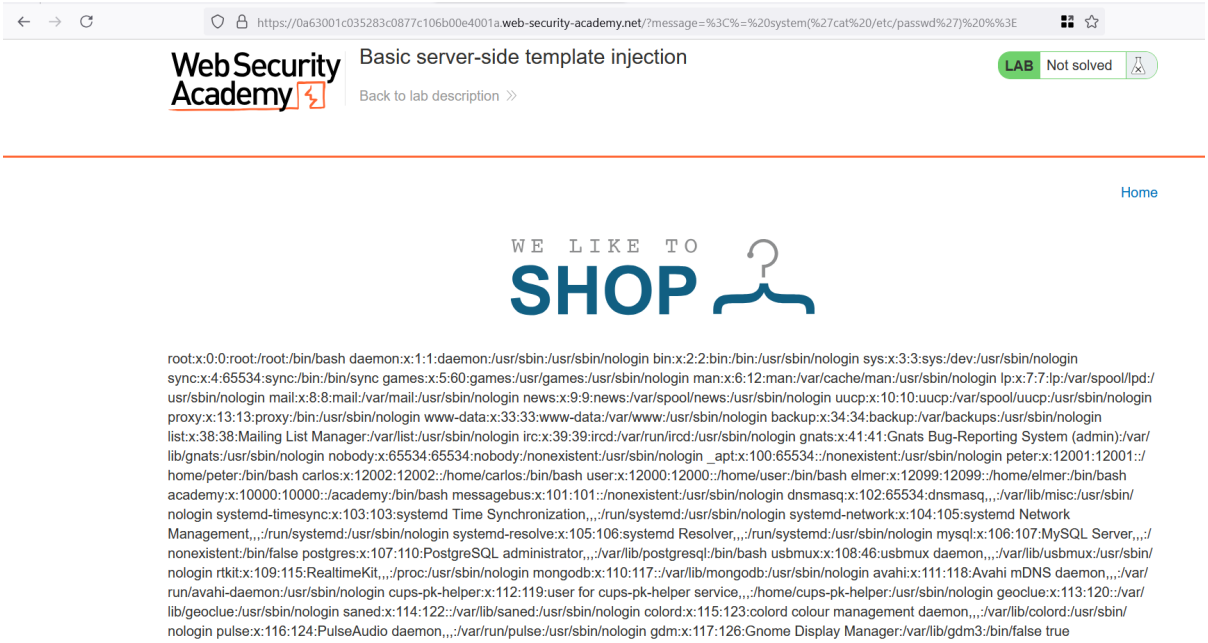
gösteren detay)



3. Sonrasında internetten ERB ssti yazıp, RCE için payload'ları deniyoruz



4. <%= system('cat /etc/passwd') %> payload'ı işe yarıyor

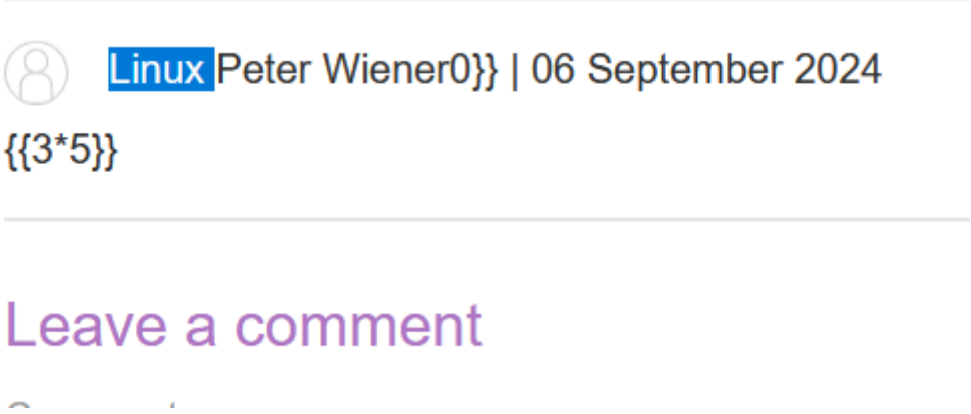


5. Lab'ı çözmek için <%= system('rm morale.txt') %> payload'ını kullanıyoruz ve lab'ı çözüyoruz

Second Lab Title -> Lab: Basic server-side template injection (code context)

- **Lab Açıklaması:**
  - Tornado template'inin unsafe kullanımından dolayı ssti açığı var. Lab'ı çözmek için tornado template'inin dokümentasyonunu inceleyip morale.txt adlı file'ı silmemiz gerekiyor. Credential -> wiener:peter
- **Lab Çözümü:**
  1. öncelikle wiener account'u ile login oluyoruz. Sonrasında /my-account directory'sindeki preferred name form'unu submit edip request'i intercept ediyoruz.
  2. Bu request'i repeater'a gönderdikten sonra, 'blog-post-author-display=' parametresinin değerini -> {{3*5}} yapıyoruz. Daha sonra home page'deki bir blog'a girip comment paylaşıyoruz.
  3. Blog'a yeniden girip yorumumuzun üstündeki isme bakınca, ssti açığını tespit ediyoruz. Sonrasında internete tornado ssti yazıp RCE'ye sebep olan payload'lara bakıyoruz.

4. Sonrasında {% import os %}{{os.system('uname')}} payload'ını deniyoruz. blog sayfasını ziyaretlerimizde bir ton hata aldıktan sonra, sonunda ilk baştaki parametre değerini 2 tane süslü parantez ile kapatıp, sonrasında RCE payloadımızı yazıyoruz. Parametrenin son değeri -> blog-post-author-display=user.name}}{%+import+os+%}{{os.system("uname")}} ve blog sayfasını ziyaret ettiğimizde, uname olan Linux'ü görüntülüyoruz.

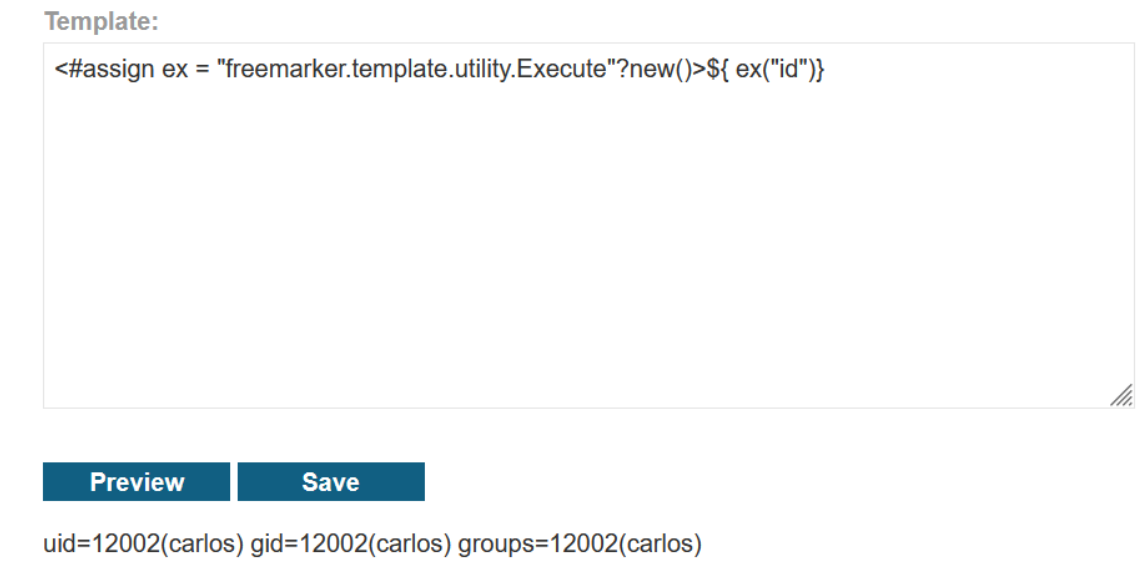


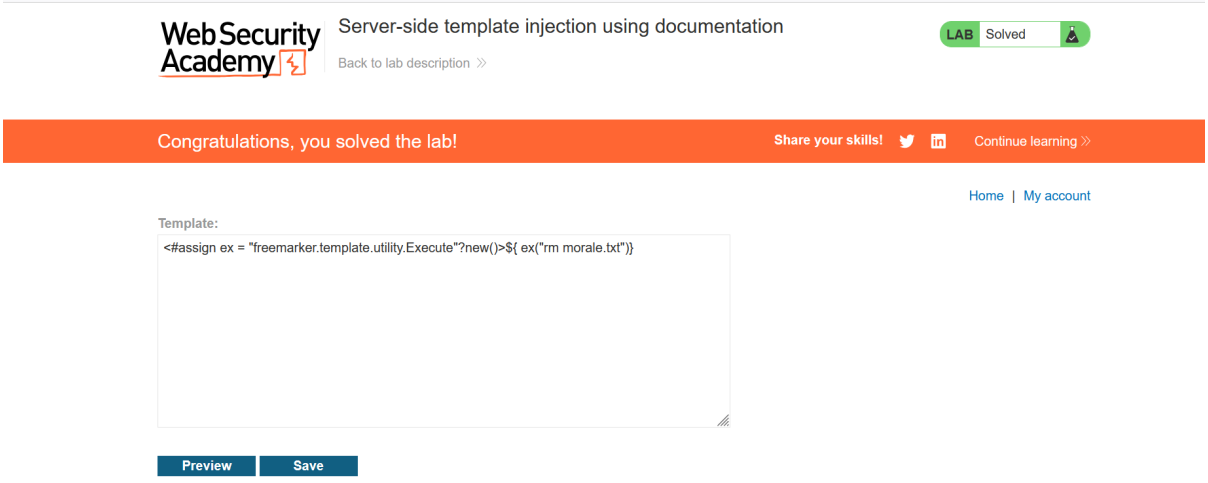
5. Son olarak morale.txt adlı file'ı silip lab'ı tamamlıyoruz. (Son payload -> blog-post-author-display=user.name}}{%+import+os+%}{{os.system("rm+morale.txt")}})

Third Lab Title -> Lab: Server-side template injection using documentation

- **Lab Açıklaması:**
  - Bu lab ssti açığı içeriyor. Lab'ı çözmek için, template engine'ini tespit edip documentation kullanarak morale.txt'yi silen RCE payloadını kullanmamız gerekiyor. Credential -> content-manager:C0nt3ntM4n4g3r
- **Lab Çözümü:**
  1. Öncelikle verilen credential'lar ile account'umuza kayıt oluyoruz. Sonrasında home page'inden herhangi bir blog post'una girip 'Edit Template' diyoruz.
  2. SSTI tespiti için payload'ları denedikten sonra, #{3*5} payload'ı işe yarıyor. Bundan sonra internete ssti yazıp, hactricks sitesindeki template engine'leri inceleyip bazı payloadları denedikten sonra, template engine'in FreeMarker olduğunu tespit ediyoruz.

  3. Bundan sonra, <#assign ex = "freemarker.template.utility.Execute"?new()>${ ex("id")} payload'ı ile RCE testini başarıyla yapıp, morale.txt file'ını siliyoruz ve lab çözülmüş oluyor.

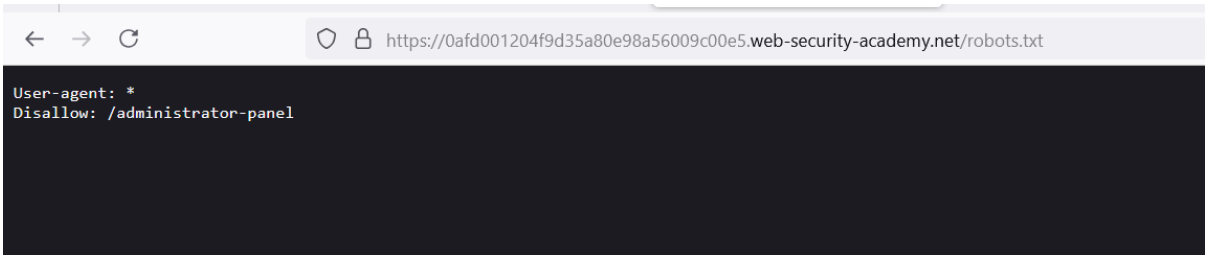




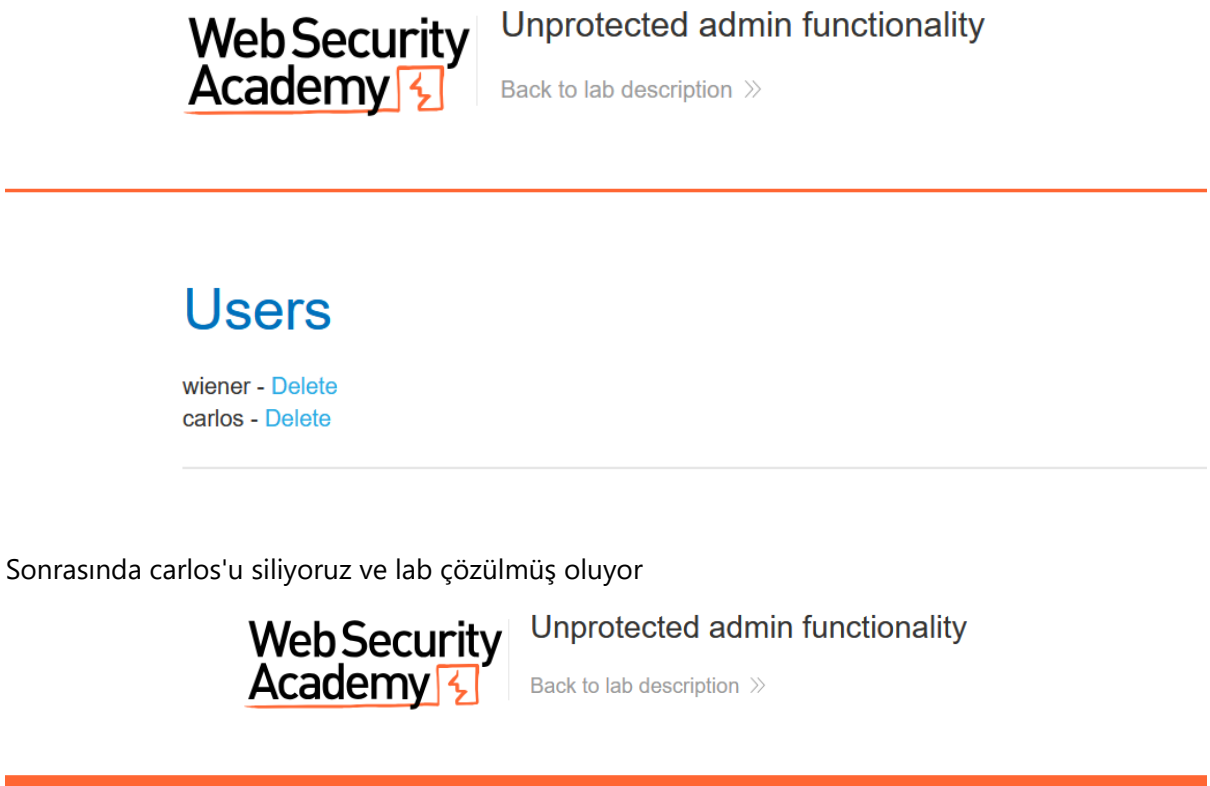
Access Control

First Lab Title -> Lab: Unprotected admin functionality

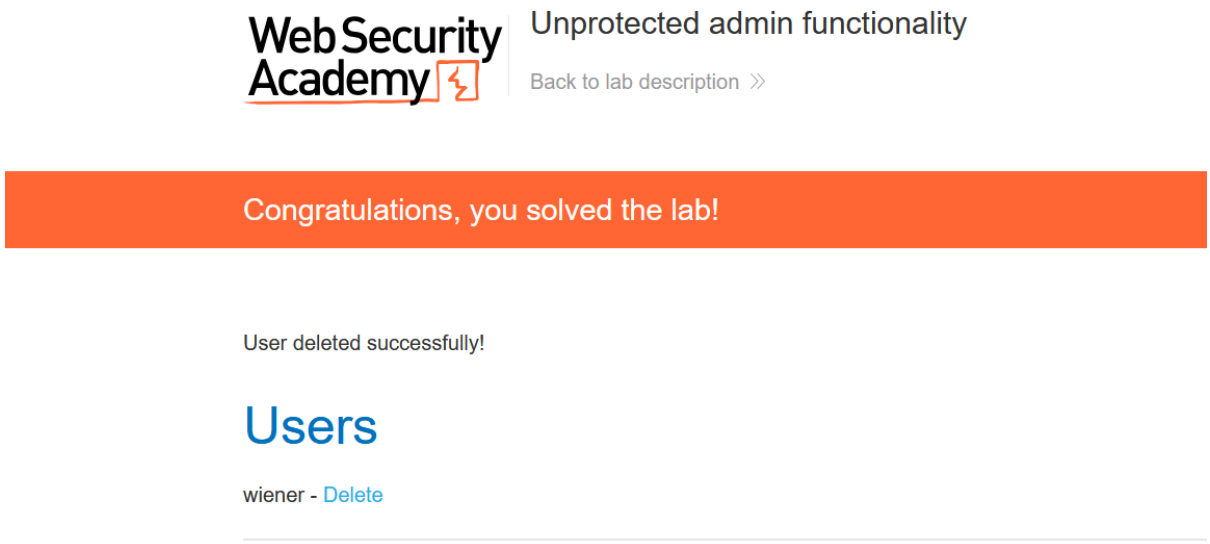
- **Lab Description:**
  - Lab unprotected admin panel'ine sahip. Lab'ı çözmek için carlos user'ını silmemiz gerekiyor.
- **Lab Çözümü:**
  1. Lab'ı açtıktan sonra /robots.txt directory'isine istek yapıyoruz ve file içeriğinde /administrator-panel adında ki directory'i görüyoruz



2. sonrasında bu directory'i ziyaret ediyoruz



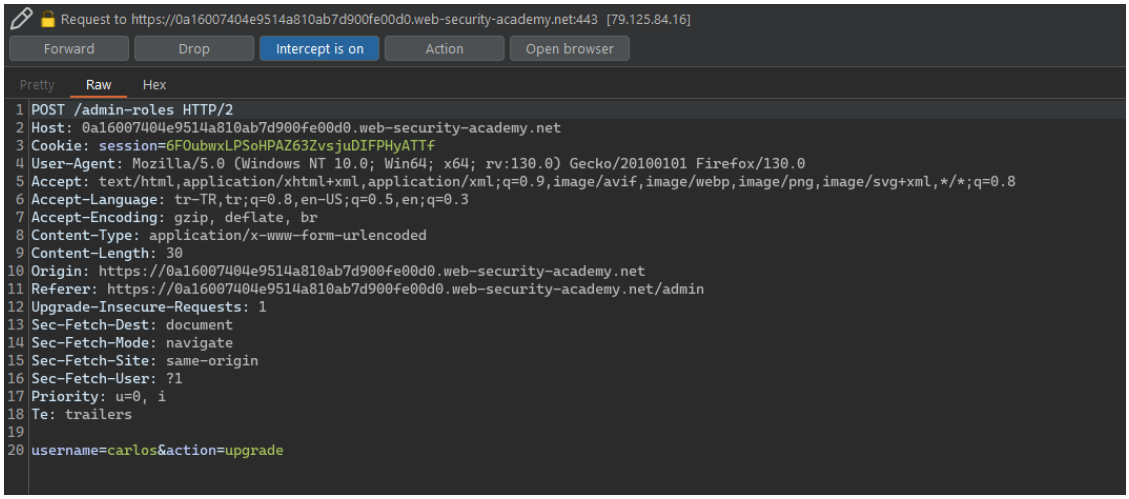
3. Sonrasında carlos'u siliyoruz ve lab çözülmüş oluyor



Second Lab Title -> Lab: Multi-step process with no access control on one step

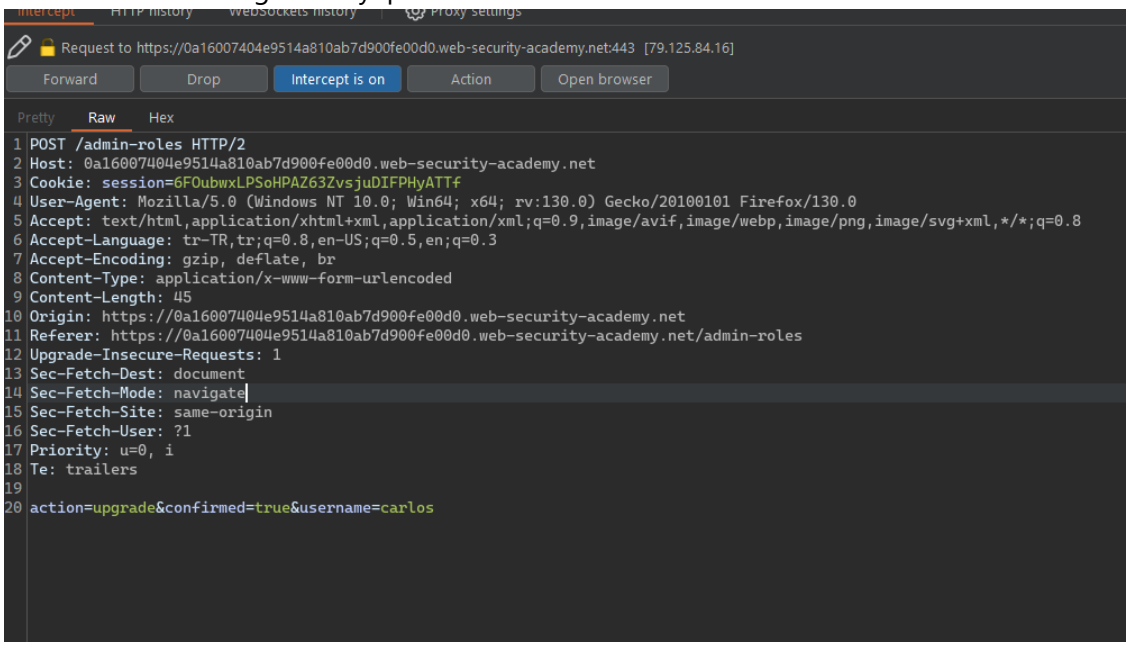
- **Lab Description:**
  - Lab, kullanıcının rolünü değiştirmek için çok adımlı bir sürece sahip bir admin paneline sahip. Lab'ı çözmek için wiener:peter account'una girip, kendi rolümüzü admin olarak değiştirmemiz gerekiyor(öncesinde admin account'una girip fonksiyonlar haşır neşir olmamız gerekiyor). Admin-Credentials -> administrator:admin
- **Lab Çözümü:**
  1. öncelikle admin ile login olup, user role değiştime fonksiyonunu inceliyoruz. Bunun için 'Upgrade user' butonuna tıklayıp isteği burp ile intercept ediyoruz

- Buton'a bastıktan sonraki First Request:

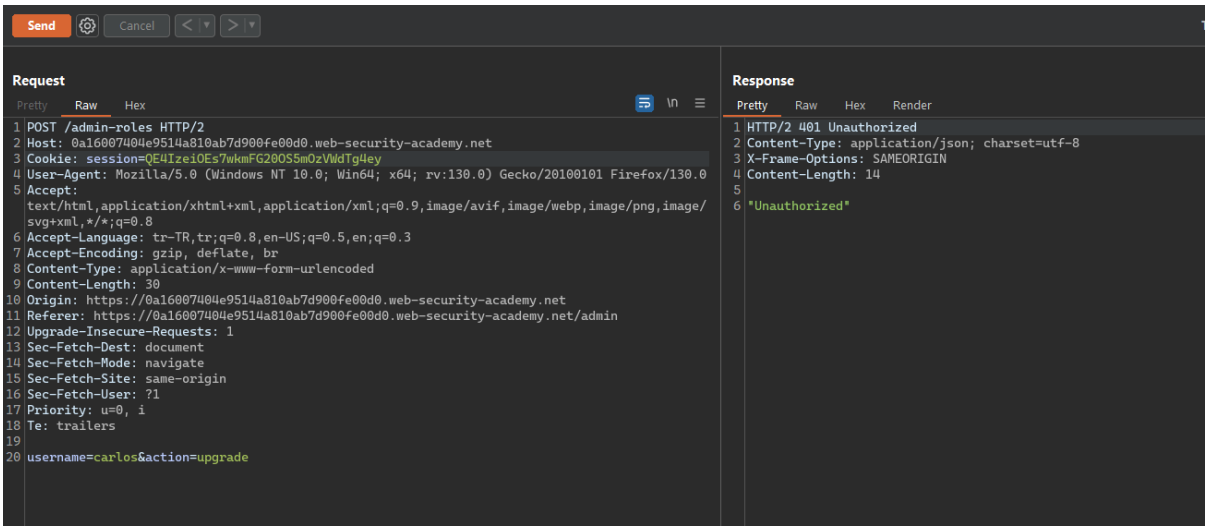


- request'i gönderdikten sonraki Emin misin sorusu:

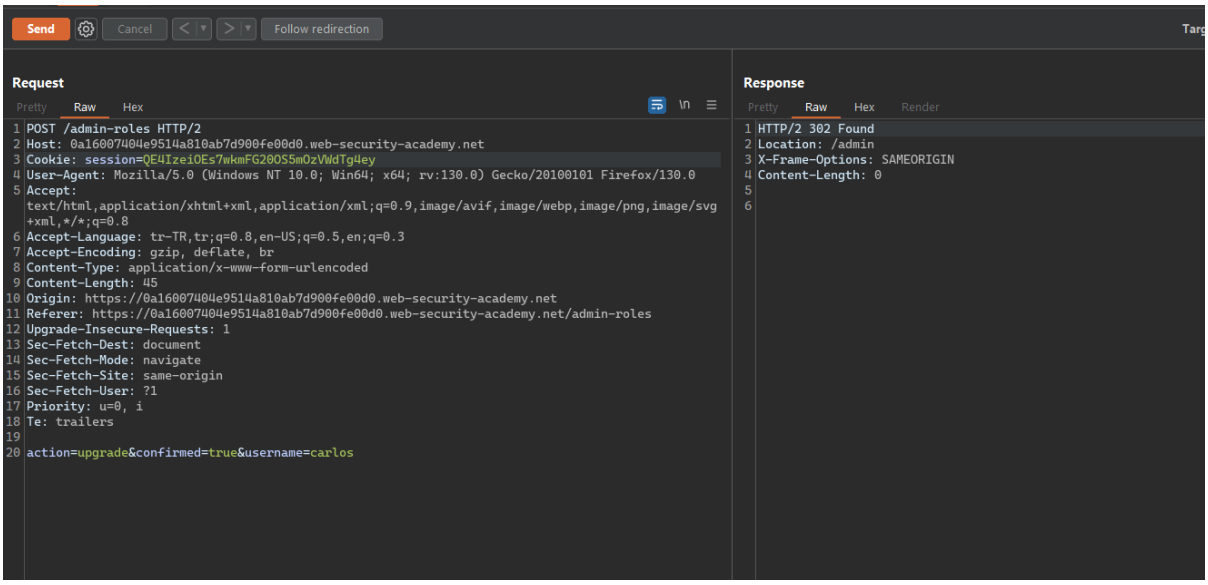
- Yes butonuna tıkladığımızda yapılan istek:



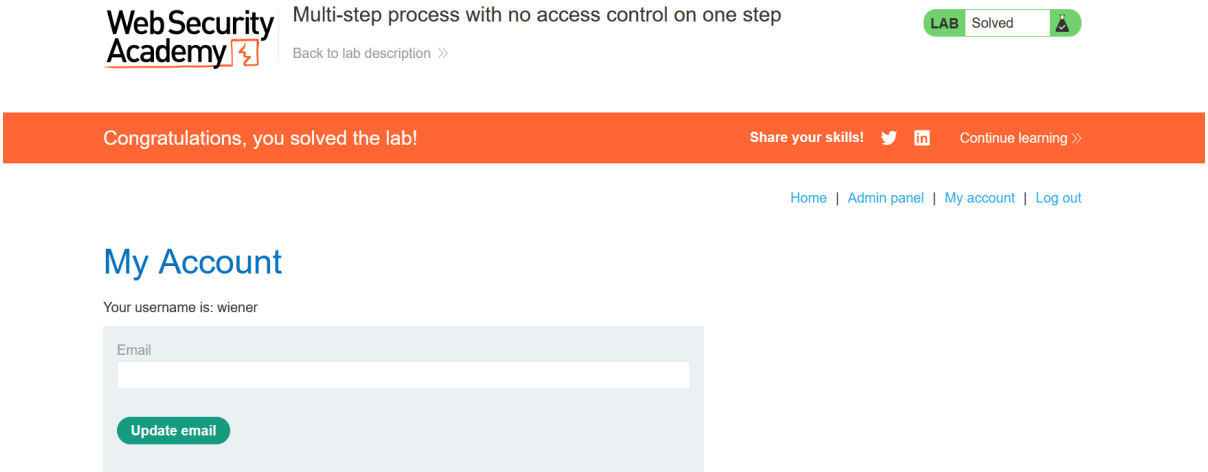
2. Bundan sonra wiener (non-admin) hesabımıza giriş yapıp, onun cookie'si ile ilk request'i tekrarlıyoruz ve 401 unauthorize response'ını alıyoruz



3. Sonrasında emin misin sorusuna yes dediğimizde yapılan request'i non-admin hesabımızın session'ı ile deniyoruz ve 302 redirect alıyoruz



4. Access control'den emin olmak için, kendi username'imiz ile requesti tekrarlıyoruz ve browser'dan sonucun olumlu olduğunu görüp, labı çözmüş oluyoruz

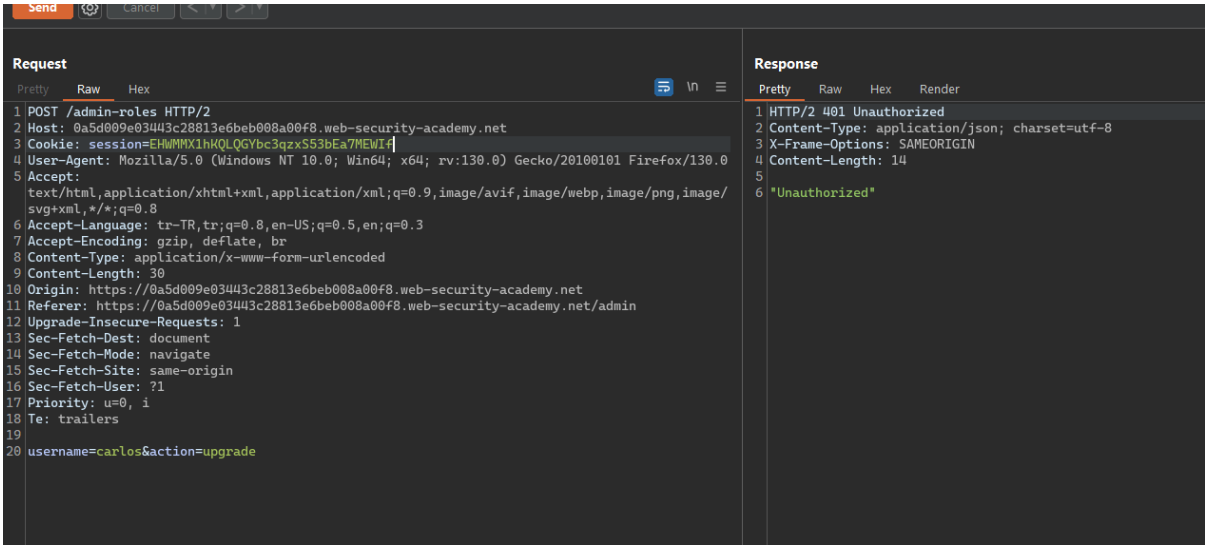


Third Lab Title -> Lab: Method-based access control can be circumvented

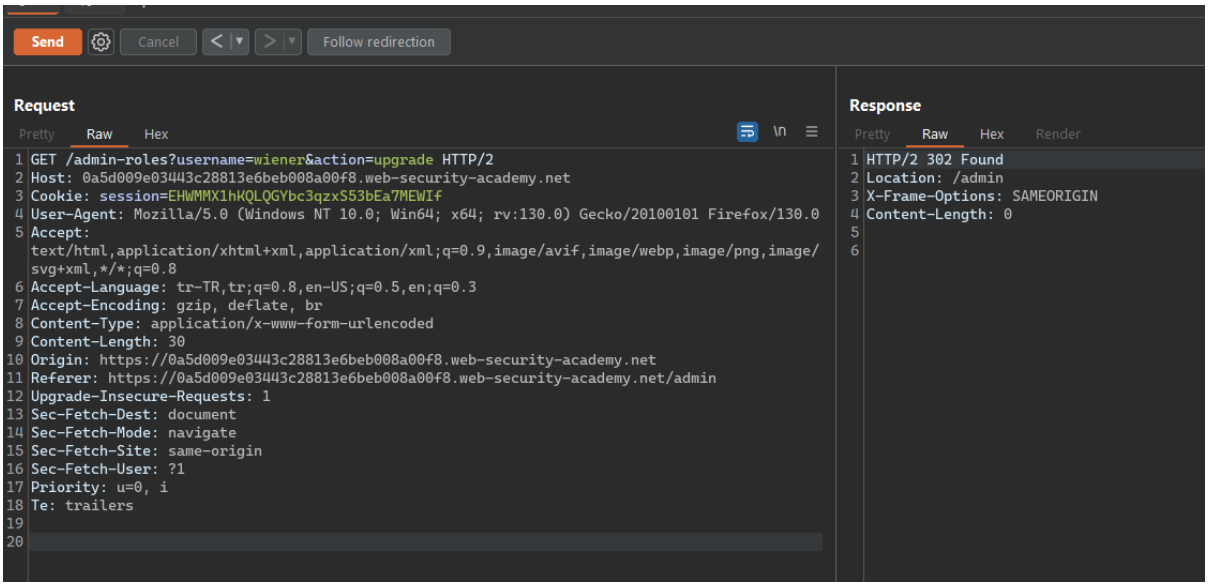
- **Lab Description:**

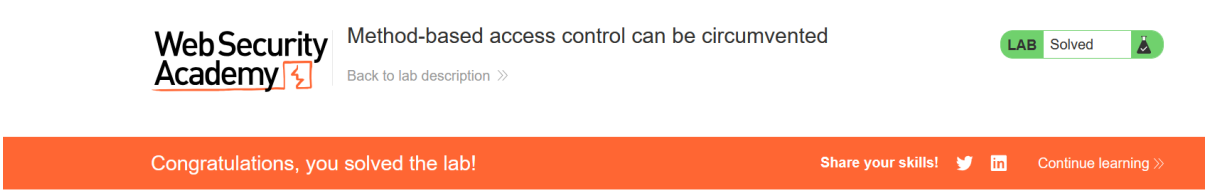
- Lab Access control'ü http header'a göre yapıyor. Lab'ı çözmek için wiener user'ı ile login olup kendimizi admin yapmalıyız. Admin credentials -> administrator:admin

- **Lab Çözümü:**
  1. Öncelikle admin olarak login olup, burp ile user upgrade request'ini intercept ediyoruz
  2. Request'i repeater'a gönderdikten sonra, wiener ile giriş yapıp onun session'ını admin'inki ile değiştirip request'i yapıyoruz ve 401 response alıyoruz

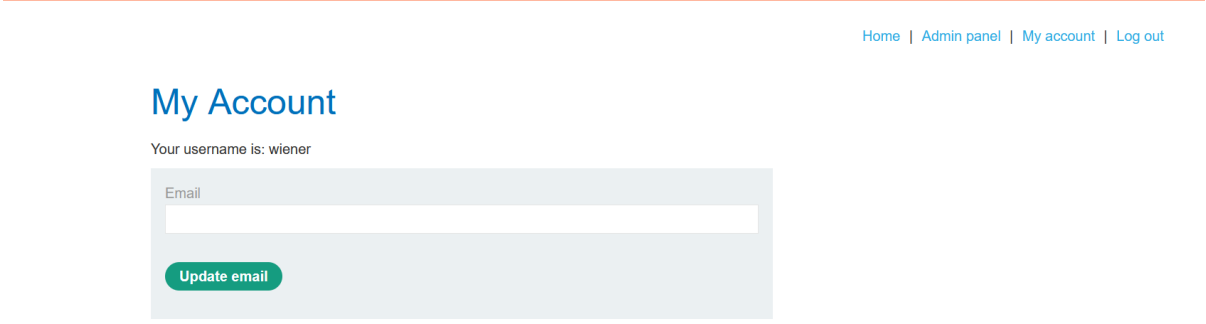


  3. Sonrasında POST header'ı GET ile değiştirip, parametreleri url'in devamına ? username=wiener&action=upgrade yaptığımızda, access control'ün doğru yapılamamasından ötürü admin olarak role'ümüzü başarıyla upgrade edip, lab'ı çözüyoruz


Send
Cancel
Follow redirection
Request
Pretty Raw Hex
GET /admin-roles?username=wiener&action=upgrade HTTP/2
Host: 0a5d009e03443c28813e6beb008a00f8.web-security-academy.net
Cookie: session=EHWMMX1hKQLQGYbc3qzxS53bEa7MEWIf
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:130.0) Gecko/20100101 Firefox/130.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/png,image/svg+xml,*/*;q=0.8
Accept-Language: tr-TR,tr;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate, br
Content-Type: application/x-www-form-urlencoded
Content-Length: 30
Origin: https://0a5d009e03443c28813e6beb008a00f8.web-security-academy.net
Referer: https://0a5d009e03443c28813e6beb008a00f8.web-security-academy.net/admin
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
Priority: u=0, i
Te: trailers
Response
Pretty Raw Hex Render
HTTP/2 302 Found
Location: /admin
X-Frame-Options: SAMEORIGIN
Content-Length: 0


Web Security Academy
Method-based access control can be circumvented
Back to lab description
LAB Solved

Congratulations, you solved the lab!
Share your skills!
Continue learning


Home | Admin panel | My account | Log out
My Account
Your username is: wiener
Email
Update email