

Not Under the Doormat

Securing your database credentials on the web application
client side

Helen Huang
Security Consultant
Aura Information Security



aura
INFORMATION SECURITY

POWERED
BY KORDIA

whoami

- Lives in Auckland
- Security Consultant / Penetration Tester at Aura Information Security
- Ex-Software Engineer
- Recently promoted to being a mum!



Thank You to Our Sponsors and Hosts!



OWASP
**NEW
ZEALAND**
owasp.org.nz



QUANTUM
SECURITY



Cyber**CX**

DATACOM



snyk



Auth0

Checkmarx



HCL AppScan

kordia



**LATERAL
SECURITY**



**MICRO
FOCUS**



Pulse Security
www.pulsesecurity.co.nz



RedShield



Flux

SEQA

Information Security



Cobalt



LACEWORK



Secure**Flag**

Without them, OWASP New Zealand Day couldn't happen





Agenda

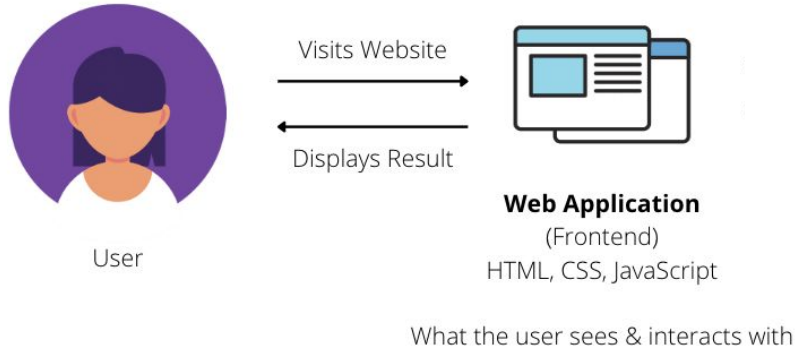
1. Background
2. Case study
3. Implementations with S3 pre-signed URLs
4. Security considerations of pre-signed URLs



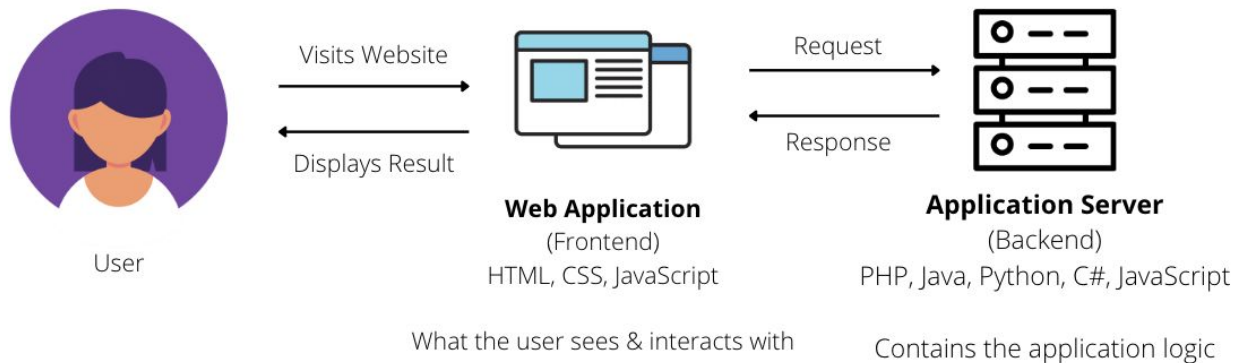
Web Application Architecture



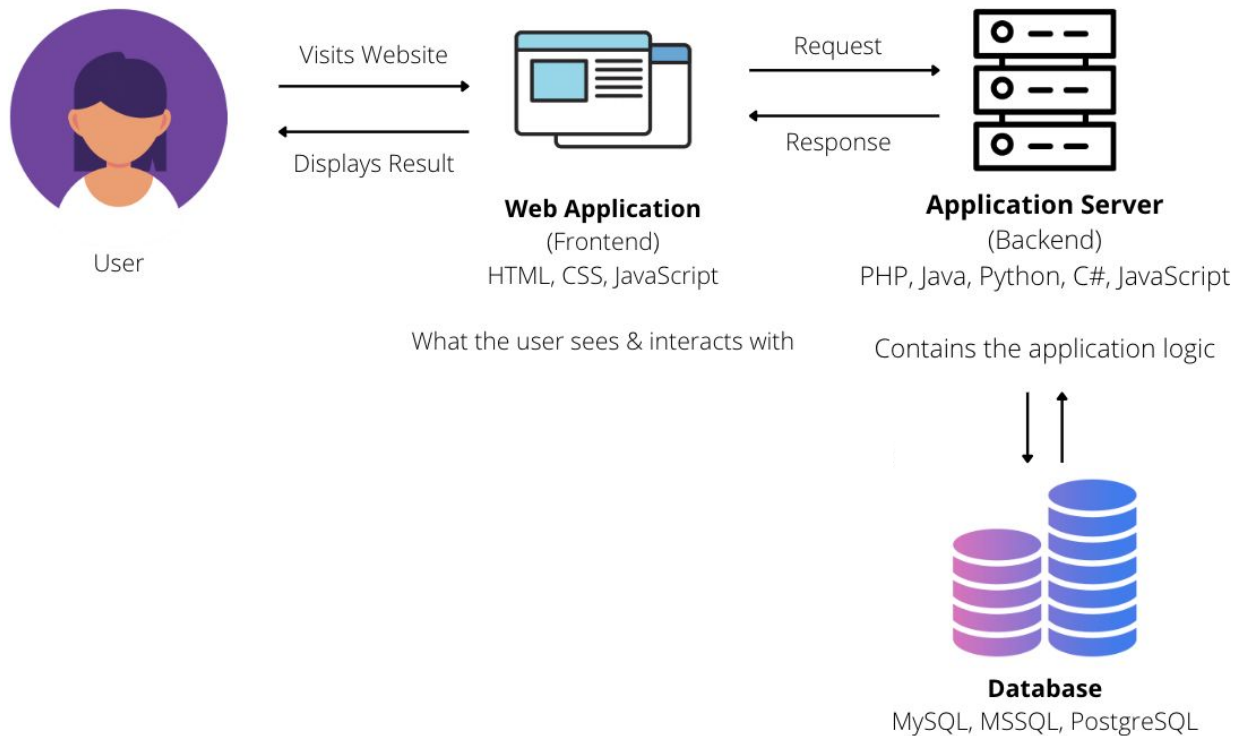
Web Application Architecture



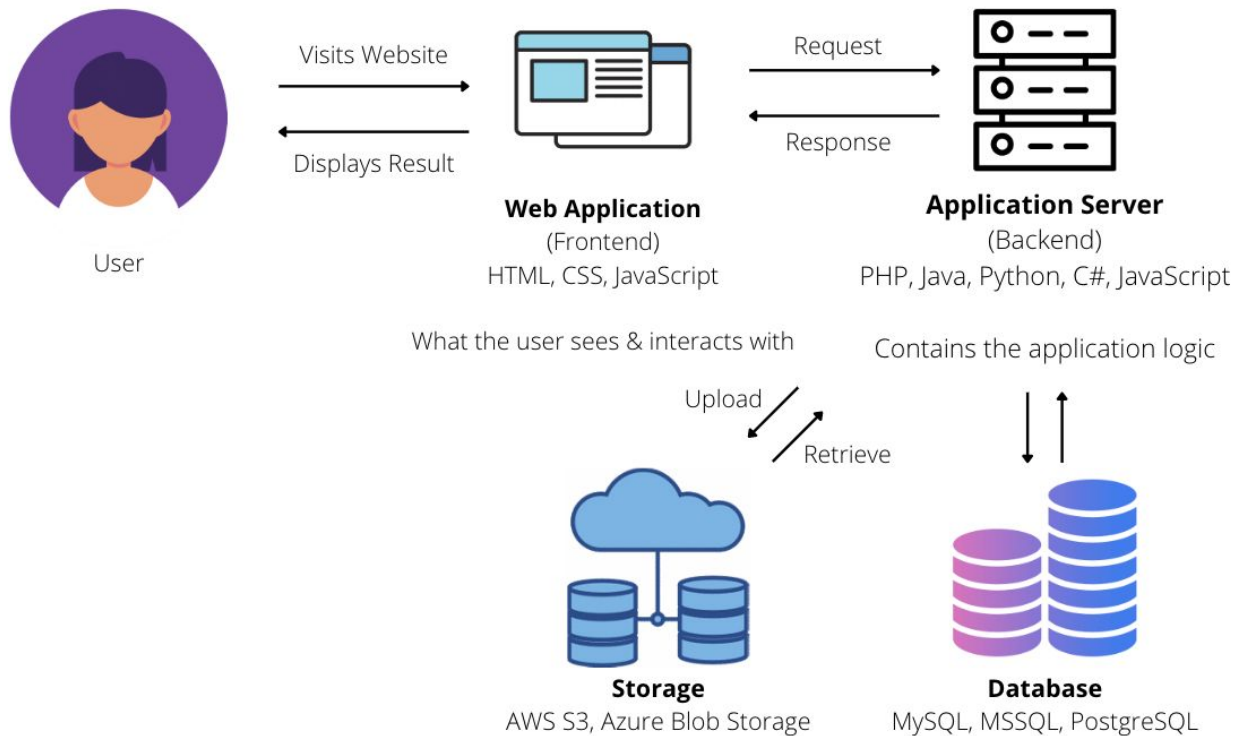
Web Application Architecture



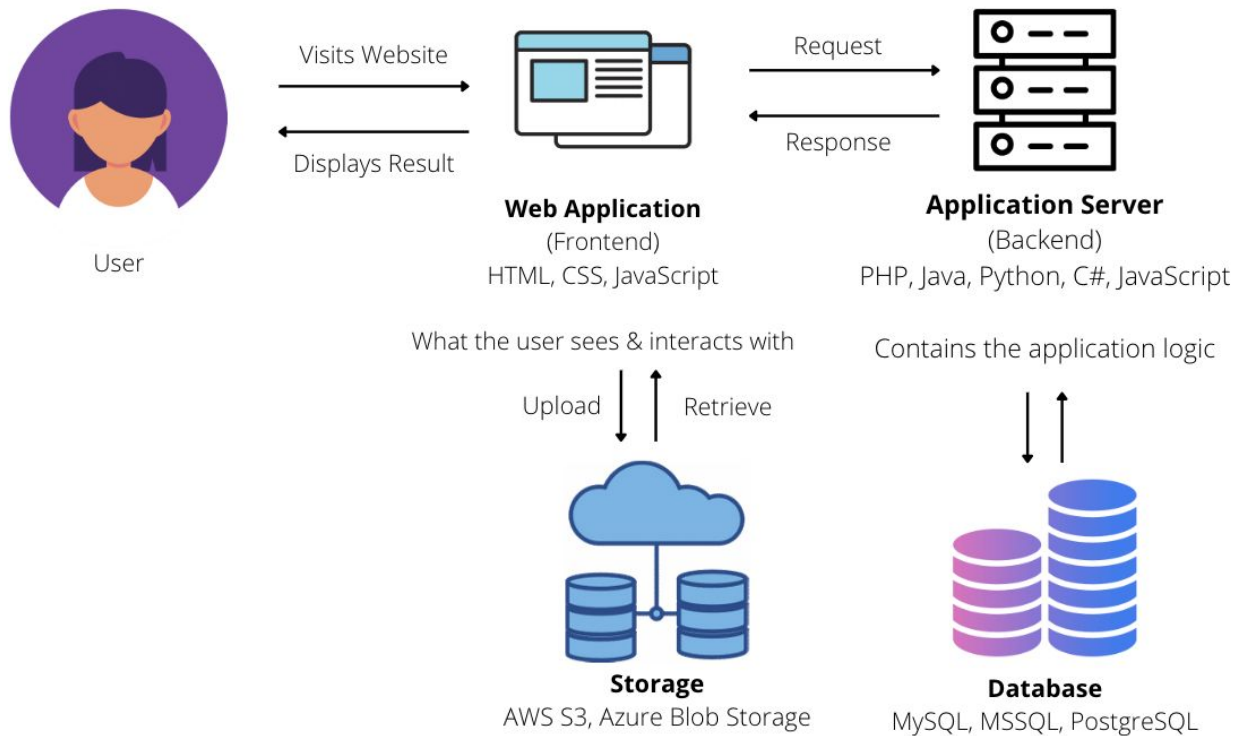
Web Application Architecture



Web Application Architecture



Web Application Architecture



Case Study



Case Study

Real engagement with a NZ company



Case Study

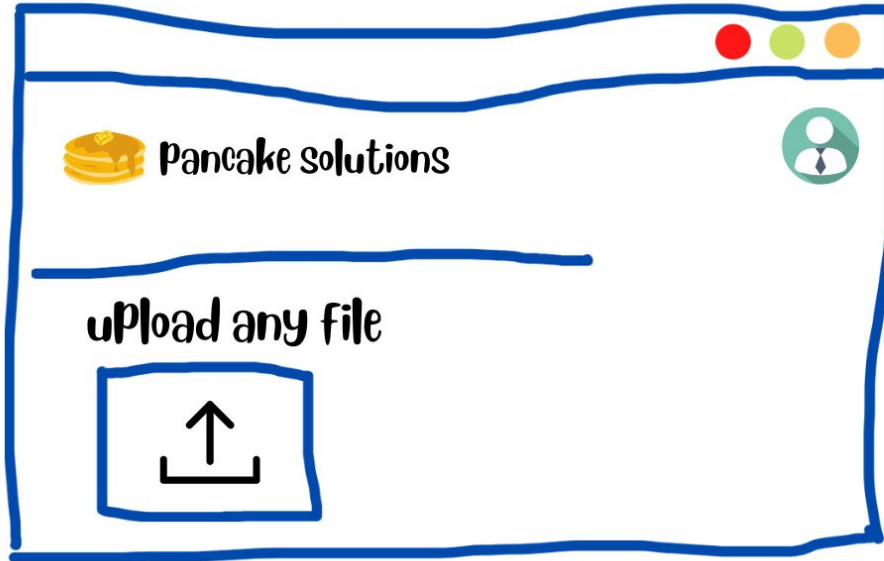
Real engagement with a NZ company

Exposed credentials -> database breach



Case Study

File upload functionality on an administrator portal



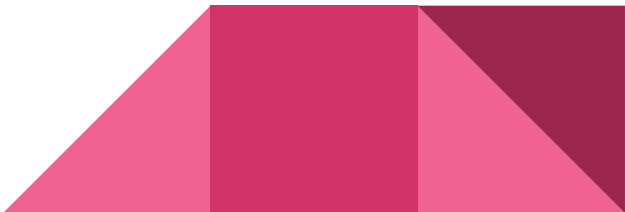
Case Study

HTTP REQUEST

```
POST /media/upload/link HTTP/1.1
```

```
Host: pancakesolutions.co.nz
```

```
name=test&description=document&purpose=material&upload_size=5&upload_name=test.txt&upload_mimetype=text%2Fplain
```



Case Study

HTTP RESPONSE

```
HTTP/1.1 200 OK
```

```
{  
  "success":true,  
  "message":"Upload url generated successfully.",  
  "payload":{  
    "storage_service":"amazon-s3",  
    "media_id":17,  
    "storage_key":"5fd93beace21b.txt",  
    "awsconfig":"eyJhY2Nlc3Nfa2V5X2lkIjoiQutJQTR0WFZXVUZDV  
FFWVTQ2WlMiLCJhY2Nlc3Nfa2V5X3NlY3JldCI6IjZ4M25ERWt3QUk1bTJl  
ZTlsNWdJZTFXQitySHc4b3AwbjVReDJwbkIiLCJidWNrZXQiOiJwYW5jYWt  
lc29sdXRpb25zLXVhdCIsInJlZ2lubiI6ImFwLXNvdXRvZWFzdC0yIn0="
```

Case Study

HTTP RESPONSE

```
HTTP/1.1 200 OK
```

```
{  
  "success":true,  
  "message":"Upload url generated successfully.",  
  "payload":{  
    "storage_service":"amazon-s3",  
    "media_id":17,  
    "storage_key":"5fd93beace21b.txt",  
    "awsconfig":"eyJhY2Nlc3Nfa2V5X2lkIjoiQutJQTR0WFZXVUZDV  
FFWVTQ2WlMiLCJhY2Nlc3Nfa2V5X3NlY3JldCI6IjZ4M25ERWt3QUk1bTJl  
ZTlsNWdJZTFXQitySHc4b3AwbjVReDJwbkIiLCJidWNrZXQiOiJwYW5jYWt  
lc29sdXRpb25zLXVhdCI6InJlZ2lubiI6ImFwLXNvdXRvZWFzdC0yIn0="
```

Case Study

Decoding the `awsconfig` variable

```
$ echo  
"eyJhY2Nlc3Nfa2V5X2lkIjoIQUtJQTR0WFZXVUZDVFFWVTQ2WlMiLCJhY2Nlc3Nfa2V5X3NlY3Jl  
dCI6IjZ4M25ERWt3QUk1bTJlZTlsNWdJZTFXQitySHc4b3AwbjVReDJwbkIiLCJidWNrZXQiOiJwY  
W5jYWtlc29sdXRpb25zLXVhdCIsInJlZ2lubiI6ImFwLXNvdXRoZWZdC0yIn0=" | base64 -d  
  
{ "access_key_id": "AKIA4NXVUFCTQVU46ZS", "access_key_secret": "6x3nDEkwAI5m2ee9  
l5gIe1WB+rHw8op0n5Qx2pnB", "bucket": "pancakesolutions-uat", "region": "ap-  
southeast-2" }
```

Case Study

Decoding the `awsconfig` variable

```
$ echo  
"eyJhY2Nlc3Nfa2V5X2lkIjoIQUtJQTR0WFZXVUZDVFFWVTQ2WlMiLCJhY2Nlc3Nfa2V5X3NlY3Jl  
dCI6IjZ4M25ERWt3QUk1bTJlZTlsNWdJZTFXQitySHc4b3AwbjVReDJwbkIiLCJidWNrZXQiOiJwY  
W5jYWtlc29sdXRpb25zLXVhdCIsInJlZ2lubiI6ImFwLXNvdXRoZWZdC0yIn0=" | base64 -d  
  
{  
  "access_key_id": "AKIA4NXVUFCTQVU46ZS",  
  "access_key_secret": "6x3nDEkwAI5m2ee9  
l5gIe1WB+rHw8op0n5Qx2pnB",  
  "bucket": "pancakesolutions-uat",  
  "region": "ap-  
southeast-2"}  
}
```

Case Study

.../admin/build/main.js

JavaScript code which uses the
awsconfig variable to access
S3 bucket and upload

```
...  
var a = JSON.parse(atob(t.payload.awsconfig));  
n.a.config.update({  
  accessKeyId: a.access_key_id,  
  secretAccessKey: a.access_key_secret,  
  region: a.region  
});  
var u = (new n.a.S3).upload({  
  Key: t.payload.storage_key,  
  Body: this.fileInput.files[0],  
  ContentType: this.fileInput.files[0].type,  
  Bucket: a.bucket,  
  ACL: "bucket-owner-full-control"  
})  
...
```

Case Study

Using AWS Command Line Interface (CLI) to access the S3 bucket

```
$ aws configure
AWS Access Key ID [None]: AKIA4NXVWUFCTQVU46ZS
AWS Secret Access Key [None]: 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
Default region name [None]: ap-southeast-2
Default output format [None]: json
```


Case Study

```
$ aws s3 ls
2019-06-13 01:19:43 869526256865-deployments
2020-06-16 17:29:01 [REDACTED] -cloudformation
2020-04-17 00:45:24 [REDACTED] -main
2020-04-17 00:45:33 [REDACTED] -main-uat
2019-06-04 04:50:57 [REDACTED] -uat-video-in
2019-06-04 04:51:48 [REDACTED] -uat-video-out
2019-05-29 00:40:32 [REDACTED] -video-in
2019-06-05 02:30:39 [REDACTED] -video-out
2020-12-09 05:46:01 [REDACTED]
2020-12-03 08:30:01 [REDACTED] -uat
2020-12-03 08:27:15 [REDACTED] -uat-video-in
2020-12-03 08:31:29 [REDACTED] -uat-video-out
2020-12-07 22:05:36 [REDACTED] -video-in
2020-12-07 22:06:39 [REDACTED] -video-out
2020-06-16 20:35:16 [REDACTED] -in
2020-06-16 20:35:18 [REDACTED] -out
2020-06-16 21:28:26 [REDACTED] -video-in
2020-06-16 21:28:27 [REDACTED] -video-out
```

```
2020-06-16 23:07:06 [REDACTED]
2020-06-16 23:07:06 [REDACTED] -dev
2020-06-16 23:07:06 [REDACTED] -dev-video-in
2020-06-16 23:07:06 [REDACTED] -dev-video-out
2020-06-16 23:07:06 [REDACTED] -staging
2020-06-16 23:07:06 [REDACTED] -staging-video-in
2020-06-16 23:07:06 [REDACTED] -staging-video-out
2020-06-16 23:07:06 [REDACTED] -video-in
2020-06-16 23:07:06 [REDACTED] -video-out
2020-06-16 23:07:06 [REDACTED]
2019-07-24 04:14:30 [REDACTED] -dev
2020-06-17 01:50:45 [REDACTED]
2020-06-17 01:50:46 [REDACTED] -uat
2020-06-17 01:50:46 [REDACTED] -uat-video-in
2020-06-17 01:50:46 [REDACTED] -uat-video-out
2020-06-17 01:50:46 [REDACTED] -video-in
2020-06-17 01:50:46 [REDACTED] -video-out
2019-06-03 21:39:16 [REDACTED] -dev
2019-06-03 21:39:51 [REDACTED] -dev-video-in
2019-06-03 21:40:09 [REDACTED] -dev-video-out
2019-06-06 02:58:19 [REDACTED] -db-backups
```


Case Study

```
$ aws s3 ls s3://[REDACTED]-db-backups
PRE 2019-06-06/
PRE 2019-06-07/
PRE 2019-06-08/
...
PRE 2020-12-09/
PRE 2020-12-10/
PRE 2020-12-11/
```



Case Study

```
$ aws s3 ls s3://[REDACTED]-db-backups
PRE 2019-06-06/
PRE 2019-06-07/
PRE 2019-06-08/
...
PRE 2020-12-09/
PRE 2020-12-10/
PRE 2020-12-11/
```

```
$ aws s3 ls s3://[REDACTED]-db-backups/2020-12-11/
2020-12-10 15:00:16 3363808 [REDACTED].sql
2020-12-10 15:00:16 1027048554 [REDACTED].sql
2020-12-10 15:00:31 25387088 [REDACTED].sql
```



Case Study

```
$ cat [REDACTED].sql

(... omitted ...)
COPY public.users (id, [REDACTED]_id, privacy_level_id, user_type_id, [REDACTED]_user_id, email,
nickname, [REDACTED]_id, password, salt, [REDACTED], [REDACTED]_completed,
event_server_connected, event_server_disconnected, created_at, updated_at, deleted_at,
remember_token, [REDACTED] [REDACTED]_id, signup_status, password_updated_at, disabled) FROM
stdin;

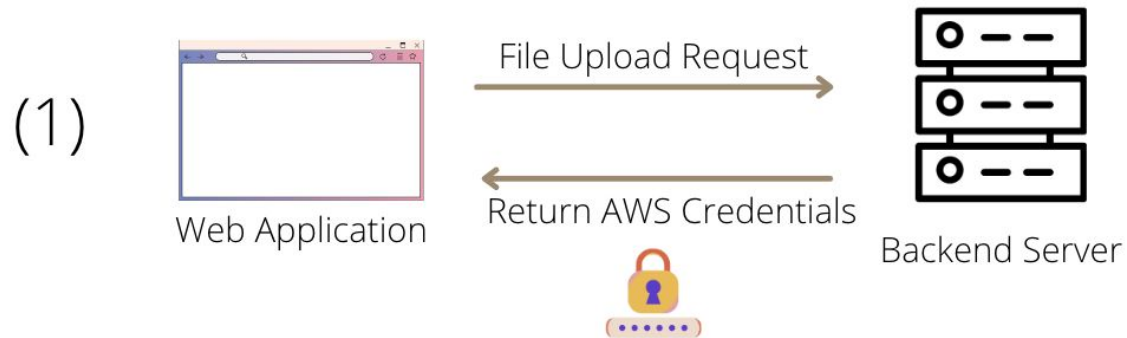
28 1 1 4 \N [REDACTED]+admin@[REDACTED].co.nz [REDACTED] \N
$2y$10$[REDACTED]Bpji
7acf9d8653ce360143f793669d3c04b4ea0a9278 \N \N \N \N 2014-05-20 16:40:40 2014-08-08
13:48:32 2014-08-08 13:48:32 \N f \N \N \N f

33 1 1 4 \N [REDACTED]+admin@[REDACTED].com [REDACTED] \N
$2y$10$[REDACTED]EqCu
54c5090d7e2c6f480363fc03c8ec49da2950fb69 \N \N 2015-08-04 13:46:46 2015-08-04 13:47:00
2014-05-20 16:40:41 2015-08-06 09:12:38 2015-08-06 09:12:38 \N f \N \N \N f

1630 42 1 4 \N [REDACTED]+admin@[REDACTED].com [REDACTED] \N
$2y$10$[REDACTED]kwMC
cef35dbc24de2100f4922066f7d3804202b228b4 \N \N 2016-05-13 14:45:28 2016-05-13 15:40:24
2016-03-17 14:20:43 2017-03-15 09:28:31 2017-03-15 09:28:31 \N f \N \N \N f

(... omitted ...)
```

Case Study



Case Study

Any user of the web application who had access to the upload functionality had access to all the database backups from this company





OWASP TOP 10 - Cryptographic Failures



OWASP TOP 10 - Cryptographic Failures

Sensitive Data Exposure -> Cryptographic Failures



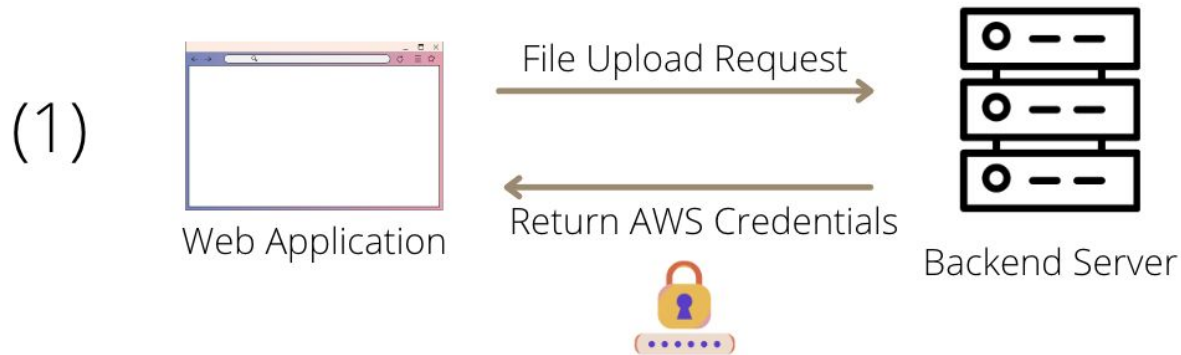
OWASP TOP 10 - Cryptographic Failures

Sensitive Data Exposure -> Cryptographic Failures

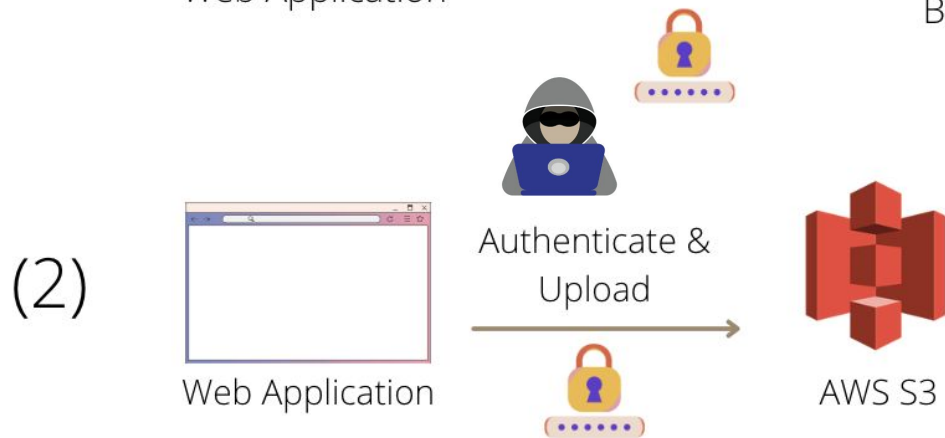
Moved up from #3 from 2017 -> #2 in 2021



OWASP TOP 10 - Cryptographic Failures



OWASP TOP 10 - Cryptographic Failures



What should we do?



What should we do?

We want:



What should we do?

We want:

Uploading directly from the client front end



What should we do?

We want:

Uploading directly from the client front end

But the client shouldn't have access to the credentials!



What should we do?

We want:

Uploading directly from the client front end

But the client shouldn't have access to the credentials!

How can the client upload without credentials?



What should we do?

AWS Pre-signed URL!



Pre-signed URLs

Grant temporary restricted access to objects in AWS S3 buckets for a predefined time period.



Pre-signed URLs

Grant temporary restricted access to objects in AWS S3 buckets for a predefined time period.

An AWS user will specifying the object they want to allow access to, the action (HTTP GET or HTTP PUT), and use their secret key to sign the URL.



Pre-signed URLs

Grant temporary restricted access to objects in AWS S3 buckets for a predefined time period.

An AWS user will specifying the object they want to allow access to, the action (HTTP GET or HTTP PUT), and use their secret key to sign the URL.



Pre-signed URLs

Grant temporary restricted access to objects in AWS S3 buckets for a predefined time period.

An AWS user will specifying the object they want to allow access to, the action (`HTTP GET` or `HTTP PUT`), and use their secret key to sign the URL.

Anyone with access to it can perform the action embedded in the URL as if they were the original signing user.



New Upload Approach



New Upload Approach

Two step process:



New Upload Approach

Two step process:

1. Obtain a pre-signed URL from the backend



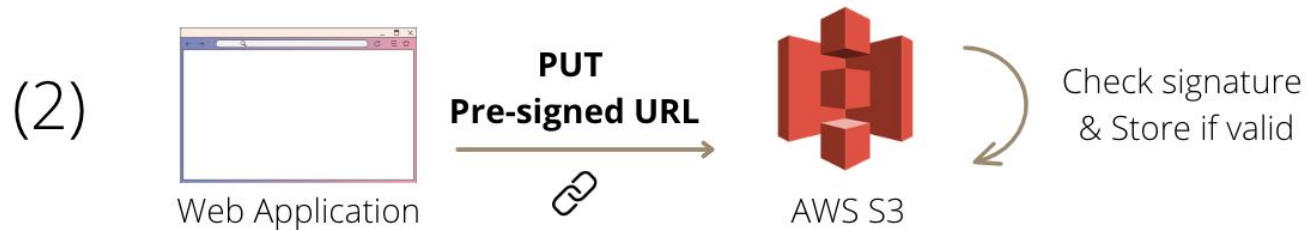
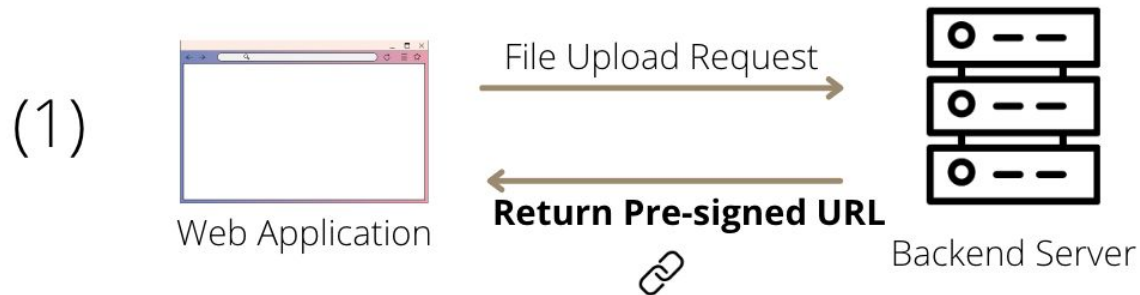
New Upload Approach

Two step process:

1. Obtain a pre-signed URL from the backend
2. Use the pre-signed URL to upload the object



New Upload Approach



Creating Pre-signed URL in Python



Creating Pre-signed URL in Python

```
import boto3

s3_client =
    boto3.client(
        's3',
        aws_access_key_id=AKIA4NXVWUFCTQVU46ZS,
        region_name=ap-southeast-2,
        aws_secret_access_key=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,
        config=Config(signature_version='s3v4'))

presigned_url =
    s3_client.generate_presigned_url(
        'put_object',
        Params={"Bucket": "test-bucket-123", "Key": "image.jpg"},
        ExpiresIn=1000)
```

Pre-signed URL

```
https://test-bucket-123.s3.ap-southeast-2.amazonaws.com/file.txt
?X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=AKIANXTQ6FVWU4UCV4ZS/20201217/ap-southeast-
2/s3/aws4_request
&X-Amz-Date=20201217T203752Z
&X-Amz-SignedHeaders=host
&X-Amz-Expires=7200
&X-Amz-Signature=a84af4b4d786ae759ece86d53453a7ed1ad33d61b882aec7c4da
c9600ee62d2
```

Securing Pre-signed URLs

Like with everything - they need to be implemented securely!



Securing Pre-signed URLs

1. Limiting expiry time



Securing Pre-signed URLs

1. Limiting expiry time

Once the presigned URL is generated, it will be valid for an unlimited amount of times before it expires.



Securing Pre-signed URLs

1. Limiting expiry time

Once the presigned URL is generated, it will be valid for an unlimited amount of times before it expires.

The default pre-signed URL expiration time is 15 minutes.



Securing Pre-signed URLs

1. Limiting expiry time

Once the presigned URL is generated, it will be valid for an unlimited amount of times before it expires.

The default pre-signed URL expiration time is 15 minutes.

Keep it to the minimum possible.



Securing Pre-signed URLs

2. Least Privilege



Securing Pre-signed URLs

2. Least Privilege

Create a separate IAM role in AWS that has the least amount of privilege.



Securing Pre-signed URLs

2. Least Privilege

Create a separate IAM role in AWS that has the least amount of privilege.

Only grant the necessary permissions (e.g. read object, put object)



Securing Pre-signed URLs

2. Least Privilege

Create a separate IAM role in AWS that has the least amount of privilege.

Only grant the necessary permissions (e.g. read object, put object)

Really good for incident response and revocation



IAM Policy

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:PutObject",
        "s3:GetObject"
      ],
      "Resource": "arn:aws:s3:::test-bucket-123"
    }
  ]
}
```

Securing Pre-signed URLs

3. Control File Content



Securing Pre-signed URLs

3. Control File Content

Once a pre-signed URL is generated, normally you don't have control over who can upload a file or what file is uploaded.



Securing Pre-signed URLs

3. Control File Content

Once a pre-signed URL is generated, normally you don't have control over who can upload a file or what file is uploaded.

Specify the `Content-MD5` header while generating the pre-signed URL with the hash of the file being uploaded.




Securing Pre-signed URLs

3. Control File Content

Once a pre-signed URL is generated, normally you don't have control over who can upload a file or what file is uploaded.

Specify the `Content-MD5` header while generating the pre-signed URL with the hash of the file being uploaded.

Enforce the presigned URL to be valid only if the specified value for this header is the same from the one specified, and the one received by the user while uploading a file.



Securing Pre-signed URLs

4. Enable Logging



Securing Pre-signed URLs

4. Enable Logging

This applies even if you don't use pre-signed URL.



Securing Pre-signed URLs

4. Enable Logging

This applies even if you don't use pre-signed URL.

Server access logging provides detailed records for the requests that are made to a bucket.



Securing Pre-signed URLs

4. Enable Logging

This applies even if you don't use pre-signed URL.

Server access logging provides detailed records for the requests that are made to a bucket.

Not enabled by default.



Securing Pre-signed URLs

1. Limiting expiry time
2. Least Privilege
3. Control File Content
4. Enable Logging



Takeaway

Remember: everything that you send to the front end client is visible to the end user

Every time when you are working with any type of credentials - think really carefully about where it's stored!



Thank you!

email: helen.jiahe.huang@gmail.com

linkedin: www.linkedin.com/in/helen-jiahe-huang/

twitter: [@__helenhuang](https://twitter.com/__helenhuang)

