Tutorial 04

Security

Introduction

- Browser tools demo
  - Elements
  - Console
  - Network
  - Storage
- Refer: MDN Web Docs (https://developer.mozilla.org/)
  - https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies

What are you watching?

- **When a website is only using HTTPS, your ISP can know which of the following:**

1. Cookies
2. Entered passwords
3. Visited website
4. Pages you visit on this website
5. Other servers you communicate with

What are you watching?

- **When a website is only using HTTPS, your ISP can know which of the following:**

1. Cookies
2. Entered passwords
3. Visited website
4. Pages you visit on this website
5. Other servers you communicate with

Also: Consider relooking at E2.3 TLS Forensics. What is visible without keylog file?

What do I know?

- **You are visiting: www.foo.com/dir/index.html#top**

  The server knows that you are visiting (select most complete answer)

1. www.foo.com
2. www.foo.com/dir
3. www.foo.com/dir/index.html
4. www.foo.com/dir/index.html#top

What do I know?

- **You are visiting: www.foo.com/dir/index.html#top**

  The server knows that you are visiting (select most complete answer)

1. www.foo.com
2. www.foo.com/dir
3. www.foo.com/dir/index.html
4. www.foo.com/dir/index.html#top

Fragment is not communicated to the server.

"Do Not Track"

- **You enable "Do Not Track" in your browser (i.e. Send websites a signal that you don't want to be tracked; navigator.doNotTrack is set to 1)**

  Does this website still use Google Analytics when you visit the page?

```
if(window.doNotTrack === "1" || navigator.doNotTrack === "1") {
 // code to add google analytics
}
```

1. True
2. False

"Do Not Track"

- **You enable "Do Not Track" in your browser (i.e. Send websites a signal that you don't want to be tracked; navigator.doNotTrack is set to 1)**

  Does this website still use Google Analytics when you visit the page?

```
if(window.doNotTrack === "1" || navigator.doNotTrack === "1") {
 // code to add google analytics
}
```

1. True
2. False

With DNT you only “signal”. It is upto the website to consider it or not.

Cookies

- **Cookies are generally transmitted via:**

1. Request/Response Body
2. Request/Response Headers
3. URL
4. Including JavaScript code

Cookies

- **Cookies are generally transmitted via:**

1. Request/Response Body
2. Request/Response Headers
3. URL
4. Including JavaScript code

Look at network requests in browser: `set-cookie` and `cookie`

Same Origin Policy (SOP)

- **Do these two URLs have same origin?**

  http://store.company.com/dir/page.html

  http://news.company.com/dir/page.html

1. True
2. False

Same Origin Policy (SOP)

- **Do these two URLs have same origin?**

  http://store.company.com/dir/page.html

  http://news.company.com/dir/page.html

1. True
2. False

Same Origin Policy considers: Protocol + Host + Port
Here, hosts are different.

Cookie Attributes

- **Match cookie attributes with the kind of attack they prevent:**

1. Secure
2. HttpOnly
3. SameSite

A. Cross-site request forgery (CSRF)
B. Man in the middle (MITM) attacks over network
C. Cookie theft via JavaScript

Cookie Attributes

- **Match cookie attributes with the kind of attack they prevent:**

1. Secure → Man in the middle (MITM) attacks over network
2. HttpOnly → Cookie theft via JavaScript
3. SameSite → Cross-site request forgery (CSRF)

Someone told me

- **Same Origin Policy prevents CSRF?**

1. True
2. False

Someone told me

- **Same Origin Policy prevents CSRF?**

1. True
2. False

You can not access the response to cross site request but the request is still sent,
which is enough for the attack

What are you watching?

- **One of the tutors proposes a new CSRF defense mechanism for CMS. This JavaScript snippet is added to every page by CMS server:**

```
document.write("<input id=csrf_token value="+Math.random()+">")
```

It adds random CSRF token on client side. This is then sent to back to server with every request. Should this be used? Why/Why not?

1. Yes
2. No

What are you watching?

- **One of the tutors proposes a new CSRF defense mechanism for CMS. This JavaScript snippet is added to every page by CMS server:**

```
document.write("<input id=csrf_token value="+Math.random()+">")
```

It adds random CSRF token on client side. This is then sent to back to server with every request. Should this be used? Why/Why not?

1. Yes
2. No

The token is random but the server has no way to know what number was actually generated on client side.

Content Security Policy

- **We are www.example.com**

  We need to run following code on our website:

```
<html>
<body>
<!-- external code -->
<script src="https://code.jquery.com/jquery-3.6.0.js"></script>
</body>
</html>
```

  What is the minimal CSP?

Content Security Policy

- **We are www.example.com**

  We need to run following code on our website:

```
<html>
<body>
<!-- external code -->
<script src="https://code.jquery.com/jquery-3.6.0.js"></script>
</body>
</html>
```

  What is the minimal CSP?

```
Content-Security-Policy: script-src
https://code.jquery.com/jquery-3.6.0.js
```

Content Security Policy

- **We are www.example.com**

  Further, we need to run following code:

```
<html>
<body>
<!-- external code -->
<script src="https://code.jquery.com/jquery-3.6.0.js"></script>
<script>
alert("Hello World");
</script>
</body>
</html>
```

  What is the minimal CSP?

Content Security Policy

```
<html>
<body>
<!-- external code -->
<script src="https://code.jquery.com/jquery-3.6.0.js"></script>
<script>
alert("Hello World");
</script>
</body>
</html>
```

What is the minimal CSP?

```
Content-Security-Policy: script-src
https://code.jquery.com/jquery-3.6.0.js 'unsafe-inline'
```

Malicious Input

- **Which of the following can be used by attacker to inject arbitrary input to a web server?**

1. Visible form fields
2. Hidden form fields
3. GET parameters
4. Cookies
5. HTTP headers

Malicious Input

- **Which of the following can be used by attacker to inject arbitrary input to a web server?**

1. Visible form fields
2. Hidden form fields
3. GET parameters
4. Cookies
5. HTTP headers

SQLi

- **There is a server that interacts with a database using user input:**

```
mysql_query("SELECT * FROM users WHERE name='".$_GET["name"]."'
AND password='".$_GET["password"]."'")
```

If we enter: `example.com/login?name=lol&password=secret`
We have `SELECT * FROM users WHERE name='lol' AND password='secret'`

Attack by using: `example.com/login?name=' OR TRUE #&password=secret`
We have: `SELECT * FROM users WHERE name='' OR TRUE #' AND password='secret'`

Is filtering ' from user inputs good fix?

SQLi

- **No, it is not a good fix.**

  We can do: `example.com/login?name=\&password= OR TRUE #`

  We will have: `SELECT * FROM users WHERE name='\' AND password=' OR TRUE #'`

Fix blind SQL

- **A student proposes to fix blind SQL injections by fixing the amount of time taken by query to provide results. Does this fix blind injections? What is still possible?**

1. Yes
2. No

Fix blind SQL

- **A student proposes to fix blind SQL injections by fixing the amount of time taken by query to provide results. Does this fix blind injections? What is still possible?**

1. Yes
2. No

It might be difficult to infer information using timing based attacks but applications might still leak information by the means of:

1. Different error codes
2. Metadata about the response

Demo Time

- **Client side reflected XSS using fragments.**
  - Use file: secret_location_ex1.html
  - Walkthrough: ex1_walkthrough.txt

Demo Time

- **Client side persistent XSS using local storage.**
  - Use file: cat_viewer_ex2.html

    cat_loader_ex2.html

  - Walkthrough: ex2_walkthrough.txt

Feedback Form

- Would like to see anything different?
- Liked it, hated it, something can be improved?

- Link: https://forms.gle/JqMrDToQgf1Utyp16