2-4-21

OWASP TOP 10

1.) Injection

↳ sql injection

↳ Command injection (in THM) — Active (results are displayed to attacker) / passive (result doesn't displayed to attacker)

2.) Broken authentication

Here in THM, Attacker is able to create a another account as the same name of victim by adding space before it (i.e) "edith" as " edith". Then, the attacker can able to access real victim account & access to his content.

(space)

3.) Sensitive data exposure

↳ In THM, db is stored in the web directory itself under /assets which can be downloaded & viewed.

4.) XXE

↳ External xml injection

↳ types

- in-bound XXE
- out-of-bound XXE

xml :

↳ xml documents mostly starts with xml prolog.

<?xml version = "1.0" encoding = "UTF-8" ?>

↳ xml is used widely coz data in xml can be easily transported

↳ Xml allows validation using DTD & Schema. It helps in avoiding Syntax error.

↳ Every xml document must have root element.

↳ We can use attributes in xml also :

eg:) <note type = "test"> this is simple test note </note>

(type: attribute; "test": value)

DTD:

- Document type definition
- It defines structure of an xml document.

Let us try to understand this with the help of an example. Say we have a file named note.dtd with the following content:

DTD (note.dtd) ←

```
<!DOCTYPE note [ <!ELEMENT note (to,from,heading,body)> <!ELEMENT to (#PCDATA)> <!ELEMENT from (#PCDATA)>
 <!ELEMENT heading (#PCDATA)> <!ELEMENT body (#PCDATA)> ]>
```

Now we can use this DTD to validate the information of some XML document and make sure that the XML file conforms to the rules of that DTD. Ex: Below is given an XML document that uses note.dtd

actual xml document ←

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE note SYSTEM "note.dtd">
<note>
   <to>falcon</to>
   <from>feast</from>
   <heading>hacking</heading>
   <body>XXE attack</body>
</note>
```

So now let's understand how that DTD validates the XML. Here's what all those terms used in note.dtd mean

!DOCTYPE note - Defines a root element of the document named note
!ELEMENT note - Defines that the note element must contain the elements: "to, from, heading, body"
!ELEMENT to - Defines the to element to be of type "#PCDATA"
!ELEMENT from - Defines the from element to be of type "#PCDATA"
!ELEMENT heading - Defines the heading element to be of type "#PCDATA"
!ELEMENT body - Defines the body element to be of type "#PCDATA"

NOTE: #PCDATA means parseable character data.

5) Broken Access Control

Websites have pages that are protected from regular visitors, for example only the site's admin user should be able to access a page to manage other users. If a website visitor is able to access the protected page/pages that they are not authorised to view, the access controls are broken.

- force browsing to unauthorized webpages
  - IDOR This are comes under Broken access control.

6) Security Misconfiguration :

It includes,

↳ Poorly configured permissions on cloud services, like S3 buckets

↳ Having unnecessary features enabled, like services, pages, accounts or privileges

↳ Default accounts with unchanged passwords

↳ Error messages that are overly detailed and allow an attacker to find out more about the system

↳ Not using HTTP security headers, or revealing too much detail in the Server: HTTP header

7.) XSS

- Popup's (<script>alert("Hello World")</script>) - Creates a Hello World message popup on a users browser.
- Writing HTML (document.write) - Override the website's HTML to add your own (essentially defacing the entire page).
- XSS Keylogger (http://www.xss-payloads.com/payloads/scripts/simplekeylogger.js.html) - You can log all keystrokes of a user, capturing their password and other sensitive information they type into the webpage.
- Port scanning (http://www.xss-payloads.com/payloads/scripts/portscanapi.js.html) - A mini local port scanner (more information on this is covered in the TryHackMe XSS room).

Note:
XSS-Payloads.com (http://www.xss-payloads.com/) is a website that has XSS related Payloads, Tools, Documentation and more. You can download XSS payloads that take snapshots from a webcam or even get a more capable port and network scanner.

<script>window.location.hostname</script> → gives your ip-addr/hostname.

</script>document.cookies</script> → gives the cookies of HTTP-only flag is not enabled.

<script>document.querySelector('#thm-title').textContent = 'I am a hacker'</script>

Select the element by id → get content

8.) Insecure Deserialization: → Explained at the end.

9.) Components with known Vulnerability

↳ Sometimes the programs used in server may have known vulnerabilities.

↳ with some amount of research, find if there are any vulnerabilities are there for the particular versions & it can be exploited. Exploit may also available on internet in most cases. (eg:) exploit DB.

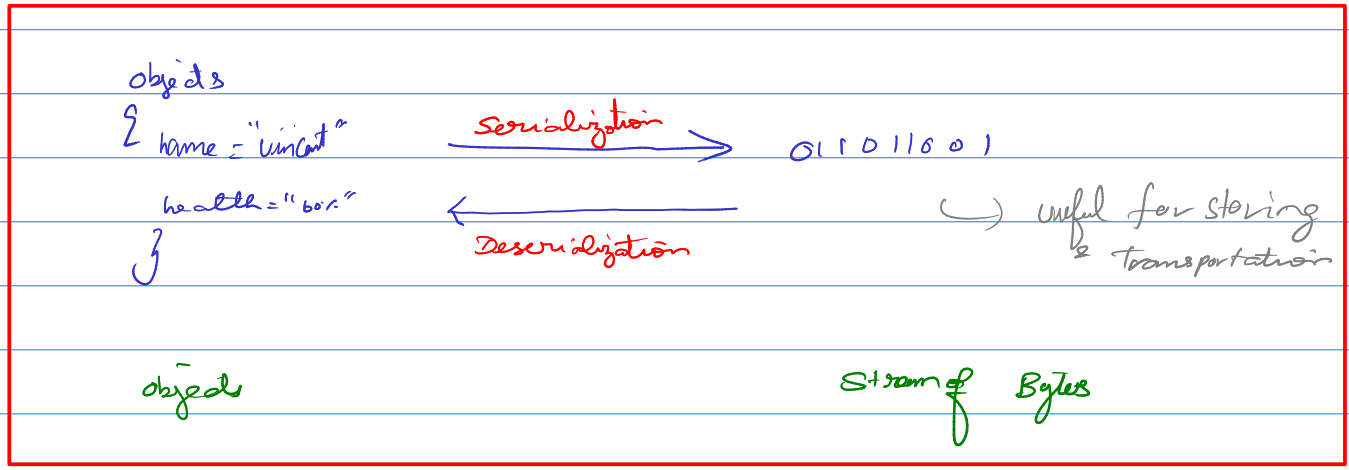
10.) Insufficient Logging & Monitoring:

↳ If there is no logging mechanism enabled, then it is easy for attacker to attack & destroy.

↳ No way to find attacker, if no logging is enabled.

↳ It also helps in preventing attacks by detecting them.

8.) Insecure Deserialization:



Insecure deserialization:

↳ Usually in browser, cookie is saved by the web server to get the state of the user.

↳ Thus cookies are nothing but object which is saved in browser as stream of bits in files.

↳ When cookies are transferred to server, server deserializing the stream of bits into object.

Attacker uses this scenario as, he made a crafted cookie and save it in browser, when server asks for it, malicious crafted cookie is transfered, At server side it is deserialized and the arbitary command is executed at the server side.

Nice Explanation : https://www.youtube.com/watch?v=jwzeJU_62IQ

Cookies : → It can be set using set-cookie method (method name & syntax may vary on different Languages.)

Cookie name, Cookie value, Secure only, Expiry, Path

- Cookie name ← name of cookie to set
- Cookie value ← value of cookie to set
- Secure only ← If it is set then, cookie will only sent over HTTPS
- Expiry ← time at which cookie is removed from browser
- Path ← cookie will only set if specific path (URL) is set

Pickle :-

↳ python library for serialization & deserialization

we use that 3 methods

- pickle.dump(value) → It serializes the value
- Pickle.load(Serialized value) → It deserializes the given value
- __reduce__ → It is called when object is serialized/dumped.

(less amount of further explanation in video)

↳ The Actual Scenario is, in web app the cookie is encoded (base64 in THM eg.) & saved in browser side (i.e) serialized & it can be sent to Server if needed.

↳ Server side cookie is deserialized. (In THM eg.) to exploit that vulnerability we encode of payload after serialization using dump method from pickle and then we set the result value as cookie.

Cookie we set has arbitrary commands like netcat which helps us to get reverse shell.

If server doesn't check the trust of the cookie & deserializes it. This is the problem Here. So that we get the shell by exploiting this.