CHANDIGARH UNIVERSITY

Discover. Learn. Empower.

INSTITUTE : UIE

DEPARTMENT : CSE

Bachelor of Engineering (Computer Science & Engineering)

WEB AND MOBILE SECURITY (Professional Elective-I)

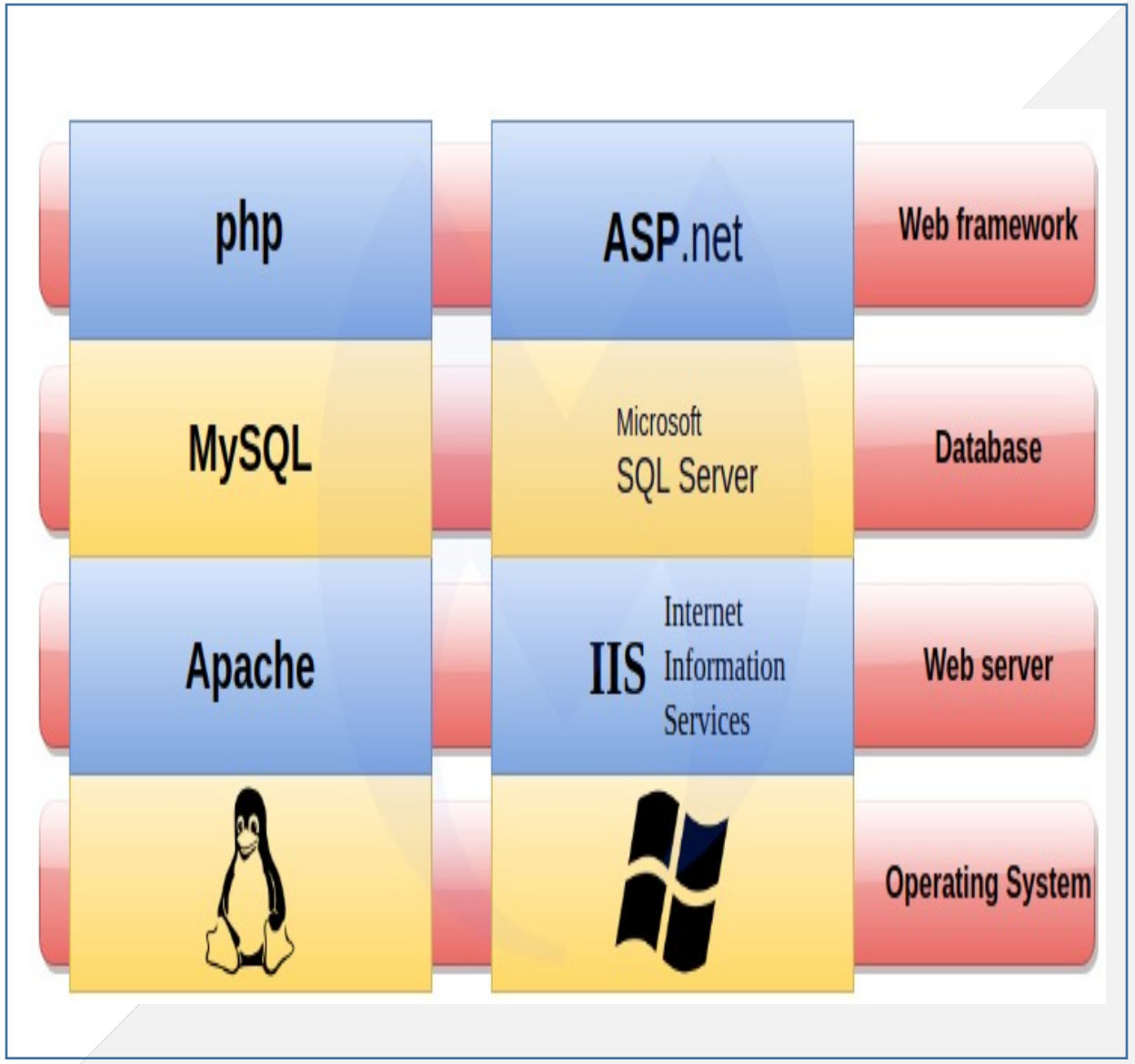
(20CST/IT-333)

TOPIC OF PRESENTATION:

Attacks, detection evasion techniques, and countermeasures for the most popular web platforms, including IIS, Apache, PHP, and ASP.NET.

Lecture Objectives

In this lecture, we will discuss:
Web platforms: attacks and solutions in PHP, and ASP.NET



Most common types of ASP.NET attack

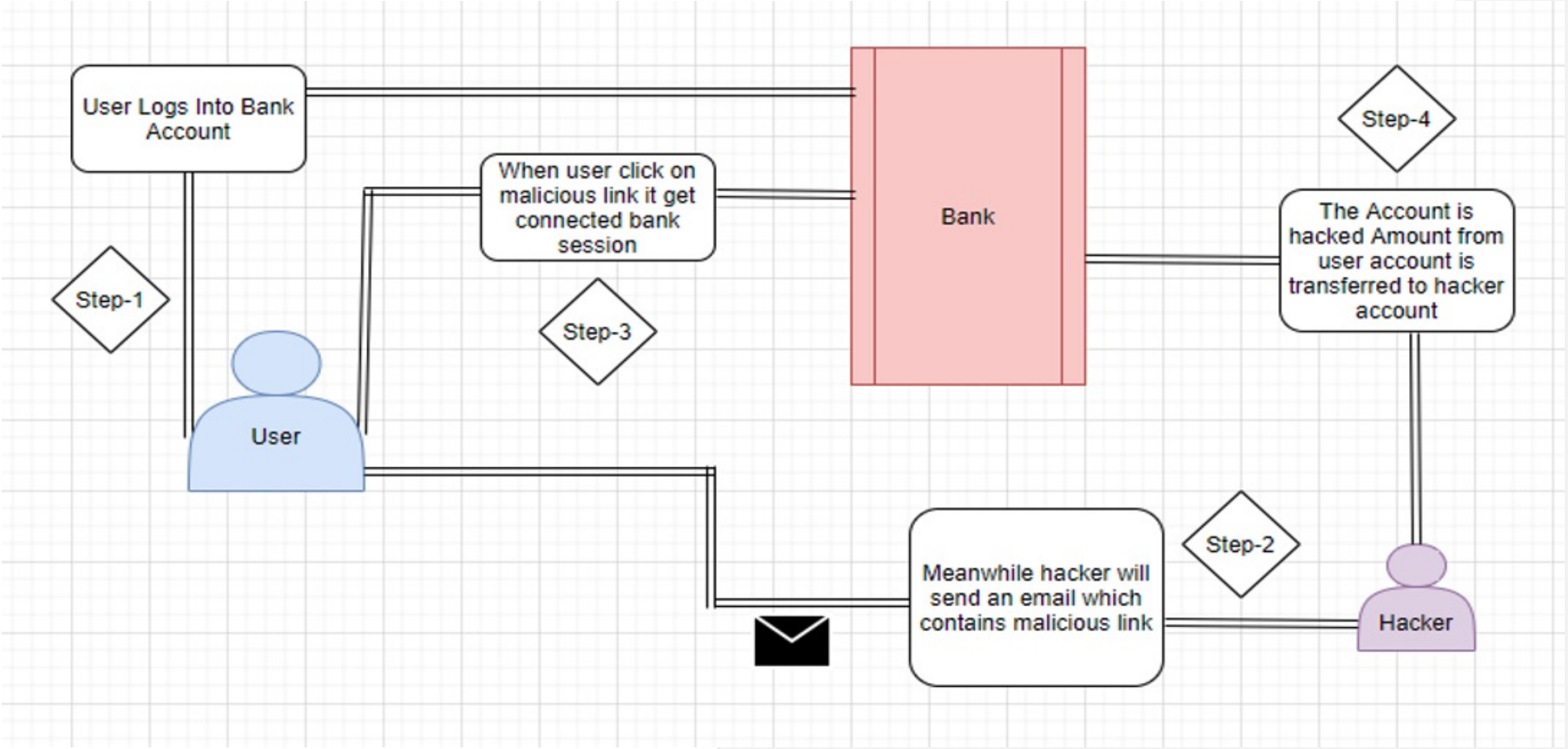
1. Security Misconfiguration (Error Handling Must Setup Custom Error Page)

- In this kind of attack, the attacker intercepts form data submitted by the end-user, changes its values and sends the modified data to the server. When the validations display errors, a lot of information on the server is subsequently revealed.

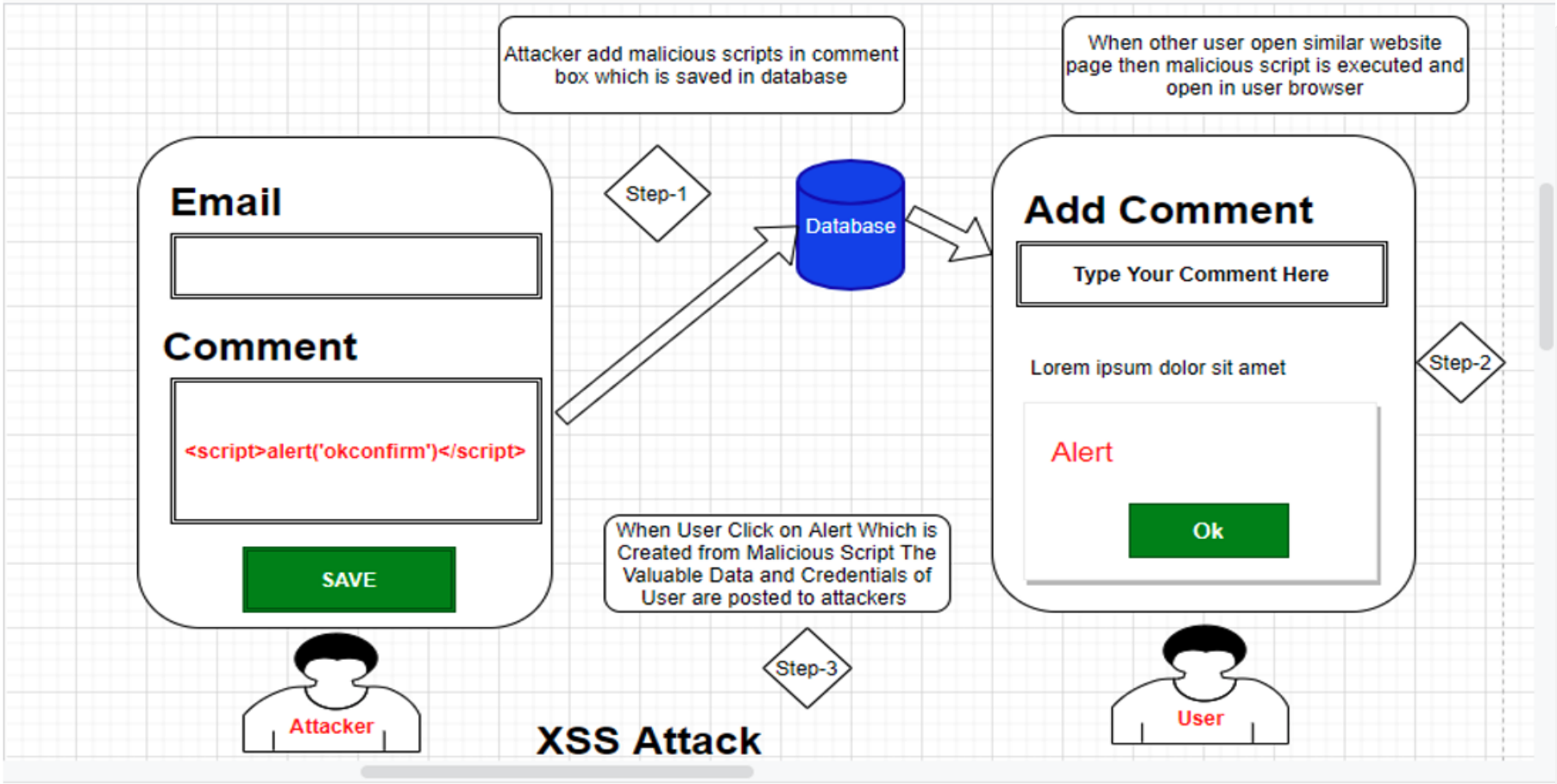
https://www.business2community.com/cybersecurity/9-ways-hackers-exploit-asp-net-and-how-to-prevent-them-02353604

2. CSRF attack

- Microsoft recognized this threat and we now have something called **AntiForgeryToken** to prevent similar attacks.

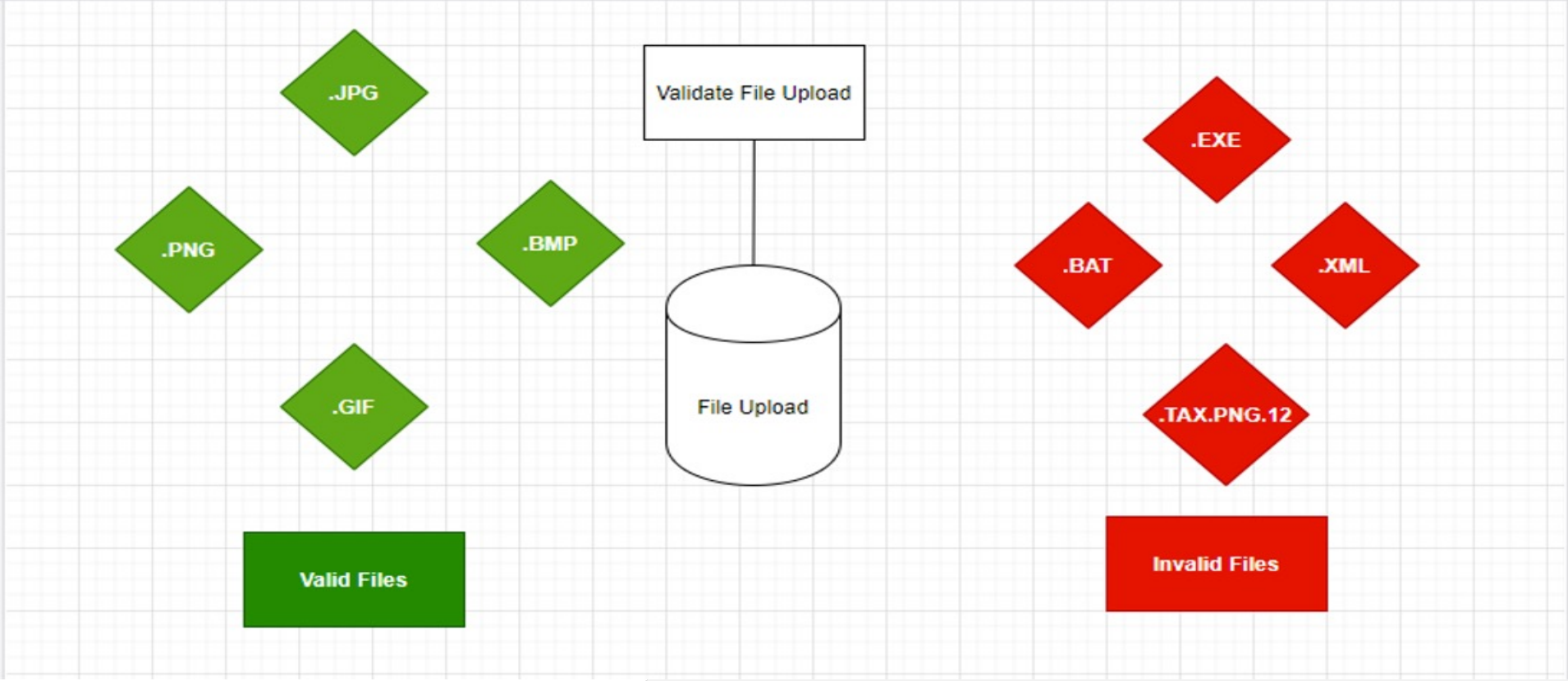


3. CSS


Attacker add malicious scripts in comment box which is saved in database
When other user open similar website page then malicious script is executed and open in user browser
Email
Comment
<script>alert('okconfirm')</script>
SAVE
Step-1
Database
Add Comment
Type Your Comment Here
Lorem ipsum dolor sit amet
Alert
Ok
Step-2
When User Click on Alert Which is Created from Malicious Script The Valuable Data and Credentials of User are posted to attackers
Step-3
Attacker
User
XSS Attack

4. Malicious File Upload

- We've already learned how to protect input fields from malicious attacks, but we haven't looked at file uploads. Attackers can change file extensions (tuto**.exe** to tuto**.jpeg**, for example) and the malicious script can be uploaded as an image file.
- It is easy to get duped this way. The lesson here for developers is to remain alert to file extensions at all times.



5. Version Disclosure

- Version information can be used by an attacker to help plan their next move. Whenever a browser sends an HTTP request to the server, in response we get a response header which contains all of the server information:
- “X-AspNet-Version” shows information on which specific Asp.Net Version Used.
- “X-AspNet Mvc-Version” shows information on which ASP.NET MVC version Used.
- “X-Powered-By” shows information on which framework your website is running on.

6. SQL Injection Attack

- An SQL injection attack is one of the most dangerous attacks, ranked first in a list of the top ten vulnerabilities as outlined by OWASP2013 [Open Web Application Security Project]. An SQL injection attack can give valuable data to the attacker that can lead to a big security breach and can also grant full access to the database server.

7. Sensitive Data Exposure

- All websites and applications have a database in which everything is stored, including user data like passwords, PAN numbers, credit card information etc. While encryption of all of this data is possible, we tend to only encrypt passwords, which can leave sensitive data exposed and vulnerable.

8. Broken Authentication and Session Management

- If authentication and session management are not properly implemented in a web application, it will leave it vulnerable to attack. Attackers can steal data due to:
- Unsecured connections (not using SSL)
- Predictable log-in credentials
- Not storing credentials in an encrypted form
- Improper application log-out

https://www.business2community.com/cybersecurity/9-ways-hackers-exploit-asp-net-and-how-to-prevent-them-02353604

9. Un-validated Redirects and Forwards

- In all web applications, we redirect from one page to another. It's also common to redirect to another application entirely. But if we don't validate those redirects, it can cause unvalidated redirects and the risk of a forwards attack. This attack is mostly used to phish the user for valuable credentials or install malicious malware.
- For example:
- **Original URL:** –http://localhost:7426/Account/Login
- **Crafter URL by Attacker:** –?return URL=https://www.google.co.in
- In this attack, the user gets an email from the attacker containing a tempting offer on an online shop, such as **http://demo.com**. However, upon closer inspection, we can see that the URL actually contains a redirect: **http://demo.com/Login/Login?url=http://mailicious.com**. If the user enters their valid credentials they'll be redirected back to the shopping website and won't suspect anything is wrong, but their details will have been stolen in an instant.

Countermeasueres

- **Input validation**

Identifying the validation needs for type, length, format or range of input data avoids hacker discovering the vulnerability in your application. An attacker can compromise your application if any such vulnerability is identified. It is a must to validate the input before processing it by your application. So, how do you know that your application is safe enough? You just need to answer whether your application trusts the input blindly. If the answer is yes, may be your application is susceptible for following threats.

- Buffer Overflow
- Cross-site scripting
- SQL injection
- Canonicalization

https://www.c-sharpcorner.com/article/securing-your-Asp-Net-web-applications/

PHP vulnerabilities and solutions

- **Why PHP Security Matters**
- The **massive PHP install base and an active development community drew security researchers and malicious actors looking for systems to exploit**. And indeed, as recent research demonstrates, **many PHP applications suffer from vulnerabilities** due to bad design and lackluster understanding of basic security practices required to secure a web application.

https://spectralops.io/blog/top-7-php-security-issues-and-vulnerabilities/

PHP Security Issues And Vulnerabilities & How To Avoid Them

- **1. SQL Injection**
- SQL is a language used to interact with databases to store and retrieve information. For example, when you fill a website signup form, the web server reads the data. The web server may relay the request to a backend database to retrieve specific information.
- **SQL injection works by fooling the server-side code into pushing an unsanitized SQL command to the database**. Such an SQL query can return information that the original web application developer did not intend to expose. The exposed information can include the entire content of the database, private information, passwords, etc.
- Preventing SQL injections is relatively simple. It requires that your code use parameterized queries and prepared statements instead of plain-text queries with no input sanitation.

2. XSS (Cross-Site Scripting)

- Cross-site scripting takes place when a malicious actor plants a script within your website to execute code from a remote site when a user visits your website.
- To eliminate XSS as a threat requires sanitizing every input your web application accepts. Sanitation is not limited to stored data but also output data such as a search result URL printing the URL's content unsanitized in a web page's body section, allowing a hacker to craft a unique link that initiates an XSS attack when clicked.

3. **Cross-Site request forgery works through social engineering and phishing tactics to exploit that you are already logged in and authenticated with a secure website**. A CSRF attack tries to convince you to open a specially crafted link that can invisibly steal session information and automatically send legitimate-looking commands to the server on your behalf.

- The recommended approach to blocking CSRF attempts is to use authentication tokens that get validated on sensitive operations. A new token is generated on each login and saved in a local cookie, making it very difficult for hackers to perform an unauthorized action on your web application without having prior access to the token.

4. Authentication Bypass

- Authentication bypass stems from developer error where an app does not validate a user's credentials correctly and inadvertently gives the user elevated access.

5. PHP Object Injection

- **PHP Object Injection exploits the fact that calling the unserialize() function with unsanitized user input can be used to inject PHP objects into memory**. Such objects can be triggered in an application that implements a PHP magic method such as __wakeup or __destruct in one of its classes.

6. Session Hijacking

- Whenever you login, the website creates a ‘session’ to maintain your logged-in status without repeatedly requesting authentication on every subsequent action. An XSS or CSRF attack can allow a hacker to copy current session information and use your web application, bypassing the login requirement or any other form of authentication.

7. Stream Injection Attacks (Local/Remote File Inclusion)

- **Stream injection attacks abuse the ability of websites to accept uploaded content such as documents and images.** Using remote file inclusion, a hacker tries to fool your PHP code into accepting a URL on another site as valid input. This action can then be used to execute malicious code. Local file inclusion can be used to get a web application to return the content of a local file.

References:

Books:

1. Hacking Exposed Mobile: Security Secrets & Solutions 1st Edition, Kindle Edition, by Neil Bergman, Mike Stanfield, Jason Rouse, and Joel Scambray
2. Hacking Exposed Web Applications, 3rd edition, Joel Scambray, Vincent Liu, Caleb Sima, Released October 2010, Publisher(s): McGraw-Hill

Relevant Videos:

https://www.youtube.com/watch?v=qz7t95qIc24

https://www.youtube.com/watch?v=wgkp6smOzi8

Reference Links:

https://spectralops.io/blog/top-7-php-security-issues-and-vulnerabilities/

https://www.it.iitb.ac.in/~madhumita/web%20services/Web%20Security%20Threats%20and%20Countermeasu_files/rightframe.htm

https://blog.securityinnovation.com/php-security-the-good-the-bad-and-the-ugly

https://www.c-sharpcorner.com/article/securing-your-Asp-Net-web-applications/

https://www.it.iitb.ac.in/~madhumita/web%20services/Web%20Security%20Threats%20and%20Countermeasu_files/rightframe.htm

THANK YOU