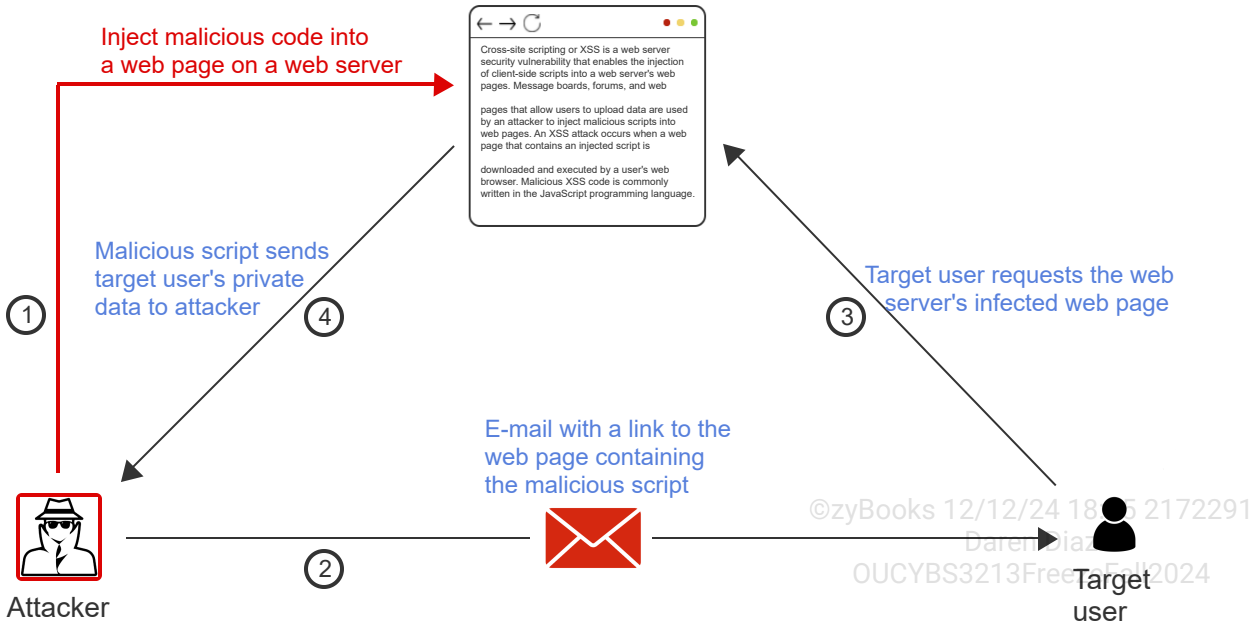
7.1 Cross-site scripting

Cross-site scripting

Cross-site scripting or ***XSS*** is a web server security vulnerability that enables the injection of client-side scripts into a web server's web pages. Message boards, online forums, and web pages that allow users to upload data can be used by an attacker to inject malicious scripts into web pages. An XSS attack occurs when a web page that contains an injected script is downloaded and executed by a web browser. The injected script can access session tokens, cookies, or other sensitive information retained by a browser for the website hosting the web page. Malicious XSS code is commonly written in the JavaScript programming language.

PARTICIPATION ACTIVITY 7.1.1: Cross-site scripting.



Animation content:

Static image: An attacker, a browser showing a webpage, a target user, and an email icon. Arrows are used to show steps. Step 1 is a red arrow labeled "Inject malicious code into a webpage on a web server" that points from the attacker to the browser. Step 2 is a black arrow labeled "E-mail with a link to the webpage containing the malicious script" that goes from the attacker, through the email icon, and to the target user. Step 3 is a black arrow labeled "Target user requests the web server's infected webpage" that points from the target user to the browser. Step 4 is a black arrow labeled "Malicious script sends target user's private data to attacker" that points from the browser to the attacker.

Animation captions:

1. An attacker finds a website with a vulnerability that enables execution of malicious scripts. The attacker uploads a malicious script to a web page on the web server.
2. The attacker sends an e-mail to a target user which contains a link to the script-injected web page on the web server.
3. The target user clicks on the link and loads the infected web page. The target user's browser executes the malicious code injected into the web page by the attacker.
4. The malicious script sends the target user's private data to the attacker.

PARTICIPATION ACTIVITY 7.1.2: Cross-site scripting.

1) XSS enables the injection of client-side scripts into a web server's web pages.
   - True
   - False

2) An XSS attack occurs when a web browser executes malicious scripts in a web page.
   - True
   - False

3) An attacker may send an email to a target user that contains a link to the infected web page.
   - True
   - False

Cross-site scripting attack types

Three cross-site scripting attack types exist:

- A ***reflected cross-site scripting***, also known as ***non-persistent cross-site scripting*** or ***reflected XSS***, is a cross-site scripting attack in which an injected script is returned to a client in response to an HTTP request by the client. An injected script is activated through a link that sends a request to a website with a vulnerability that enables execution of injected scripts. An attacker may use various methods to trick a user into clicking a maliciously crafted link, such as placing shortened URLs on an attacker-controlled website or sending the user a phishing email.
- A ***stored cross-site scripting***, also known as ***persistent cross-site scripting*** or ***stored XSS***, is a cross-site scripting attack in which a malicious script is stored on a web server's back-end database and returned to all clients that request the web pages containing the script. A stored XSS attack is self-contained within a web application and does not require an attacker to induce a user into making an HTTP request to a web page.
- A ***DOM-based cross-site scripting***, or ***DOM-based XSS***, is a cross-site scripting attack in which the source and destination of the attack is the client web browser. ***Document object model (DOM)*** is a standard object model and programming interface for HTML. DOM defines how to get, change, add, or delete HTML elements. In a DOM-based XSS attack, malicious code is inserted into DOM objects and executed by a client browser without being sent to a web server.

Table 7.1.1: Cross-site scripting attacks.

XSS type	Attack source	Vulnerability source
Reflected	client	server and client
Stored	server	server
DOM-based	client	client (DOM)

PARTICIPATION ACTIVITY 7.1.3: Cross-site scripting types.

How to use this tool

DOM-based cross-site scripting **Reflected cross-site scripting**

Stored cross-site scripting

	A cross-site scripting attack in which the injected script is returned to a client in a response to an HTTP request by the client.
	A cross-site scripting attack in which a malicious script is stored on a web server's back-end database and returned to all clients that request a page which contains the script.
	A cross-site scripting attack in which the source and destination of the attack is the client web browser.

Reset

PARTICIPATION ACTIVITY 7.1.4: Cross-site scripting attack types.

Select the cross-site scripting attack type used in each scenario.

1) A user is tricked into clicking on a link that takes the user to a web page containing an injected script.
   - ○ reflected
   - ○ stored
   - ○ DOM-based

2) All the users that visit a specific web page download and run an injected script.
   - ○ reflected
   - ○ stored
   - ○ DOM-based

3) An injected script in run by a browser without the involvement of a web server.
   - ○ reflected
   - ○ stored

- DOM-based

4) An attacker stores malicious script on a back-end database used to store web pages by a web server.
   - reflected
   - stored
   - DOM-based

CHALLENGE ACTIVITY 7.1.1: Cross-site scripting.

581480.4344582.qx3zqy7

Start

Select the term that best completes the sentence.

Pick An XSS attack occurs when a web page containing an injected script is exe

Pick An attacker sends an e-mail to a target user which contains a link to a scrip

Pick In an XSS attack, a user requests a script-injected web page on a _____.

1 2

Check Next

7.2 Cross-site and server-side request forgeries (CSRF and SSRF)

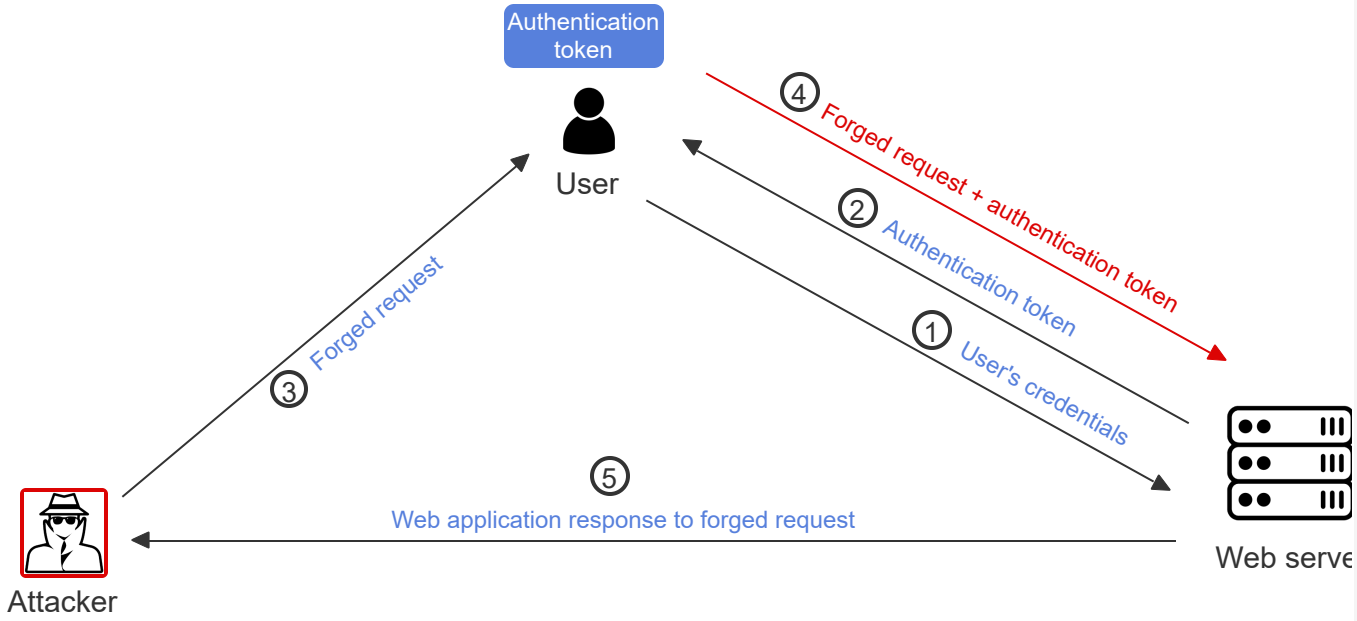
Request forgeries

A request to a web application can be forged to trick the web application into performing an unintended action. Two types of request forgeries exist: cross-site request forgery (CSRF) and server-side request forgery (SSRF).

Cross-site request forgery (CSRF)

A ***cross-site request forgery (CSRF)***, also known as ***client-side request forgery*** or ***XSRF***, is an attack in which an attacker sends a forged request to a web application on behalf of an authenticated user. A CSRF attack exploits a web application's trust in an authenticated user. Ex: A banking web application stores an authentication token on a user's device when the user authenticates to the banking web application. If the user does not logout from the banking web application, the authentication token remains on the user's device and can be used by an attacker to perform a banking transaction on behalf of the user and without the user's knowledge.

PARTICIPATION ACTIVITY 7.2.1: Cross-site request forgery (CSRF).



Animation content:

Static image: An attacker, a user, and a web server. The text "Authentication token" is above the user. Arrows are used to show steps. Step 1 is a black arrow labeled "User's credentials" that points from the user to the web server. Step 2 is a black arrow labeled "Authentication token" that points from the web server to the user. Step 3 is a black arrow labeled "Forged request" that goes from the attacker

to the user. Step 4 is a red arrow labeled "Forged request + authentication token" that goes from the user to the web server. Step 5 is a black arrow labeled "Web application response to forged request" that points from the web server to the attacker.

Animation captions:

1. A user authenticates to a web application. The web application stores an authentication token on the user's device.
2. An attacker sends a forged request to the user to exploit the web application's trust in the user. The forged request can be an embedded script or a URL.
3. User unknowingly sends the attacker's forged request to the web application. The authentication token is included with each request sent from the user to the web application.
4. Since the request is from an authenticated user, the web application responds to the forged

PARTICIPATION ACTIVITY 7.2.2: Cross-site request forgery (CSRF).

1) In a CSRF attack, a forged request is sent from _____ to a web application.
   - attacker
   - user
   - web server

2) What is the goal of a CSRF attack?
   - To prevent a user from authenticating to a web application
   - To send a malicious request from an authenticated user to a web application
   - To trick a user into authenticating to a web application

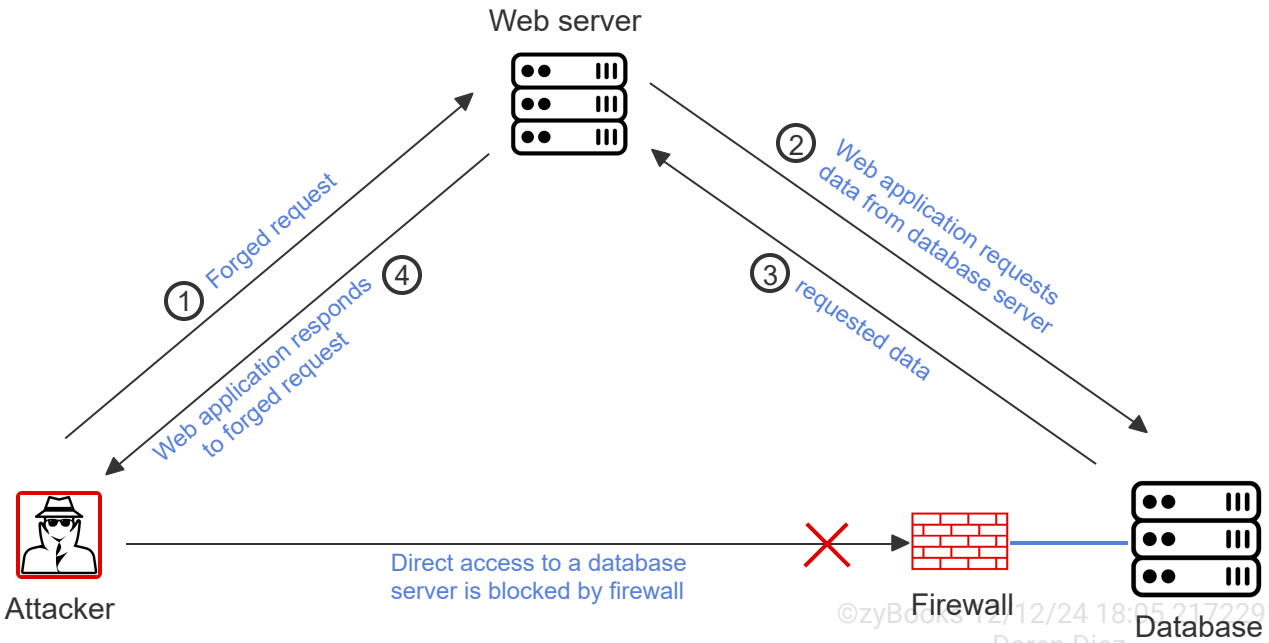
3) Why is a CSRF attack possible?
   - Because a web application fails to authenticate users
   - Because a web application trusts unauthenticated users

Because a web application cannot distinguish between
- legitimate requests and forged requests from an authenticated user

Server-side request forgery (SSRF)

A ***server-side request forgery (SSRF)*** is an attack in which an attacker sends forged requests from a web application's server to other servers. In an SSRF attack, an attacker tricks a server into making an unauthorized request. An SSRF attack uses a web server to access or modify internal resources that are protected from external networks. SSRF can also be used to access a web server's other available services through the server's loopback interface. Ex: An attacker sends a forged request to a web server to retrieve user account information from a back-end database server.

PARTICIPATION ACTIVITY 7.2.3: Server-side request forgery (SSRF).



Animation content:

Static image: An attacker, a web server, and a database server. A firewall is between the attacker and the database server. An arrow labeled "Direct access to a database server is blocked by firewall" points from the attacker to the firewall. Arrows are used to show steps. Step 1 is an arrow labeled "Forged request" that points from the attacker to the web server. Step 2 is an arrow labeled "Web application requests data from database server" that points from the web server to the database server. Step 3 is an arrow labeled "requested data" that points from the database server to the web server. Step 4 is an arrow labeled "Web application responds to forged request" that points from the web server to the attacker.

Animation captions:

1. A firewall protects back-end servers (such as a database server) that should not be accessible from an external network.
2. An attacker sends a forged request to a web application. To respond to the forged request, the web application retrieves data from the back-end database server.
3. Database server returns requested data to the web application. The web application responds to the attacker's forged request.

PARTICIPATION ACTIVITY 7.2.4: Server-side request forgery (SSRF).

1) What is the target of an SSRF attack?

- User
- Back-end server
- Network

2) In an SSRF attack, a forged request is sent to a _____.

- Web server
- Back-end server
- Web client

3) What is the aim of an SSRF attack?

- To trick a back-end server into performing an unauthorized action
- To trick a user into making an unauthorized request
- To trick a web server into making an authorized request

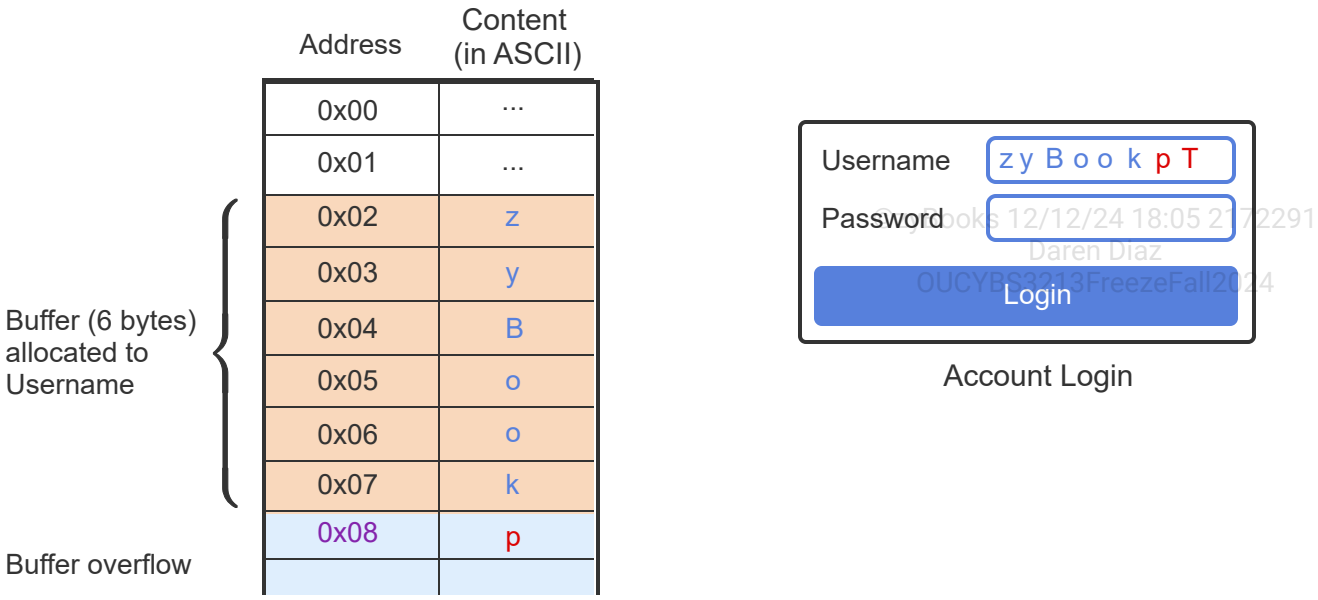
7.3 Memory vulnerabilities: Buffer and integer overflows, memory leak, and pointer dereferencing

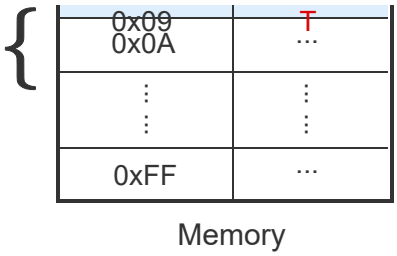
Buffer overflow

A ***data buffer***, or ***buffer***, is a memory region used for temporarily storing data while the data is being transferred from one memory location to another. Ex: A buffer is used to store a user's input to a program. A ***buffer overflow***, or ***buffer overrun***, is a programming error condition that occurs when the size of the data to be stored in a buffer is larger than the buffer size. If a buffer overflow occurs, data is written to a buffer's adjacent memory locations that are not allocated to the buffer.

Improper input handling is a program's insufficient validity checking of user input. User input validation checks should be performed to ensure the size of a user input is not larger than the buffer size allocated to store the user input. Access to memory locations beyond a buffer's bounds should also be prevented. A buffer overflow that results in overwriting executable code may cause a program to output incorrect results, generate memory access errors, or terminate unexpectedly.

PARTICIPATION ACTIVITY 7.3.1: Buffer overflow.





Memory

Animation content:

Static image: A box labeled "Account Login" with a username input field, a password input field, and a login button. The username input field contains "zyBookPT" with "zyBook" in blue font and "PT" in red font. A table titled "Memory" with columns "Address" and "Content (in ASCII)". The address column contains the addresses 0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08, 0x09, 0x0A,..., 0xFF. The rows with addresses 0x02 through 0x07 are highlighted orange and each contain one letter of "zyBook". A bracket appears to the left of the orange rows with the label "Buffer (6 bytes) allocated to Username". The rows with addresses 0x08 and 0x09 are highlighted in blue. 0x08 has content "p" in red font. 0x09 has content "T" in red font. A bracket appears to the left of the blue rows with the label "Buffer overflow".

Animation captions:

1. A 6-byte buffer is allocated to the Username field of an Account Login page.
2. Adequate buffer storage exists if user input in the Username field is 6 bytes or less.
3. User input from the Username field is stored in the buffer. Ex: zyBook is 6 bytes.
4. A buffer overflow occurs if user input in the Username field exceeds 6 bytes (zyBookpT is 8 bytes).

PARTICIPATION ACTIVITY 7.3.2: Buffer overflow.

1) A buffer overflow occurs when the size of user input is _____ the size of a buffer allocated to store that input.

   - ○ less than
   - ○ larger than
   - ○ equal to

2) How can buffer overflows be prevented?

   - ○ By preventing all user input

- By preventing access beyond a buffer's bounds
- By creating a buffer larger than user input

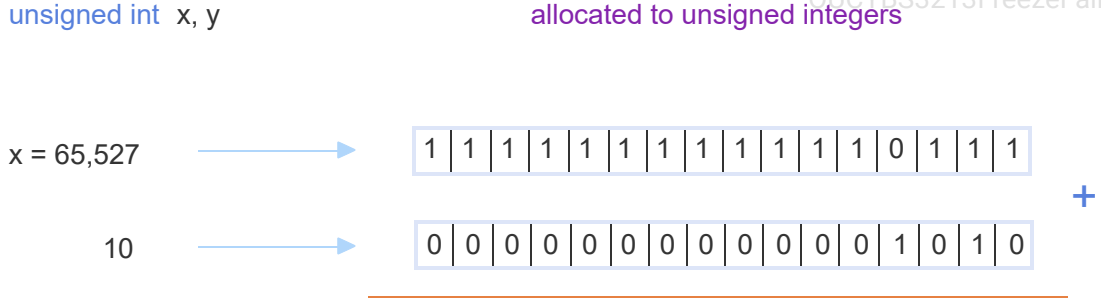
3) In the animation above, how many bytes are overwritten to the buffer's adjacent memory locations?
   - 2 bytes
   - 6 bytes
   - 8 bytes

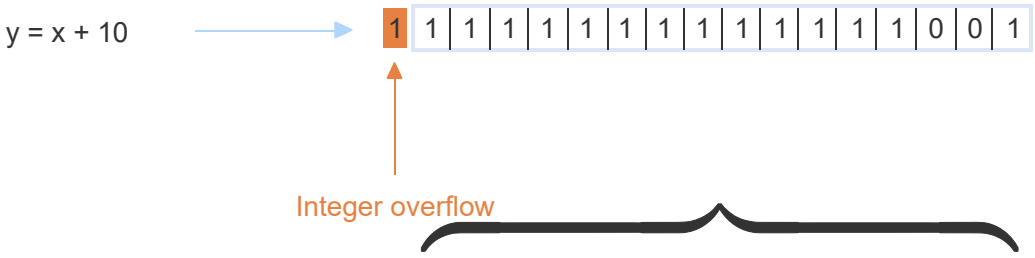
Integer overflow

A ***data type***, or ***type***, is a data attribute that specifies how the data is intended to be used in a program. Most programming languages support basic data types of floating-point numbers, integer numbers, characters, and Booleans. An ***integer overflow*** is a programming error condition that occurs when the result of an integer operation is greater than the maximum size, or less than the minimum size of the integer type used to store the result. Ex: A 16-bit variable of type integer can store integers ranging from -32,768 to 32,767. An integer overflow occurs when the result of an integer operation that is larger than 32,767 is stored in a 16-bit variable of type integer.

An integer overflow causes a program to perform incorrect computations. An integer overflow can be prevented by checking for an overflow after each integer operation where an overflowing value could potentially occur. In some programming languages such as Python, the memory space allocated to a variable is automatically adjusted at run-time to accommodate larger values that may result from integer operations.

PARTICIPATION ACTIVITY 7.3.3: Integer overflow.





Animation content:

Static image: The text "unsigned int x, y" in the top left corner. The text "x = 65,527" has an arrow pointing to the right to a row of 16 boxes representing 16 bits. Each box contains one bit of the binary representation for x, 1111111111110111. A bracket above the row of boxes has the label "16-bit memory blocks are allocated to unsigned integers". The text "10" is below "x = 65,527" with an arrow pointing to a second row of 16 boxes. The second row of boxes contains the binary representation for 10, 0000000000001010. A plus sign is between the first and second rows of boxes. The text "y = x + 10" is below "10" with an arrow pointing to a row of 17 boxes. The row of boxes contains the binary representation for 65,537, 10000000000000010. The first box is highlighted orange and has the label "Integer overflow".

Animation captions:

1. Variables x and y are declared as unsigned integers. Unsigned integers are allocated 16-bit memory blocks and can store integers ranging from 0 to 65,535.
2. The unsigned integer x is assigned 65,527 and stored in a 16-bit memory block.
3. Adding 10 to x results in 65,537, which is larger than 65,535 and cannot be stored in a 16-bit memory block. An integer overflow occurs.

PARTICIPATION ACTIVITY 7.3.4: Integer overflow.

1) An integer overflow occurs when the result of an integer operation is _____ the maximum size of the integer type used to store the result.

- less than
- larger than
- equal to

2) How can an integer overflow be prevented?

- By not using integer operations
- By using a programming language that does not increase the memory allocated to variables at run-time
- By checking for an overflow after each integer operation where an overflowing value could potentially occur

3) An integer overflow that exceeds the maximum representable value may cause undefined program behavior because _____.
   - adjacent memory locations are corrupted
   - the result goes back (wraps) to the minimum representable value
   - the result may be used in a subsequent operation

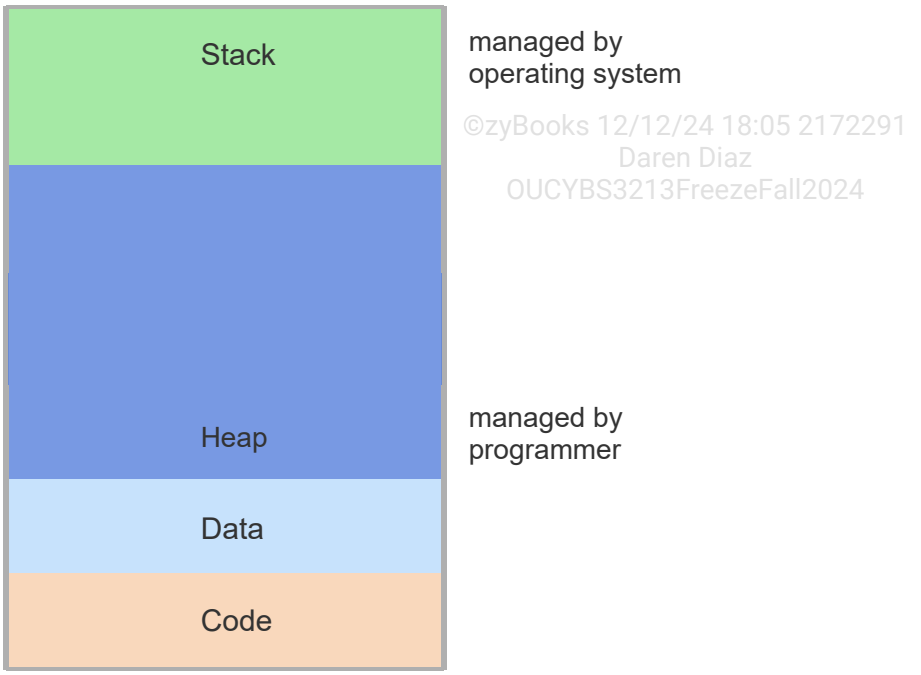
Memory leak

Memory management is the process of controlling and coordinating a computer's memory. An operating system allocates two types of memory to a program:

- ***Stack***, also known as ***temporary memory***, is memory that stores local variables and is used for static memory allocation. The allocation and deallocation of stack memory is managed by an operating system.
- ***Heap***, also known as ***dynamic memory***, is memory that stores global variables and is used for dynamic memory allocation. The allocation and deallocation of heap memory is managed by a programmer.

Heap memory is not automatically deallocated. A ***memory leak*** is an occurrence in which a program allocates heap memory but does not deallocate the memory when the memory is no longer needed. A memory leak is caused by improper memory management by a programmer. A memory leak negatively impacts a computer's performance by reducing the computer's available memory.

PARTICIPATION ACTIVITY 7.3.5: Memory leak.



Animation content:

Static image: A rectangle divided into four sections. The bottom section is tan and labeled "Code". The second section is light blue and labeled "Data". The third section is blue and labeled "Heap". The top section is green and labeled "Stack". The text "managed by programmer" is next to the heap section. The text "managed by operating system" is next to the stack section.

Animation captions:

1. An operating system allocates memory to a program's code, data, stack, heap, and memory to be used by the program during execution.
2. Allocation and deallocation of stack is managed by the operating system, and allocation and deallocation of heap is managed by the programmer.
3. Both stack and heap are allocated at program execution time. A memory leak may occur when a programmer does not deallocate memory that is no longer needed.

PARTICIPATION ACTIVITY 7.3.6: Memory leak.

How to use this tool

Heap | Memory leak | Stack

	Memory that stores local variables and is used for static memory allocation.
	Memory that stores global variables and is used for dynamic memory allocation.
	An occurrence in which a program allocates heap memory, but does not deallocate the memory when no longer needed.

Reset

PARTICIPATION ACTIVITY 7.3.7: Memory leak.

Select the memory type in each statement.

1) _____ is managed by programmer.
   - Stack
   - Heap

2) _____ is used for static memory allocation.
   - Stack
   - Heap

3) _____ may cause a memory leak.
   - Stack
   - Heap

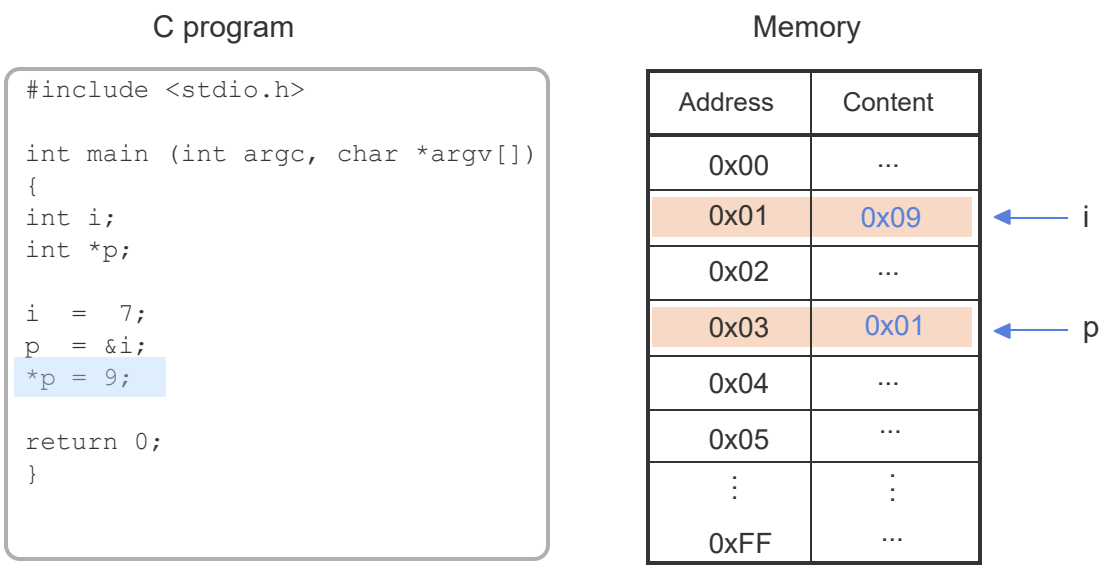
4) _____ stores local variables.
   - Stack
   - Heap

Pointer dereferencing

A program variable is used to store information that is referenced and manipulated in a program. A ***pointer*** is a program variable that stores a memory address. A pointer references a location in memory. A ***pointer dereferencing*** or ***object dereferencing*** is the act of obtaining the value stored at the memory location pointed to by the pointer. Pointers improve a computer's performance because copying and dereferencing pointers is faster than copying and accessing the data to which the pointers point.

A program stops behaving correctly and exits (crashes) if the program dereferences a pointer with an invalid memory address or a pointer that has a value of NULL. A program crash may result in a computer state that enables a user to gain unauthorized access to the computer's resources. ***Error handling*** is the process of responding and recovering from error conditions in a program. Error conditions that arise from invalid pointer dereferencing should be properly handled by a program.

PARTICIPATION ACTIVITY 7.3.8: Pointer dereferencing.



Animation content:

Static image: A box labeled "C program" with C code.
Begin C code:
#include <stdio.h>

int main (int argc, char *argv[])

```
{
int i;
int *p;

i = 7;
p = &i;
*p = 9;

return 0;
}
```


End C code.

The line "*p = 9;" is highlighted blue. A table titled "Memory" has columns "Address" and "Content". The row with address 0x01 is highlighted in orange and has content "0x09". A label "i" has an arrow pointing to the row with address 0x01. The row with address 0x03 is highlighted in orange and has content "0x01". A label "p" has an arrow pointing to the row with address 0x03.

Step 1: Memory is allocated to a variable of type integer (labeled i) and a variable of type integer pointer (labeled p).

A blue highlight appears on the line "int i;" and then the table row with address 0x01 is highlighted in orange with the label "i" pointing to the row. The blue highlight moves to the line "int *p;" and then the table row with address 0x03 is highlighted in orange with the labeled "p" pointing to the row.

Step 2: Integer 7 is stored in memory location 0x01 (labeled i).

The blue highlight moves to the line "i = 7;". The content "0x07" is added to the row with address 0x01.

Step 3: The address of memory location storing integer value 7 is stored in integer pointer p.

The blue highlight moves to the line "p = &". The text "0x01" is copied from the address column to the content column of the row with address 0x03.

Step 4: * is the dereference operator in the C language. *p dereferences p and *p = 9 stores integer 9 to the memory location pointed to by p.

The blue highlight is moved to the line "*p = 9". The content associated with the address 0x01 changes from "0x07" to "0x09".

Animation captions:

1. Memory is allocated to a variable of type integer (labeled i) and a variable of type integer pointer (labeled p).
2. Integer 7 is stored in memory location 0x01 (labeled i).
3. The address of memory location storing integer value 7 is stored in integer pointer p.
4. * is the dereference operator in the C language. *p dereferences p and *p = 9 stores integer 9 to the memory location pointed to by p.

PARTICIPATION ACTIVITY 7.3.9: Pointer dereferencing.

Refer to the above animation to answer the questions.

1) What is the value of variable *i* at the end of Step 2?

- 0x07
- 0x09
- 0x01

2) What is the result of dereferencing variable *p* at the end of Step 3?

- 0x07
- 0x09
- 0x01

3) What is the value of variable *p* at the end of Step 3?

- 0x07
- 0x09
- 0x01

4) What is the result of dereferencing variable *p* at the end of Step 4?

- 0x07
- 0x09
- 0x01

PARTICIPATION ACTIVITY 7.3.10: Pointer dereferencing.

How to use this tool

Pointer dereferencing **Error handling** **Pointer**

	A program variable that stores a memory address.

	The act of obtaining the value stored at the memory location pointed to by the pointer.
	The process of responding and recovering from error conditions in a program.

Reset

Memory injection

Memory injection *is a security vulnerability involing the manipulation of an application's memory to execute malicious code, gain unauthorized access, extract sensitive data, or disrupt behavior. Ex: Injecting code into a web application's memory could allow access to sensitive user data or control over the application's functionality.*

CHALLENGE ACTIVITY | 7.3.1: Memory vulnerabilities.

581480.4344582.qx3zqy7

Start

Select the overflow type described in each statement.

Pick — Occurs when the size of data types do not change based on computation results at run-time

Pick — An overflow condition in which data is written to an allocated memory region's adjacent memory locations

Pick — Occurs because a program does not validate user input

Check Next

7.4 Replay attack and pass the hash

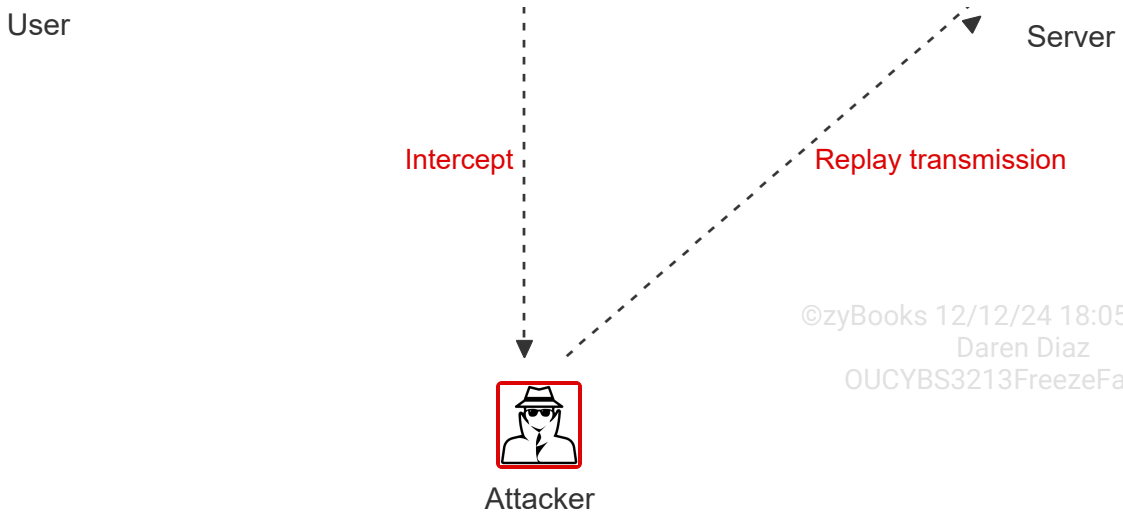
Replay attack

A ***replay attack***, also known as a ***repeat attack*** or ***playback attack***, is a network attack in which a data transmission is intercepted and maliciously repeated or delayed. A replay attack is a type of man-in-the-middle attack. Ex: A replay attack occurs when a user's login credentials sent to a system are intercepted and used by an attacker to authenticate to the system at a later time.

A replay attack is possible because a message receiver does not distinguish between identical messages received at different times. A replay attack can be prevented if a message sender appends a timestamp or sequence number to each message. The message receiver would then discard any message with a repeated timestamp or sequence number.

A ***session ID*** is a unique number assigned to an authenticated user by a web application. A session ID is valid for the duration of the user's session and is used by a web application to identify an authenticated user. A ***session replay attack*** is a replay attack in which a user's session ID is used to impersonate the user and perform a fraudulent transaction or activity.

PARTICIPATION ACTIVITY 7.4.1: Replay attack.



Animation content:

Static image: A user, a server, and an attacker. An arrow labeled "Original transmission" points from the user to the server. A dashed arrow labeled "Intercept" points from the Original transmission arrow to the attacker. A dashed arrow labeled "Replay transmission" points from the attacker to the server.

Animation captions:

1. A data transmission is sent from a user to a server.
2. An attacker intercepts the data transmission.
3. The attacker sends the intercepted transmission to the server at a later time.

PARTICIPATION ACTIVITY | 7.4.2: Replay attack.

How to use this tool

Session replay attack **Replay attack** **Session ID**

	A network attack in which a data transmission is intercepted and maliciously repeated or delayed.
	A network attack in which a user's session ID is used to impersonate the

	user and perform a fraudulent transaction or activity.
	A unique number assigned to an authenticated user by a web application.

Reset

PARTICIPATION ACTIVITY 7.4.3: Replay attack.

1) How is a replay attack conducted?

- by transmitting randomly generated data
- by requesting a data transmission from a user
- by replaying a previously intercepted transmission

2) Why is a replay attack possible?

- a server does not authenticate the user that sends the original transmission
- a server does not distinguish between identical messages received at different times
- a server does not authenticate the attacker

3) What is used in a session replay attack to impersonate a user?

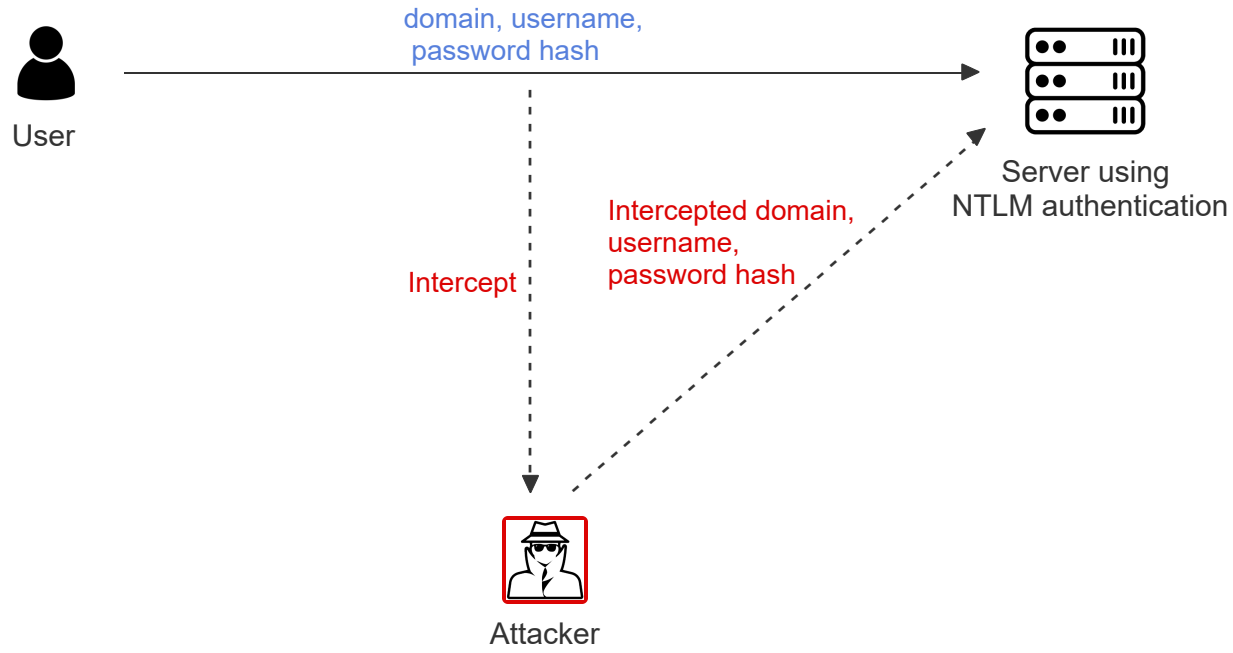
- username
- password
- session ID

Pass the hash

New technology LAN manager (NTLM) is a challenge-response authentication protocol. An NTLM credential consists of a domain name, a username, and a password hash. A user's password is not sent to a system or service using NTLM authentication.

Pass the hash is an attack in which an attacker authenticates to a server or service using the NTLM's password hash of a user. Pass the hash replaces the need to brute-force a password hash to obtain the cleartext password. The attack exploits an implementation vulnerability in NTLM's authentication protocol where a user's password hash remains static from session to session until the user's password is changed by the user. A pass the hash attack can be launched against any server or service accepting NTLM authentication.

PARTICIPATION ACTIVITY 7.4.4: Pass the hash attack.



Animation content:

Static image: A user, a server using NTLM authentication, and an attacker. An arrow labeled "domain, username, password hash" points from the user to the server. A dashed arrow labeled "Intercept" points from the Original transmission arrow to the attacker. A dashed arrow labeled "Intercepted domain, username, password hash" points from the attacker to the server.

Animation captions:

1. A user sends a domain and the user's username and password hash to a server using NTLM authentication.

2. An attacker intercepts the data transmission.
3. The attacker authenticates to the server using the NTLM's password hash of the user.

PARTICIPATION ACTIVITY 7.4.5: Pass the hash.

1) How is a pass the hash attack conducted?

- by sending a randomly generated password hash to a server that uses NTLM authentication
- by sending the domain, username, and password hash to a server that uses NTLM authentication
- by sending a user's password (obtained by brute-forcing the user's password hash) to a server that uses NTLM authentication

2) Why is a pass the hash attack possible?

- because in NTLM authentication, a user's password is transmitted in plaintext
- because in NTLM authentication, a user's password hash can be computed from a domain and a username
- because in NTLM authentication, a user's password hash remains static from session to session

©zyBooks 12/12/24 18:05 2172291
Daren Diaz
OUCYBS3213FreezeFall2024

3) What is intercepted by an attacker in a pass the hash attack?

- domain and password hash
- username and password hash
- domain, username, and password hash

7.5 Injection attacks: SQL, XML, LDAP, and DLL

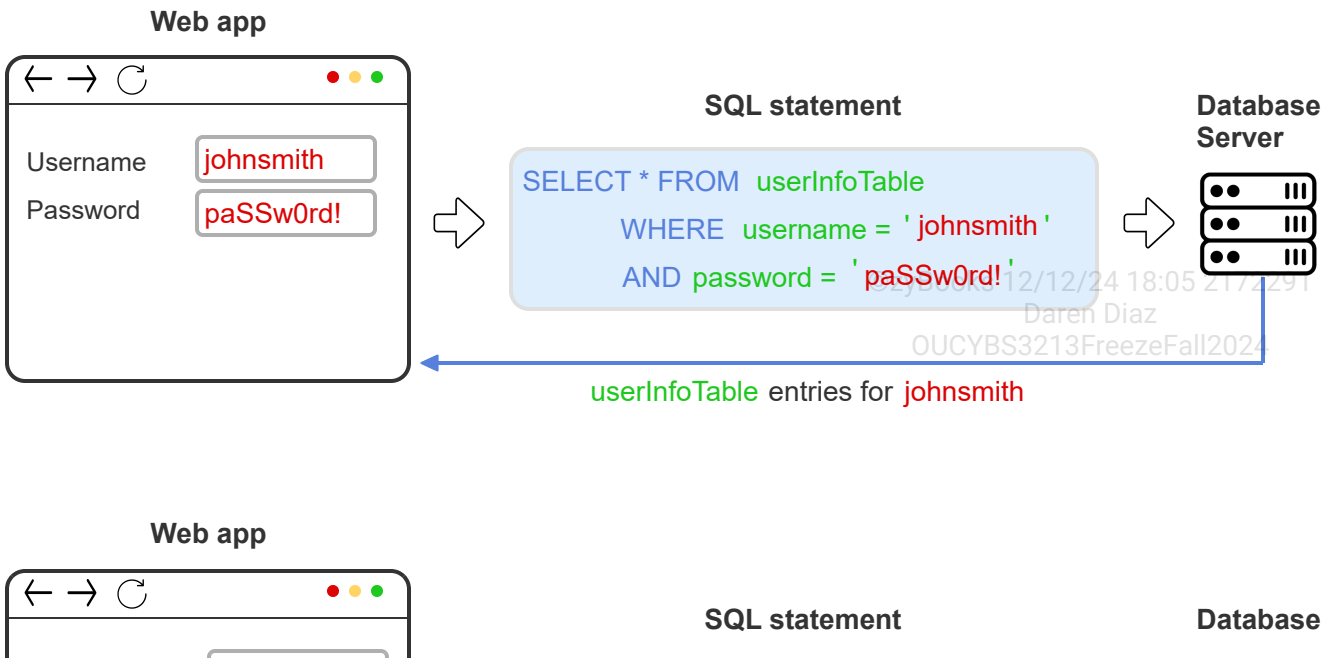
Injection attacks

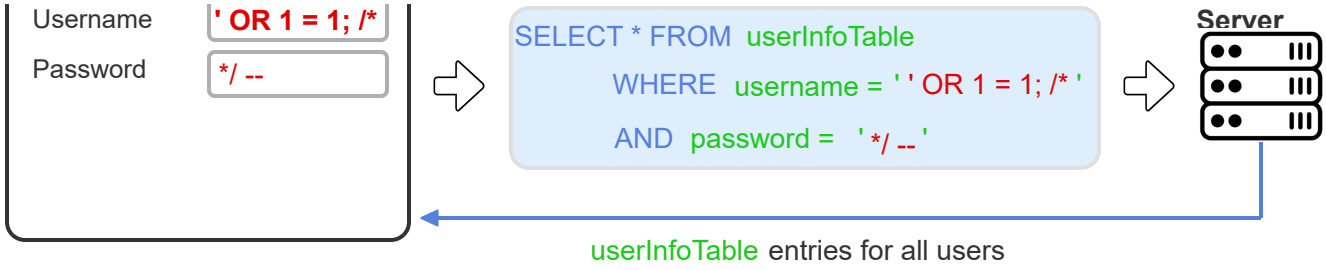
An ***injection attack***, also known as an ***input validation attack***, is an attack that injects or inserts malicious input into a program or database query. Malicious input may include code, scripts, and operating system commands. An injection attack exploits vulnerabilities created by inadequate validation of user input. Various injection attacks exist, including SQL injection, DLL injection, LDAP injection, and XML injection.

SQL injection

SQL injection, also known as ***SQLi***, is an injection attack in which an application's user input fields are used to insert a malicious SQL statement into the application. In SQL injection, an attacker enters crafted user input in an application such that a syntactically correct SQL statement is created when the user input is processed by the application. An SQL injection attack can be used against any type of SQL database, but is most commonly used against database-driven websites that allow user input. To protect against an SQL injection attack, user input should be validated before being used to construct an SQL statement.

PARTICIPATION ACTIVITY 7.5.1: SQL injection.





Animation content:

Static image: A browser labeled "Web app" with "Username: johnsmith" and "Password: paSSw0rd!". The browser has an arrow pointing to a box labeled "SQL statement" with SQL code.
Start SQL code.
SELECT * FROM userInfoTable
WHERE username = 'johnsmith'
AND password = 'paSSw0rd!'
End SQL code.
The SQL statement box has an arrow pointing to a server labeled "Database server". The database server has an arrow labeled "userInfoTable entries for johnsmith" pointing back to the web app browser. A second web app browser shows "Username: ' OR 1 = 1; /*" and "Password: */ --". An arrow points to another SQL statement box with SQL code.
Start SQL code.
SELECT * FROM userInfoTable
WHERE username = '' OR 1=1; /*'
AND password = '*/ --'
End SQL code.
An arrow points to another database server. An arrow labeled "userInfoTable entries for all users" points back to the second browser.

Animation captions:

1. A username and password entered by a user is used to construct an SQL statement that retrieves the user information stored in the userInfoTable table.
2. The database returns the userInfoTable entries for user johnsmith.
3. In SQL injection, an attacker inputs strings in username and password fields that generate a syntactically correct SQL statement by the Web app.
4. userInfoTable entries for all users are returned (WHERE clause ' OR 1 = 1; is always TRUE, everything between /* and */ is treated as multi-line comment, and -- comment literal neutralizes ').

PARTICIPATION ACTIVITY 7.5.2: SQL injection.

1) How is an SQL injection attack conducted?

- By entering user input to an application that when processed by the application, causes a back-end database to crash
- By brute-forcing a user password to authenticate to a back-end database and execute an SQL statement
- By entering user input to an application that when processed by the application, results in a syntactically correct SQL statement

2) Why is an SQL injection attack possible?

- Lack of authentication in an application that uses SQL statements to retrieve data from a back-end database
- User input to an application is not validated before being used to construct SQL statements
- User input to an application cannot be processed by an application that uses those input to construct SQL statements

3) In the animation above, why is the injection attack successful at retrieving the userInfoTable entries for all users?

- Because the user with username johnsmith is authorized to retrieve userInfoTable entries for all users
- Because the password */-- is the database administrator's

password

Because the application processes the user name of ' ' OR 1 = 1 and password of /*-- into a syntactically correct SQL statement

XML injection

Extensible Markup Language (XML) is a markup language that defines a set of rules for encoding documents in a text-based format. XML is used to describe and structure data for storage and transport. An XML document is readable by both humans and computers.

An ***XML injection*** is an attack that manipulates the logic of an XML-based application. An attacker can use XML injection to interfere with an application's processing of XML data by inserting malicious content into an XML document. The injection of malicious content into an XML message modifies the intended logic of the application that processes the XML message.

PARTICIPATION ACTIVITY 7.5.3: XML injection.

Application's input fields

Username

Password

E-mail

XML document

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<users>
    <user>
        <userid>0</userid>
        <uname>johnsmith</uname>
        <pwd>jsjs99$*</pwd>
        <mail>j.smith@example.com</mail>
    </user>
    <user>
        <userid>1</userid>
        <uname>avajones</uname>
        <pwd>ajaj77*&</pwd>
        <mail>a.jones@example1.com</mail>
    </user>
    <user>
        <userid>2</userid>
        <uname>elirashid</uname>
        <pwd>erer48&$</pwd>
        <mail>elirashid@example2.com</mail>
    </user>
    <user>
        <userid>0</userid>
        <uname>hacker</uname>
        <pwd>haha4$</pwd>
        <mail>hacker@example3.com</mail>
```

</user>
</users>

Animation content:

Static image: A box labeled "Application's input fields" has input fields for "Username", "Password" and "E-mail". A box labeled "XML document" has XML code.

Start XML code.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<users>
  <user>
    <userid>0</userid>
    <uname> johnsmith </uname>
    <pwd> jsjs99$* </pwd>
    <mail> j.smith@example.com </mail>
  </user>
  <user>
    <userid>1</userid>
    <uname> avajones </uname>
    <pwd> ajaj77*& </pwd>
    <mail> a.jones@example1.com </mail>
  </user>
  <user>
    <userid>2</userid>
    <uname> elirashid </uname>
    <pwd> erer48&$ </pwd>
    <mail> elirashid@example2.com</mail>
      </user>
      <user>
      <userid>0</userid>
      <uname>hacker</uname>
      <pwd>haha4$</pwd>
      <mail>hacker@example4.com </mail$gt;
  </user>
</users>
```

End XML code.

The blue box surrounds the following lines.

Start XML code.

```
<user>
      <userid>0</userid>
      <uname>hacker</uname>
      <pwd>haha4$</pwd>
      <mail>hacker@example4.com </mail$gt;
  </user>
```

End XML code.
Step 1: An application uses XML-based messages to perform user registration by creating and adding a new <user> node in an XML document.
A user "johnsmith" adds information into the application's input fields. The information is copied into the XML document to create the first user.
Step 2: A userid is generated for each new user and the user's Username, Password, and E-mail are added to the XML document.
A second user "avajones" adds information into the application's input fields. The information is copied into the XML document to create a second user.
Step 3: In XML injection, an attacker inserts a crafted input string in the application's E-mail input field.
A user "elirashid" adds information into the application's input fields. In the email field, the user inputs the following XML code.
Start XML code.
elirashid@example2.com</mail>
</user>
<user>
<userid>0<userid>
<uname>hacker</uname>
<pwd>haha4$</pwd>
<mail>hacker@example3.com
End XML code.
Step 4: If no input validation checks are performed by the application, a new user is added to the XML document with the attacker's input entries.
The attacker's information is copied into the XML document.
Step 5: A user with the username of hacker and userid of 0 has been injected in the XML document. A user with userid of 0 may have administrative privileges in the system.

Animation captions:

1. An application uses XML-based messages to perform user registration by creating and adding a new <user> node in an XML document.
2. A userid is generated for each new user and the user's Username, Password, and E-mail are added to the XML document.
3. In XML injection, an attacker inserts a crafted input string in the application's E-mail input field.
4. If no input validation checks are performed by the application, a new user is added to the XML document with the attacker's input entries.
5. A user with the username of hacker and userid of 0 has been injected in the XML document. A user with userid of 0 may have administrative privileges in the system.

PARTICIPATION ACTIVITY 7.5.4: XML injection.

1) What is impacted in an application by XML injection?

- Application performance
- Application logic
- Application responsiveness

2) How does XML injection interfere with an application's processing of XML data?

- By deleting XML documents that are processed by the application
- By inserting malicious content into XML documents
- By preventing an application from processing XML documents

3) In the above animation, what is the userid of the injected user?

- 0
- 1
- 2

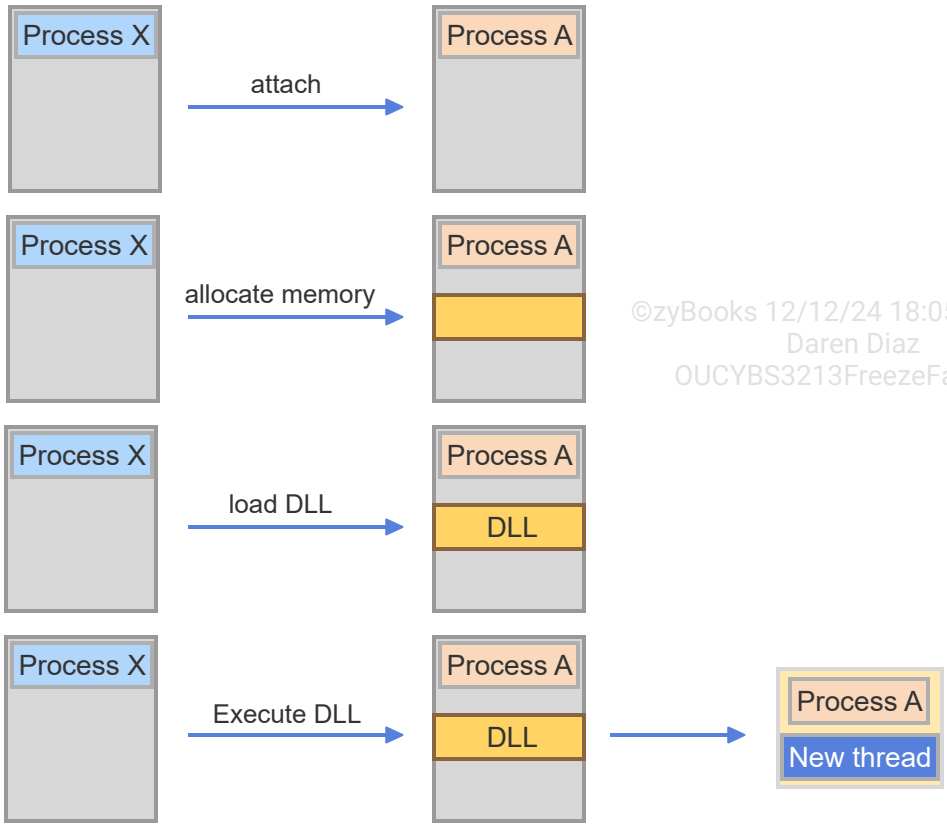
DLL injection

A ***dynamic link library*** or ***DLL*** is a library of code and data for commonly used functions in a Windows computer. A DLL can be used by more than one program at the same time. A DLL helps promote code reuse, speeds up processing, and improves memory usage. Ex: The Comdlg32 DLL contains code and data for functions related to a Windows dialog box.

DLL injection is an injection attack in which a process is forced to load and execute malicious DLL code within the process's address space. DLL injection aims to manipulate the execution of a running process and modify the process's behavior. Ex: DLL injection could be used to inject code in a running program that reads a password textbox's contents.

PARTICIPATION ACTIVITY 7.5.5: DLL injection.

Attacking process Victim process



Animation content:

Static image: Two columns labeled "Attacking process" and "Victim process". Four rows of gray boxes in each column. Each gray box in the Attacking process column contains a blue box labeled "Process X". Each box in the Victim process column contains a tan box labeled "Process A". The first row has an arrow labeled "attach" pointing from the attacking process to the victim process. The second row has an arrow labeled "allocate memory" pointing from the attacking process to the victim process. A gold box is under the Process A box. The third row has an arrow labeled "load DLL" pointing from the attacking process to the victim process. A gold box labeled "DLL" is below the Process A box. The last row has an arrow labeled "Execute DLL" pointing from the attacking process to the victim process. The gold box labeled "DLL" is still below process A. The victim process points to a new orange box containing a copy of the tan Process A box and a purple box labeled "New thread".

Animation captions:

1. The attacking process (process X) issues a Windows API call to attach to the victim process (process A).
2. A memory region is allocated within the victim process with "write" access.
3. The attacking process stores the path of the malicious DLL to the allocated memory region.
4. The attacking process creates a remote thread in the victim process's memory space.

PARTICIPATION ACTIVITY 7.5.6: DLL injection.

1) A DLL can be used by only one program at a time.
   - ○ True
   - ○ False

2) DLL injection results in the execution of malicious code in the victim process's address space.
   - ○ True
   - ○ False

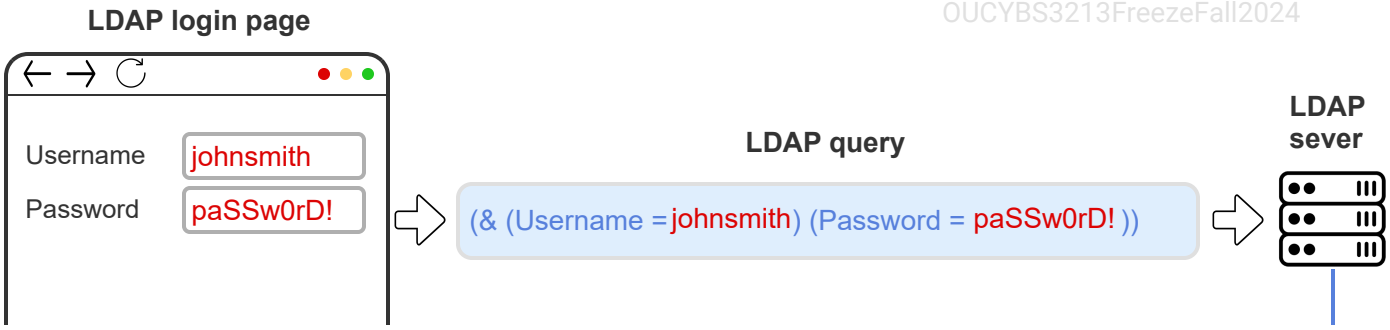
3) The aim of DLL injection is to stop the execution of the victim process.
   - ○ True
   - ○ False

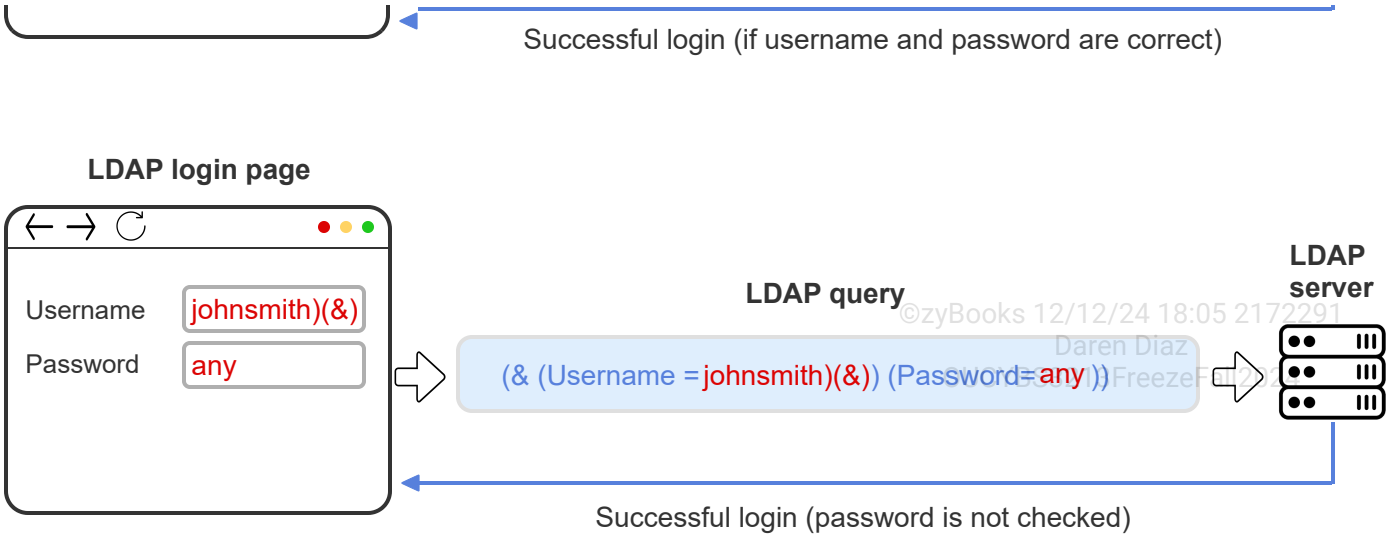
LDAP injection

Lightweight directory access protocol (LDAP) is a protocol for accessing and maintaining distributed directory information services over an IP network. A directory service enables the sharing of information about a user, system, service, or application in a network. LDAP is commonly used to provide a central location for storing usernames and passwords.

An ***LDAP injection*** is an attack in which an attacker exploits input validation vulnerabilities to manipulate an LDAP statement executed on an LDAP server. LDAP injection exploits applications that construct LDAP statements based on user input. LDAP injection may result in execution of arbitrary LDAP commands, such as viewing and modifying LDAP data or bypassing LDAP authentication.

PARTICIPATION ACTIVITY 7.5.7: LDAP injection.





Animation content:

Static image: A browser labeled "LDAP login page" with "Username: johnsmith" and "Password: paSSw0rD!". The browser has an arrow pointing to a box labeled "LDAP query" with "(& (Username = johnsmith) (Password = paSSw0rD!))". The LDAP query has an arrow pointing to a server labeled "LDAP server". The LDAP server has an arrow labeled "Successful login (if username and password are correct" pointing back to the LDAP login page. A second LDAP login page shows " johnsmith)(&)" and "Password: any". An arrow points to another LDAP query box with "(& (Username = johnsmith) (&)) (Password=any))".
An arrow points to another LDAP server. An arrow labeled "Successful login (password is not checked)" points back to the second browser.

Animation captions:

1. An LDAP login page has a username and password text box fields. To authenticate to an LDAP server, a user enters a username and a password.
2. To authenticate the user, a query is constructed using the username and password and sent to the LDAP server.
3. The user is authenticated if the username and password pair exist in the LDAP server.
4. In LDAP injection, an attacker injects special characters after a valid username. The special characters cause the LDAP server to ignore the password string.
5. "(& (Username = johnsmith)(&))" always evaluates to TRUE and the Password is not checked. The attacker successfully logs into the LDAP server.

PARTICIPATION ACTIVITY 7.5.8: LDAP injection.

1) How is an LDAP injection attack executed?

- By gaining access to the login credentials of an authorized LDAP user
- By conducting a Denial-of-Service (DOS) attack against an LDAP server
- By manipulating an LDAP query executed on an LDAP server

2) What is not a potential consequence of LDAP injection?

- Viewing LDAP data
- Modifying LDAP data
- Shutting down an LDAP server

3) In the above animation, what is the consequence of the LDAP injection?

- Failed login attempt
- Unauthenticated access to LDAP data
- Updated the LDAP server with a new password for user *johnsmith*

7.6 Race condition and resource exhaustion

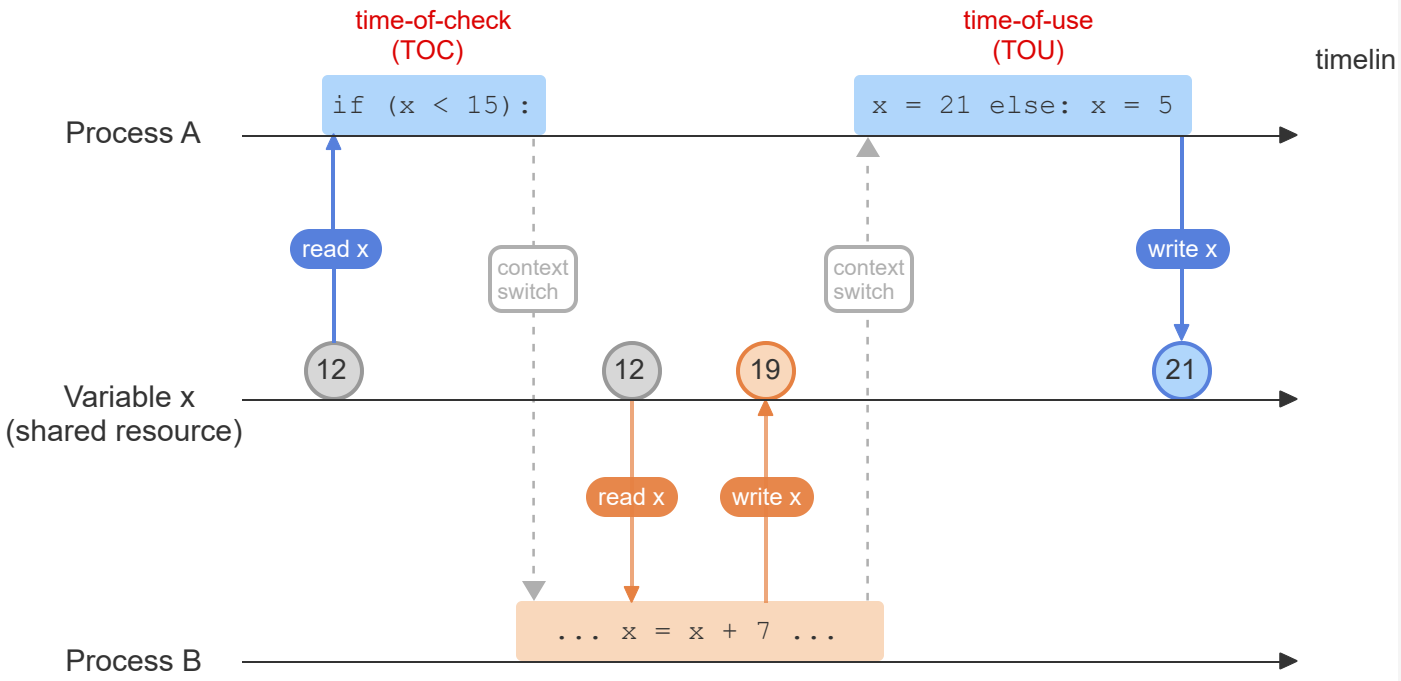
Race condition

A ***race condition*** is a software event in which multiple concurrently executing processes access a shared resource at the same time. The ***target of evaluation (TOE)*** is the shared resource being accessed, which can be a file, memory location, or program variable. Any multi-threaded program and programs that run on multi-core CPUs can be impacted by a race condition since multiple processes execute concurrently.

Time-of-check to time-of-use, or ***TOCTOU***, is a class of software errors that are caused by a race condition. ***Time-of-check (TOC)*** is the time at which a process checks the TOE's state. ***Time-of-use***

(TOU) is the time at which a process uses the TOE. A TOCTOU error occurs when the TOE's state changes between TOC and TOU in a way that invalidates the result of the resource state check. Since the TOE is in an unexpected state, a TOCTOU error causes a process to perform incorrect computations. TOCTOU errors can be exploited to gain unauthorized access to system resources, such as read/write access to files.

PARTICIPATION ACTIVITY | 7.6.1: TOCTOU error.



Animation content:

Static image: Three timelines labeled "Process A", "Variable x (shared resource)", and "Process B". A gray circle on the Variable x timeline contains "12". The 12 circle has an arrow labeled "read x" pointing toward the Process A timeline. A code block on the Process A timeline contains "if (x < 15):". A dashed gray arrow labeled "context switch" points from the end of the code block down to the Process B timeline. A code block on the Process B timeline contains "...x = x + 7...". A second gray circle containing "12" is on the Variable x timeline just above the code block on Process B. An arrow labeled "read x" points from the 12 circle down to the first "x" in the code block. An arrow labeled "write x" points from the "7" in the code block up to an orange circle with "19" on the Variable x timeline. A dashed gray arrow labeled "context switch" points from the end of the Process B code

block up to a second code block in Process A containing "x = 21 else: x = 5". An arrow labeled "write x" points from the end of the code block to a new blue circle with "21" on the Variable x timeline.

Animation captions:

1. Variable x is a shared resource between Process A and Process B. Process A reads variable x to evaluate an 'if statement'. At TOC, the value of x is 12.
2. A context switch occurs. The operating system saves the state of Process A so that the process can resume execution at a later time. Process B resumes execution and reads variable x.
3. Process B adds 7 to x and sets variable x to 19.
4. A context switch restores the state of Process A. At TOU, the value of x is 19. A TOCTOU error has occurred (variable x has changed from 12 at TOC to 19 at TOU).
5. Since variable x was 12 at TOC, Process A's 'if clause' evaluates to TRUE and variable x is set to 21.

PARTICIPATION ACTIVITY 7.6.2: Race condition.

1) A race condition is possible because multiple concurrently executing processes _____.

- access a shared resource at different times
- access a shared resource at the same time
- cannot access the same resources

2) A TOCTOU error results in incorrect computations by a process because _____.

- a shared resource is not modified by other processes between TOC and TOU
- a shared resource is modified by other processes between TOC and TOU
- a resource cannot be shared between two or more processes between TOC and TOU

3) A TOCTOU error can be exploited by _____.

- injecting malicious code into the TOE
- manipulating the TOE's state between TOC and TOU
- gaining read-only access to the TOE

4) In the above animation, what would be the final value of x if Process B finishes executing before Process A reads the value of x, and no TOCTOU errors occur?

- 5
- 12
- 19

5) In the above animation, what would be the final value of x if Process A finishes executing before Process B reads the value of x, and no TOCTOU errors occur?

- 12
- 21
- 28

Resource exhaustion

A ***resource exhaustion attack***, also known as ***resource starvation attack***, is an attack that aims to slow down or disable an application by using up the application's limited resources. A resource includes memory, CPU, file system storage, and database connection pool entries. Ex: A CPU exhaustion attack against an application consumes all of the application's allocated CPU cycles and reduces the application's responsiveness and availability.

A resource exhaustion attack is a type of denial-of-service attack that prevents legitimate users and applications from accessing an application or service. A resource exhaustion attack is possible because an application does not properly control the allocation and maintenance of the application's limited resources. Ex: A memory leak in an application can be exploited to exhaust the application's available memory and slow down the application and the application's host operating system.

PARTICIPATION ACTIVITY 7.6.3: Resource exhaustion.

1) A resource exhaustion attack against an application aims to deny the application's authorized users from accessing the application.
   - True
   - False

2) A resource exhaustion attack is possible because an application properly controls the allocation and maintenance of the application's resources.
   - True
   - False

3) An attack that makes an application to continuously write data to a solid state drive (SSD) is a resource exhaustion attack.
   - True
   - False

7.7 Driver manipulation, privilege escalation, and password attacks

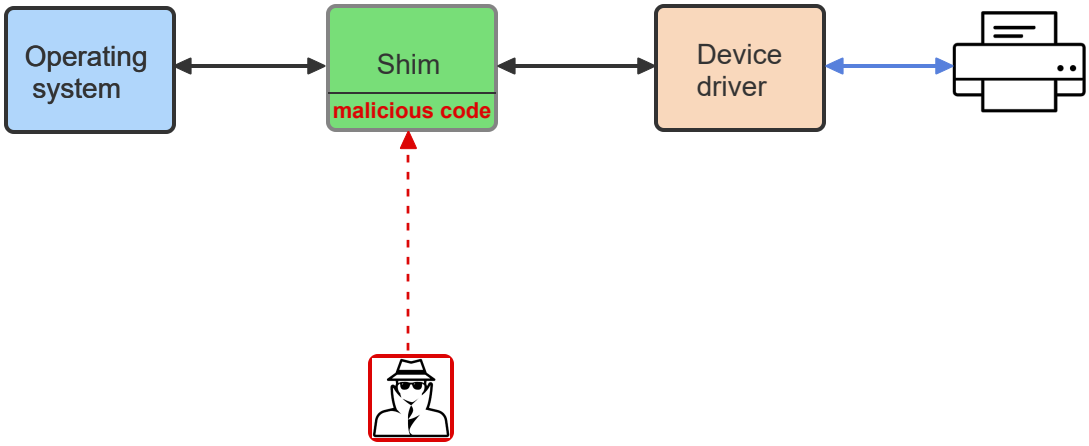
Driver manipulation

A ***device driver***, also known as ***hardware driver*** or ***driver***, is a program that controls a computer-connected hardware device. Ex: A keyboard, printer, webcam, or mouse is controlled by a device driver. A device's driver provides a software interface to the device and enables an operating system or application to access the device's functions. Ex: A printer's device driver for Windows operating system enables Windows applications to use the printer for printing documents.

A device driver can be manipulated to enable an attack against a computer or device. Two device manipulation attack methods exist:

- A ***driver shim*** is a software library that intercepts and modifies calls to a device driver. A driver shim is used to provide backward compatibility with older devices. Ex: Windows 10 provides compatibility with Windows 8 device drivers by using driver shims. A ***shim attack***, also known as ***shimming***, is an attack in which malicious code is added to a driver shim. The malicious code in a driver shim is executed when a device driver call is made by the operating system.

PARTICIPATION ACTIVITY 7.7.1: Shim attack.



Animation content:

Static image: A box labeled "Operating system", a box labeled "Shim", a box labeled "Device driver", a printer, and an attacker. A two-way arrow is between the Operating system and the Shim. A two-way arrow is between the Shim and the Device driver. A two-way arrow is between the Device driver and the printer. A dashed red arrow points from the attacker to the Shim, and "malicious code" is written in red text within the Shim box.

Animation captions:

1. A device driver is a program that controls a computer-connected hardware device, such as a printer. A device driver provides a software interface to a device.
2. A driver shim is a software library that intercepts and modifies calls to a device driver. A driver shim is used to support an old device driver and provide backward compatibility.

3. A shim attack, also known as shimming, is an attack that inserts malicious code inside a driver shim.

- ***Refactoring*** is a set of techniques used for improving a program's internal structure without changing the program's behavior. Refactoring is used for improving a program's non-functional attributes, such as reducing the program's complexity or improving the program's readability. A ***refactoring attack*** is an attack in which malicious code is added to a device driver through refactoring without changing the device driver's desired behavior. A refactored device driver supports a device's functions, but also executes the added malicious code. Ex: A refactored printer driver prints documents on a printer, but may also send the print jobs to a remote printer controlled by an attacker.

PARTICIPATION ACTIVITY 7.7.2: Driver manipulation.

How to use this tool

Shim attack **Refactoring** **Device driver** **Refactoring attack** **Driver shim**

	A program that controls a computer-connected hardware device.
	A software library that intercepts and modifies calls to a device driver.
	An attack in which malicious code is added to a driver shim.
	A set of techniques used for improving a program's internal structure without changing the program's behavior.
	An attack in which malicious code is added to a device driver through refactoring.

Reset

PARTICIPATION ACTIVITY 7.7.3: Driver manipulation.

Select the device manipulation attack method in each question.

1) A _____ attack adds a layer of code between a device driver and an operating system.
   - shimming
   - refactoring

2) A _____ attack exploits the need to improve code efficiency and performance.
   - shimming
   - refactoring

3) A _____ attack restructures device driver code without changing the device driver's external behavior.
   - shimming
   - refactoring

4) A _____ attack exploits the need to provide backward compatibility with older devices.
   - shimming
   - refactoring

Privilege escalation

A computer account has a set of access rights or privileges on the computer. Common account privileges include viewing, editing or printing files, or running a web browser. ***Privilege escalation*** is the act of gaining account privileges beyond those assigned to the account. Privilege escalation enables a user or application to perform unauthorized actions. Ex: Privilege escalation may enable a user to install malware on a computer, even though the user is not authorized to install programs on the computer. Privilege escalation may exploit vulnerabilities in application programs, flaws in account permissions, or errors in operating system configuration files.

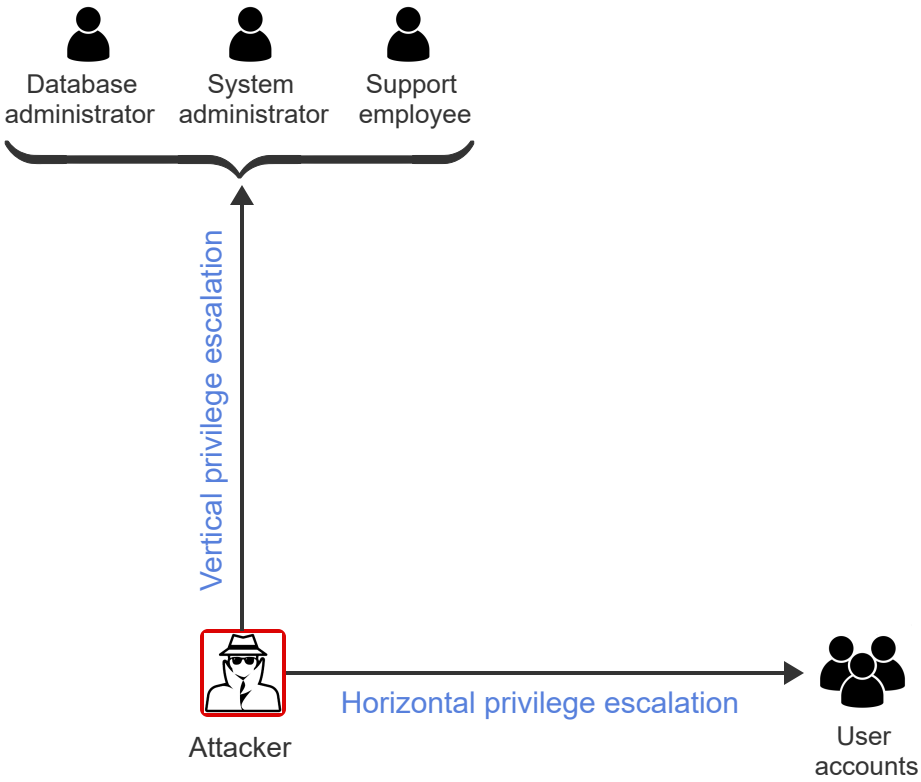
Two privilege escalation types exist:

- A ***vertical privilege escalation*** is privilege escalation in which a lower privilege user or application gains privileges assigned to another user or application with higher privileges. A

vertical privilege escalation may result in a user or application gaining administrative privileges on a computer. Ex: A vertical privilege escalation may enable an Internet banking user to access the banking site's administrative functions.

- A ***horizontal privilege escalation*** is privilege escalation in which a user or application gains privileges assigned to another user or application with the same privileges. Ex: A horizontal privilege escalation may enable an Internet banking user to access the Internet bank account of another user.

PARTICIPATION ACTIVITY 7.7.4: Privilege escalation.



Animation content:

Static image: An attacker. An arrow labeled "Horizontal privilege escalation" points to the right from the attacker to a group of people labeled "User accounts". An arrow labeled "Vertical privilege escalation" points upward from the attacker to three people labeled "Database administrator", "System administrator", and "Support employee".

Animation captions:

1. Privilege escalation can be horizontal or vertical.
2. In horizontal privilege escalation, an attacker gains access to user accounts that have the same privileges as the attacker.
3. In vertical privilege escalation, an attacker gains access to user accounts that have higher privileges than the attacker, such as administrator or support accounts.

PARTICIPATION ACTIVITY 7.7.5: Privilege escalation.

How to use this tool

Privilege escalation **Vertical privilege escalation** **Horizontal privilege escalation**

	The act of gaining account privileges beyond those assigned to the account.
	A privilege escalation type in which a lower privilege user or application gains privileges assigned to another user or application with higher privileges.
	A privilege escalation type in which a user or application gains privileges assigned to another user or application with the same privileges.

Reset

PARTICIPATION ACTIVITY 7.7.6: Privilege escalation.

Select the privilege escalation type in each scenario.

1) A non-administrative user gains access to an administrative web page in which user accounts can be deleted.
   - ○ Horizontal

- Vertical

2) A cross-site scripting (XSS) attack allows an attacker to steal a user's session cookies and access the user's account.
   - Horizontal
   - Vertical

3) A cross-site request forgery (CSRF) attack tricks a user into executing an unwanted action in a web application.
   - Horizontal
   - Vertical

4) A buffer overflow exploit enables an attacker to obtain privileged credentials for an application.
   - Horizontal
   - Vertical

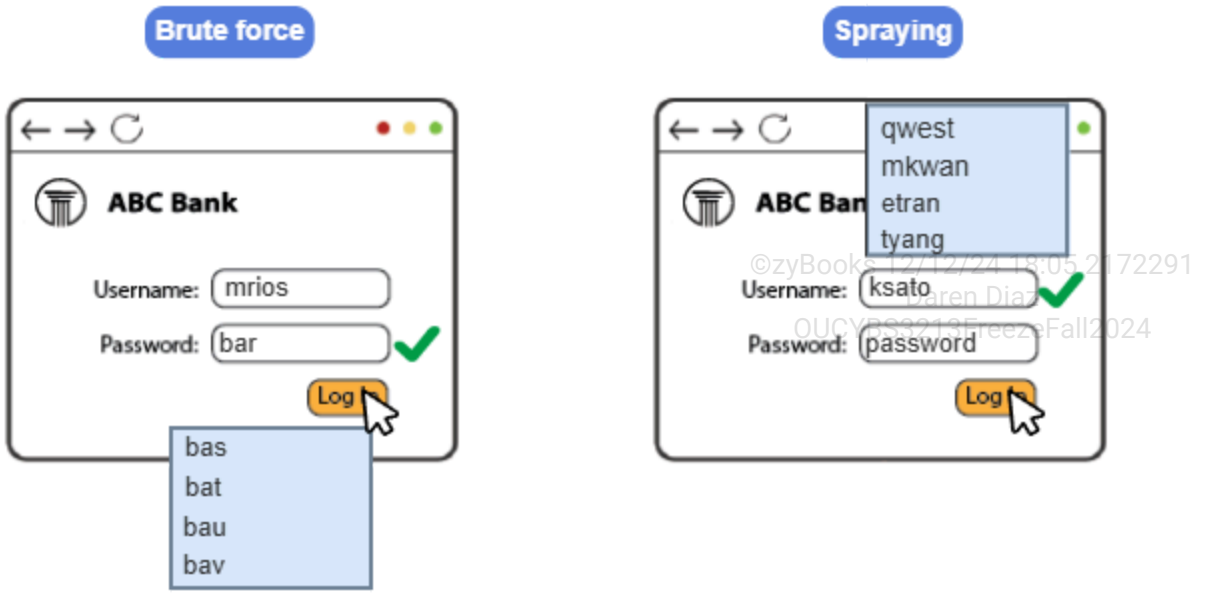
5) An attacker obtains a user's credentials through a social engineering attack and uses those credentials to access the user's bank account.
   - Horizontal
   - Vertical

Password attacks

An attacker can use brute force or spraying password attacks to gain unauthorized access to a user's account.

- A ***brute force*** attack is a password attack in which an attacker gains unauthorized access to a user's account by attempting every possible character combination to find the user's password.
- A ***spraying*** attack is a password attack in which an attacker gains unauthorized access to an account by attempting a default or common password with many accounts.

PARTICIPATION ACTIVITY 7.7.7: Brute force and spraying.



Animation content:

Static figure: One browser showing a brute force attack and a second browser showing a spraying attack.
Step 1: In a brute force attack, an attacker selects a single username and tries all possible passwords.
A "Brute force" label above a browser with a login screen. The username field contains "mrios". A box appears with a list containing "a", "b", "c", and "...". The "a" moves into the password field. A red X appears next to the password, and then the "a" disappears. The "b" moves into the password field, the red X appears, and then the "b" disappears. The "c" moves into the password field, the red X appears, and then the "c" disappears. The "..." in the box disappears, and the box contains the list "bar", "bas", "bat", "bau".
Step 2: The brute force attack succeeds when one of the passwords is accepted with the given username.
The "bar" moves into the password field. A green check mark appears next to the password field.
Step 3: In a spraying attack, an attacker selects a single password and tries several usernames.
A "Spraying" label above a browser with a login screen. The password field contains "password". A box appears with a list containing "mrios", "apeters", "ksato", and "tyang". The "mrios" moves into the username field. A red X appears next to the username, and then the "mrios" disappears. The "apeters" moves into the username field, the red X appears, and then the "apeters" disappears.
Step 4: The spraying attack succeeds when one of the usernames is accepted with the given password.
The "ksato" moves into the username field. A green check mark appears next to the username field.

Animation captions:

1. In a brute force attack, an attacker selects a single username and attempts all possible passwords.
2. The brute force attack succeeds when one of the passwords is accepted with the given username.
3. In a spraying attack, an attacker selects a single password and attempts many usernames.
4. The spraying attack succeeds when one of the usernames is accepted with the given password.

PARTICIPATION ACTIVITY 7.7.8: Brute force and spraying attacks.

1) In a spraying attack, an attacker guesses _____.
   - passwords
   - security question answers
   - usernames

2) An account with a _____ password is most vulnerable to a spraying attack.
   - default
   - short
   - long

3) An account with a _____ password is most vulnerable to a brute force attack.
   - default
   - short
   - long

4) An application can defend against a brute force attack by _____.
   - requiring users to choose a password rather than assigning a default password
   - locking user accounts after three failed login attempts

- requiring special characters in a user's password

Default credentials

Default credentials *refer to pre-configured usernames and passwords set by manufacturers or developers for devices, systems, or applications. Default credentials are intended to provide initial access for setup, configuration, and administration purposes. Widely known and easily identifiable, default credentials pose a significant security risk. Ex: 'admin' for username and 'password' for password.*

Default credentials expand an organization's attack surface, facilitating unauthorized access and malicious activities like lateral movements and privilege escalation. To mitigate security risks associated with default credentials, such credentials must be replaced with unique and strong alternatives promptly.

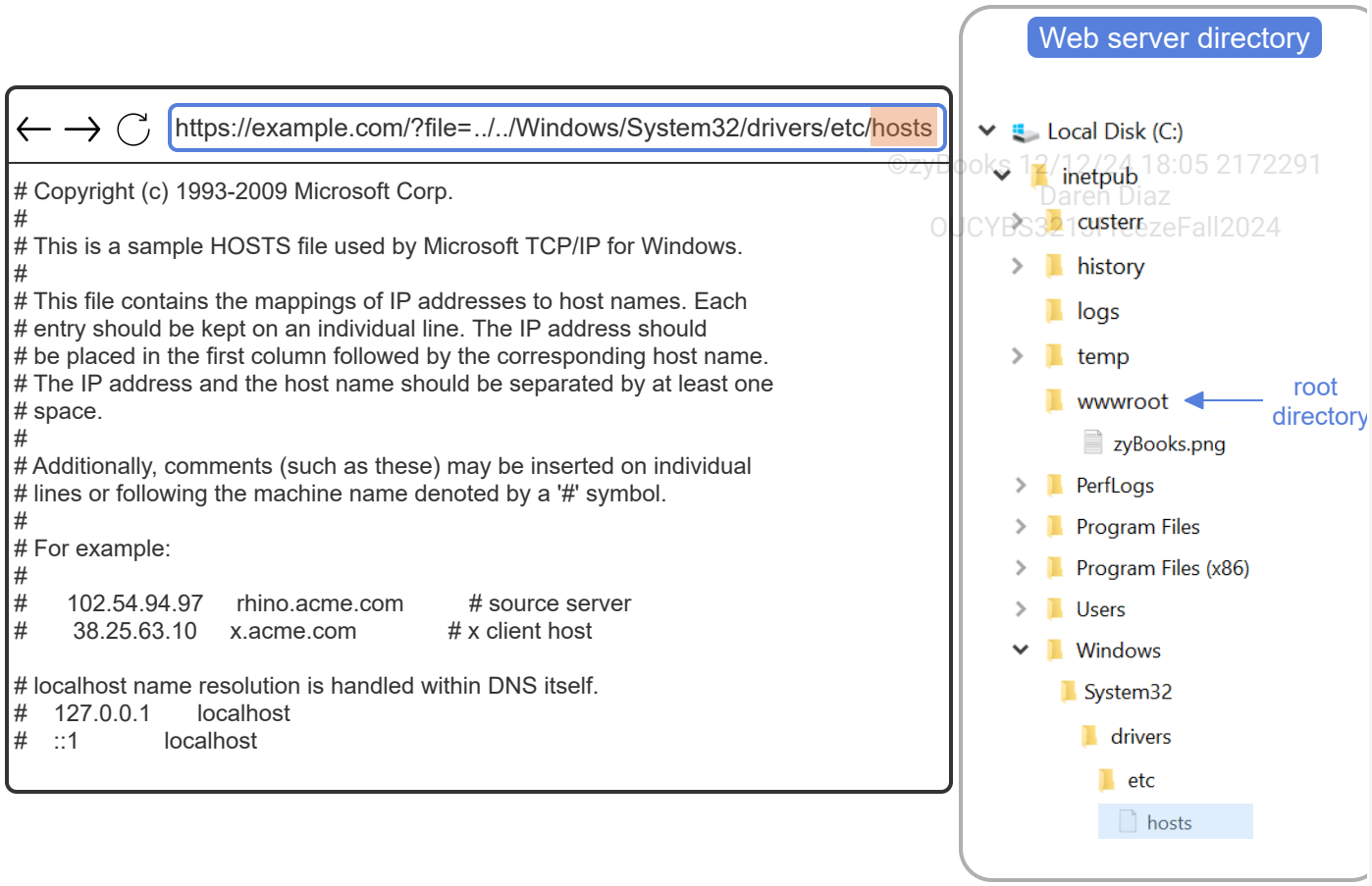
7.8 Directory traversal, API attacks, and SSL stripping

Directory traversal

A ***root directory*** is a directory on a web server's file system that the web server's users can access. Ex: The root directory of Microsoft Internet Information Services (IIS) web server is *"C:\Inetpub\wwwroot"*. ***Directory traversal***, also known as ***path traversal*** or ***backtracking***, is a web application security vulnerability that allows unauthorized users to access files outside of a web server's root directory. Since a directory traversal attack uses ../ (dot-dot-slash) to traverse a file system, a directory traversal attack is also known as a dot-dot-slash attack. A server vulnerable to directory traversal can allow a user to read file contents outside of the intended directories.

A directory traversal attack can be prevented by different methods, such as not authorizing a web server program to access files outside the web root directory. To prevent unauthorized users from traversing to system directories, the web root directory should not be on the system disk. A ***system disk*** is any media that contains part or all of an operating system.

PARTICIPATION ACTIVITY 7.8.1: Directory traversal.



Animation content:

Static image: A browser window showing a sample HOSTS file used by Microsoft TCP/IP for Windows. The web address is "http://example.com/?file=../../Windows/System32/drivers/etc/hosts". An orange highlight is on "hosts" in the web address. A box labeled "Web server directory" is next to the browser window. The web server directory shows files nested under the Local Disk (C:). A folder called "wwwroot" is nested under a folder called "inetput" and contains a folder called zyBooks.png. The label "root directory" points to the wwwroot folder. A folder called "Windows" is also nested under inetpub. A file called "hosts" is highlighted blue and nested under the Windows folder.

Animation captions:

1. A root directory is a directory on a web server's file system that the web server's users can access.
2. A directory traversal is a web application security vulnerability that allows unauthorized users to access files outside of a web server's root directory by entering crafted URLs in a web client.

3. URLs should be validated to prevent user access to files stored outside a web server's root directory.

PARTICIPATION ACTIVITY 7.8.2: Directory traversal.

1) Directory traversal allows users to _____.
   - access files outside of a web server's root directory
   - access files inside of a web server's root directory
   - access files in a different server

2) A directory traversal attack uses _____ to traverse a file system.
   - */
   - -->
   - ../

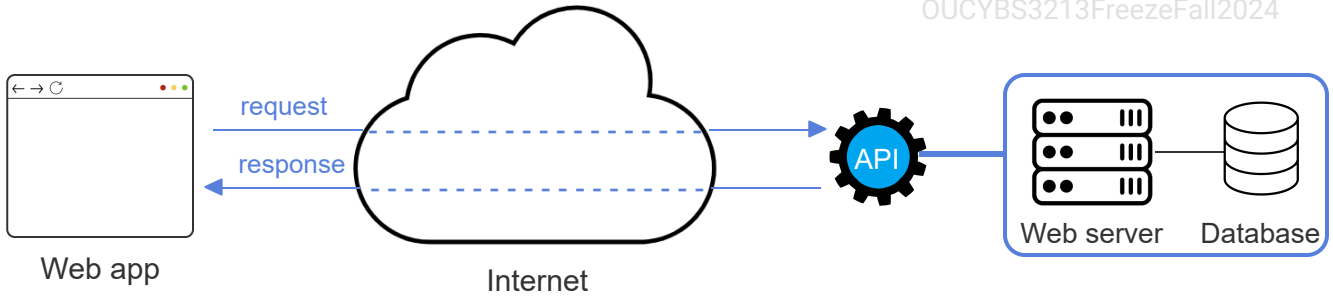
3) In the animation above, which directory should be restricted from unauthorized users?
   - All directories outside System32
   - All directories under wwwroot
   - All directories outside wwwroot

Application programming interface (API) attacks

An ***application programming interface (API)*** is a set of functions and protocols for building and integrating different applications and data sets. An API allows applications to communicate with one another. Ex: A web server's API connects an application with other systems, such as a social platform, database, or device.

APIs are a target of various attacks because APIs provide entry points to an application and enable access to application data. An ***application programming interface (API) attack*** is an attack that exploits the vulnerabilities in an application's API to attack the application or a service. The aim of an API attack is to disrupt the services an API enables or access the data an API stores or retrieves. Ex: A web server's API can be used to launch a Denial-of-Service (DoS) attack against the web server.

PARTICIPATION ACTIVITY 7.8.3: Web-based API.



Animation content:

Static image: A browser window labeled "Web app" is next to a cloud labeled "Internet". An arrow labeled "request" points from the Web app browser, through the Internet cloud, to a gear labeled "API". The API gear is connected to a box containing a web server and a database. An arrow labeled "response" points from the API gear, through the Internet cloud, back to the Web app browser.

Animation captions:

1. A web API provides an interface to a request-response message system provided through a web server.
2. A web app sends requests to the web server by using the API functions.
3. A web app receives responses from the web server by using the API functions.

PARTICIPATION ACTIVITY 7.8.4: API attacks.

1) APIs facilitate communication between applications.

   - True

- False

2) An API attack cannot be used to slow down the services an API enables.
   - True
   - False

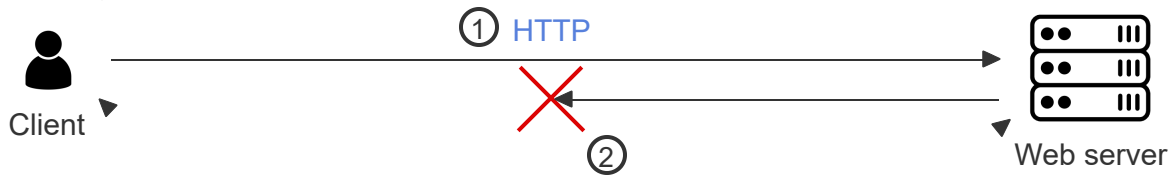
3) In the animation above, the web server API enables the web app to request data from the web server.
   - True
   - False

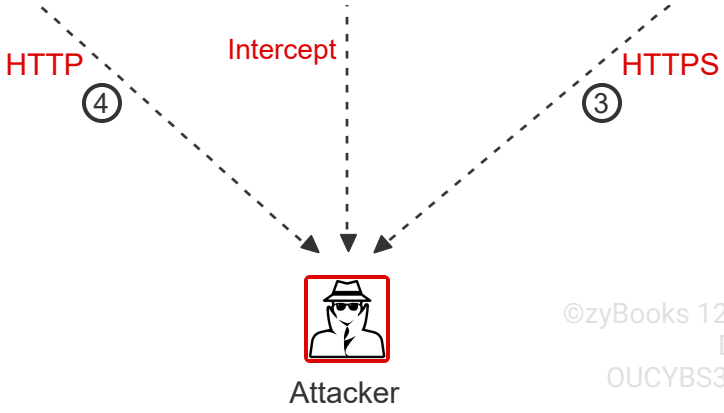
Secure Sockets Layer (SSL) stripping

A ***downgrade attack*** is an attack that attempts to shift client-server communication to a less secure encryption method. ***Secure socket layer stripping (SSL stripping)***, also known as ***SSL downgrading*** or ***HTTP downgrading***, is a downgrade attack that aims to downgrade an attempted HTTPS connection between a web server and client to an HTTP connection. Unlike an HTTPS connection, an HTTP connection does not provide data confidentiality, integrity, and authenticity. SSL stripping is a type of on-path or man-in-the-middle attack.

In SSL stripping, an attacker intercepts a web server's redirect response to a client's HTTP request. The attacker establishes an HTTP connection to the client and an HTTPS connection to the web server. Throughout the attack, the client unknowingly communicates with the attacker via the HTTP connection, and the attacker communicates with the web server via the HTTPS connection.

PARTICIPATION ACTIVITY 7.8.5: SSL stripping.





Animation content:

Static image: A client, a web server, and an attacker. An arrow labeled "HTTP" points from the client to the web server. Another arrow points from the web server to a red X halfway between the client and web server. A dashed arrow labeled "Intercept" points from the red X to the attacker. A dashed two-way arrow labeled "HTTPS" is between the attacker and the web server. Another dashed two-way arrow labeled "HTTPS" is between the client and the attacker.

Animation captions:

1. A client sends an HTTP request to a web server.
2. The web server sends an HTTP redirect response to the client with the URL of the HTTPS version of the website. The redirect response is intercepted by the attacker.
3. The attacker uses the redirect URL to establish an HTTPS connection to the web server (pretending to be the client) .
4. The attacker establishes an HTTP connection to the client (pretending to be the web server).

PARTICIPATION ACTIVITY | 7.8.6: SSL stripping.

1) The aim of SSL stripping is to conduct a _____ attack.

- denial-of-service
- man-in-the-middle
- server-side request forgery

2) What is the result of a successful SSL stripping attack?

- an unencrypted communication channel between a web server and an attacker
- an unencrypted communication channel between a client and an attacker
- an unencrypted communication channel between a web server and a client

3) In the animation above, what is intercepted by the attacker?

- the client's HTTPS request to the web server
- the web server's redirect response sent to the client
- the client's HTTP request sent to the web server

Cryptographic vulnerabilities

Cryptographic vulnerabilities, exemplified by downgrade attacks like SSL stripping, highlight the importance of implementing strong cryptographic measures to secure online communications. Cryptographic vulnerabilities may arise from weaknesses in encryption algorithms, key management, or protocol implementations, compromising the confidentiality, integrity, and authenticity of sensitive data.

Effective mitigation of cryptographic vulnerabilities requires establishing secure cryptographic configurations, keeping software and protocols up-to-date, and implementing security best practices such as secure key exchange and authentication. Additionally, proactively identifying and remediating vulnerabilities, such as weak keys or outdated encryption protocols, is essential to preventing attacks that exploit cryptographic weaknesses.

7.9 LAB: Pass the hash (Walkthrough)

IT-Labs are not printable at this time.

7.10 LAB: Cross-site scripting (XSS) attacks (Walkthrough)

IT-Labs are not printable at this time.

©zyBooks 12/12/24 18:05 2172291
Daren Diaz
OUCYBS3213FreezeFall2024