Module III

Software vulnerabilities: Buffer and stack overflow, Crosssite scripting(XSS) , and vulnerabilities, SQL injection and vulnerabilities , Phishing.

SOFTWARE VULNERABILITIES

-Software vulnerability is a glitch, flaw, or weakness **present in the software** or in an OS (Operating System). Of course, all systems include vulnerabilities. The thing is whether or not they're exploited to cause damage.

-Software vulnerability is an instance of a fault in the specification, development, or configuration of software such that its execution can violate the (implicit or explicit) security policy.

-It is generally found after the system has been released to the public

-Software vulnerabilities involve bugs in software. **Bugs** are coding errors that cause the system to make an unwanted action. All software has bugs of one form or another. Some bugs cause the system to crash, some cause connectivity to fail, some do not let a person to log in, and some cause printing not to work properly.

-Some bugs create information leakage or elevate user privileges or grant otherwise unauthorized access. These are security vulnerabilities. If all software has bugs and it is inevitable that some bugs will be security vulnerabilities, all software will have security vulnerabilities.

-It is important to consider that just about every device has software, and therefore security vulnerabilities. Operating systems are composed of software, as are web browsers, word processing programs, spreadsheets, video players, websites, and every other application. Even computer hardware includes a form of software called firmware. Networking equipment and cell phones also have software, and therefore inevitably security vulnerabilities.

Software vulnerabilities are explained by three ideal factors. These are:

a) **Existence** – The existence of vulnerability in the software.
b) **Access** – The possibility that hackers gain access to the vulnerability.
c) **Exploit** – The capability of the hacker to take advantage of that vulnerability via tools or with certain techniques.

Exploiting the weaknesses

Once an attacker identifies vulnerability, he can write a new computer program that uses that opportunity to get into a machine and take it over.

In recent years, attackers began targeting web browsers, which are allowed to connect to the internet and often to run small programs; they have much vulnerability that can be exploited. Those initial openings can give an attacker control of a target computer, which in turn can be used as a point of intrusion into a larger sensitive network.

Sometimes the vulnerabilities are discovered by the software developers themselves, or users or researchers who alert the company that a fix is needed. But other times, hackers or government spy agencies figure out how to break into systems and don't tell the company. These weaknesses are called "**zero days,"** because the developer has had no time to fix them. As a result, the software or hardware has been compromised until a patch or fix can be created and distributed to users.

The best way users can protect themselves is to regularly install software updates, as soon as updates are available.

Types of Software Vulnerabilities

- Buffer overflows
- Heap overflows
- Format string vulnerabilities
- Integer Overflow
- SQL Injection
- XSS/Cross site scripting

Buffer Overflow

A buffer is a sequential section of memory allocated to contain anything from a character string to an array of integers.

A buffer overflow, or buffer overrun, occurs when more data is put into a fixed-length buffer than the buffer can handle. The extra information, which has to go somewhere, can **overflow into adjacent memory space**, corrupting or overwriting the data held in that space.

This overflow usually results in a system crash, but it also creates the opportunity for an attacker to run arbitrary code or manipulate the coding errors to prompt malicious actions.

SQL Injection

SQL injection, also known as SQLI, is a common attack that **uses malicious SQL code for backend database manipulation** to access information that was not intended to be displayed. This information may include any number of items, including sensitive company data, user lists or private customer details.

Uncontrolled format string

It involves accepting **unchecked/unauthorized user inputs as a format string to execute**

Format string bugs most commonly appear when a programmer wishes to print a string containing user supplied data. The programmer may mistakenly write printf(buffer) instead of printf("%s", buffer). The first version interprets buffer as a format string, and parses any formatting instructions it may contain. The second version simply prints a string to the screen, as the programmer intended.

Heap overflow

A heap overflow attack is a type of a buffer overflow **attack that specifically targets the heap**, as it's name implies.

In these attacks the data in the heap is overwritten to exploit some aspect of the program.

A buffer overflow attack on a heap works by **corrupting information in the heap** in an effort to change specific things to be what they want.

Heap attacks are typically harder to perform than a Stack based attack because the presence of an overflow is not the only factor that determines the success, quite often the information in the heap must be corrupted not just overwritten.

Integer Overflow

An Integer Overflow is the condition that occurs when the result of an arithmetic operation, such as multiplication or addition, **exceeds the maximum size of the integer type** used to store it. When an integer overflow occurs, the interpreted value will appear to have "wrapped around" the maximum value and started again at the minimum value, similar to a clock that represents 13:00 by pointing at 1:00.

For example, an 8-bit signed integer on most common computer architectures has a maximum value of 127 and a minimum value of -128. If a programmer stores the value 127 in such a variable and adds 1 to it, the result should be 128. However, this value exceeds the maximum for this integer type, so the interpreted value will "wrap around" and become -128.

XSS or Cross site scripting

Cross-site scripting (XSS) is a type of injection security attack in which an attacker injects data, such as a **malicious script, into content from otherwise trusted websites**. Cross-site scripting attacks happen when an untrusted source is allowed to inject its own code into a web application, and that malicious code is included with dynamic content delivered to a victim's browser.

BUFFER OVERFLOW

A **buffer** is a temporary area for data storage. When more data (than was originally allocated to be stored) gets placed by a program or system process, the extra data overflows. It causes some of that data to leak out into other buffers, which can corrupt or overwrite whatever data they were holding.

A buffer overflow, or buffer overrun, is a common software coding mistake that an attacker could exploit to gain access to your system.

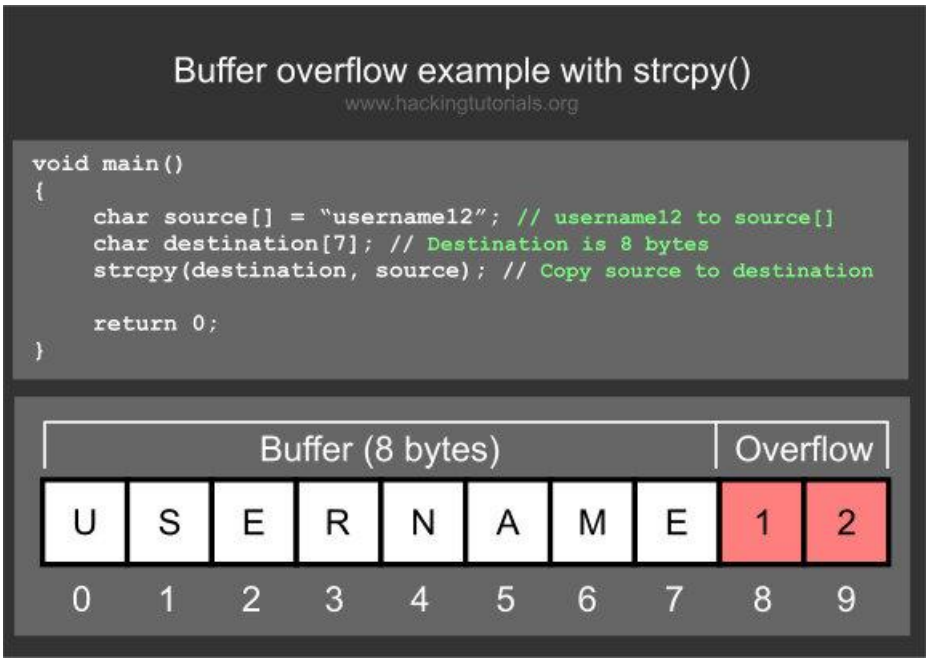
To effectively mitigate buffer overflow vulnerabilities, it is important to understand what buffer overflows are, what dangers they pose to your applications, and what techniques attackers use to successfully exploit these vulnerabilities.

Buffer overflows can be exploited by attackers with a goal of modifying a **computer's memory in order to undermine or take control of program execution.**

Programming languages like C and C++ are prone to buffer overflow attacks as they have no built-in protection against accessing or overwriting data in any part of their memory and as actors can perform direct memory manipulation with common programming constructs.

Modern programming languages like C#, Java and Perl reduce the chances of coding errors creating buffer overflow vulnerabilities, but buffer overflows can exist in any programming environment where direct

memory manipulation is allowed, whether through flaws in the program compiler, runtime libraries or features of the language itself.



This function uses 2 pointers as parameters, the source which points to the source array to copy from and the destination pointer to the character array to write to.

When the function is executed the source array of chars will be copied to the destination array and does not have a check for bounds when it does so. When the source buffer is larger than the destination buffer, than the buffer is overrun.

How do attackers exploit buffer overflows?

An attacker can deliberately feed a carefully crafted input into a program that will cause the program to try and store that input in a buffer that isn't large enough, overwriting portions of memory connected to the buffer space.

If the memory layout of the program is well-defined, the attacker can deliberately overwrite areas known to contain executable code. The attacker can then replace this code with his own executable code, which can drastically change how the program is intended to work.

How to protect against buffer overflow attacks

-Qualify all input from users and ensure input size validation

-Use strong programming languages like C#, Java etc

-Avoid functions like strcpy, strcat,gets etc

- **Address space randomization** - Randomly rearranges the address space locations of key data areas of a process. Buffer overflow attacks generally rely on knowing the exact location of important executable code, randomization of address spaces makes that nearly impossible.

- **Data execution prevention** - Marks certain areas of memory either executable or non-executable, preventing an exploit from running code found in a non-executable area.

What are the different types of buffer overflow attacks?

There are a number of different buffer overflow attacks which employ different strategies and target different pieces of code. Below are a few of the most well-known.

- **Stack overflow attack** - This is the most common type of buffer overflow attack and involves overflowing a buffer on stack
- **Heap overflow attack** - This type of attack targets data in the open memory pool known as the heap.
- **Integer overflow attack** - In an integer overflow, an arithmetic operation results in an integer (whole number) that is too large for the integer type meant to store it; this can result in a buffer overflow.
- **Unicode overflow** - A unicode overflow creates a buffer overflow by inserting unicode characters into an input that expect ASCII characters. (ASCII and unicode are encoding standards that let computers represent text. For example the letter 'a' is represented by the number 97 in ASCII. While ASCII codes only cover characters from Western languages, unicode can create characters for almost every written language on earth. Because there are so many more characters available in unicode, many unicode characters are larger than the largest ASCII character.)

STACK OVERFLOW

-A stack is contiguous block of memory containing data.

-Stack pointer (SP) – a register that points to the top of the stack.

-The bottom of the stack is at fixed address.

-Its size is dynamically adjusted by kernel at run time.

-CPU implements instructions to PUSH onto and POP off the stack.

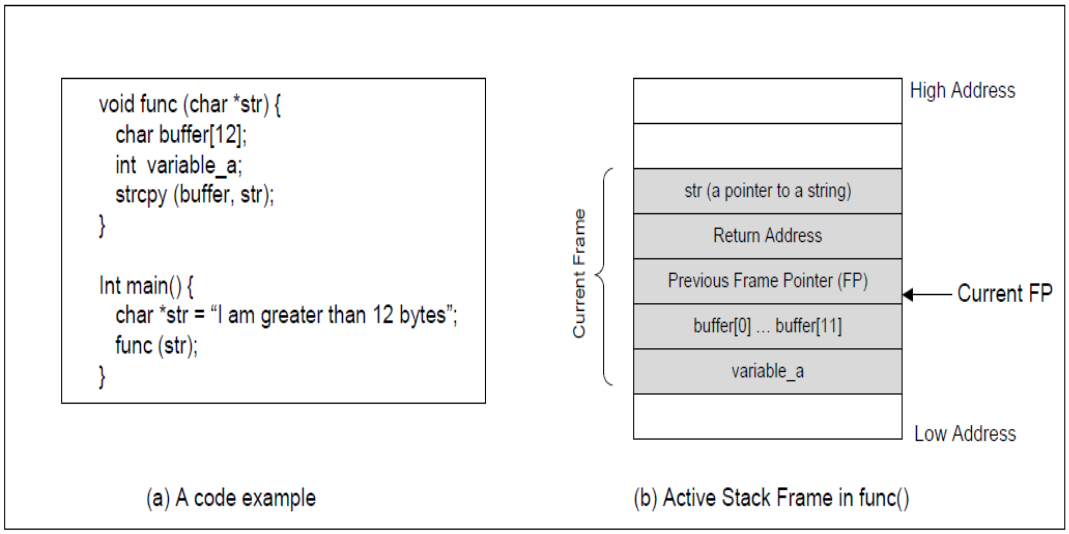
-A stack consists of logical stack frames that are pushed when calling a function and popped when returning. Frame pointer (FP) – points to a fixed location within a frame.

When a function is called, the return address, stack frame pointer and the variables are pushed on the stack (in that order).

-So the return address has a higher address as the buffer.

-When we overflow the buffer, the return address will be overwritten

The following figure shows the stack layout after the execution has entered the function func().



Stack Direction: Stack grows from high address to low address (while buffer grows from low address to high address)

Return Address: address to be executed after the function returns.

Frame Pointer (FP): is used to reference the local variables and the function parameters.

Buffer-Overflow Problem: The above program has a buffer-overflow problem.

– The function strcpy(buffer, str) copies the contents from str to buffer[].

– The string pointed by str has more than 12 chars, while the size of buffer[] is only 12.

– The function strcpy() does not check whether the boundary of buffer[] has reached. It only stops when seeing the end-of-string character '\0'.

– Therefore, contents in the memory above buffer[] will be overwritten by the characters at the end of str.

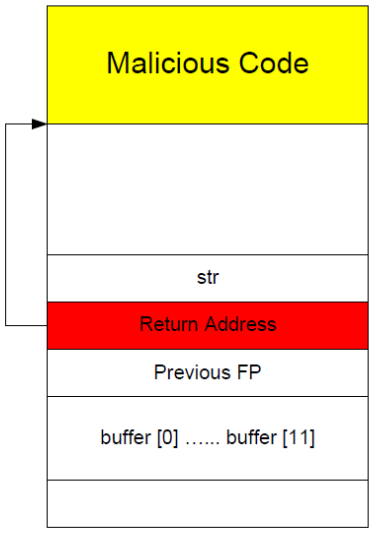
Exploit the Buffer-Overflow Vulnerability

To fully exploit stack buffer-overflow vulnerability, we need to solve several challenging problems.

1)Injecting the malicious code: We need to be able to inject the malicious code into the memory of the target process. This can be done if we can control the contents of the buffer in the targeted program.

For example, in the above example, the program gets the input from a file. We can store the malicious code in that file, and it will be read into the memory of the targeted program.

2)Jumping to the malicious code: With the malicious code already in the memory, if the targeted program can jump to the starting point of the malicious code, the attacker will be in control.



3)Writing malicious code: Writing a malicious code is not trivial. A special type of malicious code, shellcode, can be written.

Countermeasures

1. **Apply Secure Engineering Principles**

_ Use strong type language, e.g. java, C#, etc. With these languages, buffer overflows will be detected.

_ Use safe library functions.

– Functions that could have buffer overflow problem: gets, strcpy, strcat, sprintf, scanf, etc.

– These functions are safer: fgets, strncpy, strncat, and snprintf.

2.Systmetic Code Modification

StackShield: seperate control (return address) from data.

– It is a GNU C compiler extension that protects the return address.

– When a function is called, StackShield copies away the return address to a non-overflowable area.

StackGuard: mark the boundary of buffer

3.Operating System Approach

Address Space Randomization(already mentioned above)

Non-executable stack: From the attack, we can observe that the attackers put the malicious code in the stack, and jump to it. Since the stack is a place for data, not for code, we can configure the stack to be non-executable, and thus preventing the malicious code from being executed. This protection scheme is called ExecShield

SQL INJECTION

SQL injection, also known as SQLI, is a common attack that uses malicious SQL code for backend database manipulation to access information that was not intended to be displayed.

This information may include any number of items, including sensitive company data, user lists or private customer details.

A successful attack may result in the unauthorized viewing of user lists, the deletion of entire tables and, in certain cases, the attacker gaining administrative rights to a database, all of which are highly detrimental to a business.

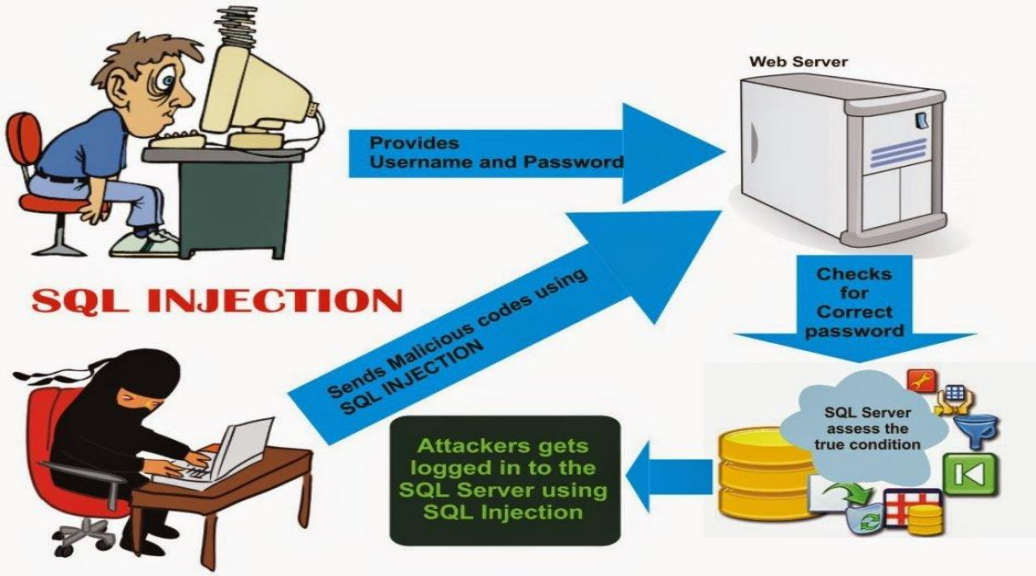
SQL injection is one of the most common web hacking techniques.

It is a type of injection attack

Most dangerous vulnerability

To perform this attack needs a web browser and some guesses to work to find important tables and field names

Once attacker realizes that the system is vulnerable to SQL Injection they are able to inject SQL Query through input field.



Attacker use SQL injection to find credentials of other users in database

To gain complete access to all data in database server

Alter data in database and add new data

Delete records from database even drop table

In some database server, you can access Operating System using database server

Attacker works on dynamic SQL statements. Dynamic SQL Statement is a statement that is generated at run time using parameters password from a web form or URI Query string

SQL injection usually occurs when you ask a user for input, like their username/userid, and instead of a name/id, the user gives you an SQL statement that you will **unknowingly** run on your database.

Look at the following example which creates a SELECT statement by adding a variable (txtUserId) to a select string. The variable is fetched from user input (getRequestString):

Example

txtUserId=getRequestString("UserId");

txtSQL = "SELECT * FROM Users WHERE UserId = " + txtUserId;

SQL Injection Based on 1=1 is Always True

Look at the example above again. The original purpose of the code was to create an SQL statement to select a user, with a given user id.

If there is nothing to prevent a user from entering "wrong" input, the user can enter some "smart" input like this:

UserId: 105 OR 1=1

Then, the SQL statement will look like this:

SELECT * FROM Users WHERE UserId = 105 OR 1=1;

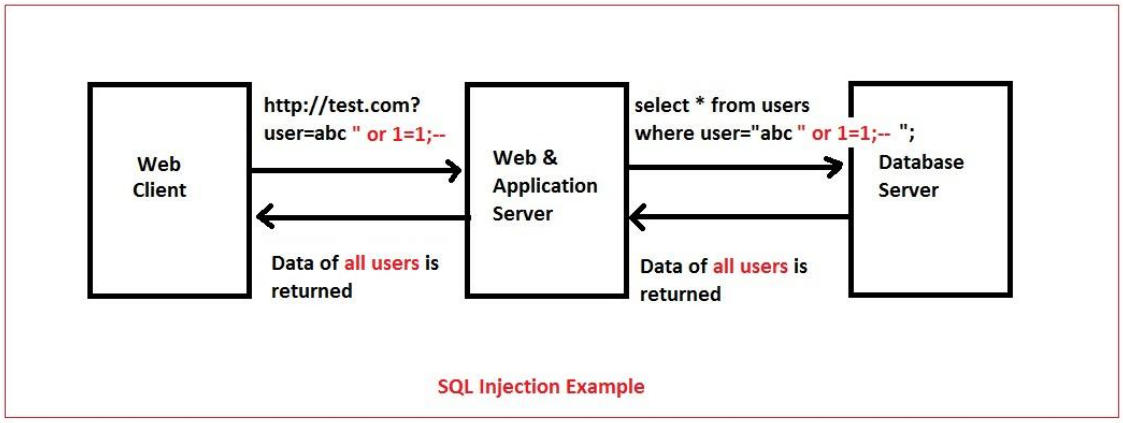
The SQL above is valid and will return ALL rows from the "Users" table, since **OR 1=1** is always TRUE.

Does the example above look dangerous? What if the "Users" table contains names and passwords?

The SQL statement above is much the same as this:

SELECT UserId, Name, Password FROM Users WHERE UserId = 105 or 1=1;

A hacker might get access to all the user names and passwords in a database, by simply inserting 105 OR 1=1 into the input field.



SQL Injection Example

SQL Injection Based on Batched SQL Statements

Most databases support batched SQL statement.

A batch of SQL statements is a group of two or more SQL statements, separated by semicolons.

The SQL statement below will return all rows from the "Users" table, then delete the "Suppliers" table.

Example

SELECT * FROM Users; DROP TABLE Suppliers

Look at the following example:

Example

txtUserId=getRequestString("UserId");

txtSQL = "SELECT * FROM Users WHERE UserId = " + txtUserId;

And the following input:

User id: 105; DROP

The valid SQL statement would look like this:

Result

SELECT * FROM Users WHERE UserId = 105; DROP TABLE Suppliers;

How to prevent SQL injection attack

1.Validating user input

You should do an input validation first to make sure the value is of the accepted type, length, format, etc. Only the input which passed the validation can be processed to the database. But remember, this method can only stop the most trivial attacks, it does not fix the underlying vulnerability.

2. Use SQL Parameters for Protection

To protect a web site from SQL injection, you can use SQL parameters.

SQL parameters are values that are added to an SQL query at execution time, in a controlled manner.

```
txtUserId=getRequestString("UserId");
txtSQL="SELECT*FROMUsersWHEREUserId=@0";
db.Execute(txtSQL,txtUserId);
```

Note that parameters are represented in the SQL statement by a @ marker.

The SQL engine checks each parameter to ensure that it is correct for its column and are treated literally, and not as part of the SQL to be executed.

3. Limiting privileges

Don't connect to your database using an account with root access unless required because the attackers might have access to the entire system. Therefore, it's best to use an account with limited privileges to limit the scope of damages in case of SQL Injection.

4. Hidding info from the error message

Error messages are useful for attackers to learn more about your database architecture, so be sure that you show only the necessary information. It's better to show a generic error message telling something goes wrong and encourage users to contact the technical support team in case the problem persists.

5. Updating your system

SQL injection vulnerability is a frequent programming error and it's discovered regularly, so it's vital to apply patches and updates your system to the most up-to-date version as you can, especially for your SQL Server.

6. Keeping database credentials separate and encrypted

If you are considering where to store your database credentials, also consider how much damaging it can be if it falls into the wrong hands. So always store your database credentials in a separate file and encrypt it securely to make sure that the attackers can't benefit much.

Also, don't store sensitive data if you don't need it and delete information when it's no longer in use.

7. Disabling shell and any other functionalities you don't need

Shell access could be very useful indeed for a hacker. That's why you should turn it off if possible. Remove or disable all functionalities that you don't need too.

Types of SQL Injection

SQL Injection can be classified into three major categories – *In-band SQLi*, *Inferential SQLi* and *Out-of-band SQLi*.

1.In-band SQLi (Classic SQLi)

In-band SQL Injection is the most common and easy-to-exploit of SQL Injection attacks. In-band SQL Injection occurs when an attacker is able to use the same communication channel to both launch the attack and gather results.

The two most common types of in-band SQL Injection are *Error-based SQLi* and *Union-based SQLi*.

Error-based SQLi

Error-based SQLi is an in-band SQL Injection technique that relies on error messages thrown by the database server to obtain information about the structure of the database. In some cases, error-based SQL injection alone is enough for an attacker to enumerate an entire database.

While errors are very useful during the development phase of a web application, they should be disabled on a live site, or logged to a file with restricted access instead.

Union-based SQLi

Union-based SQLi is an in-band SQL injection technique that leverages the UNION SQL operator to combine the results of two or more SELECT statements into a single result which is then returned as part of the HTTP response.

2. Inferential SQLi (Blind SQLi)

In an inferential SQLi attack, no data is actually transferred via the web application and the attacker would not be able to see the result of an attack in-band (which is why such attacks are commonly referred to as "blind SQL Injection attacks"). Instead, an attacker is able to reconstruct the database structure by sending payloads, observing the web application's response and the resulting behavior of the database server.

The two types of inferential SQL Injection are *Blind-boolean-based SQLi* and *Blind-time-based SQLi*.

Boolean-based (content-based) Blind SQLi

Boolean-based SQL Injection is an inferential SQL Injection technique that relies on sending an SQL query to the database which forces the application to return a different result depending on whether the query returns a TRUE or FALSE result.

Depending on the result, the content within the HTTP response will change, or remain the same. This allows an attacker to infer if the payload used returned true or false, even though no data from the database is returned.

Time-based Blind SQLi

Time-based SQL Injection is an inferential SQL Injection technique that relies on sending an SQL query to the database which forces the database to wait for a specified amount of time (in seconds) before responding. The response time will indicate to the attacker whether the result of the query is TRUE or FALSE.

Depending on the result, an HTTP response will be returned with a delay, or returned immediately. This allows an attacker to infer if the payload used returned true or false, even though no data from the database is returned.

3. Out-of-band SQLi

Out-of-band SQL Injection is not very common, mostly because it depends on features being enabled on the database server being used by the web application. Out-of-band SQL Injection occurs when an attacker is unable to use the same channel to launch the attack and gather results.

PHISHING ATTACK

Phishing is a type of social engineering attack often used to steal user data, including login credentials and credit card numbers.

It occurs when an attacker, masquerading as a trusted entity, dupes a victim into opening an email, instant message, or text message.

The recipient is then tricked into clicking a malicious link, which can lead to the installation of malware.

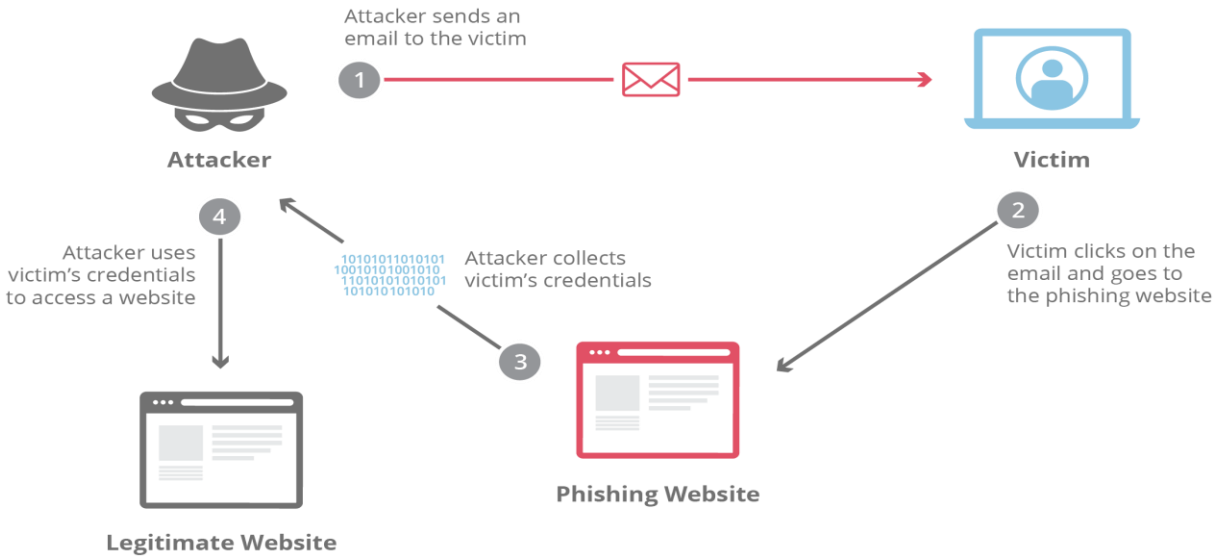
Example

Phishing email come from Bank directing the individual to a website where their username and password can be reset and collect login information.

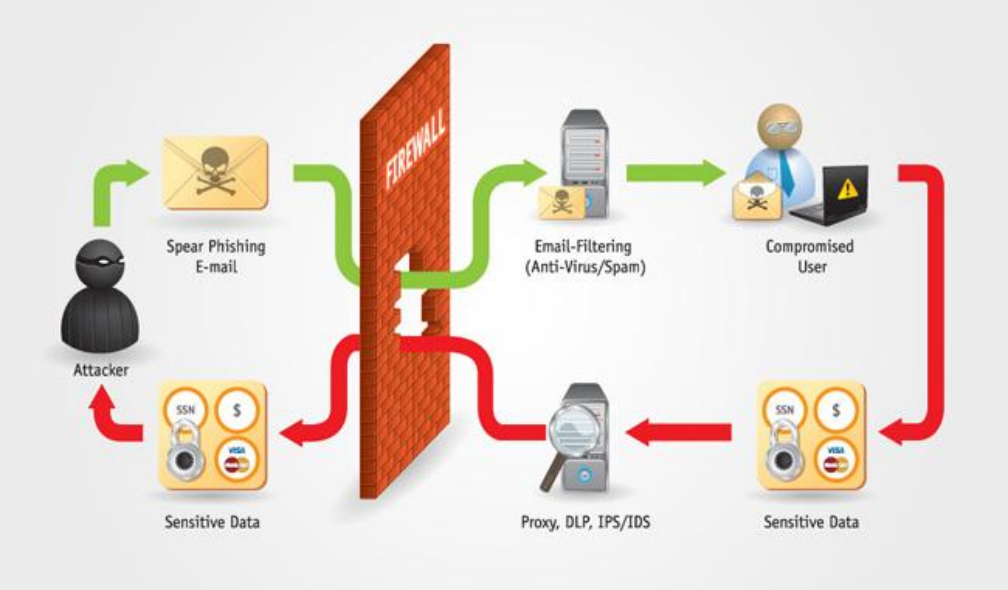
Goal of this attack is to steal sensitive data or install malware on victims machine.

What are the dangers of phishing attacks?

Sometimes attackers are satisfied with getting a victim's credit card information or other personal data for financial gain. Other times, phishing emails are sent to obtain employee login information or other details for use in an advanced attack against a specific company. Cybercrime attacks such as advanced persistent threats (APTs) and ransomware often start with phishing.



Example



How to protect against phishing attacks?

1. User education

One way to protect your organization from phishing is user education. Education should involve all employees. High-level executives are often a target. Teach them how to recognize a phishing email and what to do when they receive one. Simulation exercises are also key for assessing how your employees react to a staged phishing attack.

2.Security technology

No single cyber security technology can prevent phishing attacks. Instead, organizations must take a layered approach to reduce the number of attacks and lessen their impact when they do occur. Network security technologies that should be implemented include email and web security, malware protection, user behavior monitoring, and access control.

3. Email authentication

4. Think before you click

5. Install antiphishing tool

6. Verify site security

7. Check your online account regularly

8. Keep your browser up-to-date

9. Use firewalls

Different types of phishing

1.**Deceptive phishing**

Deceptive phishing is the most common type of phishing. In this case, an attacker attempts to obtain confidential information from the victims. Attackers use the information to steal money or to launch other attacks. A fake email from a bank asking you to click a link and verify your account details is an example of deceptive phishing.

Eg:fake email from Bank

2. Spear phishing

Spear phishing targets specific individuals instead of a wide group of people. Attackers often research their victims on social media and other sites. That way, they can customize their communications and appear more authentic. Spear phishing is often the first step used to penetrate a company's defenses and carry out a targeted attack.

3. Whaling

When attackers go after a "big fish" like a CEO, it's called whaling. These attackers often spend considerable time profiling the target to find the opportune moment and means of stealing login credentials. Whaling is of particular concern because high-level executives are able to access a great deal of company information.

4. Pharming

Similar to phishing, pharming sends users to a fraudulent website that appears to be legitimate. However, in this case, victims do not even have to click a malicious link to be taken to the bogus site. Attackers can infect either the user's computer or the website's DNS server and redirect the user to a fake site even if the correct URL is typed in.

5. **Man-in-the-Middle Phishing** is harder to detect than many other forms of phishing. In these attacks hackers position themselves between the user and the legitimate website or system. They record the

information being entered but continue to pass it on so that users' transactions are not affected. Later they can sell or use the information or credentials collected when the user is not active on the system.

6. **Search Engine Phishing** occurs when phishers create websites with attractive (often too attractive) sounding offers and have them indexed legitimately with search engines. Users find the sites in the normal course of searching for products or services and are fooled into giving up their information.

7. Clone Phishing In this type phisher creates a cloned email. He does this by getting information such as content and recipient addresses from a legitimate email which was delivered previously, then he sends the same email with links replaced by malicious ones. He also employs address spoofing so that the email appears to be from the original sender. The email can claim to be a re-send of the original or an updated version as a trapping strategy

8. Content injection

Content injection is the technique where the phisher changes a part of the content on the page of a reliable website. This is done to mislead the user to go to a page outside the legitimate website where the user is then asked to enter personal information.

9.Link Manipulation

Link manipulation is the technique in which the phisher sends a link to a malicious website. When the user clicks on the deceptive link, it opens up the phisher's website instead of the website mentioned in the link.

10. Smishing (SMS Phishing)

Phishing conducted via Short Message Service (SMS), a telephone-based text messaging service. A smishing text, for example, attempts to entice a victim into revealing personal information via a link that leads to a phishing website.

Countermeasures

DO NOT respond to any email from unknown source or emails pretend to be from known source with request for divulging confidential information especially credentials of Internet banking, credit cards, debit cards, online wallets, mobile wallets etc.

Countermeasure against phishing

The defensive mechanisms to counter the phishing technique threats.

The Client-side – this includes the user's PC and desktop.

The Server-side – this includes the business' Internet visible systems and custom applications.

Enterprise Level – distributed technologies and third-party management services.

Client side :

- At the client-side, protection against phishing can be afforded by:
- Desktop protection technologies
-User application-level monitoring solutions
-Locking-down browser capabilities
-Digital signing and validation of email

-General security awareness

Server side:

-Improving customer awareness

- Providing validation information for official communications

-Ensuring that the Internet web application is securely developed and doesn't include easily exploitable attack vectors.

-Using strong token-based authentication systems

-Keeping naming(domain name) systems simple and understandable

Enterprise level:

- Automatic validation of sending e-mail server addresses

-Digital signing of e-mail services

-Monitoring of corporate domains and notification of "similar" registrations

-Perimeter or gateway protection agents

-Third-party managed services