**1.Difference between white, Red, gray & black hacking?**

**1.White Hat Hackers:** White hat hackers are the one who is authorized or the certified hackers who work for the government and organizations by performing penetration testing and identifying loopholes in their cybersecurity. They also ensure the protection from the malicious cyber crimes. They work under the rules and regulations provided by the government, that’s why they are called *Ethical hackers* or *Cybersecurity experts*.

**2.Black Hat Hackers:** They are often called *Crackers*. Black Hat Hackers can gain the unauthorized access of your system and destroy your vital data. The method of attacking they use common hacking practices they have learned earlier. They are considered to be as criminals and can be easily identified because of their malicious actions.

**3.Gray Hat Hackers:** Gray hat hackers fall somewhere in the category between white hat and black hat hackers. They are not legally authorized hackers. They work with both good and bad intentions; they can use their skills for personal gain. It all depends upon the hacker. If a gray hat hacker uses his skill for his personal gains, he/she is considered as black hat hackers.

**4.Blue Hat Hacker**s: They are much like the script kiddies; are beginners in the field of hacking. If anyone makes angry a script kiddie and he/she may take revenge, then they are considered as the blue hat hackers. Blue Hat hackers payback to those who have challenged them or angry them. Like the Script Kiddies, Blue hat hackers also have no desire to learn.

**5.Red Hat Hackers:** They are also known as the eagle-eyed hackers. Like white hat hackers, red hat hackers also aims to halt the black hat hackers. There is a major difference in the way they operate. They become ruthless while dealing with malware actions of the black hat hackers. Red hat hacker will keep on attacking the hacker aggressively that the hacker may know it as well have to replace the whole system.

**2.Different kinds of attack on system?**

## **Web-based attacks**

These are the attacks which occur on a website or web applications. Some of the important web-based attacks are as follows-

**1. Injection attacks**

It is the attack in which some data will be injected into a web application to manipulate the application and fetch the required information.

Example- SQL Injection, code Injection, log Injection, XML Injection etc.

**2. DNS Spoofing**

DNS Spoofing is a type of computer security hacking. Whereby a data is introduced into a DNS resolver's cache causing the name server to return an incorrect IP address, diverting traffic to the attacker's computer or any other computer. The DNS spoofing attacks can go on for a long period of time without being detected and can cause serious security issues.

**3. Session Hijacking**

It is a security attack on a user session over a protected network. Web applications create cookies to store the state and user sessions. By stealing the cookies, an attacker can have access to all of the user data.

**4. Phishing**

Phishing is a type of attack which attempts to steal sensitive information like user login credentials and credit card number. It occurs when an attacker is masquerading as a trustworthy entity in electronic communication.

**5. Brute force**

It is a type of attack which uses a trial and error method. This attack generates a large number of guesses and validates them to obtain actual data like user password and personal identification number. This attack may be used by criminals to crack encrypted data, or by security, analysts to test an organization's network security.

**6. Denial of Service**

It is an attack which meant to make a server or network resource unavailable to the users. It accomplishes this by flooding the target with traffic or sending it information that triggers a crash. It uses the single system and single internet connection to attack a server. It can be classified into the following-

**Volume-based attacks**

Its goal is to saturate the bandwidth of the attacked site, and is measured in bit per second.

Protocol attacks- It consumes actual server resources, and is measured in a packet.

Application layer attacks- Its goal is to crash the web server and is measured in request per second.

**7. Dictionary attacks**

This type of attack stored the list of a commonly used password and validated them to get original password.

8. URL Interpretation

It is a type of attack where we can change the certain parts of a URL, and one can make a web server to deliver web pages for which he is not authorized to browse.

**9. File Inclusion attacks**

It is a type of attack that allows an attacker to access unauthorized or essential files which is available on the web server or to execute malicious files on the web server by making use of the include functionality.

**10. Man in the middle attacks**

It is a type of attack that allows an attacker to intercepts the connection between client and server and acts as a bridge between them. Due to this, an attacker will be able to read, insert and modify the data in the intercepted connection.

## **System-based attacks**

These are the attacks which are intended to compromise a computer or a computer network. Some of the important system-based attacks are as follows-

**1. Virus**

It is a type of malicious software program that spread throughout the computer files without the knowledge of a user. It is a self-replicating malicious computer program that replicates by inserting copies of itself into other computer programs when executed. It can also execute instructions that cause harm to the system.

**2. Worm**

It is a type of malware whose primary function is to replicate itself to spread to uninfected computers. It works same as the computer virus. Worms often originate from email attachments that appear to be from trusted senders.

**3. Trojan horse**

It is a malicious program that occurs unexpected changes to computer setting and unusual activity, even when the computer should be idle. It misleads the user of its true intent. It appears to be a normal application but when opened/executed some malicious code will run in the background.

**4. Backdoors**

It is a method that bypasses the normal authentication process. A developer may create a backdoor so that an application or operating system can be accessed for troubleshooting or other purposes.

**5. Bots**

A bot (short for "robot") is an automated process that interacts with other network services. Some bots program run automatically, while others only execute commands when they receive specific input. Common examples of bots program are the crawler, chatroom bots, and malicious bots

# **3.Interface in Java**

An interface in Java is a blueprint of a class. It has static constants and abstract methods.

The interface in Java is *a mechanism to achieve abstraction*. There can be only abstract methods in the Java interface, not method body. It is used to achieve abstraction and multiple inheritance in Java.

In other words, you can say that interfaces can have abstract methods and variables. It cannot have a method body.

## How to declare an interface?

An interface is declared by using the interface keyword. It provides total abstraction; means all the methods in an interface are declared with the empty body, and all the fields are public, static and final by default. A class that implements an interface must implement all the methods declared in the interface.

### Syntax:

1. **interface** <interface\_name>{
3. // declare constant fields
4. // declare methods that abstract
5. // by default.
6. }

**4.Use of lambda function in java?**

The three primary reasons why we need Java Lambda expressions are as follows:

* Lambda converts the code segment into an argument
* It is a method that can be created without instantiating a class
* Lambda can be treated as an Object

Now that we know the reasons for the necessity of the Java Lambda Expressions, let us continue and learn the syntax of Java Lambda Expressions.

A lambda expression can implement a functional interface by defining an anonymous function that can be passed as an argument to some method.

* Enables functional programming: All new JVM based languages take advantage of the functional paradigm in their applications, but programmers forced to work with Object-Oriented Programming (OOPS) till lambda expressions came. Hence lambda expressions enable us to write functional code.
* Readable and concise code: People have started using lambda expressions and reported that it can help to remove a huge number of lines from their code.
* Easy-to-Use APIs and Libraries: An API designed using lambda expressions can be easier to use and support other API.
* Enables support for parallel processing: A lambda expression can also enable us to write parallel processing because every processor is a multi-core processor nowadays.

## **Syntax**

(arg1, arg2...) -> { body } or(type1 arg1, type2 arg2...) -> { body }

**5.#Pragma in c/c++**

The #pragma in C is a directive that is provided by the C standard in order to provide extra required details to the C compiler. These extra details can be anything that was somehow not passed within the program or the code logic. These directives, known as pragma are prefixed by the STDC in the standard.

This pragma in C carries a special purpose and is responsible for turning on or off any feature within the program. Moving ahead, we will now learn about the standard syntax for the pragma in C along with an explanation.

This directive is a special purpose directive and is used to turn on or off some features. This type of directives are compiler-specific i.e., they vary from compiler to compiler. Some of the #pragma directives are discussed below:

**1.#pragma startup and #pragma exit:** These directives helps us to specify the functions that are needed to run before program startup( before the control passes to main()) and just before program exit (just before the control returns from main()).

**2.#pragma warn Directive:** This directive is used to hide the warning messages which are displayed during compilation. This may be useful for us when we have a large program and we want to solve all the errors before looking on warnings then by using it we can focus on errors by hiding all warnings. we can again let the warnings be visible by making slight changes in syntax.

**3.#pragma GCC poison:** This directive is supported by the GCC compiler and is used to remove an identifier completely from the program. If we want to block an identifier then we can use the #pragma GCC poison directive.

**4. #pragma GCC dependency:** The #pragma GCC dependency allows you to check the relative dates of the current file and another file. If the other file is more recent than the current file, a warning is issued. This is useful if the current file is derived from the other file, and should be regenerated.

**5. #pragma GCC system header:** This pragma takes no arguments. It causes the rest of the code in the current file to be treated as if it came from a system header.

**6.#pragma once:** The #pragma once directive has a very simple concept. The header file containing this directive is included only once even if the programmer includes it multiple times during a compilation. This is not included in any ISO C++ standard. This directive works similar to the #include guard idiom. Use of #pragma once saves the program from multiple inclusion optimisation.