



El dominio de algunos lenguajes de programación puede ser la diferencia entre detectar vulnerabilidades de manera rápida o quedarse atrás en el procedimiento. Estos lenguajes son esenciales para varias funciones como ensayos de penetración, estudio de malware y desarrollo de herramientas a medida para robustecer la seguridad.

1. Python



```
1 from math import sqrt
2 def esprimo(x):
3     if x%2 == 0:
4         if x == 2:
5             return True
6         else:
7             return False
8     # i tomará los valores de 3 a raíz de x, de dos en dos, es decir, los
    impares. Se usa int para tener solo números enteros
9     for i in range(3,int(sqrt(x))+1,2):
10         if x%i == 0:
11             return False
12     return True
13
14 # Comprobamos si varios números son primos
15 print(esprimo(2))
16 print(esprimo(4))
17 print(esprimo(524287))
```

Python es un lenguaje de alto nivel, conocido por su facilidad de uso y versatilidad.

¿Para qué lo usamos?

Es ideal para escribir scripts de automatización, análisis de tráfico de red, desarrollo de herramientas de análisis de malware y pruebas de penetración.

Ejemplo

Crear herramientas que monitorean el tráfico de red en tiempo real

```
4  #include <sys/socket.h>
5  #include <arpa/inet.h>
6  #include <unistd.h>
7
8  void scan_port(const char *ip, int port) {
9      int sock;
10     struct sockaddr_in server;
11
12     sock = socket(AF_INET, SOCK_STREAM, 0);
13     if (sock == -1) {
14         printf("No se pudo crear el socket\n");
15         return;
16     }
17
18     server.sin_addr.s_addr = inet_addr(ip);
19     server.sin_family = AF_INET;
20     server.sin_port = htons(port);
```



2. C

C es un lenguaje de programación de bajo nivel que interactúa directamente con el hardware y la memoria del sistema.

¿Para qué lo usamos?

Es crucial para la investigación de vulnerabilidades, especialmente para crear exploits o estudiar el comportamiento del sistema a nivel de memoria.

Ejemplo

Usar C para crear un exploit que aprovecha una vulnerabilidad de desbordamiento de búfer en un sistema de 32 bits.

3. C++

```
public:

float x; // Coordenadas del punto
float y;

        // Constructor
Punto() : x(0), y(0){ // Inicializamos las variables "x" e "y"
}
};

// Main para demostrar el funcionamiento de la clase

# include <iostream> // Esto nos permite utilizar "cout"

using namespace std;

int main () {
    Punto MiPunto;          // creamos un elemento de la clase Punto llamado MiPunto

    cout << "Coordenada X: " << MiPunto.x << endl;    // mostramos el valor acumulado en la variable x
    cout << "Coordenada Y: " << MiPunto.y << endl;    // mostramos el valor acumulado en la variable y
    getch(); // Le indicamos al programa que espere al buffer de entrada (detenerse)
```

C++ es un lenguaje de programación de nivel básico que ofrece una gestión de la memoria y los recursos del sistema, empleado para aplicaciones que demandan un alto rendimiento.

¿Para qué lo usamos?

Creación de exploits, análisis de malware, desarrollo de herramientas de seguridad y programas que interactúan con el hardware o el sistema operativo a nivel bajo

Ejemplo

Analizar cómo un programa manipula los datos, cómo, monitorizando la memoria del programa

```
4  #include <sys/socket.h>
5  #include <arpa/inet.h>
6  #include <unistd.h>
7
8  void scan_port(const char *ip, int port) {
9      int sock;
10     struct sockaddr_in server;
11
12     sock = socket(AF_INET, SOCK_STREAM, 0);
13     if (sock == -1) {
14         printf("No se pudo crear el socket\n");
15         return;
16     }
17
18     server.sin_addr.s_addr = inet_addr(ip);
19     server.sin_family = AF_INET;
20     server.sin_port = htons(port);
```



4. Bash

Bash es un lenguaje de scripting utilizado en sistemas Linux para ejecutar comandos en la terminal.

¿Para qué lo usamos?

Es fundamental para automatizar tareas en servidores Linux, que incluyen auditorías, escaneos de redes o análisis forense.

Ejemplo

Utilizar herramientas como nmap , theharvester para mapear vulnerabilidades y recopilar información.

5. PowerShell



```
Function Get-Adaptador {  
    Write-Host "Configuración de la IP"  
    Get-NetAdapter|ft -AutoSize  
    $script:interfaz = Read-Host "Introduzca la interfaz (IfIndex)"  
    $script:nombre = Read-Host "Introduzca el nombre (name)"  
    #Borramos datos  
    Remove-NetIPAddress -InterfaceIndex $interfaz -Confirm:$false  
    Remove-NetRoute -InterfaceIndex $interfaz -Confirm:$false  
}  
Function Ip-Fija {  
    Get-Adaptador  
    #Creamos la nueva IP  
    $ip = Read-host "Introduzca IP"  
    $mascara = Read-Host "Introduzca la máscara (nºs de unos)"  
    $gateway = Read-Host "Introduzca el gateway"  
    $dns1 = Read-host "Introduzca el primer DNS"  
    $dns2 = Read-host "Introduzca el segundo DNS"
```

PowerShell es un lenguaje de scripting creado por Microsoft, diseñado para la administración de sistemas en entornos Windows.

¿Para qué lo usamos?

Aministración y automatización de tareas en sistemas Windows, y para realizar auditorías o pruebas de penetración en entornos de servidores Windows

Ejemplo

Crear scripts que detecten contraseñas débiles o cuentas no autorizadas.

Nota

En mi perfil puedes encontrar un repositorio con ejercicios de Powershell

```
17     if (num % 2 === 0 || num % 3 === 0) return false;
18     for (let i = 5; i * i <= num; i += 6) {
19         if (num % i === 0 || num % (i + 2) === 0) return false;
20     }
21     return true;
22 }
23
24 function sumOfPrimesInRange(start, end) {
25     let sum = 0;
26     for (let i = start; i <= end; i++) {
27         if (isPrime(i)) sum += i;
28     }
29     return sum;
30 }
```



6. JavaScript

JavaScript es un lenguaje de programación que se utiliza principalmente en el desarrollo web para añadir interactividad a las páginas.

¿Para qué lo usamos?

Es útil para detectar vulnerabilidades en aplicaciones web, como XSS (Cross-Site Scripting) y CSRF (Cross-Site Request Forgery).

Ejemplo

Detectar cómo un atacante podría ejecutar código malicioso en un sitio vulnerable.

7. Go

```
1 package main
2
3 import (
4     "fmt"
5     "log"
6     "net/http"
7 )
8
9 func holaFunc(w http.ResponseWriter, r *http.Request) {
10     fmt.Fprintln(w, "Hola mundo!")
11 }
12
13 func main() {
14     http.HandleFunc("/", holaFunc)
```

Go es un lenguaje de programación desarrollado por Google, conocido por su rendimiento y capacidad para manejar tareas concurrentes.

¿Para qué lo usamos?

Es útil para escribir herramientas rápidas y eficientes, como escáneres de red, análisis de malware o exploits.

Ejemplo

Se puede usar Go para desarrollar un escáner de vulnerabilidades de alta velocidad que analice grandes redes en poco tiempo.

```
47 | end
48 end
49
50 class Duck
51 | include ActLikeADuck
52 end
53
54 class Person
55 end
56
57 normal_person = Person.new
58 captured_person = Person.new
```



8. Ruby

Ruby es un lenguaje de programación dinámico y de alto nivel, conocido por su sintaxis sencilla y legible, que permite desarrollar aplicaciones de manera rápida.

¿Para qué lo usamos?

Es usado para crear scripts y herramientas de pentesting, incluyendo exploit frameworks como **Metasploit**.

Ejemplo

Automatizar Script de automatización de ataque de fuerza bruta

9. SQL



```
1  create table partidos(  
2  codigo int(3) primary key,  
3  fecha date not null,  
4  duracion int(2) not null,  
5  codigoequipovisitante int(3),  
6  codigoequipolocal int(3),  
7  constraint fk_visitante foreign key (codigoequipovisit  
8  constraint fk_local foreign key (codigoequipolocal) re  
9  constraint ck_fecha check (format(fecha, 'MM-dd') <> '  
10
```

SQL es un lenguaje de programación utilizado para gestionar bases de datos relacionales, facilitando la manipulación de datos y su consulta.

¿Para qué lo usamos?

SQL es crucial para realizar ataques como SQL Injection, donde un atacante manipula consultas SQL para acceder a datos sensibles.

Ejemplo

Explotar una vulnerabilidad de SQL Injection en un formulario web, obteniendo acceso no autorizado a la base de datos de la aplicación

```
// Consulta para buscar el usuario en la base de datos
// Se utilizan sentencias preparadas para prevenir inyecciones SQL
$consulta_usuario = $conexion->prepare('SELECT * FROM usuarios WHERE nombre_usuario = ? LIMIT 1');
$consulta_usuario->execute([$nombre_usuario]);

// Verifica si se ha encontrado un usuario
if ($consulta_usuario->rowCount() > 0) {
    // Obtiene los datos del usuario
    $usuario_encontrado = $consulta_usuario->fetch(PDO::FETCH_ASSOC);

    // Comprobar si la contraseña enviada es correcta
    // Utiliza algoritmos seguros como BCrypt o Argon2i
    if (password_verify($contrasena, $usuario_encontrado['contrasena'])) {
        // Credenciales correctas, iniciar sesión
    } else {
        // Credenciales incorrectas, mostrar mensaje de error
    }
} else {
    // Nombre de usuario no existe, mostrar mensaje de error

    // Nótese que, por seguridad, aquí y arriba se debe mostrar un error
    // genérico como "Credenciales incorrectas", así de esta forma un posible
    // atacante no sabrá si ha ingresado el nombre de usuario correcto
```



10. PHP

PHP es un lenguaje de programación del lado del servidor utilizado principalmente en el desarrollo web, especialmente en la creación de sitios dinámicos.

¿Para qué lo usamos?

PHP se utiliza para auditar aplicaciones web, buscar vulnerabilidades en el código y crear exploits para aplicaciones mal configuradas.

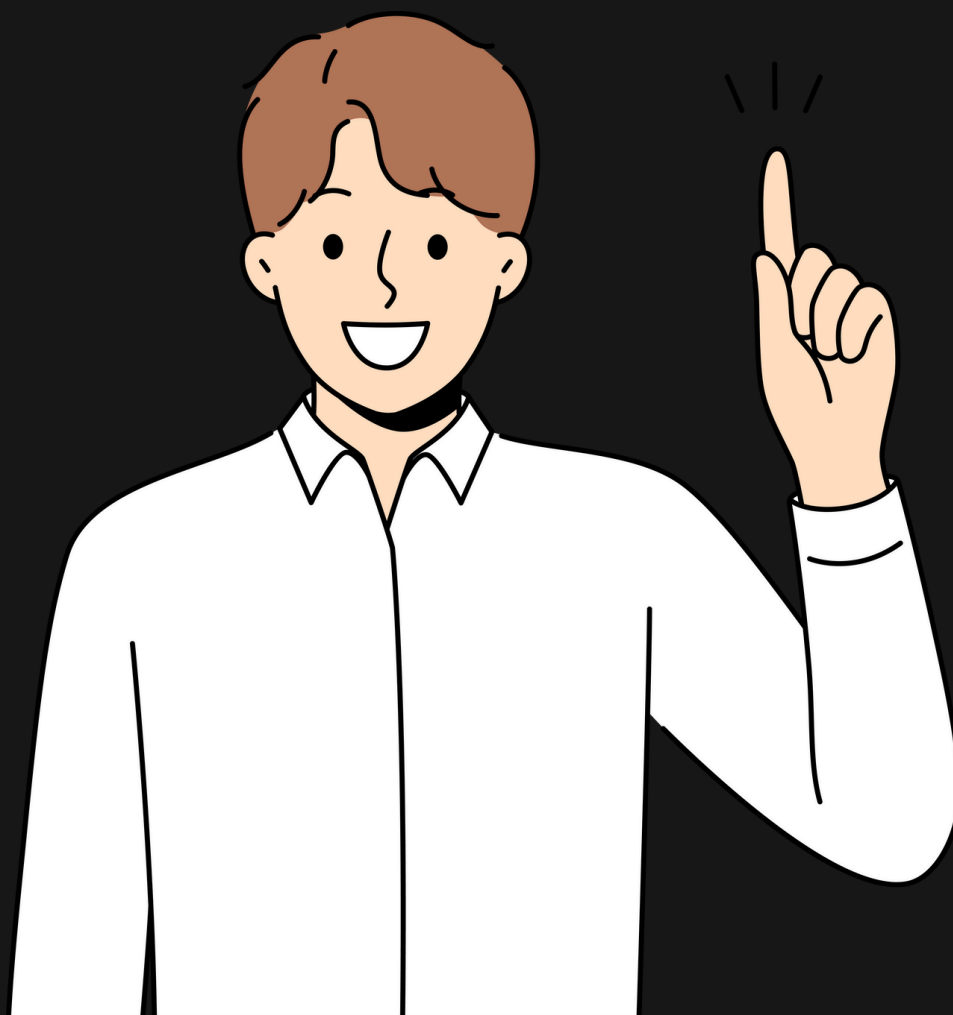
Ejemplo

Realizar un ataque de inclusión de archivos remotos en una aplicación web vulnerable, que permita ejecutar código malicioso (RFI).

Recuerda



No es necesario dominar todos estos lenguajes de programación de inmediato. La clave está en aprender de forma constante y mejorar poco a poco. No se trata de conocerlo todo, sino de **especializarse** en lo que más te interese o sea relevante para tu área de trabajo. ¡La práctica y la paciencia te llevarán lejos!



MUCHAS GRACIAS



[@saulruizplaza](#)