## Explorando Hack The Box (HTB):

# Del entrenamiento al desafío en vivo

PhiloCyber 2024

# Quién soy?

- Desde un secundario técnico, trabajando como profe de educación física y atletismo hasta ya unos años en cyber (la vida nos da sorpresas).
- Amo lo que hacemos, ciberseguridad en todas sus formas y colores.
- Tengo 6 años trabajando en diferentes empresas como Deloitte (ARG), Onapsis (ARG), Mercado Libre (ARG) y Bastion Sec. Group (NZ).
- Me gusta organizar eventos y crear contenido, aunque soy un poco (bastante) lento con los videos (falta poco para el próximo!).
- Disfruto muchísimo nadar en aguas abiertas.
- Amo los animales, especialmente los perros.

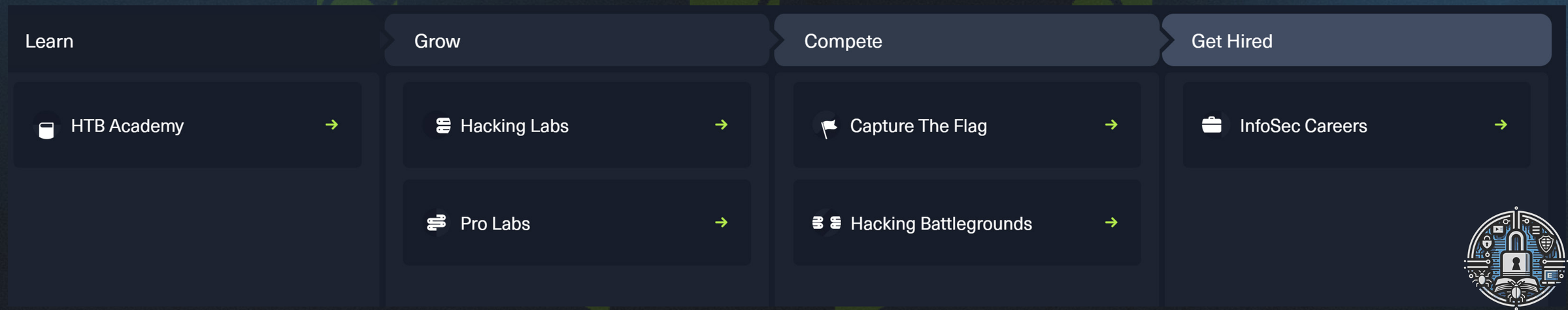
# Indice

## 1. Introducción

# ¿Qué es Hack The Box (HTB)?

Hack The Box (HTB) es una plataforma que se especializa en la **formación y evaluación** en ciberseguridad. Ofrecen una **gran variedad** de servicios diseñados para usuarios de todos los niveles, desde principiantes hasta profesionales experimentados.

Actualmente tiene **3 millones de miembros** en toda su plataforma, más de 700 Capture the Flag (CTF) creados y más de 600 laboratorios de entrenamiento.



# HISTORIA DE HTB

## Una breve visita a la historia

**2017**

- HTB fue fundado por ch4p, azik, gOblin.
- Creación del primer Pro Lab **RastaLabs**.

**2018**

- Lanzamiento de nuestra plataforma CTF.
- Nuestro servidor de Discord dedicado.

**2019**

- Primer CTF Universitario.
- Primera HTB Meetup.

**2020**

- Llegó la HTB Academy.
- Alianza con Parrot OS.
- 1er HTB Business y Community CTF anual.
- 100 Empleados cap.

**2021**

- 1 Millón de usuarios!
- Plataforma dedicada para CTFs.
- Nuevas certificaciones! CPTS y CBBH.

**2022**

- Nueva certificación Blue Team (**CDSA**).
- 2 Millones de usuarios!
- 100 Universidades registradas
- Más y mejores desafíos, CTFs, boxes y módulos de la Academy.

**2023**

**PRESENTE**

- Primera certificación avanzada(**CWEE**).
- Sherlocks y desafíos AI/ML.
- Nuevos Labs.
- Active Directory Penentration Testing, Academy Path.
- 200 Empleados cap.

# Fin

# ¿Preguntas?

# 4. Server-Side Template Injection (SSTI)

## ¿Qué es exactamente?

Esta vulnerabilidad, en español "**Inyección de plantillas del lado del servidor** ", ocurre cuando una aplicación permite a un atacante **insertar código nativo del template de forma arbitraria**.

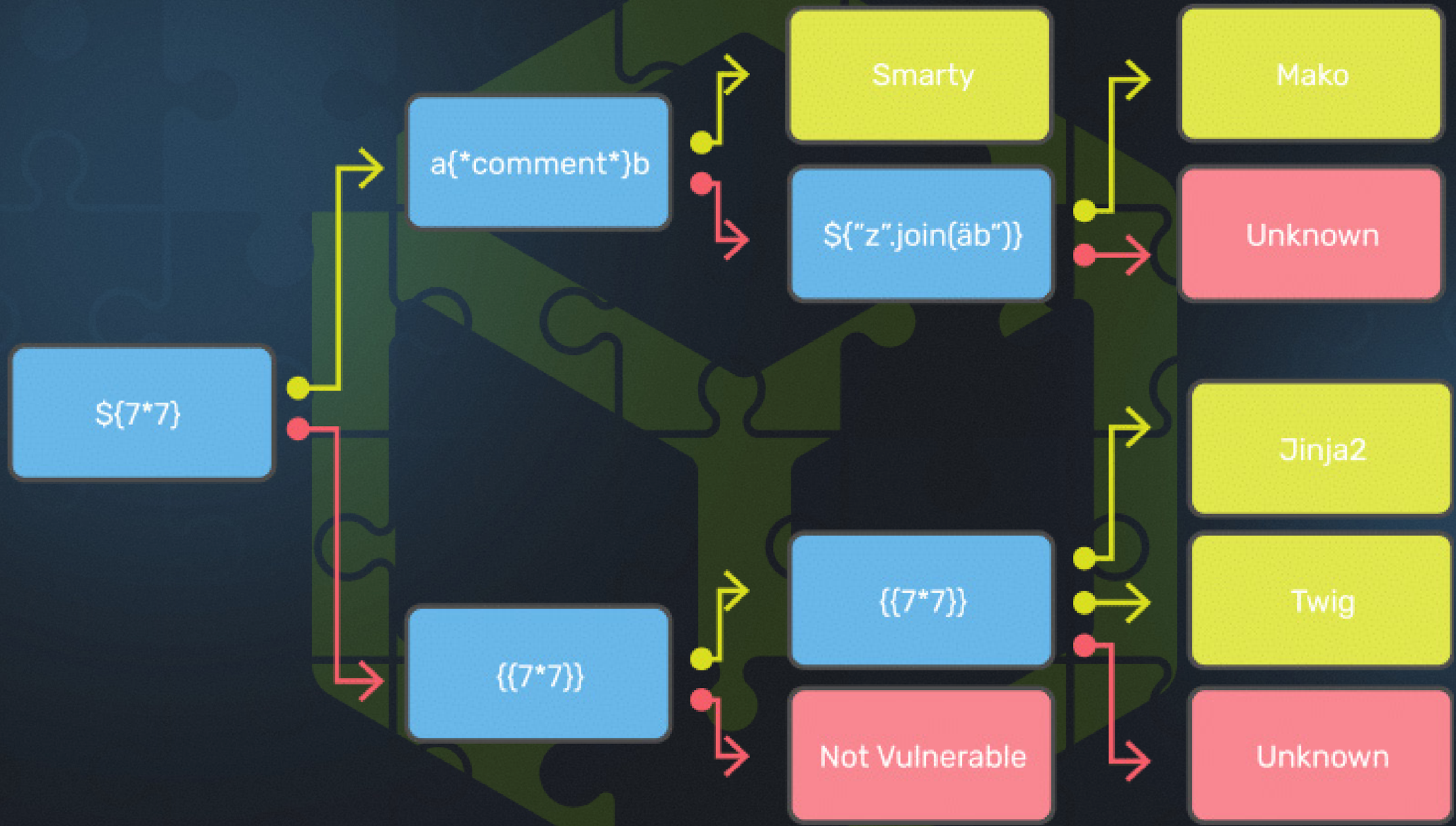
Posteriormente este código va a ser procesado por el servidor, ofreciendo al atacante **acceso no autorizado al sistema o incluso la posibilidad de ejecutar comandos en el servidor**.

# Template Engine o Motores de plantillas

Los motores de plantillas son herramientas que permiten **combinar código estático con datos dinámicos** para luego renderizar un HTML actualizado con esa nueva información.

Por ejemplo, en lugar de tener una página de registro de usuarios estática, podríamos personalizarla mostrando el nombre del usuario que se registró. Incluso se podría construir la página desde el contenido de una base de datos.

Ejemplos de motores de plantillas incluyen Jinja2 (Python), Thymeleaf (Java), y Twig (PHP)."


${7*7}
a{*comment*}b
{{7*7}}
Smarty
${"z".join("ab")}
{{7*7}}
Not Vulnerable
Mako
Unknown
Jinja2
Twig
Unknown

# Ejemplo básico de Jinja2 en código:

```
template = "Gracias por registrarte, {{ nombre }}!"
rendered = template.render(nombre="Philo")
print(rendered)  # El output va a ser: Gracias por registrarte, Philo!
```

Donde el contenido dentro de {{ }} es **evaluado como un valor dinámico a ser utilizado para renderizar el sitio**, en este caso, Jinja2. Si es vulnerable, todo lo que esté dentro de las llaves se evaluará como una expresión Python (código nativa del template) en el servidor y su resultado se insertará en la plantilla renderizada.

# ¿Pero... Cómo surge la vulnerabilidad?

El gran problema surge cuando la aplicación permite que un usuario controle o modifique las entradas utilizadas dentro de la plantilla **sin validación ni saneamiento adecuado**.

Esto podría potencialmente permitirnos inyectar código arbitrario, el cual posteriormente podría ser procesador por el template engine.

```
template = "Gracias por registrarte, {{ nombre }}!"
rendered = template.render(usuario="{{request.application.__globals__.__builtins__.__import__('os').popen('ls').read()}}")
# or
rendered = template.render(usuario="{{request.application.__globals__.__builtins__.__import__('os').popen('dir').read()}}")
print(rendered)  # El output va a ser: Gracias por registrarte, [directorio donde la app esta siendo ejecutada] !
```

# Descomponiendo el comando

1. **request.application.__globals__**: es un objeto que accede al espacio global del entorno de Jinja2, pudiendo acceder a las variables y funciones disponibles globalmente en Python

2. **__builtins__**: es un diccionario que contiene todas las funciones y clases integradas de Python (print, len, import, etc). Pudiendo importar cualquier módulo de Python que no esté explícitamente deshabilitado.

# Descomponiendo el comando

4. **__import__('os')**: importa dinámicamente el módulo 'os', proporcionando utilidades para interactuar con el sistema operativo, como la ejecución de comandos, acceso a archivos, etc.
5. **.popen('dir')**: usa el método popen del módulo os para ejecutar un comando en el sistema operativo.
6. **.read()**: lee la salida del comando ejecutado por popen y devuelve el resultado como una cadena de texto.

# Potenciales Impactos

- Acceso a variables del sistema.
- Ejecución de comandos en el servidor.
- Acceso total al servidor y a sus recursos.

# LOCAL DEMO

# HTB TEMPLATED CHALLENGE

# Mitigaciones y recomendaciones

1. Nunca pasemos los datos controlados por el usuario directamente al template engine de forma insegura, **tratar la información solo como texto**.
2. Busquemos **aislar el entorno** de ejecución del template engine, ejecutándolo en un container (Docker) con la menor cantidad de permisos.
3. **Eliminemos funciones sensibles** como eval, import, os.system, entre otras.
4. Siempre apliquemos algún tipo de **validación de datos y escape de entradas**.
5. Realizar constantes chequeos de seguridad para determinar potenciales **versiones vulnerables a ser actualizadas**.

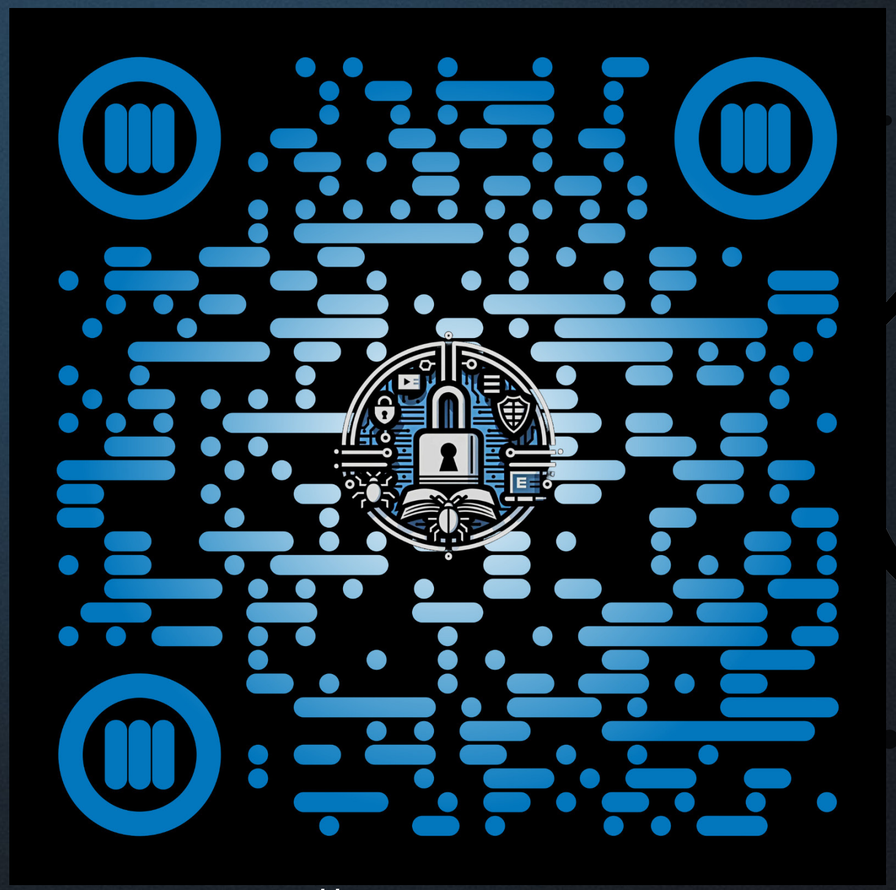
# Recursos y links útiles:

- Template Engines Injection 101
- Server-Side Template Injection research the James Kettle
- Server-side template injection | Burp Academy
- Server-side Attacks | HTB Academy
- Server Side Template Injection with Jinja2 for you
- Secure Templating with Jinja2: Understanding SSTI and Jinja2 Sandbox Environment
- Tool automatizada **tplmap**

# Mi recomendación:

HTB ACADEMY

# ¡Gracias por su tiempo!



https://www.philocyber.com

https://referral.hackthebox.com/mzwzrcC

**Ricardo Prieto**

Penetration Tester y Creador de Contenido