**Compatibilidad entre Windows y Linux:**

* En **Windows**, puedes usar PowerShell para comandos avanzados.
* En **Linux**, puedes ejecutar comandos Bash.

**Extensiones posibles:**

1. **Crear funciones específicas para comandos**: Puedes crear funciones que ejecuten comandos predefinidos para realizar tareas comunes en ambos sistemas (como listar archivos, crear carpetas, etc.).
2. **Añadir manejo de permisos**: Algunos comandos requieren permisos especiales (como sudo en Linux), y puedes agregar lógica para manejar estos casos.

 **Archivo de configuración accesible solo desde la aplicación**:

* El archivo de configuración debe ser **cifrado** y solo tu aplicación podrá leerlo y descifrarlo.

 **Contraseña para acceso**:

* Puedes implementar un **sistema de contraseña** para el descifrado, permitiendo que solo la aplicación con la clave adecuada pueda leer el archivo.

 **Cifrado de archivo**:

* Usaremos un cifrado simétrico, donde una clave secreta cifra y descifra los datos. Para ello, una opción popular es el cifrado AES (Advanced Encryption Standard).

 **Escalabilidad**:

* El archivo puede contener información como los procesos a verificar, puertos a monitorizar, y otros parámetros.

**Ajustar tu lógica PowerShell o Python**

Vamos a hacer que PowerShell lea esta configuración de un archivo YAML, una vez descifrada, y verifique los procesos, puertos y DNS de forma dinámica.

Puedes adaptar el script para que maneje otros escenarios de monitoreo como configuraciones de red, logs, entre otros.

Puedes extender este sistema para automatizar más tareas de monitoreo en servidores tanto Windows como Linux. ¿Te gustaría agregar funcionalidades adicionales?

**Arquitectura Lógica**

1. **Ckecker Device**.exe (único ejecutable) clase principal
2. **Ckecker Device** 
   1. validamos si la carpeta (Security) ya esta creada,

si esta creada, validamos la carpeta data,

si la carpeta data no existe. Debemos bajar los archivos correspondientes del servicio, realizar el flujo de codificación.

* 1. Si la carpeta data existe realizamos el proceso de decodificación para validar el identificador, debemos comparar el identificador local contra el identificador del servicio.

Si los identificadores coinciden continuar el flujo  
Si los identificadores no coinciden, se debe bajar los archivos de configuración del servicio, realizar el flujo de codificación.

1. Flujo codificación clase (Encode.py)
2. Continuación de flujo (Checker.py)