**Especificación de requisitos de software**

**Proyecto: [Comando de Control]**

Calendario

Descripción generada automáticamente con confianza media

 [octubre 2024]

# Ficha del documento

|  |  |  |  |
| --- | --- | --- | --- |
| **Fecha** | **Revisión** | **Autor** | **Verificado dep. calidad.** |
| 17/09/2024 | 1.0 | Jose Imbaquinga  Angelo Yanacallo |  |

Documento validado por las partes en fecha: 17/10/2024

|  |  |
| --- | --- |
| Por el cliente | Por la empresa suministradora |
|  |  |
| Fdo. D./ Dña. Coronel Christian Michel Vizcaino Villavicencio | Fdo. D./Dña Jose Imbaquinga  Angelo Yanacallo |

# Contenido

[FICHA DEL DOCUMENTO 3](#_TOC_250035)

[CONTENIDO 4](#_TOC_250034)

1. [INTRODUCCIÓN 6](#_TOC_250033)
   1. [Propósito 6](#_TOC_250032)
   2. [Alcance 6](#_TOC_250031)
   3. [Personal involucrado 6](#_TOC_250030)
   4. [Definiciones, acrónimos y abreviaturas 6](#_TOC_250029)
   5. [Resumen 6](#_TOC_250027)
2. [DESCRIPCIÓN GENERAL 7](#_TOC_250026)
   1. [Perspectiva del producto 7](#_TOC_250025)
   2. [Funcionalidad del producto 7](#_TOC_250024)
   3. [Restricciones 7](#_TOC_250022)
   4. [Dependencias 7](#_TOC_250021)
3. [REQUISITOS ESPECÍFICOS 7](#_TOC_250019)
   1. [Requisitos comunes de los interfaces 8](#_TOC_250018)
      1. [Interfaces de usuario 8](#_TOC_250017)
      2. [Interfaces de hardware 8](#_TOC_250016)
      3. [Interfaces de comunicación 8](#_TOC_250014)
   2. [Requisitos funcionales 8](#_TOC_250013)
      1. [Requisito funcional 1 9](#_TOC_250012)
      2. [Requisito funcional 2 9](#_TOC_250011)
      3. [Requisito funcional 3 9](#_TOC_250010)
      4. [Requisito funcional 4 9](#_TOC_250009)
      5. Requisito funcional 5
   3. [Requisitos no funcionales 9](#_TOC_250008)
      1. [Seguridad 9](#_TOC_250006)
      2. [Disponibilidad 9](#_TOC_250004)
      3. [Mantenibilidad 10](#_TOC_250003)

# Introducción

* 1. **Propósito**

#### Usar de una forma segura un malware (comando de control) en un entorno web, para gestionar y manejar dicho malware de manera grafica e interactiva. Proporcionar una guía técnica sobre el proyecto **"Comando de Control"**, desarrollado para el COCIBER.

## Alcance

#### Desarrollar una plataforma web que permite a los usuarios generar y ejecutar scripts de malware de manera controlada. Esto incluye la configuración de máquinas virtuales, la instalación de herramientas como Visual Studio y Node.js, y la implementación de un sistema que facilita la conexión entre diferentes equipos. Aborda la interacción del malware con el sistema y la visualización del estado de las redes.

## Personal involucrado

|  |  |
| --- | --- |
| Nombre | Jose Imbaquinga |
| Rol | Desarrollador Back-end |
| Categoría profesional | Pasante |
| Responsabilidades | Desarrollar, ejecutar, testear |
| Información de contacto | 0999819224 |

|  |  |
| --- | --- |
| Nombre | Angelo Yanacallo |
| Rol | Desarrollador Front-end |
| Categoría profesional | Pasante |
| Responsabilidades | Desarrollar, ejecutar, testear |
| Información de contacto | 0969636094 |

## Definiciones, acrónimos y abreviaturas

#### **COCIBER**: Comando de Ciberdefensa, la institución encargada de desarrollar y supervisar el proyecto.

#### **Malware**: Software diseñado para infiltrarse o dañar sistemas informáticos. En este proyecto, se utiliza con fines educativos y de análisis.

#### **Frontend**: La parte del sistema con la que los usuarios interactúan, hecha en React.

#### **Backend**: La parte del sistema que se encarga de la lógica, los servidores y desarrollada en Node.js.

#### **VM**: Abreviatura de Máquina Virtual, un entorno aislado donde se realizan pruebas del malware.

#### **JDK**: Java Development Kit, un conjunto de herramientas para desarrollar aplicaciones en Java.

#### **JWT**: JSON Web Token, un estándar abierto que se usa para intercambiar información de forma segura entre diferentes partes.

#### **SSH**: Secure Shell, un protocolo que permite acceder de forma segura y gestionar sistemas de manera remota

## Resumen

#### Guía completa sobre "Comando de Control", una herramienta creada para el COCIBER que permite generar y ejecutar scripts de malware en un entorno controlado. Está organizado en secciones clave: la introducción establece el propósito y el alcance del documento, además de definir términos importantes. Luego, se detalla la arquitectura del sistema, incluyendo los componentes del frontend y backend, así como las configuraciones y herramientas utilizadas. También se explican las funcionalidades del sistema, como la generación de scripts y la visualización del estado de las redes. Se incluyen pruebas y validaciones para garantizar la funcionalidad y seguridad del sistema, y se concluye con un resumen de los logros y recomendaciones para futuras mejoras.

# Descripción general

## Perspectiva del producto

#### El "Comando de Control" es una plataforma web creada para el COCIBER, dicha página permite a investigadores y analistas de ciberseguridad trabajar con scripts de malware en un entorno seguro. Puede funcionar de manera independiente, pero también está diseñada para integrarse fácilmente con otros sistemas de seguridad. La herramienta permite crear simular redes, lo que significa que se puede probar el malware sin poner en riesgo los sistemas reales de equipos físicos. Tiene una estructura modular, lo que hace sencillo añadir nuevas funciones, y cuenta con un sistema de comunicación que facilita la colaboración entre equipos. Además, su interfaz gráfica, desarrollada en React, es muy amigable, lo que hace que sea fácil de usar, y está diseñada para escalar, esto permite que más usuarios y recursos se sumen sin que el rendimiento se ralentice o se afecte.

.

## Funcionalidad del producto

#### El producto "Comando de Control" tiene varias funciones útiles para los usuarios. Permite crear scripts de malware personalizados de forma sencilla a través de una interfaz gráfica agradable, y soporta diferentes tipos de malware para adaptarse a distintos escenarios de prueba. Además, brinda un entorno seguro para ejecutar estos scripts en máquinas virtuales, evitando cualquier impacto en otros sistemas o en la red principal.

#### La interfaz, que está hecha en React, facilita la navegación y permite visualizar en tiempo real el estado de las redes y los scripts en ejecución. Los usuarios pueden monitorear cómo se comporta el malware en las máquinas virtuales y recibir alertas visuales que ayudan a entender lo que sucedió después de ejecutar los scripts. Por último, cuenta con un sistema de autenticación que garantiza un acceso seguro a la plataforma y protege las máquinas virtuales de accesos no autorizados.

#### 

#### ***Fig 1.*** Arquitectura de la página web integrada con el malware

## Restricciones

#### **Metodologías de Desarrollo:** El proyecto se desarrollará utilizando metodologías ágiles, lo que significa que habrá menos documentación formal y se necesitarán cambios frecuentes para adaptarse a nuevas necesidades.

#### **Lenguajes de Programación:** Se ha decidido usar Node.js para el backend y React para el frontend. Esto limita el uso de otros lenguajes o frameworks que no se alineen con esta elección.

#### **Normas de Seguridad:** Dado que el proyecto implica el manejo de malware, es importante tomar en cuenta normas de seguridad para garantizar de que las pruebas no afecten a sistemas externos y que los datos se manejen de forma segura.

#### **Sistema Operativo:** El sistema está diseñado para funcionar en entornos basados en Windows, lo que podría limitar su migración a sistemas operativos diferentes, como Linux o macOS.

#### **Recursos de Red:** Las pruebas del sistema pueden verse afectadas por limitaciones en la infraestructura de red disponible, como el ancho de banda o restricciones de acceso a ciertas direcciones IP.

#### **Licencias de Software:** Algunas herramientas y librerías pueden tener limitaciones de uso o requerir licencias que podrían complicar su implementación en el proyecto.

## Dependencias

#### Factores importantes que pueden afectar el cumplimiento de los requisitos del sistema "Comando de Control". Las dependencias que se deben tomar en cuenta son:

#### **Sistema Operativo:** La plataforma en la que se desarrolla y ejecuta el software depende de sistemas operativos específicos, como Windows o Linux. Si hay cambios en la compatibilidad de estos sistemas, podría afectar el rendimiento y la funcionalidad del producto.

#### **Herramientas de Desarrollo:** El uso de herramientas como Visual Studio, Node.js y JDK es muy importante para el desarrollo del proyecto. Si estas herramientas se actualizan, podríamos necesitar hacer ajustes en el código y la funcionalidad.

#### **Conexiones de Red:** La capacidad para generar y ejecutar scripts de malware depende de tener una infraestructura de red que sea estable. Cualquier cambio en la configuración de la red puede afectar la conexión entre las máquinas y el servidor.

#### **Políticas de Seguridad:** Las políticas de seguridad del COCIBER y otras entidades pueden influir en cómo se implementa y opera el sistema. Cualquier cambio en estas políticas podría requerir ajustes de gran impacto en el desarrollo y las pruebas.

# Requisitos especificados

# RF1: Loguin - Registro

# El sistema debe permitir ingresar con credenciales al sistema, a través de un Loguin, si el usuario no está registrado podrá hacerlos a través de un formulario de registro.

|  |  |
| --- | --- |
| Número de requisito | RF 1 |
| Nombre de requisito | Loguin y Registro |
| Tipo | **Requisito** Restricción |
| Fuente del requisito | Documento de especificaciones |
| Prioridad del requisito | Alta/Esencial **Media/Deseado** Baja/ Opcional |

# RF2: Generación de scripts de malware

# El sistema debe permitir a los usuarios generar scripts de malware de forma interactiva. Incluyendo opciones para seleccionar diferentes tipos de ataques y configurar parámetros específicos según sus necesidades.

|  |  |
| --- | --- |
| Número de requisito | RF 2 |
| Nombre de requisito | Generación de scripts de malware |
| Tipo | **Requisito** Restricción |
| Fuente del requisito | Documento de especificaciones |
| Prioridad del requisito | **Alta/Esencial** Media/Deseado Baja/ Opcional |

**RF 3: Soporte para múltiples tipos de scripts**El sistema debe ser capaz de generar scripts para varios tipos de malware, como troyanos, virus y gusanos, permitiendo a los usuarios elegir el tipo que desean utilizar.

|  |  |
| --- | --- |
| Número de requisito | RF 3 |
| Nombre de requisito | Soporte para múltiples tipos de scripts |
| Tipo | **Requisito** Restricción |
| Fuente del requisito | Documento de especificaciones |
| Prioridad del requisito | **Alta/Esencial** Media/Deseado Baja/ Opcional |

**RF 4: Interfaz gráfica para la generación**  
El sistema debe ofrecer una interfaz gráfica amigable que facilite a los usuarios la creación de scripts sin requerir conocimientos avanzados de programación.

|  |  |
| --- | --- |
| Número de requisito | RF 4 |
| Nombre de requisito | Interfaz gráfica para la generación |
| Tipo | **Requisito** Restricción |
| Fuente del requisito | Feedback de usuarios |
| Prioridad del requisito | Alta/Esencial **Media/Deseado** Baja/ Opcional |

**RF 5: Ejecución de scripts en entornos controlados**

Los scripts generados deben poder ejecutarse en entornos aislados, garantizando que no haya impacto en otros sistemas o redes.

|  |  |
| --- | --- |
| Número de requisito | RF 5 |
| Nombre de requisito | Ejecución de scripts en entornos controlados |
| Tipo | **Requisito** Restricción |
| Fuente del requisito | Documento de especificaciones |
| Prioridad del requisito | **Alta/Esencial** Media/Deseado Baja/ Opcional |

**RF 6: Visualización del estado de las redes**

El sistema debe incluir un módulo que permita a los usuarios visualizar en tiempo real el estado de las redes que son afectadas por los scripts que se ejecutan.

|  |  |
| --- | --- |
| Número de requisito | RF 6 |
| Nombre de requisito | Visualización del estado de las redes |
| Tipo | **Requisito** Restricción |
| Fuente del requisito | Documento de especificaciones |
| Prioridad del requisito | **Alta/Esencial** Media/Deseado Baja/ Opcional |

**RF 7: Autenticación y autorización de usuarios**  
El sistema debe implementar un sistema de autenticación robusto para asegurar que solo los usuarios autorizados tengan acceso y puedan ejecutar scripts.

|  |  |
| --- | --- |
| Número de requisito | RF 7 |
| Nombre de requisito | Autenticación y autorización de usuarios |
| Tipo | **Requisito** Restricción |
| Fuente del requisito | Normativas de seguridad |
| Prioridad del requisito | **Alta/Esencial** Media/Deseado Baja/ Opcional |

**RF 8: Registro de actividades de usuarios**  
El sistema debe mantener un registro detallado de todas las actividades realizadas por los usuarios, incluyendo la generación y ejecución de scripts, para fines de auditoría y seguridad.

|  |  |
| --- | --- |
| Número de requisito | RF 8 |
| Nombre de requisito | Registro de actividades de usuarios |
| Tipo | **Requisito** Restricción |
| Fuente del requisito | Normativas de seguridad |
| Prioridad del requisito | Alta/Esencial **Media/Deseado** Baja/ Opcional |

## Requisitos comunes de los interfaces

#### El sistema debe ofrecer interfaces que hagan más fácil la interacción del usuario con las funciones del software. Esto engloba tanto las entradas como las salidas, garantizando que los usuarios puedan gestionar y ejecutar comandos eficientemente.

### Interfaces de usuario

#### **Loguin y Registro:**

#### **Descripción:** El Loguin debe permitir ingresar credenciales validas como el nombre de usuario y contraseña, mientras que el registro debe permitir crear un usuario llenando campos como: Usuario, Correo Electrónico, Contraseña , Confirmar Contraseña.

#### **Elementos:** En el Loguin debe contar con el botón de INICIO para ingresar a la página, en cuanto al registro debe contar con el botón REGISTRAR para una vez llenados los campos validar el registro.

#### **Estilo:** Se usarán colores que se alineen con la paleta del sistema, preferiblemente tonos oscuros con texto en blanco para que sea más fácil de leer.

#### 

#### ***Fig 2.*** Loguin - Registro

#### **Página de Estado:**

#### **Descripción:** La página de "Ver Estado" debe mostrar de manera clara el estado del sistema, utilizando indicadores visuales, como semáforos, para representar el estado de las conexiones y las operaciones.

#### **Elementos:** Debe contar con botones de navegación como "ATAQUE", "GENERAR SCRIPT" y "REGRESAR".

#### **Estilo:** Se usarán colores que se alineen con la paleta del sistema, preferiblemente tonos oscuros con texto en blanco para que sea más fácil de leer.

#### 

#### ***Fig 3.*** Página de Ver Estado

#### **Página de CMD:**

#### **Descripción:** La interfaz del CMD debe permitir a los usuarios ingresar comandos y ver los resultados en un formato que sea fácil de leer.

#### **Elementos:** Debe incluir un área de texto para escribir los comandos, botones predefinidos como "whoami", "get-service", "get-process" y "get-computerinfo", así como un área de resultados que se desplace automáticamente hacia abajo.

#### **Estilo:** Se aplicará un diseño moderno con bordes redondeados para los botones y un fondo que contraste con el texto. Usar un degradado en los botones mejorará la experiencia visual.

#### Captura de pantalla de computadora Descripción generada automáticamente

#### ***Fig 4.*** Página del CMD

#### **Página de Generar Script:**

#### **Descripción:** Esta página debe permitir al usuario crear scripts personalizados, con validaciones visibles para cada entrada.

#### **Elementos:** Debe incluir campos de entrada para los parámetros del script, botones de acción y mensajes de error estilizados que muestren las validaciones.

#### **Estilo:** Se usará un fondo claro con texto oscuro, y los mensajes de error se mostrarán en blanco sobre un fondo rojo para que sean fácilmente visibles.

#### 

#### ***Fig 5.*** Página para Generar Scrip

#### **Interactividad:**

#### **Descripción:** La interfaz debe permitir a los usuarios interactuar sin demoras significativas.

#### **Elementos:** Los botones deben cambiar de color al pasar el mouse y ofrecer retroalimentación visual para las acciones realizadas, como ejecutar un comando.

#### **Estilo:** Debe incluir animaciones sutiles para las transiciones de página y efectos hover en los botones.

### Salidas del Sistema

### ****Resultados de Comandos:**** La salida de los comandos que se ejecutan en la página de CMD debe mostrarse de manera legible, con una separación clara entre cada resultado para que sea fácil de entender.

### ****Mensajes de Estado:**** La página de "Ver Estado" debe actualizarse en tiempo real para reflejar cualquier cambio en el sistema, proporcionando alertas o notificaciones cuando sea necesario.

### Interfaces de hardware

#### **Interfaz del CMD**

#### La interfaz del CMD está diseñada para interactuar con el sistema operativo y permitir la ejecución de comandos. Aquí están algunas de sus características clave:

#### **Conexión de WebSocket:** Se utiliza WebSocket para establecer una comunicación bidireccional entre el cliente y el servidor. Esto significa que se pueden enviar y recibir comandos en tiempo real, lo que asegura que las respuestas aparezcan de inmediato en la interfaz.

#### **Entradas de Comando:** La interfaz permite ingresar texto a través de un campo de entrada, así como hacer clic en botones predefinidos. Los comandos disponibles, como "whoami", "get-service", "get-process" y "get-computerinfo", están relacionados con funciones específicas en el backend.

#### **Gestión de Errores:** Se han implementado validaciones para garantizar que los comandos se ingresen correctamente. Si hay un error, se mostrará un mensaje de validación estilizado en color blanco para que sea fácil de ver.

#### **Interfaz de Visualización de Estado**

#### La interfaz de visualización de estado permite al usuario monitorear el sistema y la ejecución de comandos. Algunas de sus características incluyen:

#### **Contenedor de Desplazamiento:** La página tiene un contenedor que se ajusta y permite el desplazamiento vertical, asegurando que el último comando ejecutado siempre sea visible sin que la página se desborde.

#### **Estilos Consistentes:** Se aplican estilos CSS uniformes que mantienen la interfaz atractiva y funcional, adaptándose a la identidad visual del sistema.

#### **Interfaz de Generación de Scripts**

#### La interfaz de generación de scripts está diseñada para facilitar la creación y validación de scripts. Algunas características importantes son:

#### **Validación de Puertos:** Se han implementado validaciones para asegurarse de que los puertos sean correctos y accesibles. Si hay un error, se proporciona retroalimentación visual inmediata.

#### **Interacción con Elementos HTML:** Los mensajes de error se conectan a clases específicas en CSS, lo que permite que se muestren correctamente sobre el fondo de la tabla, mejorando la usabilidad y la experiencia del usuario.

#### **3.1.2.4 Configuración de Variables de Entorno**

#### Para optimizar el funcionamiento de la aplicación, es necesario configurar variables de entorno que faciliten la ejecución de comandos en entornos como PowerShell y Node.js. Esto incluye:

#### **Configuración Dinámica de Puertos:** Se implementará una lógica para obtener puertos de manera dinámica, lo que mejorará la conectividad y evitará conflictos.

### Interfaces de comunicación

#### **Requisitos de Comunicación**

#### El sistema establece comunicaciones tanto internas como externas, permitiendo la interacción con otros componentes y servicios. Los principales requisitos de comunicación son:

#### **Conectividad en Tiempo Real**: El sistema debe permitir la transmisión de datos en tiempo real para ejecutar y recibir comandos desde la interfaz de usuario sin demoras significativas.

#### **Seguridad en la Transmisión**: Es esencial que las comunicaciones estén cifradas y autenticadas para proteger los datos sensibles que se transmiten, especialmente en entornos de ataque.

#### **Fiabilidad**: Las comunicaciones deben ser robustas, asegurando que los mensajes se entreguen de manera efectiva, incluso en caso de errores o interrupciones temporales en la conexión.

#### **Requisitos de Comunicación**

#### **WebSocket**: Se utiliza WebSocket para la comunicación bidireccional entre el cliente y el servidor. Este protocolo permite mantener una conexión persistente, facilitando la transmisión instantánea de comandos y respuestas. Es ideal para aplicaciones que requieren interacción en tiempo real.

#### **HTTP/HTTPS**: Para las interacciones iniciales, como la carga de páginas y recursos, se utilizan los protocolos HTTP/HTTPS. HTTPS se utiliza para asegurar que los datos transmitidos entre el cliente y el servidor estén cifrados, proporcionando un nivel adicional de seguridad.

#### **TCP/IP**: El sistema hace uso del protocolo TCP/IP para establecer conexiones de red confiables. Esto asegura que los datos se envíen y reciban correctamente, manteniendo la integridad de la información durante la transmisión.

## Requisitos funcionales

## Los requisitos funcionales explican las tareas que el sistema "Comando de Control" debe llevar a cabo. A continuación, se presentan estos requisitos.

### Requisito funcional 1: Validación de entradas

### Descripción: El sistema tiene que comprobar que todas las entradas de los usuarios sean correctas antes de seguir adelante.

### Funcionalidad:

### Al registrarse o iniciar sesión, el sistema debe asegurarse de que todos los campos estén completos.

### Al crear scripts, debe validar que los datos ingresados (como direcciones IP y puertos) sean válidos.

### Si algo no está bien, se mostrará un mensaje claro que explique cómo corregirlo.

### Requisito funcional 2: Secuencia de Operaciones

### Descripción: El sistema debe realizar las tareas en un orden lógico para que todo funcione bien.

### Funcionalidad:

### Cuando se crea un script, el sistema guiará al usuario paso a paso, eligiendo la ip, el puerto, y el nombre de la red, ajustando los parámetros y ejecutando el script.

### Asegurará que el entorno esté listo, como que la máquina esté encendida y configurada.

### Requisito funcional 3: Manejo de Errores

### Descripción: El sistema tiene que manejar bien los problemas que puedan surgir al ejecutar scripts.

### Funcionalidad:

### El usuario podrá reiniciar el proceso o ajustar la configuración del script. Asegurará que el entorno esté listo, como que la máquina esté encendida y configurada.

### Si hay un error, el sistema avisará al usuario con un mensaje claro

### Requisito Funcional 4: Generación de salidas

### Descripción: El sistema debe mostrar resultados claros después de ejecutar los scripts.

### Funcionalidad:

### Tras ejecutar un script, el sistema mostrará en la pantalla si todo salió bien o si hubo errores.

### Habrá alertas visuales que indiquen el estado de la ejecución (como colores para éxito, advertencias o fallos).

### Requisito funcional 5: Autenticación y Autorización

### Descripción: El sistema necesita tener un buen sistema para identificar y autorizar a los usuarios.

### Funcionalidad:

### Los usuarios deben iniciar sesión con un sistema que valide sus credenciales.

### Debe haber una manera de crear nuevos usuarios, asegurándose de que la información sea correcta.

## Requisitos no funcionales

### Seguridad

#### El sistema "Comando de Control" necesita tener medidas de seguridad para proteger las funciones. Estos son los requisitos de seguridad:

#### **Controles de comunicación:** Se limitará la comunicación entre diferentes partes del sistema para que solo los módulos necesarios puedan interactuar.

#### **Verificaciones de integridad:** Se realizarán controles periódicos para asegurar que los datos importantes no hayan sido alterados, alertando sobre cualquier cambio sospechoso.

### Disponibilidad

#### El sistema debe estar disponible para los usuarios en todo momento. Aquí están los requisitos de disponibilidad:

#### **Alta disponibilidad:** El sistema debe estar operativo al menos el 90% del tiempo.

### Portabilidad

#### El sistema debe ser fácil de trasladar a diferentes plataformas. Aquí están los requisitos de portabilidad:

#### **Componentes dependientes:** Se buscará que menos del 20% de los componentes del sistema dependa del servidor, para mayor flexibilidad.

#### **Lenguaje de programación:** Se usará JavaScript para el frontend (React) y Node.js para el backend, ya que son ampliamente aceptados en diferentes entornos.

#### **Herramientas de desarrollo:** Se utilizarán herramientas que funcionen en varios sistemas operativos, como Visual Studio Code y Node Package Manager (NPM).