Cátedra de Sistemas Operativos II

Trabajo Práctico NºIII

Soriano, Juan

6 de junio del 2019

Introducción

En esta sección se encuentra la especificación de los requerimientos de la aplicación de desarrollo de un webserver capaz de ejecutar CGI en el backend correspondiente al trabajo N° 3 de la Sistemas Operativos II.

Propósito

El propósito del Trabajo Práctico n° 3 de Sistemas Operativos II es el de diseñar e implementar un software que sea capaz de correr en un webserver y ejecutar código CGI en el backend. Debe poder realizar 4 tareas:

1. Una página que reporte información sobre recursos varios del sistema embebido.
2. Una página con un formulario que permita definir una fecha (año y día DOI) y que al enviarlo devuelva la lista de archivos disponibles de GOES 16 en AWS (producto ABI-L2-CMIPF, canal13).
3. Página que liste los módulos instalados en el sistema y que posea un formulario que permita subir un archivo al servidor, controlar que este sea un archivo válido e instalarlo en el kernel del sistema operativo.

Referencias

1. Linux man pages: https://linux.die.net/man/

2. GNU, “5 Making The Best Use of C”, https://www.gnu.org/prep/standards/html_node/Writing-C.html

3. W. Richard Stevens, Stephen A. Rago, Advanced Programming in the UNIX Environment, 3rd Edition, Adison-Wesley

Descripción General

En la presente sección se explicarán los distintos aspectos del trabajo. Las funciones que ofrece el producto, la perspectiva del mismo, las características de los usuarios, requisitos para futuras iteraciones, entre otras cosas.

Perspectiva del Producto

El producto es una implementación de un servicio liviano hosteado en un webserver (lighttpd) que debe poder ofrecer funcionalidades a nivel de sistema operativo al usuario por medio de CGI.

Características de los Usuarios

El producto se destina a usuarios con conocimientos básicos del manejo de sistemas operativos Linux.

Restricciones

Las restricciones son muy bajas. Se necesita de una computadora con requerimientos de hardware mínimos capaz de ejecutar el programa. Para computadoras con menor capacidad de cómputo, el programa se ejecutará de la misma manera pero tomará más tiempo. Se necesitará de un compilador GCC para compilar el código. Es necesario un webserver que ofrezca la funcionalidad de ejecutar archivos CGI.

Requisitos Futuros

Queda un gran abanico de posibles funcionalidades para ser agregadas, a continuación se listan algunas de ellas:

- Mejorar la estética de las páginas utilizando CSS y Javascript.
- Mejorar la seguridad del sistema para no utilizar paths a los archivos.

Requisitos Específicos

En la presente sección se explicará con mayor lujo de detalle las entradas y salidas del sistema como así también una descripción de las funciones y requerimientos funcionales del producto.

Interfaces Externas

A continuación se ofrece una breve descripción de las interfaces implementadas.

Interfaz de usuario

Para acceder al sistema se deberá utilizar un navegador de internet y dirigirse a la IP de la Raspberry (configurada de forma estática en 192.168.1.14 para la interfaz eth0). Se ofrece un portal web.

Funciones

En esta sección se listan los requerimientos funcionales de la aplicación para la versión 1 del release del producto con fecha 6 de Junio de 2019:

El producto realiza principalmente las siguientes tareas mediante el uso del webserver:

1. Reporte de información sobre el sistema embebido, incluyendo consumo de procesador, memoria, uptime y la fecha y hora actual.
2. Devolución de la lista de archivos disponibles de GOES 16 en AWS y muestra de los mismos.
3. Página que liste los módulos instalados en el sistema y que posea un formulario que permita subir un archivo al servidor, controlar que este sea un archivo válido e instalarlo en el kernel del sistema operativo.

Requisitos de Rendimiento

En esta sección se listan los requerimientos funcionales de la aplicación para la versión 1 del release del producto con fecha 6 de Junio de 2019:

1. Requerimiento mínimos para instalar lighttpd y un sistema embebido con MMU.

Restricciones de Diseño

En esta sección se listan las restricciones de diseño de la aplicación para la versión 1 del release del producto con fecha 16 de Mayo de 2019:

1. Se utiliza HTML, CGI y Pearl o C.
2. Sintaxis de código del tipo GNU o Linux Kernel.

Atributos del Sistema

En esta sección se listan los atributos del sistema de la aplicación para la versión 1 del release del producto con fecha 6 de Junio de 2019:

1. Sistema portable, es decir, deberá ser capaz de correr en cualquier sistema operativo.
2. Sistema compilado con GCC y flags -Werror -Wall -pedantic
3. Se correrá la herramienta de análisis de código Cppcheck.

Comparación de SO y Webservers

Para la elección del SO, no se hizo una gran búsqueda. Las opciones son amplias, desde Archlinux hasta una versión de windows para Rasbperry. Se optó por Raspbian debido a su amplia comunidad y su distribución basada en Debian que la hace una opción robusta y bien probada. Dentro de las opciones de Raspbian se optó por Raspbian lite que no ofrece una interfaz gráfica ya que no es necesaria. A comparación de Archlinux, Raspbian utiliza versiones de paquetes más antiguos pero muy probados.

Para la elección de Webservers, se buscó uno que ofreciera funcionalidades CGI y no fuera pesado. En la siguiente tabla comparativa se muestran las opciones más utilizadas:

	CGI	Usuario/Kernel
Nginx	No	User
Apache	Si	User
Lighttpd	Si	User

Para una lista más comprensiva, dirigirse al siguiente enlace:
https://en.wikipedia.org/wiki/Comparison_of_web_server_software

Métodos GET y POST

Los métodos GET, se utilizan para pedir datos de una determinada fuente y es uno de los métodos HTTP más comunes. En este caso, se envía un QUERY string que va en el URL con la informacion de los pares nombre/valor codificados.

Los métodos POST, se utilizan para enviar información al servidor para crear o actualizar algún recurso. El típico caso de uso es para enviar un usuario y contraseña para autenticarse. En este caso, la información va codificada en el body del HTTP request.

Paso del archivo del host al filesystem

Cuando el host llena un formulario con un input del tipo file y un enctype del tipo multipart/form-data, se envía un HTTP post request al servidor. El servidor hace un fork para atender esta petición HTTP y utilizando CGI, es capaz de decodificar la información recibida y las variables del cuerpo del mensaje. Aquí comienza la escritura del archivo binario en el servidor. Es importante indicar que el enctype sea del tipo multipart/form-data para que no se realice ninguna codificación de caracteres y no se altere la información binaria del archivo.

Inserción de un módulo en el kernel vía CGI

El servidor, al recibir el archivo, hace un llamado a un wrapper hecho en C que otorga permisos +s al que lo ejecute. Esto quiere decir que el servidor al ejecutar dicho wrapper que hace un insmod, eleva sus privilegios a los mismos que tenía el creador del archivo, en este caso se compiló con sudo asi que los permisos son de superusuario. Esto permite hacer la llamada a sistema con insmod e insertar el módulo del kernel. Lo mismo sucede cuando se quiere remover el módulo, se utiliza el wrapper correspondiente.

Conclusiones

A modo de conclusión se puede decir que fue posible realizar el diseño e implementación de un servidor webserver que realice acciones en el backend con el sistema operativo utilizando bajos recursos en un sistema embebido. Se podría realizar una implementación parecida en una placa bastante menos potente, siempre y cuando tenga una MMU como para poder ejecutar un linux.