Notas Técnicas UHB

# Índice de Contenidos

1 Índice de Contenidos 1

2 Abreviaturas y Acrónimos 1

3 Entorno de desarrollo utilizado 3

3.1 Lenguaje de programación utilizado 3

3.2 Sistema operativo utilizado 4

4 Configuración inicial del entorno 5

4.1 Instrucciones de instalación de BSD 5

4.2 Instrucciones de instalación de DEB 5

4.3 Dependencias y servicios utilizados en BSD y DEB 5

5 Estructura de UHB 6

5.1 Flujo del Programa 6

5.2 Funcionalidades de UHB 7

6 Programas soportados por UHB 8

6.1 Discretionary Access Control (DAC) 8

6.2 Access Control Lists (ACLs) 8

6.3 Mandatory Access Control (MAC) 8

6.4 Firewall 8

6.5 Logs mediante Rsyslog 8

6.6 Auditoria mediante auditd 8

# Abreviaturas y Acrónimos

|  |  |
| --- | --- |
| **Abreviatura/Acrónimo** | **Descripción** |
| UHB | *Unified Hardening Binary* |
| VM/MV | *Virtual Machine*/Máquina Virtual |
| OS/SO | *Operating System*/Sistema Operativo |
| *distro* | *Distribution*/Distribución |
| SSH | *Secure Shell Protocol* |
| RAM | *Random Access Memory* |
| NAT | *Network Address Translation* |
| GPT | *GUID Partition Table* |
| GCC | *GNU Compiler Collection* |
| DAC | *Discretionary Access Control* |
| ACL | *Access Control List* |
| MAC | *Mandatory Access Control* |
|  |  |
|  |  |
|  |  |

# Entorno de desarrollo utilizado

El entorno utilizado en el desarrollo de UHB será la siguiente:

1. GitHub como plataforma de desarrollo colectivo y el alojamiento del proyecto.
2. Hipervisor Oracle VirtualBox VM para la creación de las máquinas virtuales de FreeBSD y Debian.
3. Se utilizará Microsoft Visual Studio Code como IDE.

## Lenguaje de programación utilizado

Se plantearon los siguientes lenguajes de *scripting*/programación para la creación de UHB:

|  |  |  |
| --- | --- | --- |
| **Opción** | **Pros** | **Contras** |
| *scripting* en *bash* | * Ideal para tareas rápidas, automatización por medio de *scripts* pequeños. * Integración directa con los OS utilizados. * Extensa documentación. | * Rendimiento limitado. * Difícil mantenimiento y depuración en aplicaciones grandes. * Falta de características avanzadas. |
| **C** | * **Alto rendimiento.** * **Estándar en aplicaciones Unix y Linux.** | * **Gestión manual de memoria.** * **Mayor tiempo de desarrollo.** |
| C++ | * Alto rendimiento. * Lenguaje orientado a OOP. | * Gestión manual de memoria. * Mayor tiempo de desarrollo. * Lenguaje orientado a OOP. |
| Python | * Sintaxis más limpia que C o C++. * Alto nivel de abstracción. * Gran cantidad de recursos y bibliotecas | * Peor rendimiento que C o C++. * Requiere un intérprete. |

Tras valorar los pros y contras, se decidió que programar UHB en C sería la mejor opción.

Para albergar el proyecto y tener un control de las distintas versiones, se ha creado un repositorio en GitHub para UHB. El repositorio será privado hasta que se termine por completo el trabajo. Se proporcionará acceso al GitHub al tutor del proyecto tras llegar a una versión inicial adecuada.

## Sistema operativo utilizado

Debido a que UHB será una utilidad centrada a OS para usos en servidores y otros productos a mayor escala, los sistemas operativos Windows y MacOS[[1]](#footnote-1) no han sido considerados y solo se han planteado las diversas familias y *distros* que ofrece Linux:

|  |  |  |
| --- | --- | --- |
| **OS** | **Pros** | **Contras** |
| **Debian** | * **Muy estable.** * **Excelente documentación.** * **Compatible con software libre.** | * **Paquetes más antiguos debido a su enfoque en estabilidad.** * **Configuración inicial compleja.** |
| Fedora | * Innovadora, incluye herramientas y tecnologías modernas. * Buen soporte para desarrolladores. * Excelente integración con SELinux. | * Ciclo de vida corto (aproximadamente 13 meses). * Menos estable que distribuciones como Debian. |
| CentOS | * Buena estabilidad. Amplio uso en servidores. * Compatible con herramientas empresariales. | * Ciclo de vida más corto que el antiguo CentOS. * Menos actualizaciones rápidas. |
| **FreeBSD** | * **Excelente rendimiento en servidores.** * **Sistema de puertos muy flexible.** * **Fuerte enfoque en seguridad.** * **Documentación sólida.** | * **Menos software precompilado disponible.** * **Menor comunidad comparada con Linux.** |

Se escogió Debian debido a su estabilidad y disponibilidad de documentación. También se escogió FreeBSD por los mismos motivos y para tener compatibilidad con sistemas BSD.

# Configuración inicial del entorno

Se utilizarán 3 máquinas virtuales en este trabajo:

1. Dos VMs en el cual se desarrollará y testeará el UHB:
   1. Un FreeBSD 14.2 STABLE, llamado “BSD”
   2. Un Debian 12.9.0, llamado “DEB”
2. Adicionalmente, se dispondrá de una tercera VM, siendo también un Debian 12.9.0, llamado “DESKTOP”. Éste servirá como intermediario entre BSD, DEB y el host para la transferencia de archivos por medio de SSH y las conexiones de red.

DEB y BSD tendrán las siguientes características en VirtualBox:

* 4 GB de RAM
* 2 procesadores
* 8 GB de memoria en una única partición, dividido en:
  + 7 GB de memoria, GPT para BSD, ext4 para DEB
  + 1 GB de memoria *swap*
* 2 adaptadores de red:
  + Un adaptador NAT para la conexión de la máquina con la red, para la descarga de los paquetes necesarios.
  + Un adaptador de red interna para la comunicación entre BSD, DEB y DESKTOP.

## Instrucciones de instalación de BSD

BSD tendrá la siguiente configuración de instalación:

## Instrucciones de instalación de DEB

DEB tendrá la siguiente configuración de instalación:

## Dependencias y servicios utilizados en BSD y DEB

Para la correcta compilación de las distintas versiones de UHB en ambos OS, se optó por utilizar el compilador GCC, requiriendo su instalación manual en ambos VMs. Adicionalmente, el *make* por defecto en FreeBSD es el BSD Make, no el GNU Make, por lo cual se necesitará instalar y utilizar *gmake* en BSD.

|  |  |
| --- | --- |
| **FreeBSD** | **Debian** |
| >>pkg install gcc  >>pkg install gmake | >>pkg install gcc |

# Estructura de UHB

La estructura inicial del proyecto será la siguiente:

1. Un fichero de código fuente llamado *source*, con los siguientes archivos .c:
   1. main.c: Punto de entrada del programa.
   2. imp\_bsd.c: Implementación de UHB para FreeBSD.
   3. imp\_deb.c: Implementación de UHB para Debian.
   4. utils.c: Funciones comunes a ambos SO.
2. Un fichero de archivos de cabecera, llamado *include*.
3. Carpeta del ejecutable, llamado *bin*, dónde se almacena el Makefile y el binario al compilarlo.
4. Una carpeta llamada *config*, dónde se almacenarán las configuraciones generadas por uhb:
   1. *uhb\_config\_ini*: Configuración inicial al ejecutarse UHB.
   2. *uhb\_config\_(timestamp)*: Configuración generada por el usuario con los cambios realizados mediante UHB.

La estructura de UHB está diseñada de forma modular, de forma que pueda adaptarse a las peculiaridades y diferencias de cada distribución Linux o BSD mediante sus archivos de implementación.

## Flujo del Programa

El flujo de funcionamiento de UHB será la siguiente:

1. *main.c* detectará si el usuario que está ejecutando UHB es root. Caso lo sea, verificará si el SO es soportado mediante *so\_detect.c*.
2. *so\_detect.c* devolverá el sistema operativo utilizado y ejecutará la implementación concreta (*imp\_bsd.c*, *imp\_deb.c*…etc.) de UHB.
3. Una vez dentro de la implementación concreta, el binario efectuará las siguientes búsquedas:
   1. Verificará y recogerá información de las particiones existentes y su tipo mediante *fstab*.
   2. Mirará la existencia y los contenidos de *rc.local* para determinar si se ejecutan comandos pertinentes a permisos en *boot-time*.
   3. Detectará la existencia o no de los programas necesarios soportados para las tareas de endurecimiento.
   4. Se volcará la información de 3.1-3. a *uhb\_config\_ini*.
4. Tras hacer las comprobaciones iniciales, se proporcionará al usuario un menú donde tendrá las opciones de endurecimiento del sistema a escoger.
5. Si el usuario elige terminar, se generará el archivo de configuración y se terminará el programa.

## Funcionalidades de UHB

El usuario dispondrá de las siguientes opciones en el menú:

1. Editar y visualizar los permisos DAC y de los ACLs en el sistema.
2. Editar permisos MAC del sistema.
3. Editar el firewall.
4. Visualizar la configuración actual.
5. Generar configuración a partir de un archivo de configuración existente.
6. Generar archivo de configuración actual y terminar el programa.
7. Terminar el programa sin generar un archivo de configuración.

# Programas soportados por UHB

La dificultad del trabajo planteado dependerá del alcance del proyecto y de los ejecutables diversos que soporte. Tanto los sistemas Unix/Linux y BSD disponen de una gran variedad de herramientas y binarios que son de utilidad para el endurecimiento de sistemas, y adicionar soporte a todos ellos puede ser contraproducente. Por tanto, se proponen los siguientes servicios que UHB dará soporte en este trabajo:

* DAC (ambos, por defecto).
* Servicio ACL: *getacl*, *setfacl* (ambos[[2]](#footnote-2)).
* Servicio MAC: mac\_bsdextended (FreeBSD), AppArmor (Debian).
* Cortafuegos: IPFW (FreeBSD), UFW (Debian).
* Servicio de manejo y envío de logs: Rsyslog (ambos).
* Servicio de auditoría: auditd (ambos[[3]](#footnote-3)).

## Discretionary Access Control (DAC)

## Access Control Lists (ACLs)

## Mandatory Access Control (MAC)

## Firewall

## Logs mediante Rsyslog

## Auditoria mediante auditd

1. Se nota que el sistema operativo que basan los MacOS actuales también es FreeBSD. [↑](#footnote-ref-1)
2. Tanto FreeBSD y Debian disponen de un servicio de listas de control de acceso llamado *acl*, pero tienen implementaciones distintas. [↑](#footnote-ref-2)
3. Ídem para *auditd*. [↑](#footnote-ref-3)