Parte I

Introducción y Objetivos

1. Introducción

Nos encontramos inmersos en la llamada era de la información, las organizaciones continúan cada vez más aprovechando las ventajas que ofrece una interconexión global y el almacenamiento de datos en formatos digitales. Actualmente la información ha pasado a ser uno de los principales activos con los que cuentan las diversas organizaciones y empresas. La información es un activo, tangible o intangible, que tiene valor en los procesos de negocio de la organización y como tal requiere una protección adecuada que incluya la protección de su confidencialidad, integridad y disponibilidad.

Nos enfrentamos pues a un escenario en el que se ha incrementado la dependencia de los sistemas y servicios de información, muchos de los cuales no fueron diseñados para ser seguros, así como el volumen de datos procesados. Hemos de afrontar también el aumento de los riesgos y amenazas, debido a la simplificación del acceso a las tecnologías de la información y al aumento de potenciales objetivos.

En términos generales, la seguridad puede entenderse como aquellas reglas técnicas y actividades destinadas a prevenir, proteger y resguardar lo que es considerado como susceptible de robo, pérdida o daño.

La seguridad de la información es un sector en alza que año tras año se afianza en el mercado de las nuevas tecnologías. Según el informe del Harvey Nash USA CIO Market Survey para 2005/2006, patrocinado por Harvey Nash Group PLC and PricewaterhouseCoopers, las inversiones en el área de seguridad informática y telemática continúan creciendo.

Son cada vez más los productos y servicios ofrecidos por las empresas especializadas del sector: cortafuegos, *proxy*, análisis de registros, detectores de intrusiones (de red NIDS y de sistemas HIDS), pasarelas antivirus, pasarelas de mensajería (*anti-spam* y otras), filtrado de contenidos y URL, filtrado de contenidos activo, redes privadas virtuales (*VPN*), firma electrónica (*PKI*), evaluación de vulnerabilidades, actualización de sistemas, correlación de eventos, gestión de usuarios, pruebas de penetración, auditorías de seguridad, cumplimiento de normas y legislaciones (LOPD y LSSI) y certificación de estándares (ISO 17799 e ISO 27001) y decenas de otros productos y derivados.

Haciendo un ejercicio de análisis y abstracción podemos sintetizar en los siguientes puntos los aspectos fundamentales de la información que deben ser protegidos:

Confidencialidad: es preciso establecer "quien" debe tener acceso a "que" información.

Posesión o control: se requiere controlar quien posee la información, incluso en el caso de que los datos no hayan sido leídos aun.

Integridad: hace referencia a la corrección de los datos. Deben evitarse los errores ya sean intencionados o no.

Autenticidad: otro de los objetivos principales consiste en ser capaces de etiquetar los datos correctamente, tan pernicioso sería tener datos erróneos como falsos.

Disponibilidad: el quinto de los principios que debemos observar es el que ataña a la disponibilidad, es preciso disponer de los datos en el momento en que sean necesarios.

Utilidad: finalmente, los datos deben ser útiles. Por ejemplo, supongamos que ciframos datos para protegerlos y olvidamos la clave para descifrarlos, de nada nos sirve asegurar la confidencialidad, control, autenticidad y disponibilidad en este caso, esos datos aunque seguros son inútiles.

De entre todos los campos de la seguridad de la información que ayudan a conseguir los principios anteriores, en este proyecto nos limitaremos a uno: ***la seguridad perimetral***.

En una arquitectura de red típica la red perimetral, también denominada a menudo DMZ (red de zona desmilitarizada) o red de extremo, se interpone entre las redes externas (la Internet) y las redes internas privadas. Un servidor de seguridad perimetral (*cortafuegos*) es un mecanismo que controla y monitoriza el tráfico de datos entre las redes externas y la red perimetral. Dentro de esta definición podemos encuadrar multitud de sistemas, según sus características, funcionalidad o su implementación, en posteriores secciones de esta memoria se expondrán estos con detalle.

En la actualidad un porcentaje muy elevado de las aplicaciones de seguridad disponibles en el mercado están desarrolladas siguiendo el modelo del software libre o del software de fuentes abiertas. Software libre (en inglés free software) es el software que, una vez obtenido, puede ser usado, copiado, estudiado, modificado y redistribuido libremente. El software libre suele estar disponible gratuitamente en Internet, o a precio del coste de la distribución a través de otros medios; sin embargo no es obligatorio que sea así y, aunque conserve su carácter de libre, puede ser vendido comercialmente. El software de fuentes abiertas (en inglés open source) comparte licencias con el software libre y en la practica viene a ser lo mismo, existiendo eso si diferencias filosóficas e ideológicas que se escapan del ámbito de este proyecto.

Las ventajas de usar software libre para desarrollar nuevas aplicaciones de seguridad son evidentes, la principal es la reducción del coste de desarrollo. En un proyecto de integración permite despreocuparse de cada módulo y centrarse en el proceso mismo de integración, asegurándonos de que las aplicaciones usadas son desarrolladas, mantenidas y auditadas por una amplia comunidad, especialmente preocupadas por los aspectos de seguridad. En el anexo A en la página 88 puede encontrarse un mapa conceptual del modelo de software libre.

En las siguientes secciones se describirá el origen de este proyecto, que finalmente se ha materializado en un sistema de Seguridad Perimetral (lo que se conoce comúnmente como "Cortafuegos", si bien en el presente proyecto se llega algo más allá) como un sistema completo, acabado y totalmente funcional.

2. Motivación

Son varias las razones que suscitaron el interés en la creación de este Proyecto, entre ellas:

- La escasez en el mercado actual de sistemas de seguridad perimetral, de dispositivos que proporcionen con la robustez de las aplicaciones Open-Source, un sistema integrado que por un lado, recoja todos estos elementos, y por otro, proporcione una interfaz común, abstracta, acoplada y *transparente* entre dichos elementos *(ver más abajo)*.
- La innovación subyacente de proporcionar una herramienta que incluya las últimas novedades disponibles en la actualidad en cuanto a capacidades y versiones de dichas aplicaciones Open-Source.
- La seguridad, robustez e integración que proporciona un sistema basado en la distribución **Debian** del sistema operativo Linux.
- ... y no menos importante, la idoneidad en cuanto a tamaño, complejidad y tiempo de desarrollo para la realización de un Proyecto de Fin de Carrera en esta rama de estudios técnicos.

Acerca del primero de estos puntos, merece las siguientes explicaciones:

Común se refiere a que la apariencia de la interfaz de usuario (e incluso su implementación) es uniforme, así como a que abarca globalmente a todas las partes del sistema (todos los módulos implicados).

Abstracta significa que lo que el usuario ve es una abstracción de los elementos y componentes software subyacentes. Esto incluye tanto la terminología (se ha intentado que sea lo más genérica y descriptiva posible, evitando términos técnicos y palabras clave específicas del producto en particular) como la granuralidad de la configuración (se han elegido principalmente aquellos parámetros que resultan indispensables para realizar la configuración desde una visión meramente funcional).

Acoplada se refiere a que los diferentes módulos, componentes y elementos software que participan en el sistema se entienden perfectamente entre ellos, ya que tienen la capacidad de comunicarse, con objeto de conseguir la mencionada transparencia.

Transparente por tanto se entiende como el resultado de las tres características anteriores, es decir, se pretende que la experiencia del usuario se centre en qué pretende conseguir, evitando tener que indagar en cómo hacer que los diferentes módulos implicados se unan para conseguir ese objetivo; pues esto será transparente para el usuario.

Así mismo, se ha pensado que el mecanismo idóneo para hacer llegar este sistema al público es mediante un paquete integrado; se aborda pues la creación de una *"distribución"* de Linux fácilmente instalable y que (como no podría ser de otro modo) tenga en cuenta también estos mismos principios mencionados.

Creemos que el valor que proporcionan estas cuestiones, tanto referentes a la interfaz, como a la distribución instalable, constituye una fuente de interés suficiente para abordar este trabajo.

3. Metodología

En un primer (y somero) análisis del sistema de información que se pretende construir, se observa que éste se divide en dos partes o subsistemas bien diferenciados:

- La interfaz de usuario
- La distribución instalable

Si bien el primero se puede abordar mediante un proceso de ingeniería claro y estricto (con la ayuda de múltiples metodologías de desarrollo existentes), la segunda no queda tan claro que se pueda acoplar a un proceso formal de este tipo (pues es una mera integración y configuración de elementos software ya construidos y existentes). Por tanto, si bien el proceso de desarrollo de la primera parte mencionada se describirá con detalle, con la ayuda de técnicas y métodos propios de las metodologías de desarrollo de software, la segunda parte se tratará de una forma más libre, aunque ello no signifique que no sea completa y exhaustiva.

4. Planificación

Mediante la planificación estimamos el tiempo (y el esfuerzo) que serían necesarios para acometer el desarrollo de este Proyecto.

Para ello, dividiremos el proyecto en fases y estimaremos el esfuerzo necesario para completar cada una de ellas. Más tarde, recogemos los datos reales (una vez realizado el trabajo) y los comparamos con los obtenidos en un principio. Por tanto, para cada fase presentaremos dos valores:

Estimación Inicial es el esfuerzo que se estimó al principio que sería necesario para completar esta fase (antes de acometer el proyecto realmente)

Estimación Final es el dato medido en el final del proyecto, esto es, el esfuerzo real que se aplicó para esta fase.

De esta forma podremos medir la desviación entre la planificación inicial y la final. Este dato aparece reflejado en la última columna de la tabla, como el *Error Relativo de la estimación* o **RE**.

Como medida del esfuerzo, utilizaremos la expresión "días" con el siguiente significado: días (o jornadas) de trabajo, de 3 horas cada uno, a realizar por 1 persona (se ha escogido esta medida ya que se acerca –si no diariamente, sí semanalmente– a la empleada en la realización de este proyecto).

Fase	Estimación Inicial	Estimación Final	RE
Búsqueda de documentación	20 días	30 días	33,3 %
Análisis de antecedentes	5 días	6 días	16,6 %
Estudio de Viabilidad	2 días	2 días	0 %
Análisis de Requisitos	10 días	7 días	-42,8 %
Diseño del Sistema	30 días	40 días	25 %
Prototipo del Diseño	10 días	15 días	33,3 %
Implementación	90 días	270 días	66,6 %
Construcción de la Distribución	20 días	30 días	33,3 %
Pruebas	10 días	3 días	-233 %
Presentación	5 días	8 días	37,5 %
Total	202 días	411 días	50,8 %

Como se puede comprobar, hay una desviación importante entre la estimación inicial y la final (superando el tiempo estimado en algo más del 50 % de lo previsto inicialmente). Esto se puede explicar por el tamaño que ha llegado a tener el proyecto, algo más complejo que lo inicialmente previsto, pero sobre todo más extenso. Así mismo, las circunstancias personales a la hora de acometer un proyecto de este tipo influyen en gran medida; en este caso, debido a una dedicación no exclusiva en la realización del mismo (es decir, teniendo que compaginar el trabajo y los estudios) influye muy negativamente, en forma de excesiva fragmentación del tiempo dedicado, o períodos de ausencia prolongada, que llevan a tener que aplicar un casi constante esfuerzo extra a la hora de "retomar" las tareas, con la pérdida de tiempo útil de trabajo que esto conlleva.

4.1. Evaluación del Coste

Ya a posteriori, una vez realizado el proyecto, conocemos el nº de líneas de código de la que se compone. En este caso han sido alrededor de 10.900 *líneas de código* (LDC), sin contar el código HTML, por haber sido realizado (en parte) con la ayuda de herramientas gráficas de diseño.

Podemos aplicar algunos modelos de medición de costes que ofrece la Ingeniería del Software para ver con qué resultados nos encontramos. En este caso emplearemos el modelo COCOMO [9].

El método COCOMO (*Constructive Cost Model*) es un modelo de estimación del coste de un Proyecto software basado en el tamaño. Introducido por Barry Boehm, es en la actualidad uno de los modelos de estimación de coste del software más utilizados en la industria. Se basa en el análisis del tamaño del proyecto, expresado en *miles de líneas de código* (KLDC).

En realidad, existen una jerarquía de modelos COCOMO, que son:

Básico: Calcula el esfuerzo del desarrollo del software en función del tamaño del programa, expresado en LDC.

Intermedio: Calcula el esfuerzo del desarrollo del software en función del tamaño del programa, y de un conjunto de "conductores de costo", que incluyen la evaluación subjetiva del producto, del hardware, del personal y de los atributos del proyecto.

Avanzado: Incorpora todas las características de la versión intermedia y lleva a cabo una evaluación del impacto de los conductores de costo en cada fase (análisis, diseño, etc.) del proceso de ingeniería del software.

Así mismo, los modelos COCOMO están definidos para tres tipos de proyectos software:

Orgánico: Proyectos relativamente pequeños y sencillos en los que trabajan pequeños equipos, con buena experiencia en la aplicación, sobre un conjunto de requisitos poco rígidos.

Semiacomplado: Proyectos de tamaño y complejidad intermedia.

Empotrado: Proyectos software que incluyen desarrollos en hardware, software y restricciones operativas muy restringidas.

Considerando para nuestro caso un modelo *básico* en un proyecto *orgánico*, la ecuación de estimación del Esfuerzo en COCOMO sería:

$$E = a\,KLDC^{b} \text{ (personas} \times \text{meses)}$$

Así mismo, la ecuación de estimación del Tiempo de desarrollo sería:

$$T = c\,E^{d} \text{ (meses)}$$

Donde *a, b, c,* y *d* son una serie de coeficientes obtenidos empíricamente de una serie de proyectos, y son para cada tipo:

Tipo de proyecto	*a*	*b*	*c*	*d*
Orgánico	2,4	1,05	2,5	0,38
Semiacoplado	3	1,12	2,5	0,35
Empotrado	3,6	1,2	2,5	0,32

Así que en nuestro caso, tomando la cifra de 10900 líneas de código (10,9 KLDC) aplicaríamos la siguiente formula para obtener el Esfuerzo:

$$E = a\,KLDC^{b} = 2,4 \times 10,9^{1,05} = 29,47 \text{ (personas–mes)}$$

Así como el Tiempo:

$$T = c\,E^{d} = 2,5 \times 29,47^{0,38} = 9,04 \text{ (meses)}$$

El Nº de personas idóneo para desarrollar el proyecto en el tiempo estimado, pues, se calcularía mediante:

$$N^{o} = \frac{E}{T} = \frac{29,47}{9,04} = 3,25 \text{ (personas)}$$

Como se puede ver, según el método harían falta 3 personas para realizar el desarrollo durante un período de algo más de 9 meses, con dedicación exclusiva. Esto nos indica que, a pesar de las circunstancias, el esfuerzo realizado finalmente para la consecución de este trabajo ha sido considerable.