Treball de Fi de Grau

GRAU D'ENGINYERIA INFORMÀTICA

Facultat de Matemàtiques
Universitat de Barcelona

Seguridad Ágil

Gonzalo Vigo Scheiner

Director: Eloi Puertas

Departament de Matemàtica Aplicada i Anàlisi. UB

AGRADECIMIENTOS

Este proyecto es la conclusión de una etapa de mi vida de firme y constante trayectoria y a la vez, la herramienta que me permitirá disfrutar y desplegar mis capacidades laborales al máximo en un ámbito de realización y enriquecimiento personal.

En el camino recorrido y en la consecución del objetivo han confluido varios factores.

Entre ellos, los valores, orientación y constante contención de mi familia, ingrediente esencial en el estricto sentido de la palabra.

Este proyecto simboliza mi despegue y en él me acompañan los esfuerzos de todos los buenos momentos desde mi infancia, en el hogar, en las calles, en los amigos...

También la férrea búsqueda consciente y afortunada de haber encontrado y podido acceder casi siempre a un entorno académico positivo en el cual tuve la oportunidad de aprender de "personas" con verdadera vocación de servicio docente, elevada formación profesional y sentido de la pedagogía y hoy entre ellos mi tutor y supervisor de este trabajo, Eloi Puertas, con su dedicación y buen entender quién me brindó respaldo y motivación para desarrollar este estudio que significó para mí un desafío desde el primer instante.

Por último, no puedo dejar de mencionar al Claustro que lo hizo posible, la *Universitat de Barcelona*. Será siempre un orgullo para mi haber sido parte de una histórica institución académica y pública que durante siglos ha fomentado y estimulado con excelencia la formación de las personas.

De todo lo anterior es consecuencia hoy mi realidad, la de poder desarrollar formado y dedicado aquello que me apasiona.

PRÓLOGO

Nací en Argentina y resido junto a mi familia en Barcelona desde el año 2002.

Desarrollé este proyecto como Trabajo Final de Grado para la Facultad de matemáticas de la *Universitat de Barcelona* en el semestre de primavera de 2015.

Como alumno del Grado de Ingeniería Informática y con 5 años de experiencia profesional en el sector de la seguridad, desarrollé este estudio a fin de alcanzar los objetivos propuestos, entre ellos el académico y como valor añadido, el adicional de concienciar a la *Universitat de Barcelona* exponiendo las vulnerabilidades existentes en algunos de sus sistemas.

SUMARIO

Este trabajo es un estudio de la seguridad en el marco de las aplicaciones web y los diferentes modelos de desarrollo, con el objetivo de desarrollar una nueva capa de seguridad sobre la metodología de desarrollo ágil SCRUM.

Como complemento práctico a esta investigación realicé una auditoría de seguridad sobre dos de las principales aplicaciones públicas de la Universitat de Barcelona.

Así, la combinación teórico-práctico facilita el entendimiento de los diferentes procedimientos y riesgos.

TABLA DE CONTENIDO

1. INTRODUCCIÓN 10
2. SEGURIDAD EN APLICACIONES WEB 11
   2.1. Amenazas, riesgos y vulnerabilidades 12
   2.2. Vulnerabilidades 15
   2.3. Common Vulnerability Scoring System (CVSS) 18
3. MODELOS DE DESARROLLO 22
   3.1. Modelos de desarrollo clásicos 24
   3.2. Modelos de desarrollo ágiles 25
4. SECURE SCRUM 30
   4.1. Responsable de seguridad 31
   4.2. Concienciación y formación de seguridad 31
   4.3. Casos de abuso 31
   4.4. Auditoría de seguridad independiente 32
   4.5. Controles de seguridad 32
5. CASO PRÁCTICO DE UNA AUDITORÍA DE SEGURIDAD WEB, HACKING UB 34
   5.1. Planificación y honorarios 37
   5.2. Resumen ejecutivo 38
   5.3. Recopilación de información 39
   5.4. Información detallada de vulnerabilidades 44
      5.4.1. HACKUB - 001 XSS (Cross Site Scripting) 44
      5.4.2. HACKUB - 002 Cifrado débil soportado 47
      5.4.3. HACKUB – 003 Exposición de información sensible del servidor 50
      5.4.4. HACKUB - 004 Cookies de sesión sin atributo ‘HttpOnly’ 54
      5.4.5. HACKUB - 005 Cookies de sesión sin atributo ‘Secure’ 56
      5.4.6. HACKUB - 006 Método HTTP TRACE habilitado 58
   5.5. Roadmap de soluciones 60
6. CONCLUSIONES 62

7. REFERENCIAS.....63
A. ANEXO.....64
A.1. MARCO DE CONTROL SECURE SCRUM.....64
A.1.1. Controles y objetivos de control.....64

TABLA DE ILUSTRACIONES

Ilustración 1 - Riesgo inherente y residual....13
Ilustración 2 - Materialización del riesgo....14
Ilustración 3 - Ejemplo de ataque SQL Injection....15
Ilustración 4 - Ejemplo de ataque XSS....16
Ilustración 5 - Ejemplo de historia de usuario....27
Ilustración 6 - SCRUM Framework....28
Ilustración 7 - Framework eXtreme Programming....29

1. INTRODUCCIÓN

En los últimos años con el auge de las nuevas tecnologías e Internet como canal principal para la comunicación, en todos los ámbitos han surgido nuevos riesgos y amenazas que provocan grandes pérdidas de activos financieros, degradación de la imagen corporativa o simplemente pérdidas de información en las organizaciones.

Es por ello que mediante este trabajo, pretendo realizar un análisis detallado de los distintos riesgos, amenazas e impactos que existen en el sector de la seguridad de la información para acabar desarrollando una nueva capa de seguridad aplicable al modelo de desarrollo SCRUM.

Por último he llevado a cabo una auditoría de seguridad sobre dos de las principales aplicaciones de la Universitat de Barcelona con el fin de exponer de forma práctica, el contenido desarrollado a lo largo del trabajo.

En esta auditoría he identificado numerosas vulnerabilidades a partir de las cuales me ha sido posible extraer información sensible de los usuarios y sistemas de la Universitat de Barcelona.

Estructura

Partí de un análisis de la seguridad de la información, introduciendo conceptos básicos que permitan al lector comprender las siguientes secciones.

A continuación presento un análisis de los diferentes modelos de desarrollo resaltando la falta de controles de seguridad existente en los mismos.

El trabajo lo concluyo con la elaboración de una capa de seguridad denominada “SecuRe Scrum” aplicable al modelo de desarrollo de software, ágil SCRUM..

Por último, en el anexo incluyo una auditoría de seguridad sobre las principales aplicaciones de la Universitat de Barcelona, a través de la cual se reflejan los riesgos existentes en las mismas y un plan de acción correctivo.

2. SEGURIDAD EN APLICACIONES WEB

La seguridad es una de esas palabras que a lo largo del tiempo se ha ido asociando a la protección o peligros sobre aquello que nos preocupa. La seguridad se refiere principalmente a la exención de peligro, de daño o de riesgo.

Existen diferentes acepciones sobre la seguridad, tantas como peligros pueden existir: la seguridad urbana pretende mitigar los riesgos asociados a las ciudades, la seguridad laboral pretender lo mismo en el ámbito del trabajo y así cada una de ellas entre las que se encuentran la seguridad jurídica, la pública, bancaria, etc.

Esta sección se enfoca en dos de las acepciones más actuales, populares y de moda que conocemos como son la Seguridad Informática y la Seguridad de la Información dentro del marco de las aplicaciones webs.

Aplicaciones web

Las aplicaciones web son la cara frontal y visual del software y los sistemas que proporcionan el acceso a la información y a los diferentes procesos de los sistemas y las organizaciones. La información contenida en ellas representa generalmente el activo de mayor valor en el contexto de la seguridad de la información.

Dicha información se almacena en soportes físicos o virtuales que conforman la infraestructura que se debe proteger y que los atacantes a diario, intentan vulnerar para conseguir acceso a aquellos datos sensibles, destruirlos o hacerlos inaccesibles.

2.1. Amenazas, riesgos y vulnerabilidades

Una amenaza es una causa de riesgo.

En el contexto de seguridad de la información son aquellas acciones que puedan poner en riesgo la información, el activo que se intenta proteger.

Estas causas pueden ser actos dirigidos como cyberterrorismo, cyberespionaje, hacktivismo, etc.

Mientras que los riesgos pueden ser una denegación de servicio a ciertas organizaciones, es decir, la inactivación o indisponibilidad de alguno de sus servicios o sistemas de forma parcial o total, robo de información por la competencia o atentados por organizaciones anónimas por motivos ideológicos entre otros, las amenazas pueden variar dependiendo de factores geográficos, sectores empresariales, etc.

Las organizaciones realizan periódicamente modelados de amenazas y análisis de riesgos con el fin de poder analizar su exposición e identificar posibles vulnerabilidades que posteriormente serán tratadas mediante medidas mitigatorias o, simplemente asumiendo ciertos riesgos de una forma controlada.

Análisis de riesgos

El análisis de riesgos consiste en un estudio con cierto grado de profundidad, que mediante entrevistas con interlocutores de las diferentes áreas de una organización que estén dentro del alcance definido para el análisis, permita calcular riesgos inherentes, es decir, sin tener en cuenta las medidas mitigatorias existentes, a partir del impacto y la probabilidad:

$$Riesgo\ inherente = Impacto * Probabilidad$$

Los criterios utilizados para asignar valor al impacto y la probabilidad, dependerán de la metodología de análisis de riesgos utilizada por la organización. Entre las más populares se encuentran PILAR, CRAM Y MAGERIT, aunque también es muy común el desarrollo de metodologías propias. Estas metodologías se caracterizan por incluir un catálogo de amenazas que facilite la identificación de las mismas.

Una vez identificados y calculados los riesgos inherentes, se procede al cálculo del riesgo residual que consiste en restarle al valor de los riesgos inherentes un factor mitigatorio, resultado de los controles mitigatorios existentes en la organización para cada uno de ellos (véase Ilustración 1 - Riesgo inherente y residual):

$$Riesgo\ residual = riesgo\ inherente - factor\ mitigatorio$$

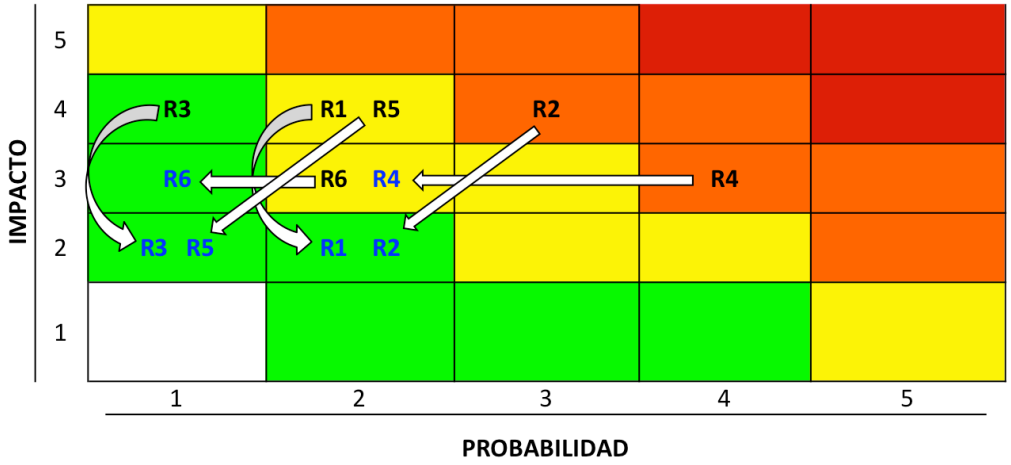


Ilustración 1 - Riesgo inherente y residual

La realización de un análisis de riesgos no es una tarea trivial, requiere un alto grado de conocimiento a nivel de seguridad y del entorno que se analiza.

Materialización de un riesgo

Para que un riesgo pueda identificarse o plasmarse dentro del análisis se deben contemplar varios elementos:

Agentes de amenaza: son aquellos actores que inician o facilitan la materialización del ataque. Entre ellos no sólo se encuentran los hackers, Ciber-terroritas, Ciber-espías o cualquier otro tipo de atacante malintencionado sino también los actores pasivos como pueden ser usuarios con malas prácticas, malware, etc.

Vectores de ataque: son los mecanismos utilizados para realizar la acción y son tan variadas como vulnerabilidades puedan existir. Por ejemplo, un vector de ataque podría ser la explotación de inyecciones SQL a través del portal web de una organización.

Debilidades de seguridad (vulnerabilidad): corresponden a las vulnerabilidades que presentan los sistemas o las aplicaciones. En el siguiente apartado se describen con mayor detalle.

Controles de seguridad: son las medidas que crean las empresas o los usuarios para proteger los activos de valor de la materialización del ataque. Pueden ser controles técnicos como la instalación de un antivirus, la implementación de un firewall (cortafuegos), regulatorios (delegación de responsabilidades mediante condiciones contractuales), formativos (concienciación a los usuarios), etc.

Impactos técnicos: son las consecuencias técnicas sobre los sistemas, aplicaciones y demás activos de información de las organizaciones o de los usuarios. Por ejemplo, la afectación a la confidencialidad de documentos sensibles (fugas o filtrados de información), afectación a la integridad de datos (perdida de información o manipulación dañina de la misma) o

afectación temporal de la disponibilidad de servicios (caída de un sistema dejando inaccesible dichos recursos).

Impactos al negocio: por último y como resultado lo anteriormente listado, se encuentran los impactos al negocio. Es decir, qué consecuencias reales provocan sobre la organización. Estos pueden ser degradación de la imagen corporativa, pérdida de activos financieros, incumplimiento legal o regulatorio, etc.

En la Ilustración 2 - Materialización del riesgo- se detalla el flujo de materialización de un riesgo partiendo del agente de amenaza y llegando al impacto a través de los vectores de ataque, debilidades y controles de seguridad e impactos técnicos.

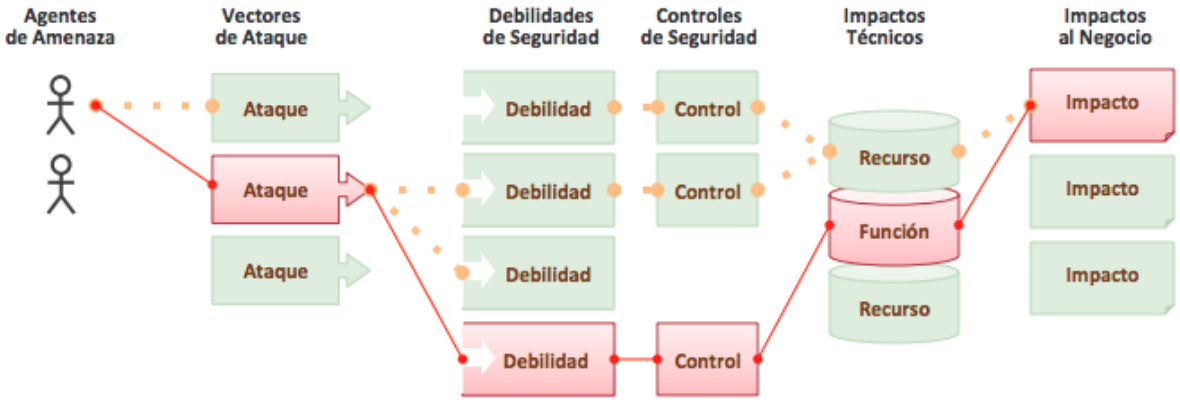


Ilustración 2 - Materialización del riesgo

2.2. Vulnerabilidades

Las vulnerabilidades son aquellos fallos o agujeros de seguridad mediante los cuales se pueden ejecutar los riesgos anteriormente comentados.

Existen multitud de vulnerabilidades que por diferentes causas exponen a los sistemas y a las aplicaciones a multitud de ataques.

Para comprender más detalladamente las causas raíces de estas nos basaremos en la clasificación que realiza sobre las mismas la organización OWASP.

El proyecto OWASP Top Ten (OWASP, 2013) es un potente documento que analiza y clasifica las vulnerabilidades más recurrentes en el mundo de la seguridad dentro del marco de las aplicaciones web.

Según dicho estudio, las 10 vulnerabilidades que mayor impacto tienen sobre las aplicaciones se pueden agrupar dentro de las siguientes categorías:

Inyecciones: las fallas de inyección tales como SQL (bases de datos), OS (sistema operativo), y LDAP (directorio activo) ocurren cuando datos no confiables son enviados por el usuario a un intérprete (servidor) como parte de un comando o consulta en la aplicación. Los datos son manipulados por el atacante a fin de engañar al servidor a la hora de ejecutarlos permitiendo así, acceder a datos no autorizados o generando errores en el servidor.

En el caso de inyecciones SQL, como se observa en la Ilustración 3 - Ejemplo de ataque SQL Injection- si un formulario de acceso a una aplicación no valida correctamente los datos de entrada (Usuarios/Contraseña) y el atacante introduce una sentencia SQL esta podría llegar a ejecutar una consulta SQL directamente en la base de datos.

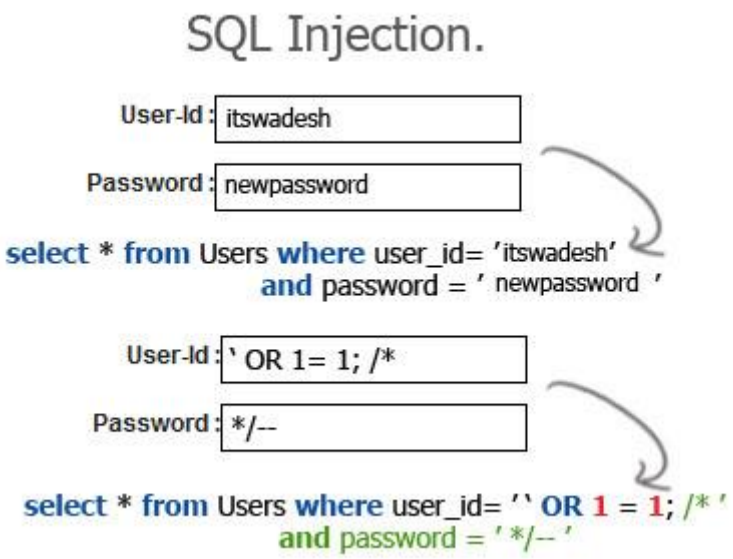


Ilustración 3 - Ejemplo de ataque SQL Injection

Pérdida de autenticación y gestión de sesiones: las funciones de la aplicación relacionadas con la autenticación y gestión de sesiones son frecuentemente implementadas de forma incorrecta permitiendo a los atacantes comprometer contraseñas, claves, token de sesiones, o explotar otras fallas de implementación para suplantar la identidad de otros usuarios.

Secuencia de comandos en sitios cruzados (XSS): las fallas XSS ocurren cada vez que una aplicación toma datos no confiables y los envía al navegador web sin una validación y codificación apropiada. XSS permite a los atacantes ejecutar secuencias de comandos en el navegador de la víctima, los cuales pueden secuestrar las sesiones de usuario, destruir sitios web, o dirigir al usuario hacia un sitio malicioso.

En el ejemplo de la Ilustración 4 - Ejemplo de ataque XSS- se describe un ataque en el que el usuario accede a una petición HTTP manipulada a partir de la incrustación de un código que comprometerá al final del proceso datos sensibles como son las cookies de sesión.

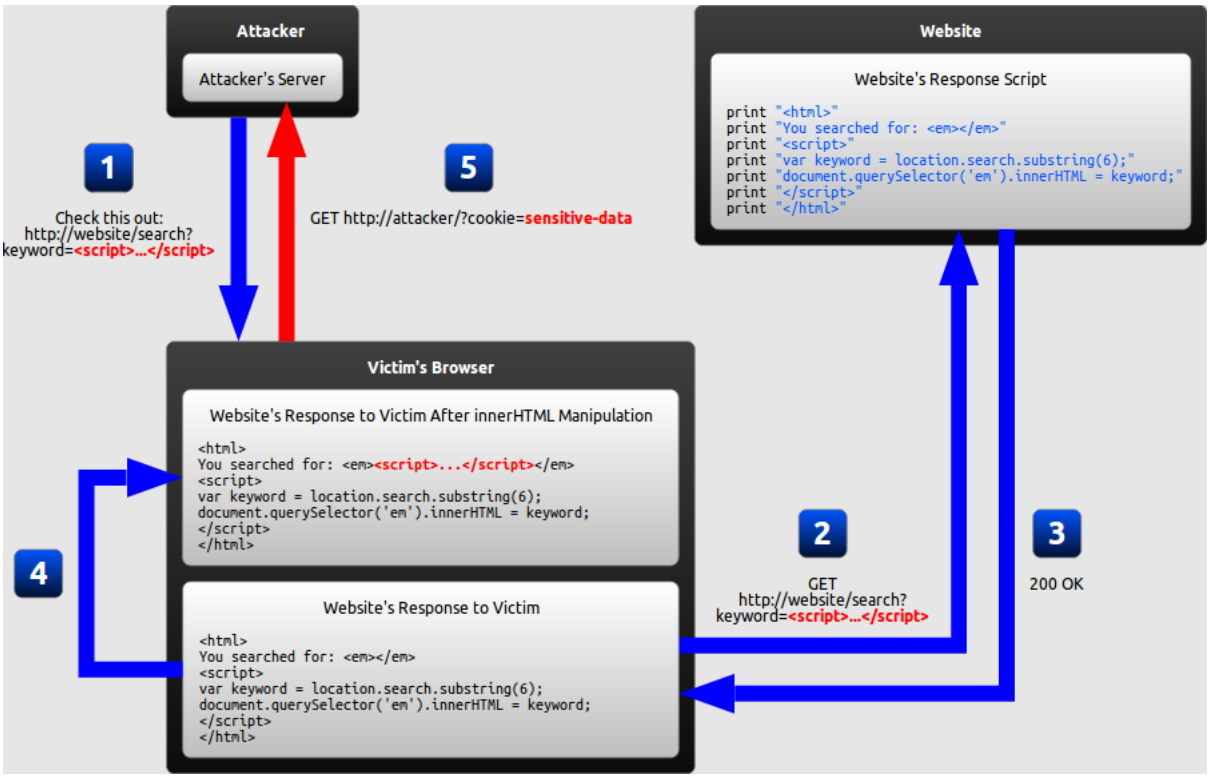


Ilustración 4 - Ejemplo de ataque XSS

Referencia directa insegura a objetos: ocurre cuando un desarrollador expone una referencia o ruta (path) a un objeto de implementación interno, tal como un fichero, directorio, o base de datos. Sin un chequeo de control de acceso u otra protección los atacantes pueden manipular estas referencias para acceder datos que no deberían.

Configuración de seguridad incorrecta: una buena seguridad requiere tener definida e implementada una configuración segura para la aplicación, marcos de trabajo, servidor de aplicación, servidor web, base de datos y plataforma. Todas estas configuraciones deben ser definidas, implementadas y mantenidas ya que por lo general son inseguras por defecto. Esto incluye mantener todo el software actualizado, incluidas las librerías de código utilizadas por la aplicación.

Exposición de datos sensibles: muchas aplicaciones web no protegen adecuadamente datos sensibles tales como números de tarjetas de crédito o credenciales de autenticación. Los atacantes pueden robar o modificar tales datos para llevar a cabo fraudes, robos de identidad u otros delitos. Los datos sensibles requieren de métodos de protección adicionales tales como el cifrado de datos, así como también de precauciones especiales en un intercambio de datos con el navegador.

Ausencia de control de acceso a las funciones: la mayoría de aplicaciones web verifican los derechos de acceso a nivel de función antes de hacer visible en la misma interfaz de usuario. A pesar de esto, las aplicaciones necesitan verificar el control de acceso en el servidor cuando se accede a cada función. Si las solicitudes de acceso no se verifican, los atacantes podrán realizar peticiones sin la autorización apropiada.

Falsificación de peticiones a sitios cruzados (CSRF): un ataque CSRF obliga al navegador de una víctima autenticada a enviar una petición HTTP falsificada, incluyendo la sesión del usuario y cualquier otra información de autenticación incluida automáticamente, a una aplicación web vulnerable. Esto permite al atacante forzar al navegador de la víctima para generar pedidos que la aplicación vulnerable piensa son peticiones legítimas provenientes de la víctima.

Uso de componentes con vulnerabilidades conocidas: algunos componentes tales como las librerías, los frameworks y otros módulos de software casi siempre funcionan con todos los privilegios. Si se ataca un componente vulnerable esto podría facilitar la intrusión en el servidor o una pérdida seria de datos. Las aplicaciones que utilicen componentes con vulnerabilidades conocidas debilitan las defensas de la aplicación y permiten ampliar el rango de posibles ataques e impactos.

Redirecciones y reenvíos no validados: las aplicaciones web frecuentemente redirigen y reenvían a los usuarios hacia otras páginas o sitios web, y utilizan datos no confiables para determinar la página de destino. Sin una validación apropiada, los atacantes pueden redirigir a las víctimas hacia sitios de phishing o malware, o utilizar reenvíos para acceder páginas no autorizadas.

2.3. Common Vulnerability Scoring System (CVSS)

CVSS, Common Vulnerability Scoring System, es un Sistema, que como su nombre índica permite clasificar las vulnerabilidades a partir de unos criterios unificados y obtener un valor como resultado final que facilitará la priorización de su resolución.

Los criterios que definen el valor CVSS son:

Vector de acceso (AV): representa la exposición de la vulnerabilidad.

Valor	Descripción	Puntuación
Local (L)	El atacante debe conseguir acceso local al sistema para explotar la vulnerabilidad.	0.395
Red interna (A)	El atacante debe conseguir acceso a la red interna o al dominio para explotar la vulnerabilidad	0.646
Internet (N)	La vulnerabilidad es explotable remotamente a través de Internet.	1.0

Complejidad de acceso (AC): refleja la complejidad de explotación de la vulnerabilidad.

Valor	Descripción	Puntuación
Alta (H)	Se requiere el conocimiento de un especialista para explotar la vulnerabilidad o el uso de técnicas de ingeniería social.	0,35
Media (M)	Existen restricciones para realizar el ataque como pueden ser límites regionales, o	0,61

	configuraciones concretas en el sistema.	
Baja (L)	No existen condiciones especiales para realizar el ataque. Es relativamente sencillo explotar la vulnerabilidad.	0,71

Autenticación (Au): refleja el nivel de autenticación requerido para conseguir explotar la vulnerabilidad.

Valor	Descripción	Puntuación
Múltiple (M)	La ejecución del ataque requiere dos o más autenticaciones por parte del atacante.	0,45
Simple (S)	El atacante necesita usar una única combinación de usuario y contraseña para atacar.	0,56
Ninguna (N)	No es necesario el uso de credenciales de acceso para explotar la vulnerabilidad.	0,704

Confidencialidad (C): refleja el impacto sobre la confidencialidad de la información del sistema tras explotar la vulnerabilidad

Valor	Descripción	Puntuación
Ninguna (M)	No hay impacto sobre la confidencialidad del sistema.	0,0

Parcial (P)	Hay una exposición considerable pero no total de la confidencialidad del sistema.	0,275
Completa (C)	Tras explotar la vulnerabilidad se puede acceder al total de la información del sistema.	0,660

Integridad (I): refleja el impacto sobre la integridad del sistema tras explotar la vulnerabilidad

Valor	Descripción	Puntuación
Ninguna (M)	No hay impacto sobre la integridad del sistema.	0,0
Parcial (P)	Existe una afectación parcial a la integridad del sistema.	0,275
Completa (C)	Hay pérdida total de la integridad del sistema. El atacante puede modificar cualquier dato del sistema..	0,660

Disponibilidad (A): refleja el impacto sobre la disponibilidad del sistema tras explotar la vulnerabilidad

Valor	Descripción	Puntuación
Ninguna (M)	No hay impacto sobre la disponibilidad del sistema.	0,0
Parcial (P)	El sistema queda temporalmente inaccesible.	0,275

Completa (C)	Tras explotar la vulnerabilidad el sistema queda totalmente inaccesible.	0,660

A partir de los valores anteriores se calcula el valor de impacto y el de explotabilidad

$$Explotabilidad = 20 * AV * AC * Au$$

$$Impacto = 10{,}41 * (1 - (1 - C) * (1 - I) * (1 - A))$$

Por último se obtiene el valor CVSS de la siguiente manera:

$$CVSS = ((0{,}6 * Impacto) + (0{,}4 * Explotabilidad) - 1{,}5) * f(impacto)$$

Siendo

$$f(impacto) \begin{pmatrix} 0, & si\ impacto = 0 \\ 1{,}176, & si\ impacto \neq 0 \end{pmatrix}$$

3. MODELOS DE DESARROLLO

Las fases del desarrollo en la mayoría de los modelos de desarrollo se pueden organizar dentro de las siguientes categorías:

Cada una de estas fases dentro del ciclo de vida del desarrollo aporta elementos diferentes pero necesarios en su conjunto para que la calidad del software desarrollado sea óptima.

En materia de seguridad, debería ocurrir lo mismo.

El problema principal que existe hoy en día respecto a la seguridad de las aplicaciones es la poca conciencia que se tiene desde los equipos de desarrollo sobre los riesgos anteriormente comentados.

Las tecnologías avanzan a pasos agigantados, surgen nuevos lenguajes o modelos de desarrollo y sin embargo la seguridad parece haberse estancado y reducido a dos elementos:

Checklist de seguridad: habitualmente dentro de las organizaciones las áreas de desarrollos no se integran con los departamentos de seguridad o viceversa. Siendo estos últimos los que obligan a los desarrolladores a seguir una serie de recomendaciones de codificación segura, insuficiente por si sola para proveer a la aplicación del marco de seguridad deseable.

Pentest: los ‘Hackings’, ‘Pentest’ o ‘test de intrusión’, son pruebas realizadas por expertos en seguridad dentro de la fase de despliegue una vez que el software ha sido desarrollado. Por este motivo, la corrección de los riesgos identificados en esta etapa suelen ser de un coste elevado o técnicamente inviables de corregir sin afectar la disponibilidad del servicio.

3.1. Modelos de desarrollo clásicos

Existen varios modelos de desarrollo denominados clásicos, los cuales destacan por dedicar gran parte del esfuerzo en las fases de planificación y diseño:

Modelo cascada

Este modelo se puede considerar como el predecesor de todos los modelos de desarrollo y ha servido de base para otros modelos.

En el modelo de cascada se organiza el proyecto a través de una serie de etapas diferentes, partiendo desde el desarrollo del concepto inicial del sistema hasta la prueba final del entregable del mismo.

En el proyecto se realiza una revisión al final de cada etapa para determinar si está preparado para pasar a la siguiente.

Modelo evolutivo

En este modelo a diferencia del de cascada, el concepto del producto se desarrolla progresivamente a medida que avanza el proyecto y no se trata en una etapa inicial.

Se comienza desarrollando los conceptos más destacados del sistema, posteriormente se presenta la parte ya desarrollada del sistema al cliente y se continúa el desarrollo del mismo en base al *feedback* que se recibe partir de los entregables.

Modelo incremental

El sistema se muestra al cliente en etapas refinadas sucesivamente.

A diferencia del modelo de prototipado evolutivo, se conoce exactamente qué es lo que se va a construir cuando se procede a construirlo.

Lo que hace diferente a este modelo es que el sistema no se entrega como un todo al final del proyecto, sino que se entrega por etapas sucesivas a lo largo del proyecto.

3.2. Modelos de desarrollo ágiles

En contraposición con los *modelos de desarrollo clásico*s anteriormente mencionados existen los denominados *modelos de desarrollo ágiles* cuyo objetivo es proporcionar al ciclo de vida del desarrollo un dinamismo que permita la interacción entre diferentes equipos o unidades de forma ágil como su nombre lo indica centrando la mayor parte del esfuerzo en la codificación y proporcionando una alta capacidad de adaptación en tiempo real durante el desarrollo para cumplir las expectativas y requerimientos funcionales.

SCRUM

El modelo de desarrollo SCRUM se caracteriza principalmente por la definición una serie de roles y prácticas que aportan a la ejecución del proyecto un alto nivel de dinamismo a través de la creación de equipos auto-organizados y una fluida comunicación.

Roles principales de SCRUM:

- ScrumMaster: es el líder del equipo y su rol consiste en velar por la cohesión del equipo, ejercer de coaching y fomentar la comunicación siempre con el objetivo de transmitir los valores y principios ágiles.
- ProductOwner: representante de los interesados internos o externos, o dicho de otra forma, es el interlocutor del cliente o inversores.
- Team: el equipo lo conforman todos aquellos que de una forma u otra participan en el desarrollo del software, desde programadores hasta técnicos de sonido pasando por técnicos de calidad o cualquier otro rol que se nos ocurra.

Principios ágiles (Agile Manifesto, 2001):

"We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value"

- **Individuals and interactions** over processes and tools,
- **Working software** over comprehensive documentation,
- **Customer collaboration** over contract negotiation,
- **Responding to change** over following a plan.

A su vez SCRUM, cuenta con **doce principios** (Principles behind the Agile Manifesto, 2001):

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.

3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

4. Business people and developers must work together daily throughout the project.

5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

7. Working software is the primary measure of progress.

8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

9. Continuous attention to technical excellence and good design enhances agility.

10. Simplicity--the art of maximizing the amount of work not done--is essential.

11. The best architectures, requirements, and designs emerge from self- organizing teams.

12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Las *historias de usuario* (scrummanager.net, 2014)*, como se observa en la* Ilustración 5 - Ejemplo de historia de usuario- son el punto de partida del proyecto en SCRUM y consisten en el desarrollo de una serie de casos de uso de la aplicación mediante las que se define las acciones, procesos y funcionalidades del sistema a desarrollar. Estas historias deben ser simples gramaticalmente y precisas, especificando claramente el actor y la acción a través de una estructura que contenta el nombre de la historia, una descripción de la funcionalidad y el criterio de validación. Por ejemplo, "El usuario puede enviar un mensaje a otro" en el caso de un programa de mensajería instantánea o "El administrador configura el catálogo de productos" en una hipotética tienda virtual de productos.

Historia de Usuario	
Número: 1	**Usuario:** Cliente
Nombre historia: Cambiar dirección de envío	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Baja
Puntos estimados: 2	**Iteración asignada:** 1
Programador responsable: José Pérez	
Descripción: Quiero cambiar la dirección de envío de un pedido.	
Validación: El cliente puede cambiar la dirección de entrega de cualquiera de los pedidos que tiene pendientes de envío.	

Ilustración 5 - Ejemplo de historia de usuario

A partir de estas historias se comienzan a desarrollar de forma iterativa los Sprint (ciclos de la aplicación):

Los *sprints* en SCRUM son los diferentes ciclos de trabajo que se ejecutan a lo largo del período que dura el proyecto como resultado del marco iterativo e incremental.

SCRUM define ciertas características que deben cumplir los *sprints*:

- El ciclo debe ser corto, no es recomendable su extensión en el tiempo más allá de un mes.
- La fecha límite del ciclo es inamovible e inextensible.
- Cada ciclo está orientado a dar respuesta a un requisito concreto del proyecto.
- Dentro de cada ciclo se escogen una serie de ítems a entregar y éstos no se pueden cambiar.
- Cada día durante el ciclo, el equipo debe reunirse para analizar el estado y los próximos pasos a seguir.
- El *sprint* acaba con una reunión del equipo, una exposición de resultados y el análisis del *feedback* del cliente a fin de definir el próximo ciclo.

En la Ilustración 6 - SCRUM Framework- se detalla el marco de desarrollo ágil SCRUM. En ella se observa cómo a partir del inversor o cliente (Product Owner) se desarrollan una serie de historias que representan los casos de uso del sistema (Product Backlog) y a partir de ahí el equipo desarrolla de forma cíclica e iterativa los diferentes sprints para obtener un producto final acorde a las necesidades expuestas por el *product owner.*

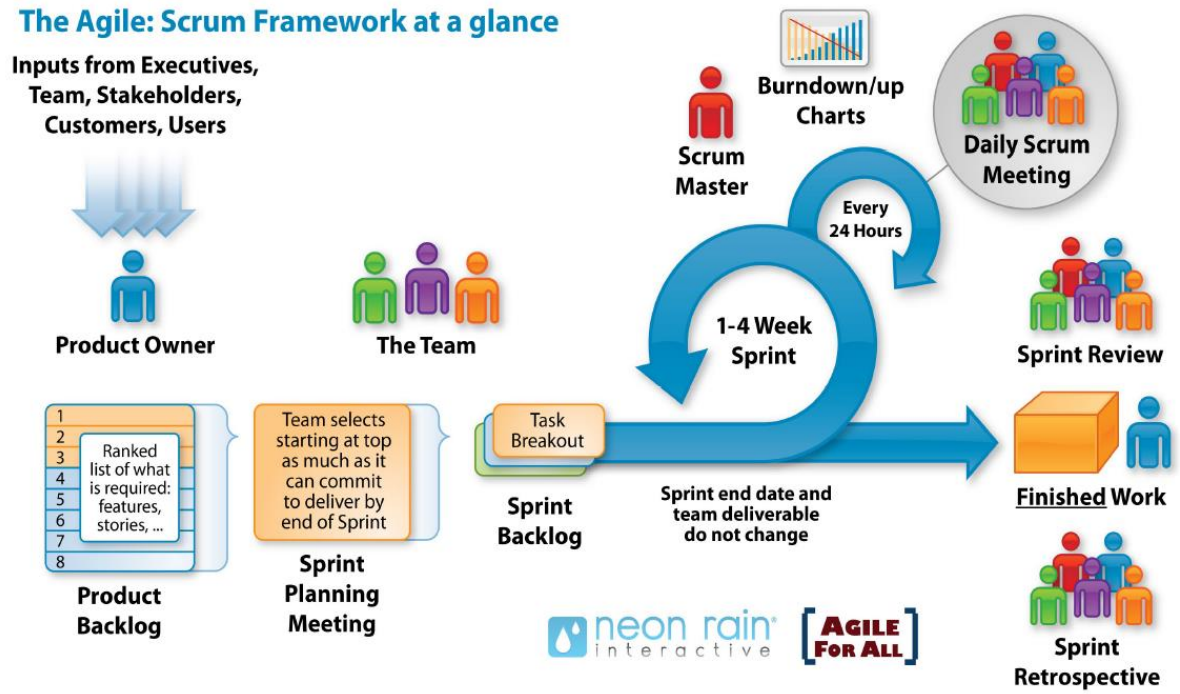


Ilustración 6 - SCRUM Framework

Programación extrema

La programación extrema, también denominada XP, del inglés eXtreme Programming, es otra metodología ágil de desarrollo (Beck, 1996).

Al igual que el resto de metodologías ágiles, consiste principalmente en la creación de historias a partir de las cuales de forma iterativa se llevarán a cabo las tareas de desarrollo, pruebas, correcciones y entregables tal como se observa en la Ilustración 7 - Framework eXtreme Programming-

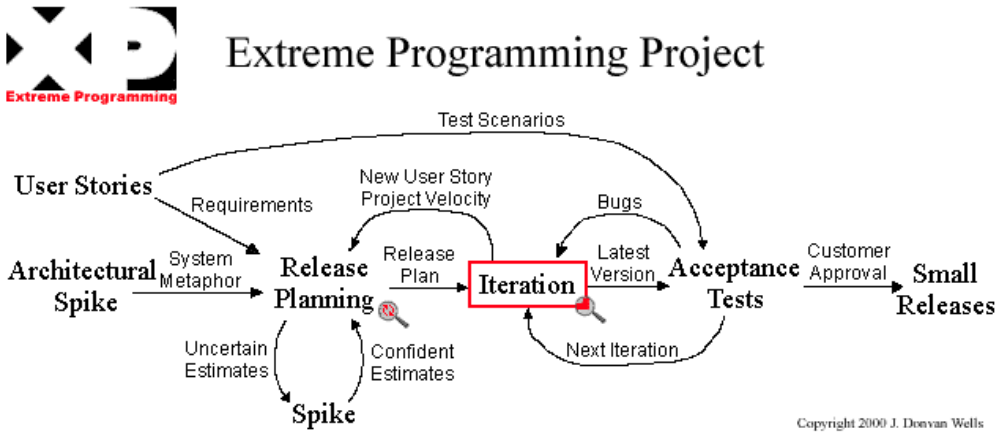


Ilustración 7 - Framework eXtreme Programming

Mikel Cohn resume en abril de 2007 (Cohn, 2007) las principales diferencias entre SCRUM y eXtreme Programming en un artículo:

- En Scrum los equipos trabajan en iteraciones llamadas Sprints que suelen durar entre dos semanas y un mes, mientras en las iteraciones en eXtreme Programming estas duran un máximo de dos semanas.

- Scrum no admite cambios dentro de los *sprints* una vez superada la etapa de planificación del mismo. eXtreme programming es más sensible a los cambios dentro de las iteraciones. En este sentido SCRUM es más intransigente con las modificaciones dentro del *sprint.*

- *eXtreme Programming es más estricto en cuanto a la priorización de las tareas a desarrollar, siendo el cliente el responsable de dicha priorización mientras que en Scrum el equipo tiene mayor autonomía dentro de la capa técnica del desarrollo para poder priorizar o personalizar diferentes características del producto.*

4. SECURE SCRUM

En esta sección se desarrolla una capa de seguridad para modelo de desarrollo SCRUM, denominado SecuRe Scrum (ver MARCO DE CONTROL , página64), que permite mejorar los controles de seguridad existentes en las aplicaciones teniendo en cuenta los actores y las fases dentro del proyecto.

SCRUM representa probablemente uno de los modelos de desarrollo ágiles con más proyección. Lleva ya un tiempo considerable poniéndose en práctica y los resultados han sido muy satisfactorios.

A diferencia de eXtreme Programming donde el cliente se implica con mucho más protagonismo en el desarrollo, en SCRUM, el equipo se siente mucho más importante, consiguiéndose así una mayor cohesión interna e implicación en el proyecto.

El desarrollo del software en SCRUM se puede analizar desde varios puntos de vista y en diferentes etapas, más allá de la naturaleza cíclica que presenta.

SecuRe SCRUM

De forma resumida esta nueva capa de seguridad sobre SCRUM introduce los siguientes elementos

- Incorporación de criterio de validación de seguridad a cada historia.
- Diseño de casos de abuso.
- Nuevo rol “Responsable de seguridad”
- Incorporación de miembros “Security Testers” al equipo del proyecto.
- Concienciación y formación.
- Buenas prácticas de codificación
- Pentest final.

4.1. Responsable de seguridad

Se establece un nuevo rol, Responsable de Seguridad, introduciendo así un actor capaz de identificar riesgos asociados al desarrollo y transmitir conceptos que permitan mejorar los controles de seguridad sobre la explicación.

Este nuevo agente, el Responsable de seguridad, debe tener la capacidad y la experiencia para transmitir al resto de participantes del proyecto los riesgos identificados y conseguir la implicación necesaria de las partes para que el entregable final sea un software robusto, seguro, de calidad y sobre todo "seguro".

Entre sus responsabilidades estarán la de acompañar al Scrum Master durante todo el proyecto para tener una visión global del estado y los requisitos en todo momento.

Definir requisitos mínimos de seguridad al inicio del proyecto en las reuniones diarias dentro de cada ciclo.

Por otro lado, se introducen al equipo uno o varios "*Security Testers",* que junto con el resto del equipo deberán llevar a cabo pruebas sobre todo el código desarrollado a fin de detectar vulnerabilidades y emitir recomendaciones.

4.2. Concienciación y formación de seguridad

En el desarrollo de software intervienen diferentes actores que Scrum resume entre Cliente, Equipo y Scrum Master. Cada uno de ellos debe tomar conciencia sobre los riesgos existentes sobre la aplicación y del impacto que éstos pueden provocar. Por esta razón el responsable de seguridad deberá llevar a cabo una serie de tareas que permitan concienciar y formar al resto de integrantes del proyecto.

4.3. Casos de abuso

Para desarrollar una aplicación de forma ágil o mediante cualquier otro modelo de desarrollo es necesario definir los casos de uso que permitan diseñar la aplicación para lograr los objetivos iniciales.

En la metodología ágil propuesta por este trabajo se debe desarrollar junto con los casos de uso, los casos de abuso.

Los casos de abuso, en contraposición con los anteriores, son aquellas acciones que un usuario legítimo o un atacante pueden llegar a realizar provocando afectaciones sobre los procesos que pongan en riesgo la disponibilidad, integridad y confidencialidad de la información.

Para cada caso de uso definido, se debe analizar que posibles casos de abuso asociados existen, y adicionalmente de forma transversal a todos ellos.

Tanto el equipo de desarrollo como el equipo de seguridad definido por el responsable de seguridad del proyecto llevarán a cabo esta tarea a la par de la definición de los casos de uso al inicio del proyecto y el responsable de la efectividad de esta definición será el responsable de seguridad del proyecto.

4.4. Auditoría de seguridad independiente

La realización de una auditoría técnica de seguridad al final del proyecto representa un punto de control que facilita la certificación del proyecto en materia de seguridad.

Es decir, si después de llevar a cabo los requisitos de SecuRe Scum, durante las fases iniciales del proyecto y los ciclos sprints, algún riesgo se ha escapado al control del responsable de seguridad del proyecto o alguno de los procesos internos de revisión no ha sido eficiente, un equipo independiente revisará la aplicación por completo identificando vulnerabilidades y emitiendo recomendaciones para mitigarlas, que deberán ser tomadas en cuenta a la hora de corregir la aplicación o transmitidas al cliente en caso de que éste estuviera dispuesto a asumir el riesgo.

La independencia del equipo de seguridad, responsable de la realización de la auditoría técnica de seguridad sobre la aplicación, aporta un valor importante sobre la legitimidad del informe redactado ya que los intereses de esté equipo no coinciden con los del equipo de desarrollo.

4.5. Controles de seguridad

Se ha desarrollado un anexo (MARCO DE CONTROL , página 64) con una serie de controles de SecuRe Scrum, para facilitar la implementación o auditoría de la capa de seguridad añadida sobre el modelo Scrum.

5. CASO PRÁCTICO DE UNA AUDITORÍA DE SEGURIDAD WEB, HACKING UB

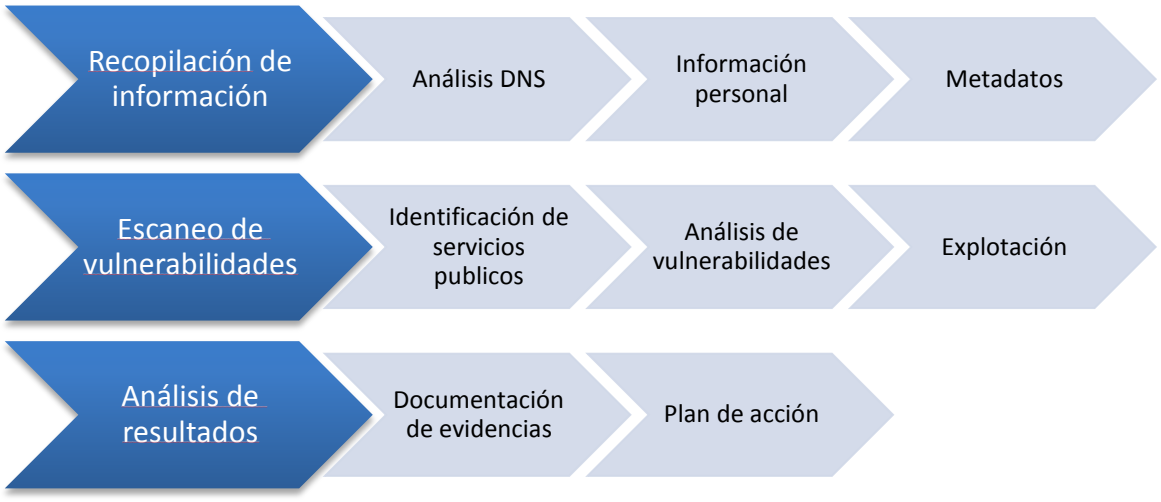
Dada la visibilidad de la Universitat de Barcelona y los riesgos de seguridad asociados a los sistemas de Información, se ha propuesto a la Universitat de Barcelona realizar una revisión de seguridad para identificar los riesgos a los que están expuestos los principales portales de la Universidad.

El presente informe de colaboración profesional reflejas los resultados obtenidos por Gonzalo Vigo Scheiner a petición de la Universitat de Barcelona y contiene información confidencial propiedad de la Universidad que deberá ser utilizada exclusivamente por la misma, prohibiéndose su reproducción o distribución sin autorización.

Objetivos y alcance

El objeto del presente informe es la realización de una auditoría técnica de seguridad a partir de la recopilación de información de la infraestructura pública de la *Universitat de Barcelona* con el fin de conocer la exposición de los servicios públicos y sus vulnerabilidades dentro del siguiente marco:

- Campus virtual UB: https://campusvirtual2.ub.edu/
- Mon UB: http://www.ub.edu/monub/



Fases de la revisión

Recopilación de información

Esta fase consiste en la adquisición de información externa de las plataformas incluidas dentro del alance. Para llevar a cabo este proceso se partirá de la denominación pública de la universidad y de las marcas comerciales que explota tratando de identificar cuál es su presencia en Internet.

Se comenzará identificando los rangos de direcciones IP y dominios registrados a nombre de la Universidad de Barcelona. Para ello se utilizarán diversas fuentes de información:

Registradores regionales:

- African Network Information Centre (AfriNIC) para Africa.
- American Registry for Internet Numbers (ARIN) para Estados Unidos, Canadá, y parte del Caribe.
- Asia-Pacific Network Information Centre (APNIC) para Asia, Australia, Nueva Zelanda y países vecinos.
- Latin America and Caribbean Network Information Centre (LACNIC) para América del Sur.
- Réseaux IP Européens Network Coordination Centre (RIPE NCC) para Europa, Rusia, y Medio Este.

Motores de búsqueda: Google, Bing, Altavista, Yahoo, Open Directory Project, redes P2P (eMule, buscadores de bittorrent), etc.

Información de infraestructura: Robtex, whois, cabeceras de correos, páginas web públicas, etc.

Información interna publicada en Internet: Casos de éxito de fabricantes, presentaciones públicas, metadatos publicados en los recursos de la Entidad, servidores de news, foros, posibles leaks de información en pastebin y webs similares, redes sociales (Facebook, LinkedIn), etc.

Herramientas de inteligencia: Se usarán herramientas especializadas como Maltego y Foca.

Se centrarán los esfuerzos en la identificación de la siguiente información:

- Direcciones de red, dominios y rangos telefónicos de la universidad.

- Direcciones de correo de los empleados y directivos.
- Nombres de usuario o determinar el formato del mismo.
- Direcciones de máquina y roles de servidores.
- Páginas web y servicios públicos expuestos por la universidad.
- Rangos de red internos y servicios internos.
- Aplicaciones y sistemas operativos usados por los ordenadores y servidores de la Entidad o entidades relacionadas.
- Información personal de los empleados y alumnos.
- Filosofía de la seguridad de la universidad.
- Entornos de desarrollo y arquitectura del software expuesto a Internet.
- Determinar los principales activos de IT y su posible ubicación.

Escaneo de vulnerabilidades

El objetivo de esta fase consiste en realizar un análisis de las aplicaciones web dentro del alcance del trabajo para identificar las debilidades que puedan existir. Se tratará de identificar aquellas condiciones que puedan afectar a la confidencialidad, integridad y disponibilidad de la información gestionada por la plataforma así como los procesos y sistemas que la engloban.

La revisión de aplicaciones requerirá una segunda iteración de recolección de información focalizada en las distintas aplicaciones, para lograr un conocimiento más detallado de los aplicativos expuestos, como tecnologías, frameworks, gestores de contenidos, etc.

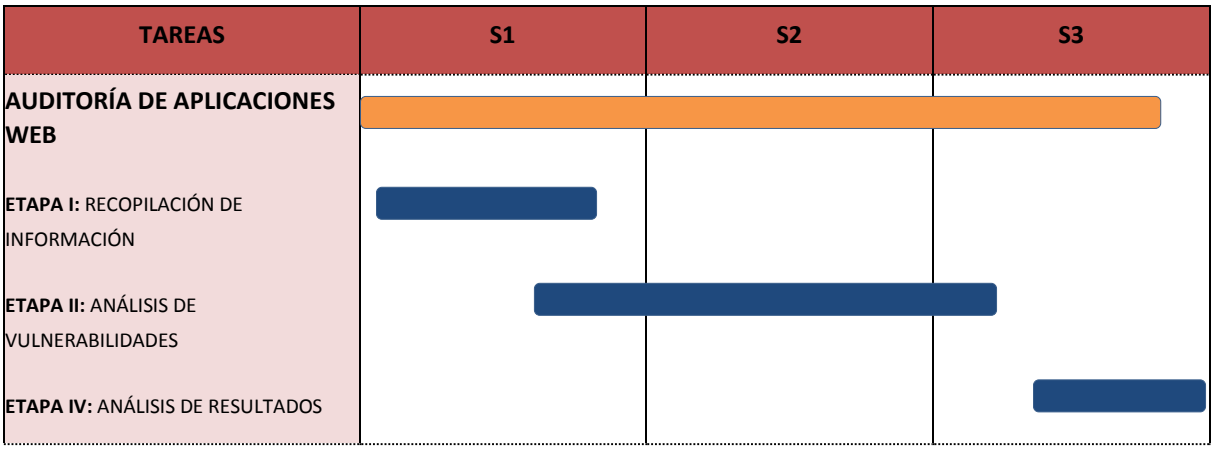
Análisis de resultados

Finalmente, se presentará adjunto al informe de resultados un plan de acción para la mitigación de los riesgos identificados que permita enfocar su implementación en base a los costes y beneficios determinados

5.1. Planificación y honorarios

De acuerdo con la propuesta realizada, se estima que el comienzo de la auditoría se realizará el día 1 de abril de 2015.

Para la realización del proyecto de Revisión de Seguridad Interna se ha estimado un total de 3 semanas de trabajo según el siguiente calendario:



Las pruebas se han llevado cabo por Gonzalo Vigo Scheiner, especialista en seguridad, de forma remota desde la ubicación geográfica de Barcelona

Honorarios

Los honorarios para la elaboración de este trabajos están basados en la experiencia del personal asignado, y en el tiempo necesario para la realización del mismo teniendo en consideración el alcance definido

Los honorarios serán facturados conforme avanzan los trabajos, en base al siguiente calendario de pagos:

- 40% al inicio del proyecto (inicio de la primera semana del proyecto).
- 40% al final del análisis de seguridad (estimada para el final de la segunda semana del proyecto).
- 20 % a la entrega de los borradores de los informes (finalización del proyecto).

Los honorarios han sido dispuestos en función de la información facilitada por la Universitat de Barcelona y la estimación de Gonzalo Vigo Scheiner. Cualquier cambio en el alcance del trabajo o de las circunstancias actuales, que motivan los presentes honorarios puede dar

lugar, previo acuerdo por escrito entre las partes, a una renegociación de los citados honorarios. Los honorarios ascenderán a 8.500 euros.

Dichos honorarios no incluyen el correspondiente Impuesto sobre el Valor Añadido, ni tampoco otros gastos (dietas, desplazamientos, traducciones, etc.), que serán facturados adicionalmente. Las facturas correspondientes serán pagadas en un plazo de treinta días tras su fecha de emisión.

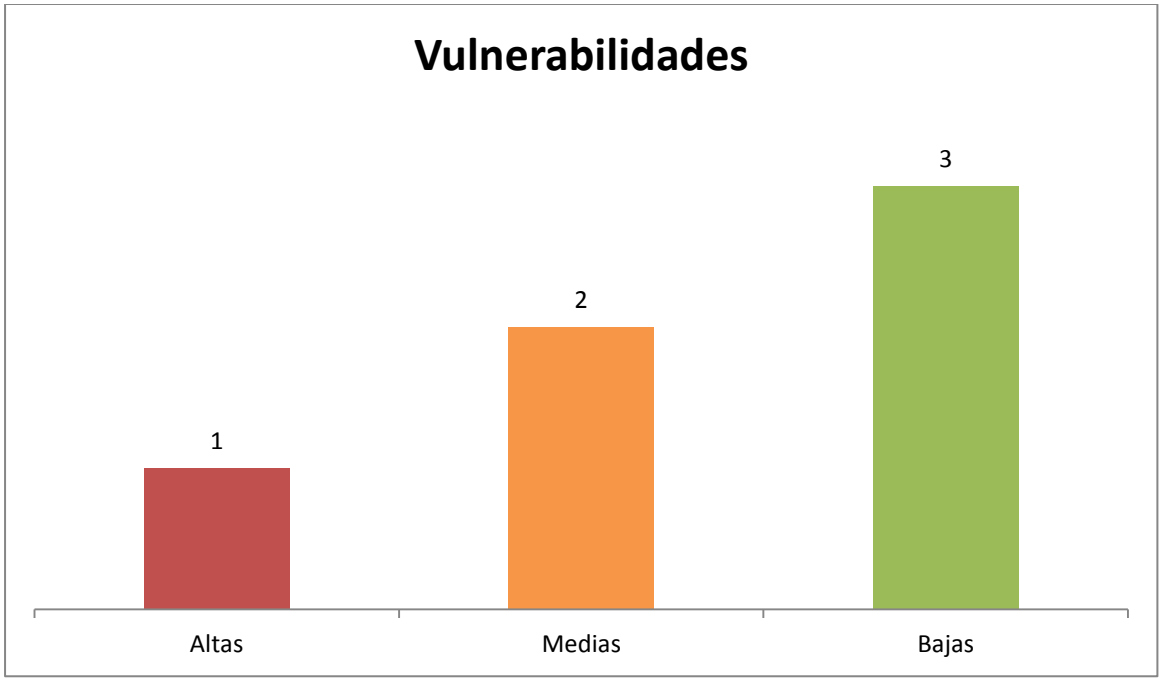
5.2. Resumen ejecutivo

Se ha realizado un análisis de seguridad con enfoque de caja negra, es decir, sin conocimiento previo de la plataforma, sobre las aplicaciones dentro del alcance de esta revisión. La ejecución del proyecto ha sido realizada remotamente desde la ubicación geográfica de la ciudad de Barcelona.

En base a los resultados del trabajo se puede concluir que el nivel de la infraestructura de las plataformas “Campus Virtual 2” y “Mon UB” es Bajo, existiendo una serie de áreas de mejora en las que debe hacerse hincapié.

Las principales vulnerabilidades identificadas corresponden a **fallos de diseño y codificación en la fase de desarrollo** o **debilidades en la configuración del sistema**

En resumen, las vulnerabilidades identificadas son:



Los principales riesgos identificados corresponden a **exposición de información sensible**.

Durante la revisión se ha demostrado que **es posible inyectar y ejecutar código remotamente en el navegador de los usuarios.**

5.3. Recopilación de información

Exposición de servicios

Se han analizado los servicios públicos accesibles desde Internet para la infraestructura de la Universitat de Barcelona incluida dentro del alcance de la auditoría obteniendo los siguientes resultados:

ub.edu (161.116.100.2)

Puerto	Protocolo	Estado	Servicio	Versión
80	tcp	open	http	Apache httpd
110	tcp	open	tcpwrapped	
143	tcp	open	Imap	
443	tcp	open	http	Apache httpd
993	tcp	open	imaps	
995	tcp	open	pop3s	

campusvirtual2.ub.edu (161.116.22.36)

Puerto	Protocolo	Estado	Servicio	Versión
80	tcp	open	http	Apache httpd 2.4.7 ((Unix OpenSSL 1.0.0l PHP/5.5.9)
110	tcp	open	tcpwrapped	
143	tcp	open	Imap	
443	tcp	open	http	Apache httpd 2.4.7 ((Unix OpenSSL 1.0.0l PHP/5.5.9)

993	tcp	open	imaps	
995	tcp	open	pop3s	

Análisis de metadatos

Tras la descarga y análisis de metadatos de 1640 documentos públicos de la universidad de Barcelona se ha obtenido la siguiente información:

Relación dominios-IP de servidores		
greav.ub.edu	161.116.7.203	Apache
www.publicacions.ub.edu	161.116.18.149	Windows 2003 IIS 6.0
stel.ub.edu	161.116.36.139	Ubuntu Apache 2.2.22
clic.ub.edu	161.116.36.185	Apache 2.2.2
ocw.ub.edu	161.116.37.71	ZServer 1.1 Zope Plone 3.2.3
eim.ub.edu	161.116.37.38	Linux Suse Apache 2.2.12
mat.ub.edu	161.116.37.92	Apache
bd.ub.edu	161.116.37.111	Linux Debian Apache 2.2.16 Perl 2.0.4 DAV/2 Python 3.3.1 Perl 2.0.4 PHP 5.3.3.7 SVN 1.6.12
icc.ub.edu	161.116.55.71	Apache
www.am.ub.edu	161.116.78.1	Apache Server

atlas.mat.ub.edu	161.116.84.74	Apache
ub.edu	161.116.100.2	Apache
www2.ub.edu	161.116.100.3	Apache
www1.ub.edu	161.116.100.4	Apache
www.giga.ub.edu	161.116.100.28	Apache
cvirtual2.il3.ub.edu	161.116.100.35	Apache 2.2.23 Linux PHP 5.3.5
www4.ub.edu	161.116.100.179	Apache
tcub.ub.edu	161.116.107.22	Linux Suse Apache 2.2.10
il3.ub.edu	161.116.107.22	Apache
www.pcb.ub.edu	161.116.196.196	Perl 2.0.3
comercioexterior.ub.edu	161.116.196.196	Apache
mmb.pcb.ub.edu	161.116.222.59	Apache
alumni.ub.edu	No localizado	Apache
am.ub.edu	No localizado	Apache
bib.ub.edu	No localizado	Apache 2.2.16
giga.ub.edu	No localizado	Apache 2.0.48 PHP 4.3.5 Linux
pcb.ub.edu	No localizado	Apache
publicacions.ub.edu	No localizado	Apache

solidaritat.ub.edu	No localizado	Apache
diposit.ub.edu	161.116.100.2	Apache

Listado de ficheros.txt

Usuarios

Tras analizar los metadatos de todos los documentos públicos de la Universitat de Barcelona se ha obtenido un total de 8 usuarios:

Usuarios
usuari
PURI
jarrese
jlain
docent
MARPAVE
Jmbemudo
Carles Ameller

5.4. Información detallada de vulnerabilidades

5.4.1. HACKUB - 001 XSS (Cross Site Scripting)

Identificador	HACKUB - 001
Activo afectado	http://www.ub.edu/monub
Categoría OWASP	Secuencia de comandos en sitios cruzados (XSS)
Vector CVSS	AV:N/AC:M/Au:N/C:C/I:P/A:N/E:H/RL:W/RC:UR
Criticidad CVSS	**7 (Alta)**
Dificultad de resolución	Media
Número de ocurrencias	17

RIESGO

Es posible ejecutar software malicioso en el navegador de los usuarios que accedan al portal MonUB (http:/www.ub.edu/monub).

DETALLES

Se ha detectado durante la revisión que, mediante la manipulación de algunos parámetros enviados al servidor a través de scripts, es posible ejecutar software de forma transparente en el navegador del usuario.

A continuación se listan los scripts y parámetros afectados:

- http://www.ub.edu/monub/cgi/interventor.pl
  - Assoc
  - Centre
  - Correu
  - Dni
  - Eleccio

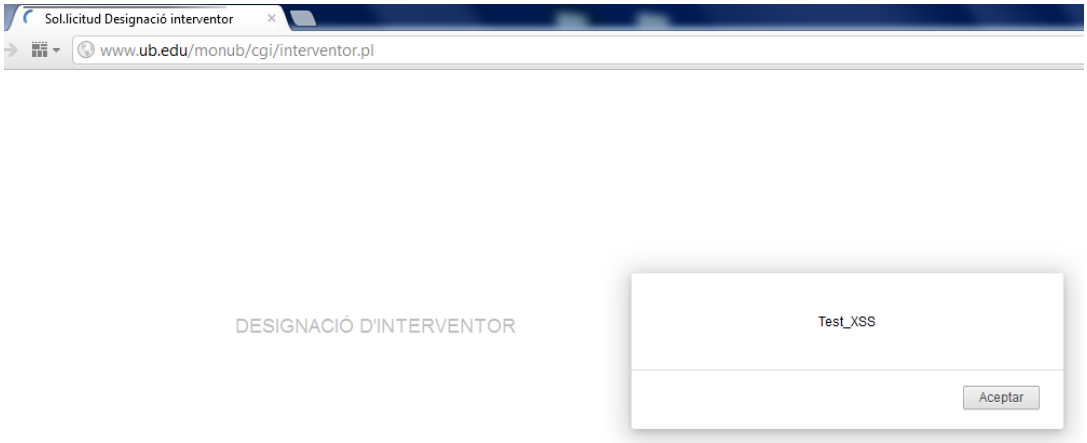
    - Niub
    - Nom
    - Tel
- http://www.ub.edu/monub/cgi/vot_anti.pl
    - Adreca
    - Centre
    - Correu
    - Correu_urgent
    - Dni
    - Ense
    - Niub
    - Nom
    - Tel

En ambos casos las peticiones se envían mediante el método HTTP POST, lo que implica que el contenido de dichos parámetros es transparente para el usuario. Disminuyendo así la capacidad del éste de percatarse del ataque.

Para entender con más detalle la vulnerabilidad se ha realizado una prueba de concepto con el parametro 'adreca' del script http://www.ub.edu/monub/cgi/interventor.pl:

```
POST /monub/cgi/vot_anti.pl HTTP/1.1
Content-Length: 305
Content-Type: application/x-www-form-urlencoded
Cookie: UBSess=34176801687954285 2356; UBSes3=521359942943912852356
Host: www.ub.edu
Connection: Keep-alive
Accept-Encoding: gzip,deflate
User-Agent: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; WOW64;
Trident/5.0)
Accept: */*

adreca=<script>alert("Test_XSS");</script>&bustia_email=eleccions.estudiants%40ub.ed
u&centre=Escola%20Universit%e0ria%20d%27Infermeria&correu=1&correu_urgent=SI&d
ni=1&ense=Administraci%f3%20i%20Direcci%f3%20d%27Empreses&niub=1&nom=avxq
ukij&Submit=Confirmar&tel=555-666-0606
```

En la siguiente captura se puede observar como el contenido del parámetro 'adreca' ha sido procesado fuera de su contexto obteniendo como resultado la ejecución de una ventana emergente, en JavaScript 'alert()', en el contexto del navegador del usuario.



Como consecuencia de esta vulnerabilidad, se podría ejecutar cualquier tipo función JavaScript a través de los parámetros mencionados o incluso llamar mediante una función 'include()' a un script malicioso disponible en cualquier otro recurso en internet para llevar a cabo ataques más elaborados.

RECOMENDACIÓN

Es recomendable revisar los procesos de validación de datos de entrada a través de los parámetros vulnerables.

Se debería controlar el contenido de los parámetros para asegurar que éstos contengan exactamente el tipo de datos y longitud esperada y descartar automáticamente su contenido en caso de que se detecten parámetros de código JavaScript o cualquier otro lenguaje.

5.4.2. HACKUB - 002 Cifrado débil soportado

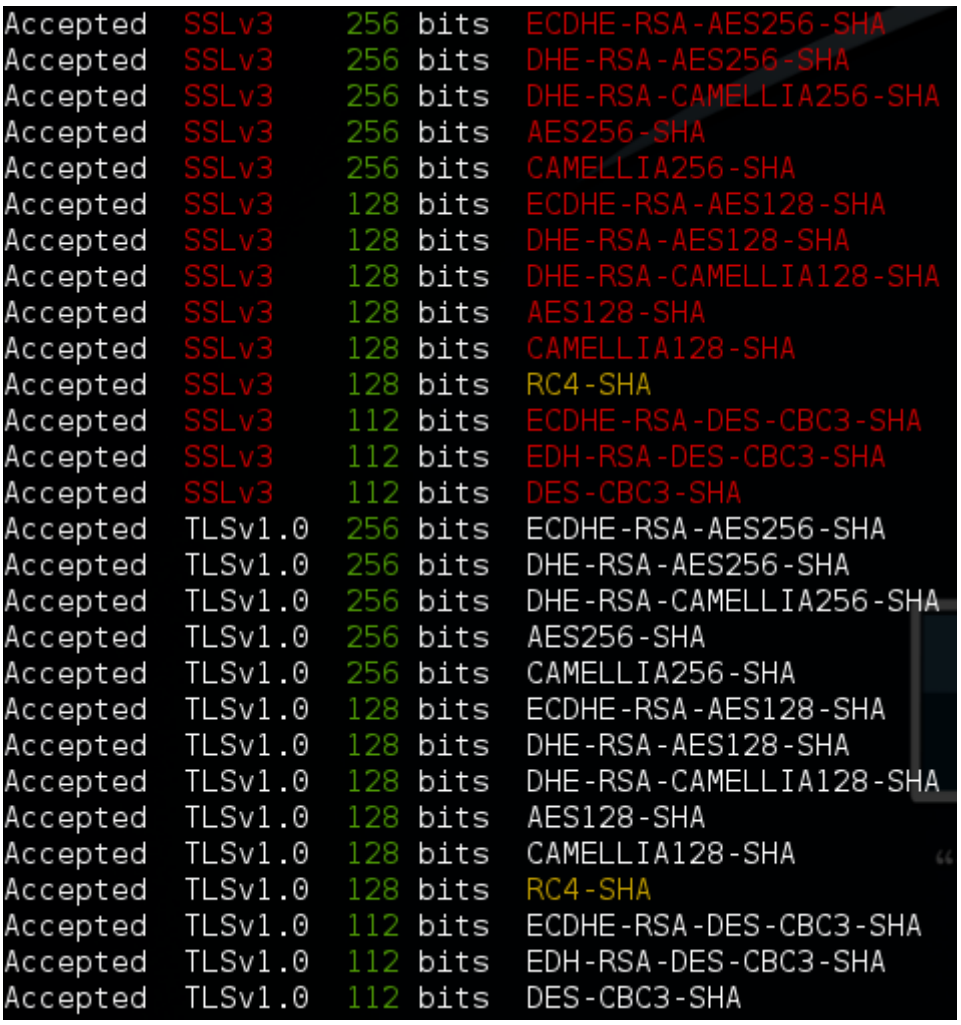
Identificador	HACKUB - 002
Activo afectado	https://www.ub.edu/monub https://campusvirtual2.ub.edu
Categoría OWASP	Configuración de seguridad incorrecta:
Vector CVSS	(AV:N/AC:M/Au:S/C:C/I:N/A:N
Criticidad CVSS	**6,3 (Media)**
Dificultad de resolución	Media
Número de ocurrencias	1

RIESGOS

Un atacante podría aprovechar la debilidad del certificado configurado en el servidor https://campusvirtual2.ub.edu para comprometer las comunicaciones entre el cliente y el servidor web exponiendo también al servidor y los usuarios a ataques 'Man in the middle', es decir, donde el atacante consigue interponerse en la comunicación del cliente y el servidor.

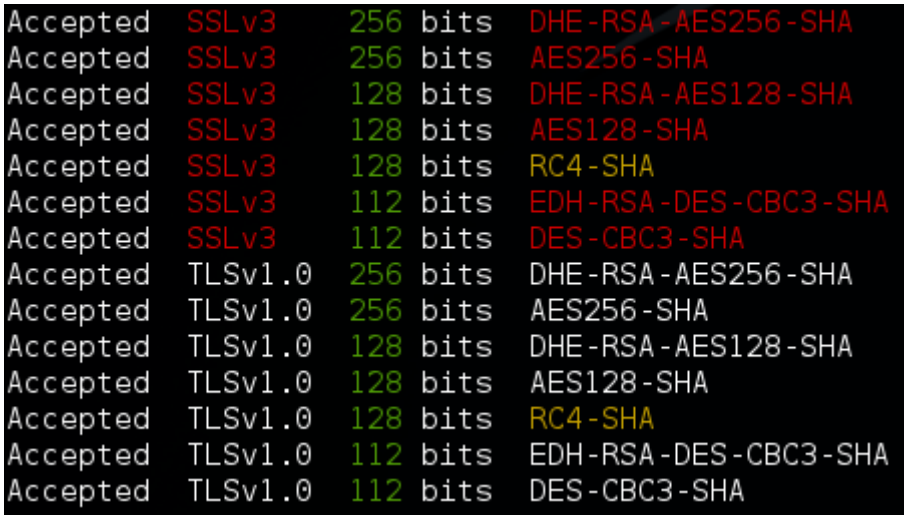
DETALLES

Se ha detectado que el servidor web https://campusvirtual2.ub.edu soporta varios certificados débiles como RC4-SHA:



Adicionalmente, el protocolo SSLv3 está activado para el establecimiento de comunicaciones seguras. Este protocolo ha sido detectado como vulnerable desde el año 2013.

Lo mismo ocurre con el servidor web del portal monUB (https://www.ub.edu/monub).



Se puede obtener más información sobre la vulnerabilidad POODLE que afecta a SSLv3 en el siguiente enlace:

- https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-3566

RECOMENDACIONES

Se recomienda reconfigurar el servidor para deshabilitar el soporte a cifrados SSL, permitiendo únicamente aquellos que provean un alto nivel de seguridad protegiendo así la confidencialidad e integridad de las comunicaciones.
Además, se recomienda hacer uso del protocolo TLSv1.2 para establecer en todo momento comunicaciones seguras entre el usuario y el servidor.

5.4.3. HACKUB – 003 Exposición de información sensible del servidor

Identificador	HACKUB - 003
Activo afectado	http://www.ub.edu/monub https://campusvirtual2.ub.edu
Categoría OWASP	Exposición de datos sensibles
Vector CVSS	AV:N/AC:H/Au:S/C:P/I:P/A:P
Criticidad CVSS	**4,6 (Media)**
Dificultad de resolución	Baja
Número de ocurrencias	1

RIESGO

Mediante el análisis del contenido de las cabeceras HTTP de las peticiones enviadas al servidor web y sus respuestas es posible obtener datos concretos sobre productos y versiones instaladas en él.

Esta información podría ser aprovechada por un atacante para diseñar un ataque dirigido a los productos instalados en el servidor web, incluso aprovechando la existencia de vulnerabilidades reconocidas por el fabricante para las versiones expuestas.

DETALLES

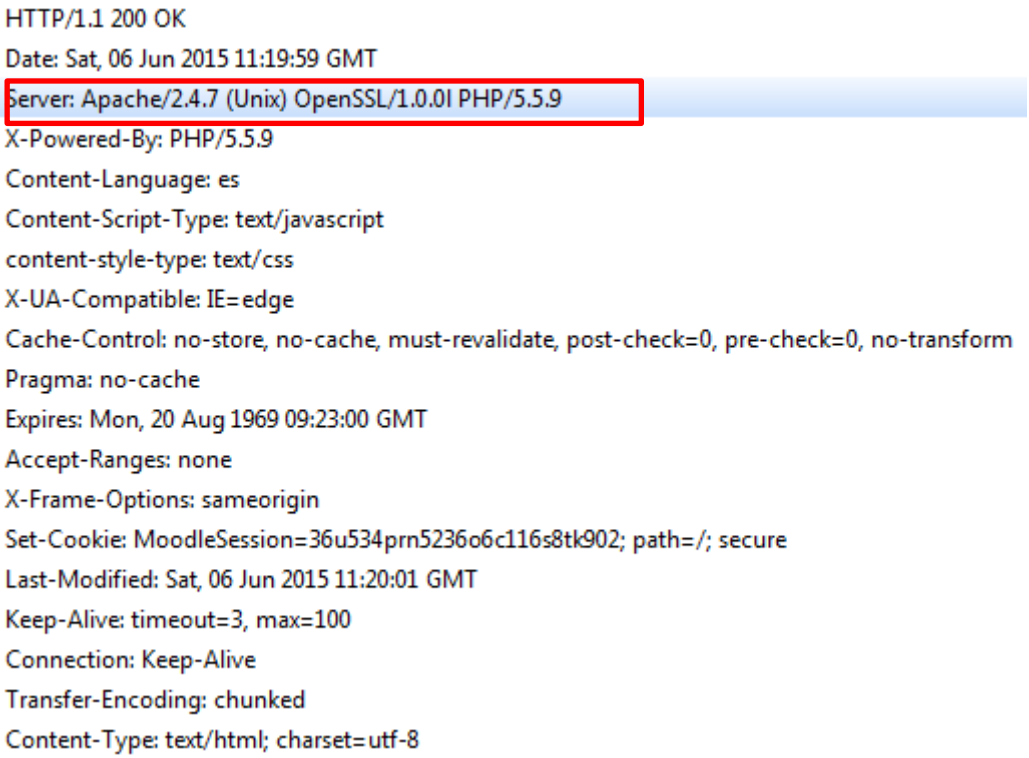
Es posible obtener la versión del servidor web que soporta el portal http://www.ub.edu/monub analizando las cabeceras Http como se observa a continuación:

```
https://www.ub.edu/portals/monUB/

GET /portals/monUB/ HTTP/1.1
Host: www.ub.edu
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:18.0) Gecko/20100101 Firefox/18.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: es-ES,es;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Cookie: JSESSIONID=a174659530de77e147daa94048f0be67ca62b533c7be.e34Mb38SbxeNe3qTchaRaNuQbi0; UBSess=2653499372108828546...
Connection: keep-alive

HTTP/1.1 200 OK
Date: Sat, 06 Jun 2015 10:52:23 GMT
Server: Oracle-Application-Server-10g/10.1.3.1.0 Oracle-HTTP-Server
Content-Length: 30991
Expires: Tue, 01 Jan 2007 1:00:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0,false
Pragma: no-cache
Content-Type: text/html;charset=iso-8859-1
Set-Cookie: JSESSIONID=a174659530de77e147daa94048f0be67ca62b533c7be.e34Mb38SbxeNe3qTchaRaNuQbi0; path=/portals/monUB
Keep-Alive: timeout=3, max=100
Connection: Keep-Alive
```

De igual forma es posible obtener información sensible del portal web https://campusvirtual2.ub.edu por ejemplo enviando la siguiente petición:

```
GET / HTTP/1.1
Host: campusvirtual2.ub.edu
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:18.0) Gecko/20100101 Firefox/18.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: es-ES,es;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Cookie: __utma=270291637.1066360475.1433583689.1433583689.1433587946.2; __utmz=270291637.1433583689.1.1.utmccn=(...
Connection: keep-alive
```

Ya que obtendriamos la siguiente respuesta donde se observa claramente la versión del servidor web instalada:



RECOMENDACIONES

Es recomendable no transmitir a través de las cabeceras información sensible, como lo son datos acerca de productos y versiones instaladas en el servidor web.
Una forma de correcta de configurar el parametro Server de la cabecera HTTP sería estableciendo en él un valor como “Servidor Web” o directamente eliminar el campo de la cabecera.

5.4.4. HACKUB - 004 Cookies de sesión sin atributo 'HttpOnly'

Identificador	HACKUB - 004
Activo afectado	http://www.ub.edu/monub
Categoría OWASP	Configuración de seguridad incorrecta:
Vector CVSS	AV:N/AC:M/Au:S/C:P/I:N/A:N
Criticidad CVSS	**3,5 (Baja)**
Dificultad de resolución	Baja
Número de ocurrencias	1

RIESGO

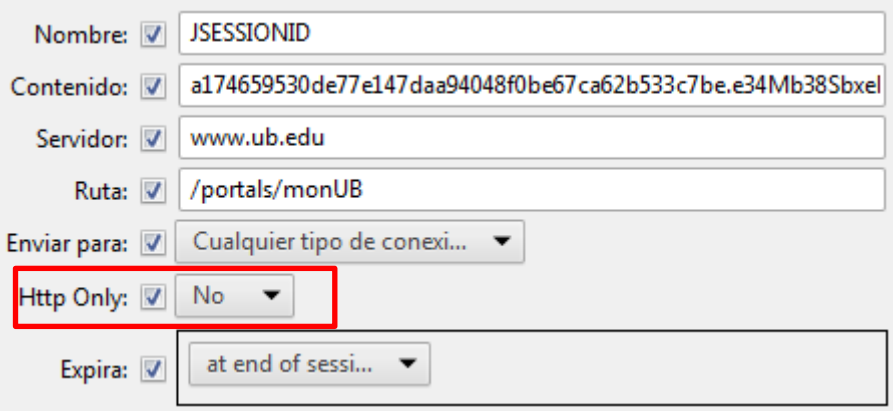
Algunas cookies han sido configuradas sin el atributo 'HttpOnly'. Lo que significa que se podría acceder a dichas cookies mediante scripts JavaScript.

Si un usuario malicioso consigue ejecutar en las páginas del sitio web afectado scripts manipulados podría acceder a dicha información obteniendo datos sensibles sobre las sesiones de los usuarios.

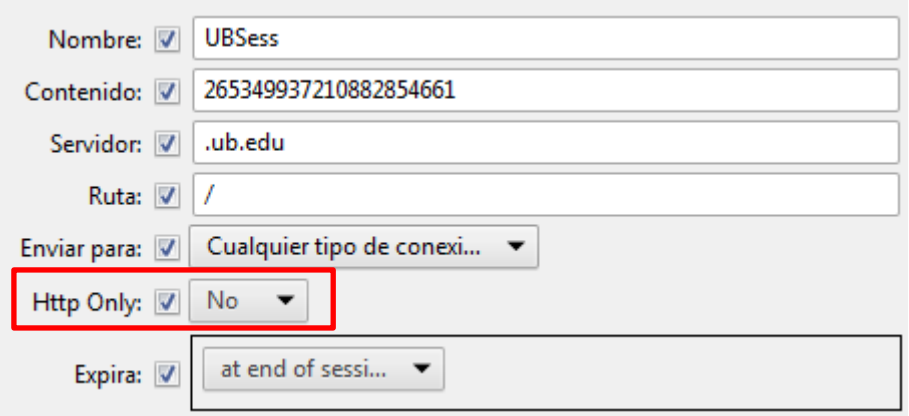
DETALLES

Se ha observado durante la revisión que las siguientes cookies de sesión del portal monUB (http://www.ub.edu/monub) no se han configurado con el atributo 'HttpOnly':

- JESSESIONID



- UBSess



RECOMENDACIONES

Se recomienda configurar las cookies de sesión de forma tal que no puedan ser accesibles mediante código JavaScript ejecutado en el navegador del usuario. Para ello es necesario activar el atributo 'HttpOnly de las cookies'.

5.4.5. HACKUB - 005 Cookies de sesión sin atributo 'Secure'

Identificador	HACKUB - 005
Activo afectado	http://www.ub.edu/monub
Categoría OWASP	Configuración de seguridad incorrecta:
Vector CVSS	AV:N/AC:M/Au:S/C:P/I:N/A:N
Criticidad CVSS	**3,5 (Baja)**
Dificultad de resolución	Baja
Número de ocurrencias	1

RIESGO

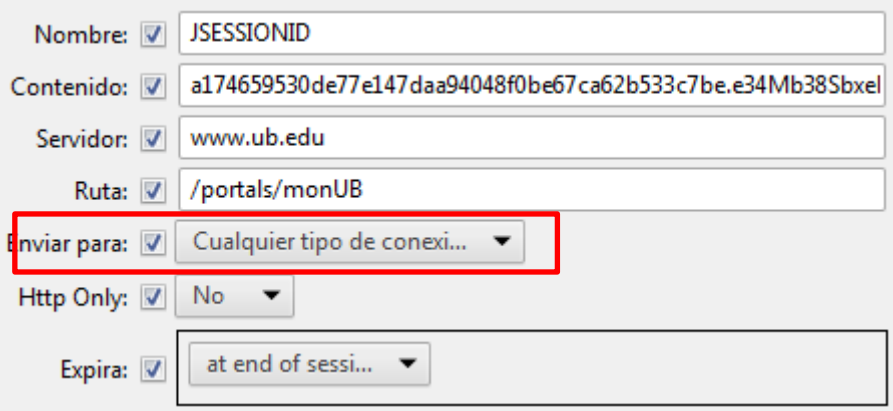
Algunas cookies han sido configuradas sin el atributo 'Secure'. Lo que significa que se acepta el envío de dichas cookies mediante protocolos no seguros (sin cifrados).

Si un usuario malicioso consigue interceptar la comunicación entre el usuario y el servidor web, podría obtener el contenido de dichas cookies sin ninguna restricción ya que se envían en texto plano (sin cifrar).

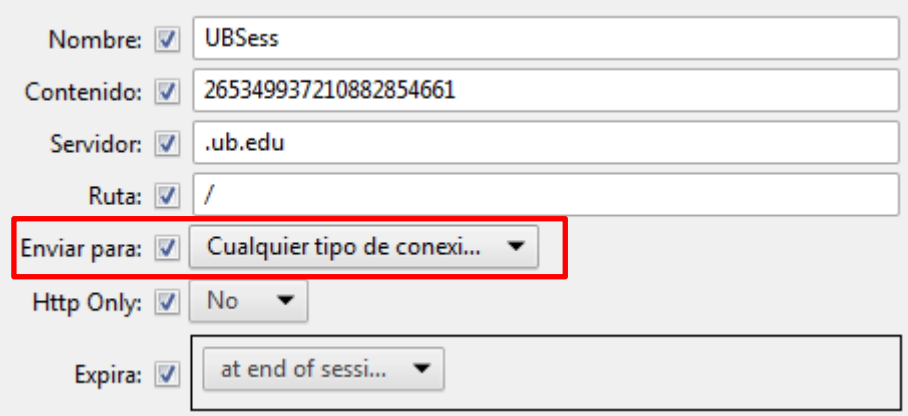
DETALLES

Se ha observado durante la revisión que las siguientes cookies de sesión del portal monUB (http://www.ub.edu/monub) no se han configurado con el atributo 'Secure':

- JESSESIONID



- UBSess



RECOMENDACIONES

Se recomienda configurar las cookies de sesión de forma tal que no puedan transmitidas mediante protocolos seguros de comunicación (HTTPS).
Para ello es necesario activar el atributo 'Secure' de las cookies vulnerables.

5.4.6. HACKUB - 006 Método HTTP TRACE habilitado

Identificador	HACKUB - 006
Activo afectado	https://campusvirtual2.ub.edu
Categoría OWASP	Configuración de seguridad incorrecta:
Vector CVSS	AV:N/AC:H/Au:S/C:P/I:N/A:N
Criticidad CVSS	**2,1 (Baja)**
Dificultad de resolución	Baja
Número de ocurrencias	1

RIESGO

El método HTTP TRACE se encuentra activado en el servidor web. Si existiera una vulnerbilidad 'cross-domain', en el mismo servidor esto provocaría que fuera posible obtener información sensible de las cabeceras HTTP de los usuarios por parte de un atacante.

DETALLES

El método HTTP Trace está activado en el servidor web de https://campusvirtual2.ub.edu como se observa a continuación.

Si se realiza la siguiente petición al servidor web:

```
TRACE /SnbuJBKcHq HTTP/1.1
Cookie: MoodleSession=t8e9vql4rtm9rb5n8t8au8guq6;
UBSess=644217013920752851868; UBSes3=708500801443252851868
Host: campusvirtual2.ub.edu
Connection: Keep-alive
Accept-Encoding: gzip,deflate
User-Agent: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; WOW64;
Trident/5.0)
Accept: */*
```

Se obtiene como respuesta el mensaje HTTP 200, lo que confirma que la petición ha sido realizada correctamente y se devuelve el recurso solicitado:

```
HTTP/1.1 200 OK
Date: Thu, 04 Jun 2015 11:09:03 GMT
Server: Apache/2.4.7 (Unix) OpenSSL/1.0.0l PHP/5.5.9
Keep-Alive: timeout=3, max=99
Connection: Keep-Alive
Content-Type: message/http
Content-Length: 321
```

RECOMENDACIONES

Se recomienda deshabilitar el método TRACE del servidor web a no ser que sea estrictamente necesario para alguna función concreta.

5.5. Roadmap de soluciones

Para una solución integral de las vulnerabilidades y los principales riesgos identificados a partir de la auditoría se ha diseñado el siguiente plan de acción:

1. Seguridad perimetral

   La presenta auditoría ha sido realizada de forma remota a través de Internet. Durante el periodo que ha durado el proceso de revisión no se ha identificado ningún dispositivo de seguridad perimetral en la infraestructura de la universidad.

   Se recomienda valorar la posibilidad de integrar las siguientes soluciones en base a los resultados obtenidos del análisis de riesgos (Punto 3):

   - Mecanismos de protección Anti-DDoS

     Los ataques DDoS (Denegación de servicio destruida) son cada vez más comunes. Representan un 20% de los incidentes reportados por las organizaciones y pueden dejar inactivos los sistemas de la infraestructura por horas e incluso días.

     Para ello existen diferentes productos capaces de filtrar o descartar las peticiones realizadas de forma remota a través de diferentes protocolos para aceptar únicamente el tráfico legítimo.
   - Firewalls (cortafuegos): Estos dispositivos permiten configurar una lista blanca o lista negra de puertos asociados a direcciones IP o rangos de IP bloqueando y filtrando la exposición de los servicios internos de la Universidad.
   - IDS (Intrusion Detection System): Sistemas de detección de tráfico malicioso a partir de expresiones regulares y algoritmos heurísticos aplicados a la comunicación.
2. Desarrollo seguro (SecuRe Scrum)

   Como se ha observado en el informe detallado de las vulnerabilidades (apartado Información detallada de vulnerabilidades) las mayoría de las vulnerabilidades identificadas corresponden a fallos de diseño y codificación.

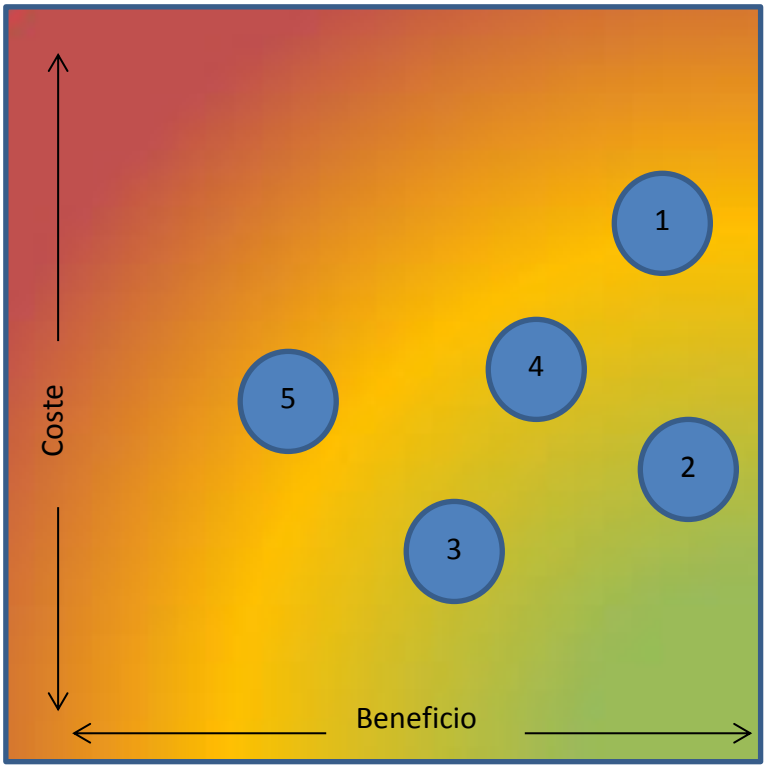
   Se recomienda implementar un modelo de desarrollo ágil con la capa de seguridad "SecuRe Scrum" (ver apartado SECURE SCRUM). Dicho modelo consiste en la incorporación de una capa de seguridad al desarrollo capaz de prevenir, identificar y mitigar riesgos en las fases de diseño y codificación, lo que implica una reducción considerable a los costes correctivos antes de la publicación del sistema.
3. Análisis de riesgos

   Se recomienda llevar a cabo un análisis de riesgos periódico sobre las principales áreas de la Universitat de Barcelona con el objetivo de identificar posibles riesgos no contemplados y realizar un plan de acción acorde a fin de mitgarlos.
4. Corrección de vulnerabilidades de criticidad media y alta:
   - HACKUB - 001 XSS (Cross Site Scripting)

- HACKUB - 002 Cifrado débil soportado
- HACKUB – 003 Exposición de información sensible del servidor

5. Corrección de vulnerabilidades de criticidad baja
   - HACKUB - 004 Cookies de sesión sin atributo ‘HttpOnly’
   - HACKUB - 005 Cookies de sesión sin atributo ‘Secure’
   - HACKUB - 006 Método HTTP TRACE habilitado

El siguiente gráfico expone las soluciones en función de su ratio coste-beneficio e identifica aquellos considerados “quick-win” (sombreado verde):



6. CONCLUSIONES

Las amenazas y los riesgos existentes en los sistemas y en las aplicaciones varían de forma constante, aparecen nuevas vulnerabilidades y nuevos atacantes que ponen en jaque a las organizaciones. Por esta razón ningún análisis o auditoría podrá certificar, por más acotado y definido que sea el alcance y las tareas a realizar, la completa seguridad del entorno.

Los procesos definidos a lo largo de este trabajo son aplicables a momentos concretos dentro del ciclo de vida de los sistemas y las revisiones deben ser periódicas e incrementales. Es decir, no basta con una sola revisión.

Si bien, Scrum Secure, pretende añadir una capa de seguridad o, en otras palabras, establece un sistema de control seguro al proceso de desarrollo ágil, siempre debe ser aplicado el sentido común a la idiosincrasia de cada proyecto para aportar un verdadero valor.

En el apartado **¡Error! No se encuentra el origen de la referencia.**, se han podido reflejar varias vulnerabilidades y se ha expuesto el riesgo asociado a las aplicaciones analizadas. Independientemente del modelo de desarrollo utilizado para las aplicaciones, siempre existen y existirán riesgos. Y la solución de estas vulnerabilidades siguiendo las recomendaciones expuestas no será suficiente. La repetición de pruebas y alternancia de equipos auditores con independencia entre ellos es un requisito casi indispensable.

Para la cerrar completamente el ciclo de seguridad alrededor de las aplicaciones analizadas sería interesante ampliar en un futuro el alcance de éste estudio incorporando aspectos de la seguridad de los sistemas, seguridad perimetral, procesos de detección y reacción ante incidentes y de continuidad o restauración de los sistemas.

7. REFERENCIAS

Agile Manifesto. (Febrero de 2001). Obtenido de Manifiesto por el Desarrollo Ágil de Software: http://agilemanifesto.org/

Beck, K. (6 de 3 de 1996). *eXtreme Programming.* Obtenido de eXtreme Programming: http://www.extremeprogramming.org/

Cohn, M. (6 de Abril de 2007). *Differences Between Scrum and Extreme Programming.* Obtenido de Mountain Goat Software: https://www.mountaingoatsoftware.com/blog/differences-between-scrum-and-extreme-programming

OWASP. (2013). *Owasp Top Ten.* Obtenido de OWASP: https://www.owasp.org/images/5/5f/OWASP_Top_10_-_2013_Final_-_Espa%C3%B1ol.pdf

Principles behind the Agile Manifesto. (Febrero de 2001). Obtenido de Agile Manifesto: http://agilemanifesto.org/principles.html

scrummanager.net. (23 de Marzo de 2014). *Historia de usuario.* Obtenido de http://www.scrummanager.net/: http://www.scrummanager.net/bok/index.php?title=Historia_de_usuario

A. ANEXO

A.1. MARCO DE CONTROL SECURE SCRUM

Gonzalo Vigo Scheiner y la Universitat de Barcelona han desarrollado un marco de control de seguridad aplicable al modelo de desarrollo ágil SCRUM.

La adopción de SecuRe Scrum es una decisión estrátegica de las organizaciones dedicadas al desarrollo de aplicaciones aplicando SCRUM.

El establecimiento y la implementación de SecuRe Scrum y los procesos derivados de su apliación está influenciada por las necesidades de la organización y el equipo de desarrollo y sus objetivos.

SecuRe Scrum pretende preservar la confidencialidad, integridad y disponibilidad de la aplicaciones desarrolladas y la información contenida en ella mediante la introducción de procesos operativos y tácticos para la gestión del riesgo del sistema.

Es importante integrar SecuRe Scrum con los procesos de desarrollo de la organización y su estructura.

Los controles de SecuRe Scrum, pueden ser utilizados también, por equipos internos o externos a la organización que los integra para analizar la madurez del ciclo de vida de desarrollo en materia de seguridad.

A.1.1. Controles y objetivos de control

C.1 Equipos y personas		
Objetivo: asegurar que los miembros del equipo y demás actores del proyecto comprendan sus responsabilidades y estén capacitados para desarrollar su rol.		
C.1.1	Capacitación	Control Se debe llevar a cabo una verificación sobre los perfiles de los empleados miembros del equipo con el objetivo de verificar la adecuación de sus capacidades y experiencia para cumplir con los objetivos del proyecto en materia de seguridad.
C.1.2	Formación	Control

		El responsable de seguridad del proyecto debe proveer al equipo de desarrollo el acceso a la formación necesaria sobre seguridad que permita desarrollar el proyecto de una forma segura.
C.1.3	Concienciación	Control El responsable de seguridad del proyecto debe desarrollar un plan de concienciación enfocado al cliente e integrantes del equipo mediante el cual se transmitan los riesgos asociados al desarrollo y el impacto de estos.

C.2 Diseño del proyecto		
Objetivo: definir aspectos de seguridad básicos dentro de la fase de diseño que permitan integrar la totalidad de controles de SecuRe Scrum en el ciclo de vida del proyecto		
C.2.1	Criterios de validación de historias	Control Las historias de usuario definidas al inicio del proyecto deben contener criterios de validación de seguridad que permitan medir la madurez de su aplicación una vez finalizado el respectivo sprint.
C.2.2	Casos de abuso	Control El responsable de seguridad del proyecto junto con el equipo de seguridad debe identificar los casos de abusos asociados a las historias de usuarios y emitir una serie de recomendaciones de seguridad al equipo de desarrollo a fin de mitigar los riesgos.
C.2.3	Análisis de riesgos	Control Se debe llevar a cabo un análisis de riesgos que permita identificar los puntos débiles de la aplicación y emitir al equipo de desarrollo una serie de recomendaciones de

		seguridad que permitan mitigar los riesgos residuales.

C.3 Desarrollo seguro		
Objetivo: asegurar que durante el desarrollo de los diferentes *sprints* del proyecto se programe de forma segura.		
C.3.1	Buenas prácticas de codificación	Control Los equipos de desarrollo deben programar en función a buenas prácticas de seguridad para los diferentes lenguajes de programación utilizados.
C.3.2	Pruebas de seguridad	Control Antes de finalizar el Sprint los "Security Tester" deben validar que el código fuente es seguro contrastando el programa con los criterios de validación de seguridad definidos en las historias de usuarios.

C.4 Pentest		
Objetivo: realizar un test de seguridad final que permita certificar que el entregable del proyecto es seguro		
C.4.1	Auditoría de seguridad	Control Se debe llevar a cabo una prueba de seguridad, realizada por un equipo independiente que permita identificar posibles vulnerabilidades no contempladas por los miembros del equipo..
C.4.2	Mitigación de vulnerabilidades	Las vulnerabilidades identificadas por el equipo de seguridad independiente en el C.4.1 deben ser mitigadas o su riesgo debe ser asumido por el cliente.