

¿QUÉ SON LAS VULNERABILIDADES DEL SOFTWARE?

Las vulnerabilidades son la piedra angular de la seguridad, puesto que suponen el origen del que derivan numerosos fallos de seguridad. Una vulnerabilidad en un programa informático o software es simplemente un error, un problema en su código o en su configuración. Es muy probable –por no decir que se produce siempre– que los programas contengan errores, puesto que han sido creados por seres humanos. Esto es especialmente frecuente en el caso de las aplicaciones muy complejas (como por ejemplo, un sistema operativo), que tienden a contener errores de manera exponencial. La peculiaridad que convierte un simple fallo en una vulnerabilidad es la posibilidad de que el "abuso inteligente" de este defecto pudiera llevar a un riesgo de seguridad que compromete todo el sistema sobre el que se ejecuta esa aplicación.

En los siguientes epígrafes se analizarán los aspectos básicos de las vulnerabilidades: por qué ocurren y cómo gestionarlas.

I **Qué es una vulnerabilidad**

Una vulnerabilidad es un fallo en un programa o sistema informático. Pero no cualquiera, sino un fallo de seguridad. Es necesaria esta distinción puesto que no todos los errores de programación derivan en fallos de seguridad. Un error en un programa puede llevar a que no funcione correctamente o que su comportamiento no sea el esperado, pero no todos estos tipos de problemas pueden considerarse fallos de seguridad. Según la capacidad de aprovecharse de este defecto, la vulnerabilidad será más o menos grave.

Una vulnerabilidad se define, básicamente, por cinco factores o parámetros que deben identificarla.

1. **Producto**

Para definir una vulnerabilidad, lo primero que es necesario conocer es a qué productos afecta. Dentro de un mismo programa, incluso, puede afectar a una sola versión, a toda una rama o incluso a programas totalmente diferentes que compartan un mismo fallo. Este último supuesto ocurre cuando la aplicación afectada por la vulnerabilidad reside en sistemas operativos diferentes. Por ejemplo, todas las distribuciones de Linux comparten una buena cantidad de software, entre ellos, el propio núcleo del sistema. Una sola vulnerabilidad en el *kernel* puede afectar a todas las distribuciones que lo utilicen, desde *Debian* a *Mandriva* pasando por *OpenSuSE*.

Ilustración 1: Alerta de Microsoft Windows donde se especifica producto, impacto y gravedad de las vulnerabilidades

Affected Software

Operating System	Maximum Security Impact	Aggregate Severity Rating
Windows XP Service Pack 3	Elevation of Privilege	Important
Windows XP Professional x64 Edition Service Pack 2	Elevation of Privilege	Important
Windows Server 2003 Service Pack 2	Elevation of Privilege	Important
Windows Server 2003 x64 Edition Service Pack 2	Elevation of Privilege	Important
Windows Server 2003 with SP2 for Itanium-based Systems	Elevation of Privilege	Important
Windows Vista Service Pack 1 and Windows Vista Service Pack 2	Elevation of Privilege	Important

Fuente: INTECO

Para identificar correctamente una vulnerabilidad debe especificarse a qué productos o versiones de productos afecta concretamente.

2. Dónde

Dentro de un mismo programa, una vulnerabilidad se localiza habitualmente en un componente o módulo. Los programas suelen componerse de varios módulos que interactúan entre sí. Una vulnerabilidad puede encontrarse en un módulo concreto del programa o bien por utilizar una configuración concreta. Por ejemplo, puede existir una vulnerabilidad en el módulo de interpretación de ficheros en formato RTF en *Microsoft Word* sin afectar al módulo que procesa otro tipo de ficheros. O en el módulo de procesamiento de ficheros MP3 en el programa de reproducción *Winamp*. Es posible que la vulnerabilidad no pueda ser aprovechada si este módulo no se encuentra activo. Por ejemplo, el módulo de procesamiento de *JavaScript* en documentos PDF no se encuentra activo por defecto en *Adobe Reader*.

Si, por el contrario, el fallo se encuentra en un componente intrínseco al programa, no existe posibilidad de deshabilitar componentes. Esto puede ocurrir por ejemplo si se encuentra un fallo en el explorador de *Windows* o en su propio núcleo.

3. Causa y consecuencia

¿Cuál es el origen del problema? ¿En qué falló su programador? Esto se refiere al fallo técnico concreto que cometió el programador a la hora de crear la aplicación que es el origen de la vulnerabilidad. Por ejemplo, puede que no comprobarse bien qué valores alojaba una variable, los límites de la memoria, o que olvidara establecer unos permisos adecuados a unos ficheros.

Las consecuencias técnicas de estos fallos suelen ser diferentes. Desde el desbordamiento de memoria hasta el consumo excesivo de memoria. Estas consecuencias suelen ser siempre las mismas derivadas de los mismos descuidos, y es lo que buscarán los cazadores de vulnerabilidades.

Un ejemplo claro es, que una falta de comprobación de caracteres en una aplicación web (causa), lleve a una posible inyección SQL (consecuencia). Otro ejemplo: Una falta de comprobación de límites en una variable (causa), puede llevar a un desbordamiento de memoria intermedia (consecuencia). Un último ejemplo: Un descuido a la hora de establecer los permisos de un servicio en Windows (causa) puede llevar a un salto de restricciones (consecuencia).

Normalmente la causa de una vulnerabilidad es un fallo técnico de programación, una falta de comprobación que permite que se den circunstancias indeseadas en el código durante su ejecución.

4. Impacto

El impacto es lo que puede conseguir un atacante que aprovechase la vulnerabilidad. Por ejemplo, si existe un desbordamiento de memoria intermedia, es posible que el atacante pueda conseguir ejecutar código. Si la consecuencia de la vulnerabilidad es que el programa comienza a consumir recursos, es posible que el atacante pueda llegar a conseguir una denegación de servicio (hacer que el programa deje de responder). Si no se han comprobado bien los permisos del programa (causa) puede que se produzca un salto de restricciones (consecuencia) y que el atacante consiga elevar privilegios en el sistema (impacto).

El impacto define en gran medida la gravedad de la vulnerabilidad. La ejecución de código arbitrario supone la mayor gravedad puesto que significa que el atacante podrá ejecutar cualquier programa en el sistema de su víctima. Por tanto, podría realizar cualquier acción. En estos casos, se dice que el sistema queda "comprometido" porque ha quedado en manos de la voluntad de un tercero.

5. Vector

A la forma que tiene el atacante de aprovechar la vulnerabilidad se le conoce como "vector de ataque". Un vector de ataque común es el envío de información especialmente manipulada a un puerto concreto del sistema. Otra forma de conseguir aprovechar una vulnerabilidad es creando un fichero manipulado que será procesado por ese programa. Por ejemplo, si se encuentra una vulnerabilidad en *Word*, es muy probable que el vector de ataque sea un archivo en formato *.doc* que aproveche la vulnerabilidad. Si la víctima lo procesa con un *Word* vulnerable, el atacante conseguirá el impacto deseado.

Otros vectores de ataque pueden ser muy sencillos de llevar a cabo: como hacer que la víctima visite un enlace. Por ejemplo, muchas de las vulnerabilidades encontradas en los navegadores son aprovechadas por atacante creando una página web adulterada que, al ser visitada con el navegador vulnerable, aprovecha la vulnerabilidad. Por tanto, enviar un enlace a la potencial víctima, sería el vector de ataque en este caso, y el impacto, podría ser la ejecución de código.

Llegados este punto, pongamos un ejemplo concreto para unir todos estos parámetros que definen una vulnerabilidad.

La aplicación de diseño gráfico *MSPaint* necesita conocer el tamaño de la imagen que va a abrir antes de procesarla. Para conseguir esto, *mspaint.exe* se ayuda de fichero llamado *image_size.dll* que está especializado en esta función.

El programador de la aplicación *MSPaint 7* omitió la comprobación correcta de un parámetro que define la longitud que debe tener una imagen, en la función *SetImageSize* del fichero *image_size.dll*. Así pues, como no se comprueba que el valor del parámetro coincida con el tamaño real de la imagen, se produce un desbordamiento de memoria. Explicado de forma sencilla: la memoria del ordenador llega a sitios donde no debería. Por tanto, un atacante puede retocar la cabecera de una imagen para modificar su atributo de tamaño, añadir a esa imagen qué código quiere ejecutar y aprovecharse de la vulnerabilidad.

Vamos a extraer la información necesaria que define la vulnerabilidad:

- **Producto y versión:** *MSPaint 7* es la versión afectada.
- **Causa / Consecuencia:** Se omite la comprobación del parámetro tamaño (causa) y se produce un desbordamiento de memoria intermedia (consecuencia)
- **Dónde / Módulo:** En la función *SetImageSize* del componente *image_size.dll*. Es una librería en la que se apoya *mspaint.exe*.
- **Impacto:** Lo que podría conseguir un atacante es ejecutar código arbitrario. O sea, ejecutar cualquier programa que desee: malware, virus, etc.
- **Vector:** ¿Cómo puede un atacante explotarla? Un atacante debería enviar un archivo de imagen manipulado (con el código que quiere ejecutar incrustado en su interior, y el parámetro incorrecto retocado) a la víctima, y ésta abrirla con la versión de *MSPaint* vulnerable. En ese momento se ejecutaría el código arbitrario con el que se ha manipulado la imagen y la víctima quedaría comprometida.

II Gestión de vulnerabilidades

Las vulnerabilidades son difíciles de gestionar. Se descubren decenas día a día, y clasificarlas es una tarea compleja. Para ello, la organización MITRE (www.mitre.org) – organismo con más de 40 años de experiencia en el mundo de la investigación y las tecnologías que está especializado en la prestación de servicios al Gobierno de EEUU – creó un sistema, imprescindible hoy en día para estandarizar las vulnerabilidades, para poder hacer referenciarlas y conocer su gravedad de forma objetiva. Además, este permite responder, siempre que sea posible, a todas los parámetros que definen la vulnerabilidad.

CVE (Common Vulnerabilities and Exposures)

El CVE es un estándar (administrado por MITRE que se encarga de identificar unívocamente a las vulnerabilidades. Se puede decir que es DNI de una vulnerabilidad. Su formato es el siguiente:

CVE-2011-1234

CVE, seguido del año en el que se asignó el código a la vulnerabilidad, seguido de un número de cuatro cifras. Los grandes fabricantes normalmente toman lotes de CVE válidos pero no usados, que MITRE les adjudica. A medida que van encontrado vulnerabilidades, se los van asignando. Con los creadores de software más pequeños, el propio MITRE se encarga de dicha asignación a medida que se descubren vulnerabilidades.

El CVE ha tenido gran aceptación entre todos los fabricantes porque la mayor parte de las veces es muy complejo saber a qué vulnerabilidad nos estamos refiriendo solo por ciertas características. Es necesario disponer de una especie de número de identidad único para cada fallo, puesto que en ocasiones las vulnerabilidades son tan parecidas entre sí, tan complejas o se ha ofrecido tan poca información sobre ellas que la única forma de diferenciar las vulnerabilidades es por su CVE.

Ilustración 2: Ejemplo de descripción de vulnerabilidad y CVE en la página del Mitre. Esta vulnerabilidad era aprovechada por el conocido malware Stuxnet

CVE-ID	
CVE-2010-2568 (under review)	Learn more at National Vulnerability Database (NVD) • Severity Rating • Fix Information • Vulnerable Software Versions • SCAP Mappings
Description	
Windows Shell in Microsoft Windows XP SP3, Server 2003 SP2, Vista SP1 and SP2, Server 2008 SP2 and R2, and Windows 7 allows local users or remote attackers to execute arbitrary code via a crafted (1) .LNK or (2) .PIF shortcut file, which is not properly handled during icon display in Windows Explorer, as demonstrated in the wild in July 2010, and originally reported for malware that leverages CVE-2010-2772 in Siemens WinCC SCADA systems.	
References	

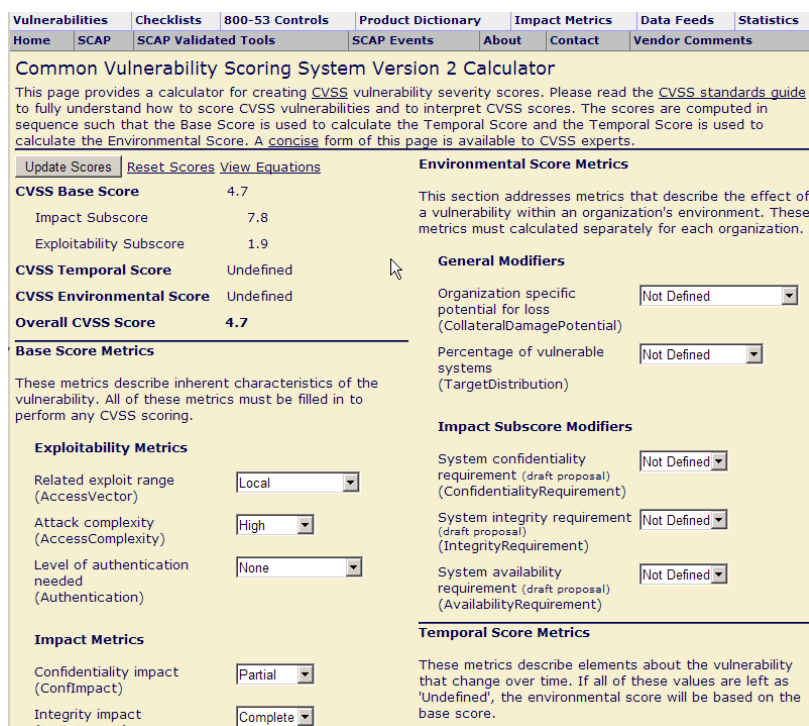
Fuente: INTECO

CVSS (Common Vulnerability Scoring System)

El CVSS es también un estándar que gradúa la severidad de manera estricta a través de fórmulas establecidas. De esta forma los administradores o usuarios pueden conocer de manera objetiva (a través de un número) la gravedad de los fallos que afectan a sus sistemas. Esto permite dar prioridad a la hora de parchear.

CVSS clasifica la facilidad de aprovechar el fallo y el impacto del problema teniendo en cuenta los tres pilares de la seguridad de la información: qué nivel de compromiso de la **confidencialidad, integridad y disponibilidad** de los datos se podrían obtener aprovechando la vulnerabilidad.

Ilustración 3: Calculadora online de CVSS



Fuente: INTECO

Para calcular la puntuación base de una vulnerabilidad se toman nueve parámetros, divididos en tres grupos de tres parámetros cada uno. Los tres grupos principales son: Explotabilidad, Impacto y Temporal.

- **Explotabilidad** define cuán complicado puede llegar a ser para un atacante aprovechar el fallo. Cuenta a su vez con tres parámetros, *Vector*, *Complejidad* y *nivel de Autenticación*. Cada uno a su vez con varios valores posibles.
- El grupo **Impacto** define qué grado de acceso a los datos podría obtener el atacante. Se define a través de tres valores a su vez, *confidencialidad*: (si el

atacante puede leer aquello que no debería). *Integridad* (si el atacante puede escribir o modificar aquello que no debería) y *Disponibilidad* (si el servicio o máquina puede seguir funcionando después de ser atacado).

- El grupo **Temporal** cuenta también con tres parámetros. Mide ciertas propiedades que influyen en cómo se percibe la vulnerabilidad a lo largo del tiempo y por tanto, son susceptibles de cambios. *Explotabilidad*: Indica si es sencillo aprovechar la vulnerabilidad. *Nivel de Remedio*: Si existe o no existe una solución y *Confianza de la información*: O sea, si la información sobre la vulnerabilidad es oficial o no.

Ilustración 4: Ejemplo de cómo la empresa de seguridad Cisco utiliza CVSS para valorar la gravedad de una vulnerabilidad encontrada en sus productos

The screenshot shows a web browser displaying a Cisco security advisory page. On the left, there is a search form with fields for 'Name' and 'E-mail', and a 'Submit' button. The main content area is titled 'Details' and 'Vulnerability Scoring Details'. It contains text explaining that Cisco provides scores based on the Common Vulnerability Scoring System (CVSS) version 2.0. A table is provided to calculate the environmental score for the vulnerability 'CSCsx47543 - AAA account-override-ignore allows VPN session without correct password'. The table shows a CVSS Base Score of 7.8 and a CVSS Temporal Score of 6.8.

CSCsx47543 - AAA account-override-ignore allows VPN session without correct password					
Calculate the environmental score of CSCsx47543					
CVSS Base Score - 7.8					
Access Vector	Access Complexity	Authentication	Confidentiality Impact	Integrity Impact	Availability Impact
Network	Low	None	Complete	None	None
CVSS Temporal Score - 6.8					

Fuente: INTECO

CVRF (Common Vulnerability Reporting Framework)

Finalmente, el CVRF se trata de un estándar reciente, que pretende dar uniformidad a la forma en la que se avisa de vulnerabilidades de software a un programador o compañía que crea un programa. Con este método se persigue que, cuando un investigador o empresa cree haber encontrado un fallo de seguridad en un programa, se le proporcione al fabricante la información precisa, rigurosa y adecuada para que pueda confirmarlo, entenderlo y sobre todo, parchearlo de forma eficaz.

Este estándar es muy reciente y su adopción es todavía limitada. Hoy en día, cada investigador tiene su propio criterio a la hora de reportar una vulnerabilidad, pero los datos que se proporcionan suelen ser los descritos más arriba: causa, consecuencia, impacto, módulo, etc.

III Parches y actualizaciones

Cuando se descubre una vulnerabilidad, lo más importante desde el punto de vista del usuario es saber cómo defenderse. Evitar una vulnerabilidad conocida puede pasar por varios estadios y requiere de un seguimiento constante:

- Conocer la vulnerabilidad: Es importante mantenerse informado en listas de seguridad sobre los nuevos fallos que van apareciendo. Existen listas públicas y gratuitas a las que los usuarios pueden suscribirse, como por ejemplo los boletines de INTECO, Hispasec, etc. En este aspecto, informarse cuanto antes es vital para reducir la ventana de tiempo de riesgo en la que se es vulnerable.

Ilustración 5: Descripción de vulnerabilidad del CERT de INTECO



The screenshot shows a web interface with a sidebar on the left and a main content area on the right. The sidebar contains a 'Destacados' section with four items: '¿Qué es un CERT?', 'Útiles Gratuitos', 'Menores Protegidos', and 'Catálogo de Seguridad'. The main content area displays a vulnerability report for 'Vulnerabilidad en controladores kernel-mode de win32k.sys de Microsoft Windows Vista SP1 y SP2, Windows Server 2008 Gold, Service Pack 2, R2 y R2 SP1, y Windows 7 Gold y SP1 (CVE-2011-1877)'. The report includes the following details:

- Tipo:** No Disponible/Otro tipo
- Gravedad:** Alta (indicated by five red bars)
- Fecha de publicación:** 13/07/2011 **Última modificación:** 14/07/2011
- Descripción:** Vulnerabilidad de uso después de liberación en los controladores kernel-mode de win32k.sys de Microsoft Windows Vista SP1 y SP2, Windows Server 2008 Gold, Service Pack 2, R2 y R2 SP1, y Windows 7 Gold y SP1 permite a usuarios locales conseguir privilegios a través de una aplicación diseñada que aprovecha el manejo incorrecto de objetos del controlador, también conocido como "Vulnerabilidad de uso después de la liberación en Win32k".

Fuente: INTECO

- Si no existe parche, aplicar contramedidas: La mayoría de vulnerabilidades se solucionan cuando el programador arregla el error en el programa y lo publica de nuevo. Esto se llama parche o actualización. Pero esto no siempre se consigue a tiempo. Si el fabricante no ha creado todavía un parche o actualización para solucionar el fallo es importante deshabilitar el módulo del programa vulnerable o bien dejarlo de usar hasta que exista parche. Se pueden utilizar programas alternativos que cumplan la misma función, mientras tanto.
- Si existe parche, aplicarlo: Si el fabricante crea un parche o una actualización del programa para solucionar el fallo, es importante instalarlo cuanto antes. En entornos críticos (servidores) es importante realizar pruebas previas, puesto que introducir un parche para arreglar un problema puede a su vez introducir otro fallo y volver inestable un servidor.

IV Cómo protegerse de las vulnerabilidades

Crear software invulnerable es imposible. Lo máximo a lo que se puede tender es a minimizar el riesgo de las vulnerabilidades que, sin lugar a dudas, aparecerán en cualquier software más o menos complejo.

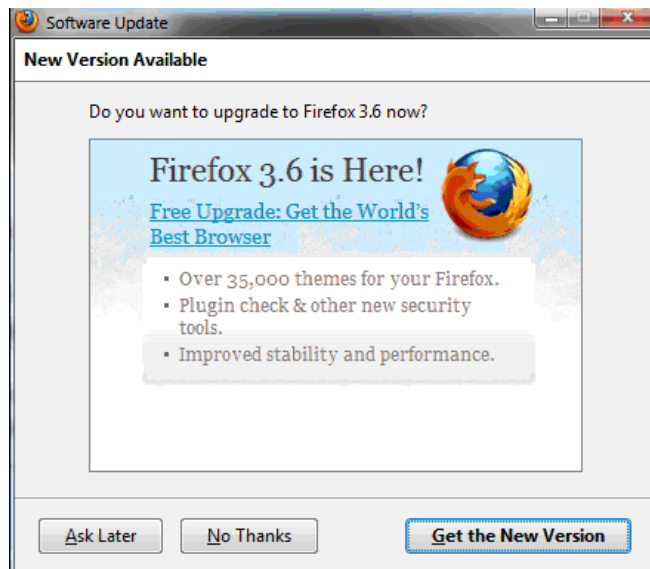
La lucha contra las vulnerabilidades debe comenzar desde el punto de vista del propio programador. El equipo que codifica el programa debe seguir unas normas básicas de seguridad a la hora de crear líneas de código que están documentadas en infinidad de manuales de buenas prácticas de seguridad. Por ejemplo, se debe comprobar siempre los límites antes de utilizar variables, procesar la línea de entrada del usuario para eliminar caracteres extraños, etc. Si el programador se centra en crear software seguro, eliminará de raíz una buena parte de las potenciales vulnerabilidades.

No todas las compañías pueden seguir esta regla puesto que una codificación segura implica realizar muchas más pruebas, invertir más tiempo y por tanto, dinero para compañías que pueden verse limitadas por un presupuesto limitado, o incluso un límite de fecha para entregar el producto.

Desde el punto de vista del usuario, para protegerse de las vulnerabilidades lo más efectivo es estar atento a las actualizaciones de software que debe publicar el propio creador del programa.

Muchas aplicaciones vienen ya de serie con módulos destinados automáticamente a comprobar si se dispone de la última versión de la aplicación y alertar al usuario en caso contrario. Sin embargo, no es el caso de la mayoría de programas y por tanto, el usuario debe estar atento para conocer las últimas actualizaciones que pueden corregir problemas de seguridad visitando regularmente la página del producto, o pendiente de las alertas de seguridad sobre ese producto que pueden aparecer en listas y noticias sobre seguridad.

Ilustración 6: Ejemplo de cómo Mozilla Firefox advierte a los usuarios sobre la disponibilidad de una nueva versión



Fuente: download.cnet.com

Otra medida de prevención que puede seguir el usuario es la regla de la mínima exposición, esto es: cuantos menos módulos se utilicen de un programa, aplicación o sistema operativo, menos expuesto se estará a potenciales vulnerabilidades que puedan ocurrir. Por tanto, una regla básica de seguridad es desactivar todo lo que no se utilice.

V INTECO y la gestión de vulnerabilidades

Con el objetivo de informar y advertir sobre las vulnerabilidades en los sistemas informáticos, desde INTECO se pone a disposición de los usuarios, una base de datos con información sobre cada una de las vulnerabilidades informáticas públicamente conocidas.

Este repositorio, que gratuitamente ofrece INTECO-CERT a través en su sitio web (<http://cert.inteco.es>), actualmente contiene más de 40.000 registros traducidos al español y se basa en la metabase NVD (*National Vulnerability Database*) – <http://nvd.nist.gov> – a disposición de INTECO fruto de un acuerdo de colaboración con el *National Institute of Standards and Technology* (NIST), agencia perteneciente Gobierno estadounidense.

La mayor parte de la información se extrae de fuentes de prestigio en el campo de la seguridad en Tecnologías de la Información: recomendaciones del CERT (www.cert.org), ISS X-Force (www.iss.net), Security Focus (www.securityfocus.com), y boletines de seguridad de fabricantes como Microsoft (www.microsoft.com/security/default.msp). En ocasiones el listado mostrará vulnerabilidades que aún no han sido traducidas al español

debido a que aparecen en el transcurso de tiempo en el que el equipo de INTECO-CERT realiza el proceso.

INTECO emplea el estándar de nomenclatura de vulnerabilidades CVE con el fin de facilitar el intercambio de información entre diferentes bases de datos y herramientas.

Cada una de las vulnerabilidades enlaza a diversas fuentes de información así como a distintos parches o soluciones. Es posible realizar búsquedas avanzadas teniendo la opción de seleccionar diferentes criterios como el tipo de vulnerabilidad, el fabricante y tipo de impacto, entre otros, con el fin de acotar los resultados.

Ilustración 7: Buscador avanzado de vulnerabilidades de INTECO-CERT



The screenshot displays the 'Buscador de vulnerabilidades' (Vulnerability Searcher) interface. It features a search bar at the top right with a 'Buscar' button and a 'Buscador avanzado' link. Below the search bar are links for 'NUEVO USUARIO' and 'USUARIO REGISTRADO'. The main content area includes a search form with fields for 'Texto', 'Fecha' (with 'Desde' and 'Hasta' sub-fields), and several dropdown menus for 'Fabricante', 'Producto', 'Gravedad', 'Rango de explotación', 'Tipo de pérdida', and 'Tipo de Vulnerabilidad'. Each dropdown menu has a 'Cargar' or 'Ayuda' button. To the right of the search form, there is a section titled 'Última Hora' and 'Últimas Vulnerabilidades Traducidas' which lists several CVE entries with their respective publication dates.

Fuente: INTECO

 <http://www.facebook.com/ObservaINTECO>

 <http://www.twitter.com/ObservaINTECO>

 <http://www.inteco.es/blog/Seguridad/Observatorio/BlogSeguridad/>

 <http://www.youtube.com/ObservaINTECO>

 <http://www.scribd.com/ObservaINTECO>

 <http://www.slideshare.net/ObservaINTECO>

 observatorio@inteco.es