Ciencias de la Computación
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS UNIVERSIDAD DE CHILE

La Ingeniería Inversa y su Rol en la Seguridad Informática

Dr. Alejandro Hevia
Departamento de Ciencias de la Computación,
Universidad de Chile

Marzo 2008

En este documento el autor presenta y discute la necesidad de permitir la ingeniería inversa con fines de investigación en seguridad informática en Chile. En particular, el autor explica el rol fundamental de la ingeniería inversa en el desarrollo de programas computacionales (software) confiables y de calidad. Actualmente, se encuentra en tramitación legislativa un proyecto de ley que actualiza la ley de propiedad intelectual en el cual la posibilidad de prohibir o limitar la ingeniería inversa es planteada.

Introducción

El propósito del proyecto de ley actualmente tramitado en el congreso que pretende actualizar la ley de propiedad intelectual es ciertamente loable y necesario en sociedades modernas. Proteger e incentivar la innovación a través de cautelar los derechos de los autores de obras y contenido cultural, y al mismo tiempo proveer de facilidades para la sociedad en su conjunto pueda acceder en forma no discriminatoria a material de conocimiento son dos objetivos importantes para nuestro país. En este contexto, la ley discute el tema de la *ingeniería inversa* la cual seria prohibida por ley excepto en determinados casos. Es este tema precisamente el tema de este documento: qué es y cúal es el rol de la ingeniería inversa en la investigación académica en general, y en el área de la seguridad del software en particular.

En un comienzo, la ley contemplaría al menos dos excepciones donde sí se permitiría el uso de la ingeniería inversa: para garantizar interoperabilidad y para fines de investigación incluyendo investigación en seguridad computacional o informática. Al parecer, también existirían iniciativas cuyo objetivo es eliminar y/o reducir estas previsiones. En este documento, el autor argumenta la necesidad de permitir ingeniería inversa con fines de investigación en seguridad computacional, en particular cuando dicha investigación es aplicada al desarrollo de software seguro.

Blanco Encalada 2120
Piso 3 y 4
Casilla Postal: 837-0459
Santiago · Chile
Tel.: (56-2) 689 27 36
(56-2) 978 43 62
Fax: (56-2) 689 55 31
www.dcc.uchile.cl
www.fcfm.uchile.cl

usar el software y, ciertamente, como efectivamente usaran el software.[1] Desarrollar software de calidad ("que funcione bien") es una empresa difícil aún para programadores, analistas, e ingenieros en computación avezados. Y una de las características de un software de calidad, es que sea seguro. ¿Qué significa esto? Un software es seguro si esencialmente satisface dos propiedades: (1) su operación fue diseñada para tener en cuenta la privacidad de los datos y la funcionalidad deseada por el cliente, y (2) en su diseño no hay errores, fallas que permitan a un tercero malicioso (ya sea el mismo cliente o un adversario externo), quebrantar la funcionalidad del software a través de hacer uso explícito de dichos errores. A un error de este último tipo se le denomina una vulnerabilidad.

Vulnerabilidades y Ataque computacionales. Explicaremos el concepto de vulnerabilidad con un ejemplo: supongamos que existe un software de contabilidad de una empresa financiera. Para operar correctamente, este software no sólo debe hacer los cálculos de manera correcta (sin equivocarse ni revelar montos ni números de cuenta a quien no esté autorizado para verlos) sino que debe carecer de errores de programación que afecten la correcta operación o seguridad del programa. Errores de este tipo se denominan vulnerabilidades. Una vulnerabilidad es un error del programa que puede ser utilizado como "vía de entrada" al sistema, permitiéndole a un usuario externo influir indebidamente en él. Por ejemplo, virus y gusanos típicamente utilizan vulnerabilidades del sistema operativo del computador para "infectarlo" remotamente. El proceso de "infectar" o atacar un software ocurre cuando un hacker malicioso detecta o encuentra una vulnerabilidad en dicho software, e intenta "explotarla" (gatillarla) usando una cierta combinación de mensajes y acciones cuidadosamente pensadas para lograr que dicho error sea activado. Una vez logrado activar la vulnerabilidad en el escenario escogido, el hacker malicioso toma control del computador. Por ejemplo, un observador externo podría explotar una vulnerabilidad en el software de la empresa para obtener ganancias financieras indebidas.

Ilustrando el problema con un ejemplo: Supongamos por ejemplo, que el software financiero de nuestro caso anterior tuviera un error en el cual las personas cuyo nombre tuviera el carácter ' (comilla simple) automáticamente tuvieran crédito ilimitado para hacer transferencias de dinero. Por ejemplo, el nombre "Bernardo O'Higgins" podría gatillar tal situación.[2] Claramente este es un error garrafal, el cual si es detectado por un observador externo pudiera causar graves perdidas e incluso llevar a la quiebra a la institución financiera.

El rol de la empresa fabricante del software: Es importante destacar que una vulnerabilidad no aparece por una acción intencional de quienes diseñan el software. Mas

[1] A aquellos familiares con el tema, no les debiera sorprender el que "cómo se debiera usar" un software y "cómo se usa" frecuentemente no son iguales. Esta diferencia se encuentra en el corazón de esencialmente todos los problemas de seguridad de software.
[2] Aunque suene improbable, errores aún más sorprendentes que éste son lamentablemente comunes.

Blanco Encalada 2120
Piso 3 y 4
Casilla Postal: 837-0459
Santiago · Chile
Tel.: (56-2) 689 27 36
(56-2) 978 43 62
Fax: (56-2) 689 55 31
www.dcc.uchile.cl
www.fcfm.uchile.cl

a alguna institución o autoridad técnica competente (en EEUU es el US-CERT o *Computer Emergency Response Team*). Luego de hecho el reporte, la empresa fabricante del software dispone de un período acotado de tiempo (usualmente unos 3 meses) para estudiar la vulnerabilidad, corregirla (si así amerita) y publicitar la actualización o arreglo. Si la empresa fabricante no respondiese en dicho plazo, el profesional que encontró la vulnerabilidad debe reportarla a la comunidad, posiblemente dando suficiente información como para permitir la verificación de la existencia de la vulnerabilidad, pero no su explotación. Dar un plazo perentorio para el anuncio final es contra intuitivo, pero fundamental para proteger a los usuarios afectados. La idea es que ellos puedan conocer acerca de la vulnerabilidad, así como realizar las modificaciones y tomar los resguardos apropiados para prevenir ser atacados, *aún si la empresa fabricante se niega o es incapaz de solucionar la vulnerabilidad reportada*. De omitir este ultimo paso, nos arriesgamos a que criminales informáticos encuentren la vulnerabilidad de todas maneras y la exploten sin la posibilidad que los usuarios posiblemente afectados se puedan proteger.

¿Qué pasaría si se prohibiera "*analizar como funciona*" el software, esto es la ingeniería inversa? Cabe preguntarse si la vulnerabilidad habría sido encontrada si no se permitiese a los investigadores y/o profesionales de seguridad hacer ingeniería inversa y, por ende, examinar el contenido del software. La respuesta es sí, seguramente habría sido encontrada, pero no por quienes hubiéramos esperado. La vulnerabilidad no habría sido encontrada por los profesionales honestos, sino por criminales informáticos quienes no habrían comunicado la existencia del problema a la empresa, sino más bien, habrían explotado la vulnerabilidad con fines ilícitos. En este caso, los profesionales honestos no tienen la motivación ni el soporte legal como para detectar el problema antes que los criminales lo hagan.

Pero, si un investigador encuentra una vulnerabilidad, ¿no le está haciendo un favor a los criminales? Este es un malentendido muy común, y falso como argumento. Basta considerar la sofisticación y el conocimiento actual de los criminales informáticos, quienes pueden encontrar vulnerabilidades en el software en forma independiente, aún si los profesionales honestos no están buscándolas. Por supuesto, a un criminal no le importa si su acción implica cualquier acción potencialmente "ilegal" como potencialmente sería la ingeniería inversa. Un criminal informático que encuentra una vulnerabilidad con seguridad la utilizara para un ilícito, no así un profesional informático honesto, quien la reportará para que sea arreglada. Obviamente, es claramente mejor para la sociedad que las vulnerabilidades sean encontradas por profesionales honestos y responsables que por los criminales. Sólo en el primer caso la empresa de seguridad tendrá la oportunidad de arreglarlas y así proveer un mejor servicio a sus clientes.

Responsabilidad social de profesionales de la seguridad computacional. Por otro lado, cabe preguntarse si los profesionales e investigadores de seguridad debiéramos examinar

Blanco Encalada 2120
Piso 3 y 4
Casilla Postal: 837-0459
Santiago · Chile
Tel.: (56-2) 689 27 36
(56-2) 978 43 62
Fax: (56-2) 689 55 31
www.dcc.uchile.cl
www.fcfm.uchile.cl

"*Secure Digital Music Initiative*" (SDMI) solicitó estudiar y atacar su sistema de protección de "sellos de agua" digitales. Aunque el grupo del prof. Felten efectivamente quebró el sistema de protección de SDMI, no pudo presentar sus resultados en una conferencia académica por amenazas de demandas legales por parte de SDMI. (Referencia [2])

(b) En el año 2005, J.A. Halderman, un estudiante de postgrado de Stanford University, en EE.UU. encontró varias vulnerabilidades en el sistema de protección de copia usado por la empresa Sony-BMG en varios de sus CDs de música. El sistema de protección de copia de Sony-BMG tenía un error de diseño dejaba el computador susceptible de ser atacado en forma externa. J.A. Halderman decidió retrasar la publicación de sus resultados por temor a ser perseguido en base a la ley DMCA, poniendo en riesgo a millones de usuarios del software por varias semanas. (Referencia [3])

(c) Otros ejemplos, son las amenazas de SunnComm contra J.A.Halderman en 2003 (por mostrar como desactivar un sistema de protección de copia por medio de *sólo presionar la tecla "mayúsculas"*), las amenazas de Hewlett Packard contra SNOsoft en 2002 (por identificar vulnerabilidades en el Sistema Operativo HP Tru64 de Hewlett Packard), o las amenazas de Blackboard Inc. contra Billy Hoffman y Virgil Griffith por identificar vulnerabilidades en el software Blackboard ID. (Referencias [4,5,6])

Un compendio de casos es descrito en [1].

Censura y seguridad. Intuitivamente, una ley que limite el contenido o características del informe de seguridad que un investigador puede realizar sobre un cierto software puede ser abusada para efectivamente censurar ciertas "malas noticias", para que nunca sen publicadas. El problema es que dichas noticias pueden ser "malas" sólo para una de las partes interesadas pero no para el resto de la comunidad. Por ejemplo, un fabricante de software pudiera abusar de tal legislación para impedir la publicación en una conferencia académica de una vulnerabilidad de su producto, o una empresa que desarrolla productos de control de acceso (ej. cortafuegos) de mala calidad puede demandar a un profesional por "hackear" su programa cuando lo que el profesional hizo fue reportar una vulnerabilidad del cortafuegos. En tales casos, la sociedad sufre significativamente. Al no poder reportar una vulnerabilidad el profesional honesto se ve obligado a dejar a toda la población de usuarios afectados totalmente indefensos ante ataques usando dicha vulnerabilidad.

Excusa para no resolver problemas de seguridad debido a los costos asociados. Asimismo, prohibiciones o limitaciones de ingeniería inversa pueden dar una excusa legal a empresas que no les importe maximizar sus beneficios económicos a costa de la seguridad de sus clientes. Por ejemplo, una empresa pudiera no querer encontrar ni arreglar posibles errores y vulnerabilidades en sus productos para no tener que solventar el costo (horas hombre, reputación) asociado. Aunque tal comportamiento es poco ético, es imposible de

Blanco Encalada 2120
Piso 3 y 4
Casilla Postal: 837-0459
Santiago · Chile
Tel.: (56-2) 689 27 36
(56-2) 978 43 62
Fax: (56-2) 689 55 31
www.dcc.uchile.cl
www.fcfm.uchile.cl

Resumen

La posibilidad de realizar ingeniería inversa con fines de seguridad informática es crucial para el surgimiento y fortalecimiento de una industria de desarrollo de software de calidad. Esto porque el proceso de examinar un programa computacional usando ingeniería inversa esta intrínsecamente relacionado con el proceso de búsqueda y reporte de vulnerabilidades de software. En consecuencia prohibir o limitar la posibilidad de realizar ingeniería inversa en programas computacionales efectivamente puede causar un perjuicio significativo a quienes utilicen programas computacionales posiblemente defectuosos o, peor aun, constituirse en un mecanismo legal donde empresas inescrupulosas puedan encubrir y prolongar el consumo de productos "nocivos" para la sociedad a cambio de beneficios económicos.

Referencias

[1] "Unintended Consecuences: Seven Years under the DMCA", Electronic Frontier Foundation, disponible en http://www.eff.org. Version 4.0, Abril 2006.

[2] "Anticurcumvention Rules: Threat to Science", Pamela Samuelson, 293 Revista Science 2028, Sept. 14, 2001.

[3] Comments of Edward Felten and J.Alex Halderman, RM 2005-11, Exception to Prohibition on Circumvention of Copyright Protection Systems for Access Control Technologies, 1 de Diciembre, 2005, pags. 6-7, http://www.freedom-to-thinker.com/doc/2005/dmcacomment.pdf

[4] "Student faces suit over keys to CD locks", John Borland, CNET News, 9 de Oct. 2003.

[5] "Security Warnings Draws DMCA Threat", Declan McCullagh, CNET News, 30 de Julio, 2002.

[6] "Court Blocks Security Conference Talk", CNET News, 14 de Abril 2003.

Blanco Encalada 2120
Piso 3 y 4
Casilla Postal: 837-0459
Santiago · Chile
Tel.: (56-2) 689 27 36
(56-2) 978 43 62
Fax: (56-2) 689 55 31
www.dcc.uchile.cl
www.fcfm.uchile.cl