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

Para entender la necesidad de la ingeniería inversa en el contexto del la seguridad de software debemos primero interiorizarnos de otros conceptos, tales como qué es la ingeniería inversa y cómo funciona el proceso del desarrollo de software seguro. En particular, a continuación discutiremos qué es la ingeniería inversa de software, qué es la seguridad computacional, que significa que el software sea seguro, y finalmente cómo la ingeniería inversa juega un rol crucial para producir software seguro. Por razones de claridad, la siguiente exposición es mantenida a un nivel técnico más bien informal.

¿Que es ingeniería inversa?

El término se refiere al proceso de analizar la estructura, funcionamiento, características y, en general, los fundamentos técnicos de un sistema o dispositivo ya sea mecánico o electrónico, e incluso un programa computacional (software). La actividad de ingeniería inversa usualmente requiere examinar, desarmar y analizar los componentes del dispositivo usualmente para crear otro dispositivo que pueda ya sea replicar o mejorar su funcionamiento, o bien que pueda interactuar ("conversar") con el dispositivo analizado. En el caso de un programa de software, este proceso usualmente involucra examinar el programa *compilado* (o ejecutable) e incluso ejecutarlo bajo condiciones controladas. También puede incluir un análisis del código fuente, esto es, el programa correspondiente escrito en un lenguaje de programación de alto nivel. Intuitivamente, la ingeniería inversa es una técnica que – cuando es exitosa -- permite a un investigador o profesional determinar "cómo" opera un programa y sus características.

Cabe mencionar que realizar ingeniería inversa a un programa computacional no es necesariamente una tarea difícil. Por ejemplo, examinar cómo un programa responde a datos incorrectos (de mayor largo que el esperado, o con caracteres inesperados, o en momentos inesperados, por ej.) efectivamente califica como ingeniería inversa. Asimismo, cambiar una o dos bytes estratégicamente ubicados en el programa computacional para "experimentar" con su comportamiento es también una técnica usual de ingeniería inversa.

Ingeniería inversa con fines de Investigación en seguridad

Para entender como la ingeniería inversa puede ser utilizada durante el proceso de desarrollo de software, primero revisaremos como funciona el proceso de crear o desarrollar programas computacionales.

El proceso de desarrollo de software. Desarrollar software es un proceso difícil, el cual requiere conocimientos específicos y la capacidad de entender con precisión las necesidades particulares de los potenciales usuarios (clientes), esto es, cómo les gustaría

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

bien, es simplemente producto de un proceso de mala calidad. Este error termina siendo "explotable" por un usuario externo sin la colaboración ni participación de la empresa financiera, en perjuicio evidente para la empresa.

En principio, pareciera suficiente obligar a las empresas de software a "hacer mejor software" y para que el problema desaparezca. Esto es, obligar a las empresas a disponer de un proceso exigente de verificación de testeo y aseguramiento de calidad del software. Aunque tal exigencia efectivamente ayudaría a disminuir el número de vulnerabilidades existentes en productos del mercado, lamentablemente no sería una solución definitiva. La historia ha mostrado que el problema de diseñar software seguro no es fácil ni simple en la práctica. Es incluso parte de la percepción pública que desarrollar software resistente a intentos maliciosos por "atacarlo" es extremadamente difícil, aún para las empresas más dedicadas y con altos recursos. Todos los fabricantes de software sin excepción, desde las grandes empresas como Microsoft, Apple, y Google, hasta proyectos colaborativos de escala mundial (ej. Linux), han tenido problemas de seguridad. Ciertas empresas han batallado contra el desprestigio de los problemas de seguridad por años, aún a costa de invertir grandes recursos, sin por ello prevenir totalmente el problema de las vulnerabilidades.

Detección de una vulnerabilidad. En la práctica, la principal sino la única herramienta contra las vulnerabilidades de software han sido la experiencia de profesionales honestos y avezados, quienes a través de examinar los contenidos del programa - esto es, haciendo ingeniería inversa - logran encontrar una vulnerabilidad. Luego de encontrarla, estos profesionales comunican su existencia responsablemente al fabricante, quien corrige el error, protegiendo así a sus clientes. Estos profesionales frecuentemente no son parte del equipo de desarrolladores del software y su accionar es la gran mayoría de las veces desinteresado. En nuestro ejemplo, un profesional honesto y de sólidos conocimientos técnicos descubre el problema por accidente (al escribir su nombre, por ejemplo, digamos "Bernardo O'Higgins") y rápidamente se lo comunica a la empresa que desarrollo el software financiero, quienes lo corrigen y comunican a sus clientes previniendo futuros ataques. Posiblemente, el Sr. O'Higgins luego es contratado por esta y otras empresas de software para mejorar la seguridad de sus productos.

La importancia de los plazos para reportar, corregir y anunciar una vulnerabilidad. El proceso de encontrar, reportar, arreglar y anunciar una vulnerabilidad debe ser cuidadosamente llevado a cabo, a fin de evitar abusos y engaños a los posibles afectados. A lo largo de los años, la comunidad de empresas y profesionales de seguridad computacional han desarrollado diversas políticas de cómo llevar este proceso a cabo, siendo la más globalmente aceptada la conocida como *notificación responsable* (o *responsible disclosure*, en Inglés). En la política de notificación responsable, quien descubre la vulnerabilidad debe reportarla en forma privada lo antes posible al fabricante del producto así como (idealmente)

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

software del cual no participamos en su creación y, donde sus desarrolladores y clientes no han expresado ningún interés que lo examinemos. Quizás deberíamos contestar esta pregunta con una analogía basada en otros ámbitos similares de nuestra sociedad, como son la industria farmacéutica y automotriz. ¿Qué haría un químico profesional, por ejemplo, si detectase que un cierto jarabe para la tos de niños actualmente en venta contiene un elemento nocivo para la salud? O el ingeniero automotriz que detecta que la combinación de bencina, aditivos y algunos residuos del aire de Santiago (por dar un ejemplo ficticio) puede causar una combustión espontánea de un cierto modelo de auto? En ambos casos, uno esperaría que dicho profesional diera aviso al fabricante del producto y/o a las autoridades competentes, aún si nadie le ha pedido hacerlo. Más aún, uno esperaría que la legislación *no le impidiera* a dicho profesional experimentar ni ganar conocimientos en forma responsable en su área. Penalizar el uso de ingeniería inversa en el contexto de seguridad computacional equivale a prohibirle a un químico profesional examinar los compuestos químicos que contiene un cierto medicamento del mercado, o prohibirle a un ingeniero automotriz estudiar la operación de un cierto automóvil vendido en el país. Cualquier autoridad que valore la necesidad de una fiscalización independiente no debiera considerar tales prohibiciones como adecuadas.

Consecuencias de prohibir o limitar la ingeniería inversa de programas computacionales

Limitaciones en la investigación en seguridad informática y en la libertad de expresión: Ilegalizar la ingeniería inversa puede tener un efecto disuasivo y de autocensura para investigadores y profesionales del área de seguridad, quienes por su trabajo deben examinar programas y detectar errores o vulnerabilidades. Tales actividades por definición caerían en el concepto de ingeniería inversa. Como se ha argumentado anteriormente, prohibir o limitar tal tipo de investigación solo traería beneficios para los criminales, quienes obviamente no se atendrían a la ley y efectivamente estudiarían y explotarían el software, con el consiguiente perjuicio para los usuarios del software explotado.

Ejemplos de este problema hay lamentablemente muchos. Aunque en EE.UU. la legislación en la práctica es mucho menos restrictiva que lo que sería una propuesta donde la ingeniería inversa fuera prohibida o limitada, ha causado una multitud de problemas y cuestionamientos. La ley en cuestión se denomina *Digital Millennium Copyright Act*, o DMCA (ver más abajo por detalles). Dicha legislación ya ha sido utilizada para limitar la investigación en seguridad, y aún más, el derecho a la libertad de expresión. Casos emblemáticos son, entre otros,

(a) En Sept. del 2001, el profesor Edward Felten junto a un grupo de investigadores y estudiantes decidieron participar en un concurso donde la asociación detrás de la

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

dejar en evidencia si quienes pudieran detectar las vulnerabilidades no están autorizados a examinar el software! Los únicos perjudicados son en definitiva los usuarios del software quienes nunca son advertidos del problema de seguridad que conlleva usar dicho software.

Datos versus programa computacional. Para entender la magnitud de la consecuencia recién descrita, es importante destacar que la diferencia entre un programa computacional (o software) y un *archivo* de datos es muy cada vez menos significativa. En un archivo de datos puede almacenarse datos que determinan de manera crítica el funcionamiento de un programa y por ende, observar dicho archivo es en efecto una forma de ingeniería inversa! Ejemplos de ello son documentos con “macros” incluidas, por ejemplo. Con el advenimiento de código móvil (por ejemplo, código usado en la mayoría de las aplicaciones web) es prácticamente imposible delimitar en forma precisa donde termina el programa en ejecución y donde comienzan los datos.

Antecedentes en otros países:

Europa. En la Unión Europea la ingeniería inversa es un proceso legal y legítimo tanto en el contexto de permitir interoperabilidad e innovación, así como en el contexto de investigar y garantizar la seguridad computacional.

EEUU. En los Estados Unidos, el proceso de ingeniería inversa es considerado legal y legitimo en varios contextos (interoperabilidad y fomento de la innovación). Sin embargo, y pese a que practicar la ingeniería inversa no es ilegal, existen lamentables limitaciones en el área de la investigación en seguridad computacional. En efecto, el año 1998 se dictó la ley denominada “Digital Millenium Copyright Act” (DMCA), la cual penaliza la acción de *evadir* cualquier mecanismo de control de acceso puesto por el dueño del software o material protegido, así como la distribución de herramientas y tecnologías útiles para dicha acción. Aunque las prohibiciones de la ley DMCA son mucho menos demandantes que una posible prohibición de la ingeniería inversa, aún así la efectividad y adecuación de ley DMCA ha sido altamente cuestionadas exactamente por las razones aquí ya expuestas.

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