



Igualdad, equivalencia y plagios en software

A finales del pasado mes de enero se hizo público un comunicado oficial¹ de Steven Edwards que ha puesto en primera línea informativa a una curiosa iniciativa llamada ReactOS². El motivo de esta intervención pública es poner freno a ciertos comentarios que acusan, más o menos veladamente, a este proyecto de estar utilizando código fuente que no es suyo y que circuló por Internet tras la última "fuga" de información reconocida por Microsoft. Los argumentos principales que se esgrimen para levantar la sospecha sobre los desarrolladores de ReactOS, es que algunos de ellos pudieran no haber seguido al pie de la letra el procedimiento de salas limpias o salas blancas a la hora de realizar sus desarrollos dentro del proyecto.

Recientemente, la iniciativa ReactOS ha sido puesta en tela de juicio y ha decidido pararlo todo y revisar concienzudamente lo que hasta ahora ha desarrollado. Detrás de todo esto está el posible uso de códigos que pueden ser de Microsoft y que han circulado ampliamente por la red, así como los conceptos de secreto industrial, derecho y propiedad sobre los productos software. Sin embargo, la pregunta importante es otra: ¿es posible detectar siempre la reutilización no autorizada de software? ¿se puede definir y detectar el plagio en productos software?

dimiento conocido como "ingeniería inversa de sala blanca".

Cuando se habla de este tipo de ingeniería inversa ("clean-room reverse engineering") lo que se quiere decir es que una persona descubre por ingeniería inversa la implementación de un componente del software y escribe una documentación detallada con lo que descubre y, más tarde, otra persona distinta, lee esa documentación e implementa un nuevo

saludables, este incidente pone en el tapete varios temas que son muy importantes, en tanto que atañen a toda la informática en su conjunto y que, además, resultan muy interesantes porque no son de fácil solución, aunque haya quienes así quieran crearlo. Algunos de los aspectos a considerar en esta historia son:

—¿Hasta qué punto queda "infectado" un programador por el simple hecho de haber llegado a ver, en algún momento, un código que no es suyo?

—En este caso, habría que ver cómo considerar la "infección" de algunos de los programadores si ésta se produce como consecuencia de un fallo en la custodia de dicha información por parte de su propietario, y ese código ha circulado y ha sido visto por un nutrido y populoso número de programadores, como es el caso de las "fugas" de los códigos de Windows NT y 2000.

—También habría que plantearse en términos más científicos y cuantitativos la validez de la hipótesis esencial que justifica la separación de funciones en el modelo norteamericano de ingeniería inversa de salas blancas. Habría que preguntarse lo diferentes o similares que son, y que pueden ser, dos códigos que implementan exactamente la misma funcionalidad.

patenta: un texto literal (las líneas concretas de código C, por ejemplo), un contenido semántico (las líneas de código y todas sus perifrasis), o la misma funcionalidad de ese software (lo que es en sí el objeto informático).

Antes de entrar a plantearnos cuestiones más profundas que van asociadas con el caso ReactOS, es conveniente llamar la atención sobre el riesgo que supone el concepto de "infección" que parece estar merodeando en todo este asunto. La idea de que se pueda perseguir a alguien porque pueda haber visto un código y que, con ello, haya sido capaz, o incluso no haya podido evitar aprenderlo e integrarlo en su modo de hacer las cosas, es muy peligrosa. Si alguien queda "marcado" por ver o por haber tenido acceso a los códigos de Microsoft y eso le retira de la profesión, esos códigos podrían utilizarse al estilo de una nueva cabeza de Medea, y los más interesados en que los códigos fuente circulasen por Internet, a modo de neurotóxico, serían los propios ejecutivos de la gran multinacional.

Si alguien, viendo o estudiando los códigos de Microsoft o de cualquier otro, lograra aprenderlos, memorizarlos, digerirlos o hacerlos suyos mediante cualquier otro tipo de asimilación mental, sería un hecho tan excepcional que saldría en los periódicos y, sin duda, lo contrataría la empresa burlada.

Otra cosa muy distinta es que se tenga acceso digital al código sustraído y que se utilice éste directamente como parte no declarada de otros desarrollos; en este caso estaríamos ante una apropiación indebida del esfuerzo de otros y eso

El problema con el que se enfrentan los promotores de ReactOS al decidir revisar lo que tienen escrito de código, es que no van a poder decir qué es suyo y qué no lo es.

En síntesis, ReactOS es un proyecto de software libre que persigue implementar, cooperativamente y "ab initio", un sistema operativo completo que sea totalmente compatible con Microsoft Windows XP, y alcanzar también la compatibilidad binaria completa con las aplicaciones y los controladores de dispositivo propios de los sistemas operativos Windows NT y 2000. Esta compatibilidad binaria sólo se puede conseguir si se dispone de un conocimiento muy íntimo de los códigos con los que se quiere ser compatible, y dado que los códigos fuentes de Windows son un secreto comercial, este objetivo de compatibilidad sólo se puede conseguir a través de un cierto grado de ingeniería inversa. Para poder hacer esto sin infringir un montón de leyes mercantiles y de derechos de autor, lo que se suele hacer es seguir el proce-

código que la ejecuta. El objetivo de este proceder es impedir que sea la misma persona la que analiza el código y la que lo regenera. Este modelo está basado en la hipótesis de que quien nunca vio el código original, nunca podría escribirlo "igual" por casualidad.

Para evitar habladurías, descritos y posibles consecuencias legales futuras, los promotores de la iniciativa ReactOS han decidido: 1) especificar el método de salas blancas para cualquier actividad de ingeniería inversa como requisito inviolable del proyecto, 2) auditar todo lo escrito hasta el momento, y reescribir todo código que no se haya desarrollado según la exigencia anterior, y 3) obligar a todos sus desarrolladores a someterse al procedimiento antes mencionado.

A pesar de estas medidas tan

La identidad o falta de identidad en el código ejecutable no es una medida válida para dirimir este tipo de conflictos.

—Por último, también habría que retomar la discusión de la patentabilidad del software en general y, en particular, qué es lo que realmente se

no está bien y, además, es delito. Sin embargo, en este punto del discurso se abre ante nosotros un problema más profundo: ¿Qué se puede consi-

¹ http://www.reactos.org/xhtml/en/news_page_14.html

² <http://www.reactos.org>

derar como plagio informático? ¿Se puede realmente evitar?

La Ingeniería del Software tiene como uno de sus objetivos estudiar cómo establecer reglas y procedimientos que permitan crear aplicaciones informáticas cada vez mejores. Lo que las reglas y las metodologías favorecen es la homogeneización y estandarización del código que se genera, y esto apunta hacia la completa despersonalización de los códigos informáticos; si por la Ingeniería del Software fuese, los resultados de la programación serían completamente independientes del programador que los ha creado; a pesar de ello, la escritura de software sigue estando muy pegada a la creatividad de cada uno de sus autores.

La programación informática, como cualquier otra actividad "literaria", tiene estilos, trucos y modismos que el programador va aprendiendo con el tiempo y con las posibilidades de conocerlos y/o estudiarlos que se tenga. Esta experiencia previa, aunque no siga metodología alguna y sea claramente heterodoxa, conforma un estilo de programación y éste lo es, en tanto que reduce el número de formas que realmente utiliza para programar cada cosa.

Por un lado van las metodologías y la ortodoxia, por otro los estilos de programación, pero juntos son los factores responsables de que, en términos de probabilidad, sean pocas las posibilidades distintas, razonables y equivalentes que hay para escribir un programa con una funcionalidad dada. Al reducirse el número de posibilidades, lo que se consigue es que no sea despreciable la probabilidad de que dos programadores independientes, que nunca se han visto y que jamás han colaborado juntos, terminen escribiendo segmentos de código muy similares y completamente equivalentes para una misma funcionalidad dada. La hipótesis básica de la ingeniería inversa "de salas limpias" no es tan cierta como podría parecer, y no resulta eficaz como defensa frente a acusaciones y sospechas de un plagio que muy bien podría producirse de forma espontánea y no detectada.

Junto con la tendencia marcada por la ortodoxia y el estilo, hay que tener en cuenta que los sistemas informáticos se construyen sobre códigos que son colecciones de comandos, de asignaciones y evaluaciones, de tomas de decisiones y saltos en el flujo de ejecución.

El catálogo de operaciones donde elegir es bastante reducido, si lo comparamos con otros lenguajes no informáticos, por lo que la riqueza de los lenguajes de programación es escasa. Esta limitación no sólo está en el lenguaje en sí, sino también en la propia arquitectura de la máquina encargada de ejecutar esas instrucciones que los programadores dan "a alto nivel", y que los compiladores, enlazadores y ensambladores, se encargan de traducir y optimizar para obtener un código a bajo nivel pegado a la arquitectura del chip que lo ha de ejecutar.

La identidad literal y extensa de sus códigos con los de Microsoft no puede ser el criterio para determinar si hay plagio o no, ya que es fácil evitar que se dé esa coincidencia, puesto que un plagiador real se habría encargado de generar una de las variantes, funcionalmente idénticas, del código plagiado.

Que un lenguaje no tenga mucho vocabulario no significa que no se puedan escribir con él y para un mismo código, numerosísimas versiones funcionalmente idénticas, pero literalmente diferentes. A cualquier nivel de descripción de un programa informático siempre hay, por ejemplo, secuencias de operaciones cuyo orden de ejecución no altera el resultado final. También puede haber otras secuencias que puedan agruparse y otras desagruparse, o bien ser sustituidas por otras distintas, sin que nada de esto cambie el resultado de ejecución. Cuáles de estas posibilidades se dan en un código ejecutable concreto bien podría ser una característica exclusiva de ese caso particular. De hecho, esta

Hacer reposar la propiedad sobre las funcionalidades es algo muy delicado y se terminaría dando "dueños" a las ideas más básicas, más sencillas y a todas las sabias recomendaciones del sentido común.

diversidad esencial de los códigos ejecutables por una CPU permite construir "huellas dactilares" (*fingerprints*) específicas que identifiquen de modo único a cada instancia de un programa y así poder vincularlo con sistemas de gestión de los derechos de autor, de licencias de uso, etc.

Una confección humana no formalizada, no educada y absolutamente independiente de un software daría lugar a alguna de las muchas posibilidades que hay para conseguir el mismo

resultado final y, en este caso, la probabilidad de que los códigos resultantes fuesen idénticos, literalmente idénticos, es muy baja, prácticamente despreciable. Esta misma diversidad es la que hace que, dado un determinado código, la operación de encontrar alguna de esas versiones funcionalmente equivalentes, y literalmente distintas, es algo sencillo para agentes humanos y, por supuesto, siempre puede automatizarse desarrollando y utilizando algo que podríamos llamar "compiladores aleatorios o estocásticos".

Si las formas distintas de escribir formal y educadamente un código fuente se limitan a un reducido grupo de estilos, aceptar que una entidad puede ser propietaria en exclusiva de un determinado código y sus análogos, equivale a aceptar que dicha entidad sea la propietaria de la idea que representa. Dicho de otro modo, una cosa es reconocer que aquello de "Érase un hombre a una nariz pegado..." es algo escrito por D. Francisco de Quevedo y que es fruto de su ingenio, pero otra muy distinta sería decir que tan ilustre autor es el propietario de todas las "composiciones ordenadas en catorce versos endecasílabos distribuidos en dos cuartetos y dos tercetos..."; es decir, de todos los sonetos.

En este ejemplo, no hay problema en limitar la propiedad a la secuencia de palabras concretas que se han utilizado para construir el soneto, ya que el uso de cualesquiera sinónimos, con mucha probabilidad, no cumplen las exigencias de la métrica y la rima, y el cambio de orden en las palabras o bien hacen al texto incorrecto, o bien pierden los valores estéticos de la obra original. Sin embargo, en el caso de un código informático, si sólo reconocemos la autoría y propiedad por parte del au-

tor por las secuencias de comandos que se encadenan dentro del código, entonces se está dejando fuera del erario del autor todas las "perifrasis" y códigos equivalentes que se puedan construir, y que tienen el mismo sentido y significado. En el caso de la poesía las perifrasis suelen ser significativamente menos bellas y acertadas, por lo que no tendrán valor; sin embargo, dentro de la CPU de un ordenador, cualquiera de esas "perifrasis" va a terminar exactamente en el mismo estado que el código de referencia, y son igual de válidas que él.

El problema con el que se enfrentan los promotores de ReactOS al decidir revisar lo que tienen escrito de código, es que no van a poder decir qué es suyo y qué no lo es. La identidad literal y extensa de sus códigos con los de Microsoft no puede ser el criterio para determinar si hay plagio o no, ya que es fácil evitar que se dé esa coincidencia, ya que un plagiador real se habría encargado de generar una de las variantes, funcionalmente idénticas, del código plagiado. Si para evitar este ardid del oponente los auditores pasan a buscar coincidencias e identidades locales, de extensión reducida, la alarma por plagio está asegurada ya que habrá varios conjuntos de comandos y asignaciones que aparezcan en el código de ReactOS y en el de Microsoft, y que realmente sean consecuencia de una programación independiente pero atenta a los estándares y a la ortodoxia actual.

El problema de fondo está 1) en que resulta muy difícil marcar los límites entre coincidencia y plagio, 2) en que la identidad o falta de identidad en el código ejecutable no es una medida válida para dirimir este tipo de conflictos, 3) que trabajar en niveles superiores de organización y complejidad no resuelve el problema, pues sólo lo transporta, isomorfo, a esas esferas más abstractas, 4) que hacer reposar la propiedad sobre las funcionalidades es algo muy delicado y se terminaría dando "dueños" a las ideas más básicas, más sencillas y a todas las sabias recomendaciones del sentido común. ■

JORGE DÁVILA MUÑOZ

Consultor independiente
Director

Laboratorio de Criptografía

**LSIIS – Facultad
de Informática – UPM**
jdavila@fi.upm.es