Verificación estática de confidencialidad en un sistema de múltiples niveles de seguridad basado en Java bytecode

Ricardo Medel, César Martínez Spessot, Juan Carlos Vázquez, Ignacio Giagante

Departamento de Ingeniería de Sistemas de Información
Facultad Regional Córdoba
Universidad Tecnológica Nacional
Maestro M. López esq. Cruz Roja Argentina
Ciudad Universitaria (X5016ZAA)
Córdoba, Argentina

CONTEXTO

La línea aquí presentada es la línea principal de investigación del grupo, centrado en la seguridad informática a través del análisis estático del código. Esta línea de investigación también tiene como objetivo la consolidación del grupo de investigación y su integración en el sistema de ciencia y técnica de la UTN, debido a que dicho grupo es de reciente creación. Este grupo, junto con otros grupos de reciente creación, forman parte de un programa de fomento de la investigación científica puesto en marcha hace un par de años por del Departamento de Ingeniería de Sistemas de Información de la UTN-FRC.

RESUMEN

Debido al incremento en la cantidad e importancia de la información manejada por sistemas informáticos, también se incrementa la importancia de controlar la difusión de información privada de los usuarios. Los mecanismos de control del acceso y las técnicas criptográficas aseguran que la información sólo es accedida por entidades autorizadas, pero resultan insuficientes puesto que una vez que la información es accedida no existe control sobre su distribución.

Extender el sistema de tipos de un lenguaje de programación agregando niveles de seguridad permite analizar estáticamente, antes de su ejecución, cómo un programa accede y utiliza la información confidencial. El creciente uso de código móvil –software obtenido de la red y ejecutado localmente, generalmente en Java *bytecode*– resalta la importancia de aplicar estos análisis de confidencialidad a lenguajes de bajo nivel.

En este proyecto desarrollaremos un sistema de tipos para Java *bytecode* extendido con información de seguridad e implementaremos un chequeador de tipos que verifique la confidencialidad de la información local accedida por código móvil. También estudiaremos la desclasificación controlada de información, a fin de incluir en nuestro análisis a programas que inevitablemente revelan información confidencial, por ejemplo, al responder negativamente ante una contraseña inválida

Palabras clave: Seguridad informática, Confidencialidad, Sistemas de tipos, Lenguaje Ensamblador, Java *bytecode*

1. INTRODUCCION

En un sistema distribuido, el software puede ser importado de la red y ejecutado localmente. Este tipo de software es llamado "código móvil" y usualmente necesita acceder a los datos almacenados localmente. A fin de proteger los datos locales de accesos no autorizados y de modificaciones ilegales, es necesario establecer mecanismos de seguridad que

aseguren que las políticas de confidencialidad e integridad de los datos locales se cumplen cuando son accedidos por código móvil. Distintos mecanismos han sido desarrollados para proteger distintos aspectos del acceso a la información, incluyendo mecanismos de control de acceso y técnicas criptográficas. Sin embargo, estos mecanismos no pueden asegurar la confidencialidad de la información una vez que los datos son accedidos, ya que no hay ningún control sobre su distribución posterior.

A fin de controlar el manejo que un programa hace de la información confidencial, se pueden utilizar técnicas de análisis del flujo de la información [1]. Para ello, los sistemas con múltiples niveles de seguridad (MLS, por sus siglas en inglés) asignan a cada dato, objeto o fragmento de información manipulado por el programa un rótulo de seguridad indicando su nivel de confidencialidad, mientras que a cada entidad computacional que puede acceder o almacenar dicha información se le asigna un rótulo de seguridad indicando su nivel de acceso. De esta manera, los rótulos permiten analizar si los programas cumplen con las políticas de seguridad que indican a qué información pueden acceder las entidades y cómo pueden distribuirla [2, 3]. Las políticas de seguridad se pueden dividir entre "de confidencialidad" (información con nivel de confidencialidad *c* sólo pueden ser accedidas por entidades con nivel de acceso *a* si $a \geq c$) y "de integridad" (la información en entidades con nivel de acceso *a1* puede ser modificada sólo por entidades con nivel de acceso *a2* si $a2 \geq a1$).

La formalización de la noción de "no-interferencia" [4], que establece que los resultados públicos (con bajo nivel de confidencialidad) de un programa no puede depender de los datos secretos (con alto nivel de confidencialidad) a los que el programa accede, permitió el desarrollo de sistemas de tipos para analizar el flujo de la información en programas escritos en lenguajes de programación de alto nivel [5, 6]. Estos sistemas de tipos permiten realizar un análisis estático del código del programa, antes de que éste sea ejecutado, por medio de simples programas verificadores de tipos. Los cuales a su vez pueden ser verificados completamente e incluidos en el sistema básico de seguridad (llamado TCB – *Trusted Computing Base*) de la máquina local sin incurrir en costos computacionales elevados.

Aunque algunos sistemas de tipos para lenguajes de bajo nivel (ensamblador y *bytecode*) han sido desarrollados en la última década [7, 8], el análisis del flujo de información en dichos lenguajes era un problema abierto [1] hasta recientemente, cuando se desarrollaron sistemas de tipos para la confidencialidad de datos accedidos por programas en lenguajes de bajo nivel [9, 10, 11, 12]. Sin embargo, todos los trabajos sobre Java *bytecode* asumen que el código es analizado previamente por el *Java Verifier*, una herramienta que asegura, entre otras cosas, que la pila de ejecución es utilizada correctamente. Nuestra investigación apunta a eliminar ese requisito de modo de evitar la necesidad de incluir el *Java Verifier* en la TCB.

2. LINEAS DE INVESTIGACION y DESARROLLO

Nuestro proyecto aplicará las técnicas estáticas de verificación de confidencialidad desarrolladas para lenguajes ensambladores [11, 12, 20] a la creación de un sistema de tipos para Java *bytecode* que asegure, sin intervención de verificadores externos, que los programas descargados desde la red siguen las políticas de confidencialidad del usuario de dichos programas.

Además, estudiaremos cómo permitir que cierto monto de información confidencial pueda ser observada por entidades de menor nivel de seguridad mientras se mantiene la confidencialidad del sistema. Este fenómeno se denomina "desclasificación", y su inclusión en las técnicas de

verificación permite analizar programas que por su naturaleza no poseen la propiedad de no-interferencia, tales como los programas que verifican contraseñas, los cuales responden negativamente si la palabra ingresada no es la contraseña esperada, mostrando información de bajo nivel de confidencialidad en base a información de alto nivel (la contraseña).

Las técnicas aplicables a MLS fueron desarrolladas originalmente para sistemas militares, pero con el aumento exponencial de la cantidad de programas distribuidos por Internet y ejecutados en computadoras locales, cada vez es más probable que la información confidencial de usuarios, empresas y gobiernos quede expuesta por ataques malintencionados o errores en el software [13]. Nuestro proyecto apunta a disminuir la incidencia de estos ataques, mediante la incorporación de un simple verificador de tipos en el sistema computacional local que verifique que el software bajado de la red cumple con las políticas de confidencialidad del usuario.

El grupo de trabajo original, sito en el Stevens Institute of Technology (Hoboken, New Jersey, EE. UU.), ha desarrollado técnicas para la verificación de la confidencialidad de los datos locales accedidos por programas en lenguaje ensamblador [11, 19, 20]. Dichas técnicas han sido implementadas en los lenguajes SIF [19] y SIFTAL [11], ambos basados en tecnología RISC. Se ha desarrollado un simple compilador y verificador de tipos, escrito en el lenguaje de programación funcional Haskell, para el lenguaje SIF. Además, ha sido definida una función de compilación utilizando un lenguaje imperativo simple similar a C como lenguaje fuente y SIFTAL como lenguaje objeto. Esta función está en proceso de ser implementada. Cabe acotar que un grupo de investigación en el Laboratorio de Investigación y Formación en Informática Avanzada (LIFIA) de la Universidad Nacional de La Plata, ha implementando un verificador de tipos para SIFTAL, el cual está siendo utilizado en nuestro proyecto.

3. RESULTADOS OBTENIDOS/ESPERADOS

El objetivo de desarrollo inmediato es crear un compilador de un lenguaje simple a SIFTAL para alimentar el verificador desarrollado en el LIFIA. Luego se aplicará esa experiencia en el desarrollo de un verificador de tipos de seguridad para Java *bytecode*.

Paralelamente, se está estudiando la posibilidad de aplicar esta técnica a una parte del desarrollo de un sistema de "*e-mall*". Este sistema, en desarrollo por otro grupo de I&D de la UTN-FRC y Microsoft, permite a un proveedor de *e-mall* ofrecer a comercios instalados en un centro de compras un *framework* para proveer a sus clientes información y compras virtuales a través de un sistema *wireless* instalado en el centro comercial. Los comercios utilizan este *framework* para crear muy fácilmente sus servicios web y ponerlos a disposición de los clientes. A su vez, los clientes acceden a esos servicios por medio de sus dispositivos de Internet móviles al entrar al centro comercial. Nuestra técnica podría ser utilizada por el proveedor del *framework* a fin de asegurar que los servicios en línea creados por los comercios cumplen con las políticas de confidencialidad de los datos de los clientes, las cuales han sido establecidas por el proveedor en el contrato aceptado por el cliente cuando se inscribe por primera vez en el servicio.

Como una extensión al proyecto en curso, este desarrollo para "*e-mall*" podría ser utilizado para una comparación empírica de la técnica estática con el reciente renacimiento de las técnicas dinámicas a través de monitores. Se ha postulado [21] que los monitores tienen un poder de análisis de confidencialidad equivalente a los sistemas de tipos. La aplicación de nuestro enfoque al sistema de "*e-mall*" y la creación de un sistema de monitoreo

dinámico para el mismo permitirán un análisis empírico que permita contribuir al estudio de las similitudes y diferencias de estos dos enfoques desde el punto de vista práctico.

4. FORMACION DE RECURSOS HUMANOS

Dado que este es un grupo de investigación de reciente creación, se realizará una primera etapa de estudio dirigido de las técnicas de análisis estático basadas en tipos de datos, a fin de nivelar los conocimientos de los nuevos integrantes y consolidar el grupo. Como resultado de esta etapa se implementará un compilador de un lenguaje imperativo simple al lenguaje SIFTAL, mencionado anteriormente, y se integrará dicho compilador con el verificador desarrollado en el LIFIA.

En lo que respecta a la relación de la formación de recursos humanos con las asignaturas de la carrera asociada al Departamento, este proyecto ampliará los conocimientos del grupo de investigación en temas de las cátedras de Sintaxis y Semática del Lenguaje, Tecnología de Software de Base y Paradigmas de Programación. Dichos conocimientos serán transferidos a estas cátedras a través de seminarios y cursos sobre los temas tratados en el proyecto de investigación.

También se formarán estudiantes avanzados de la carrera de Ingeniería en Sistemas de Información, quienes serán invitados a iniciar su formación en investigación científica y tecnológica, profundizando sus conocimientos en temas de seguridad de sistemas de información y técnicas de seguridad a través de lenguajes de programación.

A fin de transferir los resultados de este proyecto al ámbito de la industria de desarrollo de software, se escribirán artículos científicos para ser presentados en congresos de la especialidad, se presentarán pósters en jornadas profesionales y estudiantiles, se escribirán artículos de difusión general que se publicarán en revistas técnicas, y se organizarán cursos abiertos a estudiantes, docentes, investigadores y profesionales.

5. BIBLIOGRAFIA

[1] A. Sabelfeld y A. Myers, "Language-based information-flow security", IEEE Journal on Selected Areas in Communications, 21(1), 2003.

[2] David Bell y Len LaPadula, "Secure computer systems: Mathematical foundations and model", Technical Report MTR 2547 v2, MITRE, November 1973.

[3] Dorothy E. Denning y Peter J. Denning, "Certification of programs for secure information flow", Communications of the ACM, 20(7):504-513, July 1977.

[4] Joshep A. Goguen y Jose Meseguer, "Security policy and security models", en Proceedings of the Symposium on Security and Privacy, págs. 11-20, IEEE Press, 1982.

[5] Dennis Volpano, Geoffrey Smith y Cynthia Irvine, "A sound type system for secure flow analysis", Journal of Computer Security, 4(3):167-187, 1996.

[6] Aninda Banerjee y David Naumann, "Secure information flow and pointer confinement in a java-like language", en Proceedings of 15th IEEE Computer Security Foundation, págs. 253-267, June 2002.

[7] G. Morrisett, D. Walker, K. Crary y N. Glew, "From System F to Typed Assembly Language", ACM Transactions on Programming Languages and Systems, 21(3):528-569, May 1999.

[8] David Aspinall y Adriana B. Compagnoni, "Heap bounded assembly language", Journal of Automated Reasoning, Special Issue

on Proof-Carrying Code, 31(3-4):261-302, 2003.

[9] G. Barthe, A. Basu y T. Rezk, "Security types preserving compilation", en Proceedings of Verification, Model-Checking and Abstract Interpretation - VMCAI '04, LNCS 2937, Springer-Verlag, 2004.

[10] Samir Genaim y Fausto Spoto, "Information Flow Analysis for Jaba Bytecode", en Proceedings of the 6th International Conference VMCAI '05, Paris, France. January 2005, LCNS 3385, págs. 346-362.

[11] E. Bonelli, A. Compagnoni y R. Medel, "Information-Flow Analysis for Typed Assembly Languages with Polymorphic Stacks", CASSIS 2005, Nice, France. March 8-11 2005, LNCS 3956, Springer-Verlag, 2006.

[12] Dachuan Yu y Nayeem Islam, "A typed assembly language ofr confidentiality", en Proceedings of the 2006 European Symposium on Programming (ESOP'06), Vienna, Austria, March 2006, LNCS 3924.

[13] Ross Anderson, "Security Engineering: A Guide to Building Dependable Distributed Systems", Wiley, 2001.

[14] Edward Amoroso, "Fundamentals of Computer Security Technology", Prentice-Hall PTR, 1994.

[15] Jean-Francois Bergeretti y Bernard A. Carré, "Information-flow and data-flow analysis of while-programs", ACT Trans. Program. Lang. Syst., 7(1):37-61, 1985.

[17] Karl Crary, Aleksey Kliger y Frank Pfenning, "A monadic analysis of information flow security with mutable state", Journal of Functional Programming, 15(2):249-291, 2005.

[18] Mads Dam y Pablo Gambiagi, "Confidentiality for mobile code: The case of a simple payment protocol", en 13th IEEE Computer Security Foundations Workshop, 2000, CFW-13, págs. 233-244, 2000.

[19] R. Medel, A. Compagnoni y E. Bonelli, "Non-Interference for a Typed Assembly Language", en 9th Italian Conference ICTCS '05, Certosa di Pontignano, Italy, 12-14 October 2005. LNCS 3701, Springer Verlag, 2005.

[20] Ricardo Medel, "Typed Assembly Languages for Software Security", PhD Thesis, Stevens Institute of Technology, January 2007.

[21] Alejandro Russo y Andrei Sabelfeld, Chalmers University of Technology, comunicación personal, 2009.