Betriebssystemunterstützung für applikationsspezifische Sicherheitspolitiken

Abstract

Marius Schlegel
Fachgebiet Verteilte Systeme und Betriebssysteme
Technische Universität Ilmenau
marius.schlegel@tu-ilmenau.de

Motivation. IT-Systeme mit umfangreichen Sicherheitsanforderungen setzen zunehmend problemspezifische Sicherheitspolitiken zur Spezifikation, Analyse und Durchsetzung strategischer Sicherheitskonzepte ein. Angesichts der kritischen Bedeutung der Korrektheit der Sicherheitspolitiken in solchen Systemen haben sich Politikentwicklungsansätze basierend auf formalen Methoden und Sicherheitsmodellen etabliert [1, 10, 12, 18, 24], die eine verifizierte Spezifikation und Implementierung von Sicherheitsmechanismen und -architekturen ermöglichen.

Im Kontext von Betriebssystemen wird nunmehr seit über einem Jahrzehnt die Idee verfolgt, Sicherheitspolitiken direkt in politikgesteuerte Betriebssysteme zu integrieren und systemweit rigoros durchzusetzen [23]. Die Kontrolle von Operationen auf Betriebssystemobjekten (z. B. Dateien, Prozesse, Sockets) wird durch die Erweiterung von Betriebssystemkernen (z. B. Linux, FreeBSD, Android) um obligatorische Zugriffssteuerungs- und Informationsflusskontrollmechanismen realisiert [6, 11, 13, 20–22].

Applikationen arbeiten oftmals mit Objekten, die auf einem anderen, wesentlich höheren Abstraktionsniveau liegen als vom Betriebssystem verwaltete Objekte. Beispielsweise wenden ERP-Systeme rollenbasierte Zugriffssteuerungspolitiken an, die die Organisationsstrukturen eines Unternehmens reflektieren [4, 19], Dokumentenmanagementsysteme setzen Informationsflusspolitiken zum Schutz von Vertraulichkeit oder Integrität ein [3, 5], Datenbanksysteme nutzen Label-basierte Sicherheitspolitiken zur Kontrolle von Zugriffen auf Relationen und Sichten [9, 16] und Online-Social-Network-Plattformen verwenden für die Zugriffssteuerung auf Nutzerdaten Politiken basierend auf Nutzerbeziehungen [7, 8].

Dieser Vergleich zeigt, wie stark sich die Art der Objekte und demzufolge auch die Semantik der Politikregeln bezüglich dieser Objekte zwischen Betriebssystemen und Applikationen unterscheidet. Damit wird deutlich, dass eine einzige systemweite Sicherheitspolitik für die Anforderungen von Applikationen nicht ausreicht. Unterschiedliche Applikationen benötigen nicht nur jeweils eigene Sicherheitspolitiken, sondern zudem unterschiedliche Klassen von Politiken, um ihre Sicherheitsziele zu erfüllen. Folglich ist es erforderlich, die auf Betriebssystemebene erprobten Technologien zur Ausstattung und Durchsetzung von Sicherheitspolitiken auch für jede individuelle Applikation bereitzustellen.

Diese Erkenntnis motivierte zur Erweiterung politikgesteuerter Betriebssysteme, beispielsweise SELinux, um die Möglichkeit, Anwendungsobjekte in der Laufzeitumgebung des Kernels zu registrieren und dort Applikationspolitiken gemeinsam mit der systemweiten Sicherheitspolitik durchzusetzen. Hierzu bietet SELinux eine begrenzte Auswahl möglicher Politikparadigmen: Type Enforcement, Multi-level Security und Role-based Access Control.

Mit diesem Ansatz sind die Regeln der Anwendungspolitiken Teil der systemweiten Betriebssystempolitik, da diese im Laufzeitsystem des Betriebssystems durchgesetzt werden. Aufgrund der Unterstützung einer Vielzahl an Politikklassen besitzt das Kernel-Laufzeitsystem einen entsprechend großen Funktionsumfang; das Laufzeitsystem ist mit jeder Applikation, auch wenn diese nur einen einzigen Politiktyp benötigt, assoziiert. Dadurch ergibt sich ein Widerspruch zum Leitprinzip der geringen Komplexität und des geringen Funktionsumfangs der TCB.

Trotz erste Ansätze für dieses Problem, wie SELinux, bieten heutige Betriebssysteme bislang noch keine adäquate Unterstützung für applikationsspezifische Sicherheitspolitiken.

Idee. Thematischer Kontext dieses Beitrags ist die Entwicklung einer neuen Betriebssystemabstraktion für applikationsspezifische Sicherheitspolitiken. Den Ausgangspunkt für diesen Ansatz bilden die Referenzmonitorprinzipien als qualitätsleitende TCB-Designprinzipien. Diese besagen, dass es eine TCB-Kernkomponente zur Politikdurchsetzung gibt, den Referenzmonitor, der (1.) sicher vor unautorisierter Manipulation ist, (2.) unumgehbar bei allen politikrelevanten Objektinteraktionen involviert ist sowie (3.) hinreichend klein und wenig komplex, um formale Verifikation zu ermöglichen [2].

Das Prinzip zur Herstellung der Manipulationssicherheit einer applikationsspezifischen Sicherheitspolitik gegenüber Fehlern innerhalb der mit ihr assoziierten Applikation ist zunächst, Sicherheitspolitik und Applikations-

logik voneinander zu isolieren, sodass es außer an definierten Politikdurchsetzungspunkten keinerlei wechselseitige Beeinflussung zwischen einer Applikation und ihrer Sicherheitspolitik gibt. Realisierungstechnisch reicht hier das Spektrum, je nach Angreifermodell und dem benötigtem Grad der Manipulationssicherheit, von sprachbasierten Isolationsmechanismen (Politik als Instanz einer dynamisch beigebundenen Klasse) über betriebssystemgestützte Isolation (Ausführung der Politik in separatem virtuellem Adressraum) bis hin zu hardwaregestützten Isolationsmechanismen (Ausführung der Politik in Intel SGX Enclave) sowie Container- und Virtualisierungstechniken (Politikausführung in Docker-Container oder VM).

Um die vollständige Kontrolle durch die Sicherheitspolitik umzusetzen, müssen Politikdurchsetzungspunkte (Policy Enforcement Points) bei allen sicherheitsrelevanten Objektinteraktionen gesetzt werden. Mithilfe von Hooks im Applikationsprogramm wird die Isolation von einer Applikation und ihrer Sicherheitspolitik kontrolliert überwunden und die Politik involviert. Im Unterschied zur Manipulationssicherheit kann die vollständige Interaktionskontrolle nicht mehr ausschließlich im Betriebssystem realisiert werden, weil hierfür Wissen über die Applikationslogik erforderlich ist; die Hook-Platzierung kann jedoch zum Entwicklungszeitpunkt durch Algorithmen unterstützt werden [14, 15].

Für die Erfüllung des Prinzips geringer Größe und Komplexität sollten für den Ablauf einer anwendungsspezifischen Sicherheitspolitik nur jene Funktionen implementiert werden, die hinreichend und notwendig zur Umsetzung dieser Politik sind. Wie durch frühere Arbeiten gezeigt werden konnte [17], lassen sich diese Funktionen aus den Komponenten einer modellierten Sicherheitspolitik ableiten.

Beitrag. Dieser Beitrag thematisiert die für die Implementierung der Abstraktion für applikationsspezifische Sicherheitspolitiken notwendige Betriebssystemunterstützung. Um die Eigenschaften des Konzepts einfach und effektiv umzusetzen, ist es die Idee, eine konfigurierbare Schnittstelle für die dynamische Generierung der Umgebung zum Ablauf einer applikationsspezifischen Sicherheitspolitik und Mechanismen für die Assoziation dieser Politik bzw. ihrer Umgebung mit einer Applikation bereitzustellen. Die Konfiguration ist dabei möglich hinsichtlich (1.) der zugrundeliegenden Politikklasse und (2.) der geforderten Manipulationsschutzmaßnahmen. Zur Bestätigung der Praxistauglichkeit des Ansatzes werden erste Ergebnisse der Evaluierung im Rahmen einer Linux-basierten Umgebung präsentiert.

Literaturreferenzen

[1] Peter Amthor, Winfried E. Kühnhauser und Anja Pölck. „WorSE: A Workbench for Model-based Security Engineering“. In: *Comp. & Secur.* 42 (2014), S. 40–55.

[2] James P. Anderson. *Computer Security Technology Planning Study*. Tech. Rep. ESD-TR-73-51. U.S. Air Force Electronic Systems Division, 1972.

[3] D. Elliott Bell und Leonard J. LaPadula. *Secure Computer System: Unified Exposition and Multics Interpretation*. Tech. Rep. MTR-2997-REV-1. Bedford, MA, USA: MITRE Corporation, 1976.

[4] Rafae Bhatti, Arif Ghafoor, Elisa Bertino et al. „X-GTRBAC: An XML-based Policy Specification Framework and Architecture for Enterprise-wide Access Control“. In: *ACM Trans. on Inf. and Syst. Secur.* 8.2 (2005), S. 187–227.

[5] K. J. Biba. *Integrity Considerations for Secure Computer Systems*. Tech. Rep. ESD-TR-76-372 MTR-3153, Rev. 1. Bedford, MA, USA: MITRE Corporation, 1977.

[6] Sven Bugiel, Stephan Heuser und Ahmad-Reza Sadeghi. „Flexible and Fine-Grained Mandatory Access Control on Android for Diverse Security and Privacy Policies“. In: *Proc. 22nd USENIX Secur. Symp.* 2013, S. 131–146.

[7] Philip W. L. Fong. „Relationship-Based Access Control: Protection Model and Policy Language“. In: *Proc. 1st ACM Conf. on Data and App. Secur. and Priv.* ACM, 2011, S. 191–202.

[8] Philip W. L. Fong, Mohd M. Anwar und Zhen Zhao. „A Privacy Preservation Model for Facebook-Style Social Network Systems“. In: *Proc. 14th European Symp. on Research in Comp. Secur.* Springer, 2009, S. 303–320.

[9] IBM. *Db2 11.1 – Label-based access control (LBAC) overview*. `https://www.ibm.com/support/knowledgecenter/en/SSEPGG_11.1.0/com.ibm.db2.luw.admin.sec.doc/doc/c0021114.html`. 2016.

[10] Somesh Jha, Ninghui Li, Mahesh Tripunitara et al. „Towards Formal Verification of Role-Based Access Control Policies“. In: *IEEE Trans. on Dependable and Secure Computing* 5.4 (2008), S. 242–255.

[11] Maxwell N. Krohn, Alexander Yip, Micah Z. Brodsky et al. „Information Flow Control for Standard OS Abstractions“. In: *Proc. 21st ACM Symp. on Operating Systems Principles*. ACM, 2007, S. 321–334.

[12] Winfried E. Kühnhauser und Anja Pölck. „Towards Access Control Model Engineering“. In: *Proc. 7th Int. Conf. on Inform. Syst. Secur.* Bd. 7093. Springer, 2011, S. 379–382.

[13] Peter A. Loscocco und Stephen D. Smalley. „Integrating Flexible Support for Security Policies into the Linux Operating System“. In: *Proc. FREENIX Track: 2001 USENIX Ann. Techn. Conf.* 2001, S. 29–42.

[14] Divya Muthukumaran, Trent Jaeger und Vinod Ganapathy. „Leveraging ‚Choice‘ to Automate Authorization Hook Placement“. In: *Proc. 2012 ACM Conf. on Comp. and Comm. Secur.* ACM, 2012, S. 145–156.

[15] Divya Muthukumaran, Nirupama Talele, Trent Jaeger et al. „Producing Hook Placements to Enforce Expected Access Control Policies“. In: *Proc. 7th Int. Symp. on Engineering Secure Software and Syst.* Springer International Publishing, 2015, S. 178–195.

[16] Oracle. *Oracle Label Security Administrator's Guide, 18c*. `https://docs.oracle.com/en/database/oracle/oracle-database/18/olsag/index.html`. 2018.

[17] Anja Pölck. „Small TCBs of Policy-controlled Operating Systems“. Dissertation. Universitätsverlag Ilmenau, 2014.

[18] Marius Schlegel. „Analyse und Simulation dynamischer RBAC-Zugriffssteuerungsmodelle“. In: *Proc. D·A·CH Security 2015*. syssec Verlag, 2015.

[19] Wei She und Bhavani Thuraisingham. „Security for Enterprise Resource Planning Systems“. In: *Information Systems Security* 16.3 (2007), S. 152–163.

[20] Stephen Smalley und Robert Craig. „Security Enhanced (SE) Android: Bringing Flexible MAC to Android“. In: *Proc. 20th Ann. Network and Distributed System Secur. Symp.* The Internet Society, 2013.

[21] Wai Kit Sze, Bhuvan Mital und R. Sekar. „Towards More Usable Information Flow Policies for Contemporary Operating Systems“. In: *Proc. 19th ACM Symp. on Access Control Models and Technol.* ACM, 2014, S. 75–84.

[22] Chris Vance und Robert N. M. Watson. *Security-Enhanced BSD.* Techn. Rep. Network Associates Laboratories, 2003.

[23] Robert N. M. Watson. „A Decade of OS Access-control Extensibility“. In: *ACM Queue* 11.1 (2013), 20:20–20:41.

[24] Giorgio Zanin und Luigi Vincenzo Mancini. „Towards a Formal Model for Security Policies Specification and Validation in the SELinux System“. In: *Proc. 9th ACM Symp. on Access Control Models and Technol.* ACM, 2004, S. 136–145.