تسهیل توسعه امن سیستم‌ها از طریق متصل نمودن مدلسازی و اعمال سیاست‌های کنترل دسترسی

سعید پارسا[†]، مرتضی دامن افشان[‡]

چکیده

علی‌رغم تأکید بر حذف شکاف موجود بین مدلهای نرم‌افزاری و کدهای پیاده‌سازی، تلاش‌های کمی در مورد بکارگیری ابزارهای نرم‌افزاری جهت اعمال خودکار مدل‌های کنترل دسترسی بطور مستقیم در داخل متن برنامه‌ها صورت گرفته است. در این مقاله به شرح طراحی و پیاده‌سازی یک محیط اعمال سیاست‌های کنترل دسترسی پرداخته شده است. در این محیط، سیاست‌های مدل شده کنترل دسترسی بر مبنای دیدگاه یا همان (VBAC (View-Based Access Control که به فرمت (XML Metadata Interchange) XMI هستند، تبدیل به زبان سیاست دیدگاه یا (VPL (View Policy Language می‌شود.کدهای تولید شده به زبان سیاست دیدگاه، به راحتی قابل تبدیل به دستورالعمل‌های زبان جاوا است. در نهایت دستورالعمل‌های جاوای متناظر با کدهای زبان سیاست دیدگاه به داخل برنامه‌هایی که قرار است امن شوند، اعمال می‌شوند. این روش شکاف موجود بین مدلسازی و اعمال سیاست‌های کنترل دسترسی را از بین می برد و تضمین می‌کند که کد برنامه همانطور که طراح برنامه قصد داشته، امن شده است.

کلمات کلیدی

مهندسی امنیت، کنترل دسترسی، توسعه سیستم، مدلسازی سیاست‌ها، اعمال سیاست‌ها.

Facilitating Secure Development of Systems through Connecting the Modeling and Enforcement of Access Control Policies

Saeed Parsa, Morteza Damanafshan
Department of Computer Engineering
Iran University of Science and Technology, Tehran, Iran
parsa@iust.ac.ir, morteza_damanafshan@comp.iust.ac.ir

Abstract

Despite the emphasis on removing the gap between software models and implementation code, there has been made little effort to apply software tools to enforce access control models directly into program code. In this paper, the design and implementation of an access control policy enforcement environment is described. Within this environment, the modeled VBAC (View-Based Access Control) policies defined in XMI (XML Metadata Interchange) format are translated into VPL (View Policy Language). This VPL code can then be easily translated into Java primitives. Finally, these Java primitives are enforced into the Java program code to be secured. This approach removes the gap between modeling and enforcement of access control policies, and can ensure that the code is secured as desired by the designer.

Keywords

Security Engineering, Access Control, System Development, Policy Modeling, Policy Enforcement.

[†] دانشیار گرایش نرم‌افزار، عضو هیأت علمی دانشگاه علم و صنعت ایران، دانشکده مهندسی کامپیوتر، parsa@iust.ac.ir.
[‡] دانشجوی کارشناسی ارشد، دانشگاه علم و صنعت ایران، دانشکده مهندسی کامپیوتر، morteza_damanafshan@comp.iust.ac.ir.

۱- مقدمه

کنترل دسترسی به عنوان یکی از پایه‌های علم امنیت رایانه، در حال تبدیل شدن به یکی از مهمترین مباحث موجود در بسیاری از زمینه‌های علوم رایانه است. در گذشته یک برنامه کاربردی موقعی امن تلقی می‌شد که از تکنیک‌هایی همانند رمزنگاری و پروتکل‌های امن در راستای اهداف خود استفاده می‌کرد. همگام با پیشرفت تکنولوژی نرم‌افزار، حملات جدید و آسیب پذیری‌های جدیدی [8,13] ظهور کرده‌اند که مقابله با آنها با استفاده از روش‌های سنتی تقریباً غیرممکن است و دقیقاً به همین دلیل است که تنها تکیه بر تکنیک‌های سنتی با شکست روبرو می‌شود [7]. در حقیقت این تکنیک‌ها تنها اطلاعات جاری بر روی شبکه را از گزند دسترسی‌های غیرمجاز مصون نگاه می‌دارند. بعلاوه این تکنیک‌ها برنامه‌ها را به عنوان جعبه‌های سیاهی در نظر می‌گیرند و قادر نیستند از مسایلی که در داخل برنامه‌ها روی می‌دهند، مطلع شوند.

در حالت کلی امن ساختن برنامه‌ها از طریق اعمال سیاست‌های کنترل دسترسی یک فرآیند دو مرحله‌ای است. مرحله اول بدست آوردن و مدلسازی سیاست‌های کنترل دسترسی و مرحله دوم اعمال آن سیاست‌ها در داخل برنامه‌هاست. زبان ها و ابزارهای بسیاری همانند آنچه که در [4,11,16-20,22,23,26,27] مطرح شده‌اند، برای مدلسازی سیاست‌های کنترل دسترسی بوجود آمده است. همچنین روش‌های متعددی برای اعمال سیاست‌های کنترل دسترسی در داخل برنامه‌ها پیشنهاد شده است. برخی از این روش‌ها سیاست‌های کنترل دسترسی را در سطح کدهای اسمبلی و بایتکد اعمال می‌کنند [24,28]. برخی دیگر نیز اعمال این سیاست‌ها را در سطوح بالاتر انجام می‌دهند[3,12,25] .

مهمترین مسأله‌ای که در اینجا وجود دارد این است که باید هر دوی این مراحل (مدلسازی و اعمال سیاستهای کنترل دسترسی) را با هم در نظر گرفت. این بدین مفهوم است که ما نمی‌توانیم تنها به یکی از این دو مرحله توجه داشته باشیم و حال آنکه دیگری را ندیده بگیریم. اگر ما تنها بر روی مدلسازی سیاست‌های کنترل دسترسی تمرکز کنیم، در اینصورت اتخاذ یک روش نامناسب برای اعمال این سیاست‌ها در داخل برنامه‌ها باعث کاهش شدید کارایی برنامه امن شده خواهد شد. از سویی دیگر اگر تنها به نحوه اعمال سیاست‌های امنیتی پرداخته شود و مباحث مدلسازی کمتر مورد توجه واقع شود مشکل دیگری بوجود می‌آید. توجه کنید که سیاست‌های کنترل دسترسی و امنیتی بیشتر به این دلیل نقض می‌شوند، که مورد استفاده صحیح قرار نمی‌گیرند [1] . این امر بیشتر به دلیل پیچیدگی سیاست‌های کنترل دسترسی روی می‌دهد. مدل کردن این سیاست‌ها در مرحله طراحی سیستم می‌تواند از بروز چنین مشکلاتی جلوگیری کند. به عبارت دیگر هر چقدر سطح انتزاع سیاست‌های کنترل دسترسی بالاتر باشد، احتمال اینکه مشکلات و ناسازگاری بین آنها راحت‌تر کشف و حل شود نیز بیشتر می‌شود. بنابراین سیاست‌های کنترل دسترسی را نمی‌توان بصورت کورکورانه در سیستم‌های حساس امنیتی قرار داد، و بایستی کل فرآیند توسعه سیستم، نحوه مدلسازی و اعمال آن سیاست‌ها را در نظر بگیرد [1,6].

علی‌رغم اینکه بسیاری از محققین بر روی هر یک از مراحل مدلسازی و اعمال سیاست‌های کنترل دسترسی تمرکز کرده‌اند، ولی با این حال یک نکته اصلی در اینجا فراموش شده است، و آن پلی است که این دو فاز را به هم متصل می‌سازد. علی‌رغم اینکه هدف غایی هر یک از این دو مرحله ساخت سیستم‌های امن است، ولی با این حال اگر این پل به دقت ساخته نشود این هدف نمی‌تواند کاملاً محقق شود. در واقع ابزارها و زبان‌های بسیاری جهت مدلسازی سیاست‌های کنترل دسترسی مطرح شده‌اند، ولی با این وجود هیچکدام از این ابزارها و زبان‌ها، سیاست‌های کنترل دسترسی مدل شده در درون خود را به کد مناسبی که بتوان آن را مستقیماً به داخل برنامه‌ها اعمال کرد، تبدیل نمی‌کنند. در حقیقت قبل از اعمال کدهای تولید شده توسط ابزارهای مدلسازی به داخل برنامه‌ها، نوعی عمل پیش پردازش روی این کدها لازم است. پس از انجام عمل پیش پردازش، می‌توان مرحله اعمال سیاست ها را شروع کرد. کدهای تولید شده توسط ابزارهای مدلسازی، بایستی به فرمتی تبدیل شوند تا با کدی که می‌خواهیم امن شود سازگار باشد. غالباً وجود ناسازگاری از آنجا ناشی می‌شود که سطح انتزاع کدهای تولید شده توسط ابزارهای مدلسازی، متفاوت با سطح انتزاع کدهایی است که مکانیزم‌های اعمال سیاست‌های کنترل دسترسی قادرند آنها را در برنامه‌ها اعمال کنند. ابزارهای مدلسازی عموماً مدلها را تبدیل به کدهای سطح بالایی می‌کنند که به سادگی قابل اعمال توسط مکانیزم‌های اعمال کنترل دسترسی نیستند. در کنار ابزارهای موجود برای مدلسازی سیاست‌های کنترل دسترسی و مکانیزم‌های موجود برای اعمال آنها، برخی چرخه‌های توسعه امن[1] [10,14] نیز جهت مجتمع کردن و ادغام سیاست‌های امنیتی در طول فرآیند توسعه نرم‌افزار مطرح شده‌اند، ولی با این حال هیچ‌یک از آنها روی اعمال خودکار سیاست‌های کنترل دسترسی مدل شده به داخل کد برنامه‌ها تأکیدی نداشته‌اند.

هدف اصلی این مقاله تبدیل خودکار سیاست‌های کنترل دسترسی مدل شده به زبان UML به کدهای مناسب و سپس ادغام مناسب آنها به داخل کدهای برنامه‌هایی است که قرار است امن شوند. در این راستا، ما محیطی را طراحی و پیاده‌سازی کرده‌ایم تا سیاست‌های کنترل دسترسی بر مبنای دیدگاه[2] را بصورت خودکار در داخل کد برنامه‌های نوشته شده به زبان جاوا اعمال می‌کند. در واقع محیط توسعه داده شده پلی است که مرحله مدلسازی را به مرحله اعمال سیاست‌های کنترل دسترسی متصل می‌کند و فرآیند امن‌سازی سیستم‌ها را یکپارچه و بدون درز می‌سازد.

بخش‌هایی که در ادامه این مقاله به آنها خواهیم پرداخت به قرار زیر هستند: در بخش ۲ جایگاه سیاست‌های کنترل دسترسی در فرآیند توسعه امن شرح داده خواهد شد. در بخش ۳ محیط توسعه داده شده مورد ارزیابی قرار می‌گیرد، و در نهایت در بخش ۴ نتایج بدست آمده از مقاله را جمع‌بندی خواهیم کرد.

۲- جایگاه سیاست‌های کنترل دسترسی در فرآیند توسعه امن سیستم‌ها

در این بخش مراحل مدلسازی تا اعمال خودکار سیاست‌های کنترل دسترسی در قالب یک فرآیند توسعه امن نرم‌افزار مطرح شده است. از آنجاییکه برای مدلسازی سیاست‌ها، مدل کنترل دسترسی برمبنای دیدگاه انتخاب شده است، لذا ابتدا نیازمندی‌های خاص این مدل بایستی به ترتیب برطرف شوند و سپس سیاست‌های مدل شده در داخل متن برنامه‌ها اعمال شوند.

در اینجا باید خاطرنشان کرد که مدل کنترل دسترسی بر مبنای دیدگاه روی هر منبع تعدادی دیدگاه[۳] تعریف می‌کند و سپس تعدادی از این دیدگاه‌ها را به نقش‌های مختلف سیستم انتساب می‌کند. بدین ترتیب هر نقش جهت تعامل با اشیا بایستی از طریق آن دیدگاه‌ها وارد عمل شود. در این مقاله کلاس‌های استفاده شده در برنامه جاوا به عنوان منابع در نظر گرفته می‌شوند و دیدگاه‌هایی بر اساس سیاست‌های دسترسی روی آن کلاسها تعریف می‌شوند، بدین طریق هر نقش بسته به اینکه چه دیدگاه‌هایی در مورد آن کلاس به وی انتساب داده شده باشد، به بخشی از متدهای آن کلاس دسترسی پیدا می‌کند. همچنین جهت توصیف مفاهیم مربوط به مدل کنترل دسترسی بر مبنای دیدگاه از زبان سیاست دیدگاه[۴] یا VPL [2,5,9] استفاده شده است.

۲-۱- مرحله مدلسازی سیاست‌های کنترل دسترسی

فعالیت‌های این مرحله شامل : "طراحی نقش‌ها"، "طراحی کلاس‌های سیستم و نمودارهای واسطها[۵]"، "طراحی نمودارهای دیدگاه"، "انتساب دیدگاه‌های اولیه به نقشها"،و "طراحی نمودارهای شما[۶] " می‌باشد.

۲-۲- مرحله پیاده‌سازی و اعمال سیاست‌های کنترل دسترسی

این مرحله شامل چهار فعالیت است که عبارتند از: "پیاده‌سازی کلاس‌های سیستم"، "تبدیل نمودارهای کنترل دسترسی به XMI "، "تبدیل کدهای XMI به کدهای VPL "،" تبدیل کدهای VPL به کدهای جاوا و درج آنها در مکان‌های مناسبی از کلاس‌های مورد نظر". بجز فعالیت "پیاده‌سازی کلاس‌های سیستم"، بقیه فعالیت‌های این مرحله بصورت خودکار انجام می‌گیرد.

۳- ارزیابی

روش‌ها و ابزارهای زیادی جهت مدلسازی سیاست‌های کنترل دسترسی موجود می‌باشند [4,16-20,22,23,26,27]. حتی برخی از این ابزارها امکاناتی اضافی همانند یافتن تناقض‌ها در سیاست‌های مدل شده را علاوه بر امکانات ایجاد و ویرایش سیاستهای کنترل دسترسی فراهم می‌کنند [15,21]. اما با این حال هیچ‌گونه امکاناتی جهت تبدیل سیاستهای مدل شده به کدهایی که بتوان مستقیماً به متن برنامه‌ها اعمال کرد، ندارند. با استفاده از محیط توسعه داده شده در این مقاله، بسادگی می‌توان مستندات XMI متناظر سیاست‌های کنترل دسترسی مدل شده با ابزارهایی همانند [29,31] را تبدیل به کدهای قابل اجرا نمود و به داخل برنامه‌ها اعمال کرد. مزیت دیگر محیط توسعه داده شده، استفاده از VPL به عنوان کدمیانی است، که بعدها می‌توان این کد میانی را به هر زبان دلخواه دیگری تبدیل کرده و به برنامه‌هایی به همان زبان اعمال نمود.

مکانیزم‌هایی [3,12,25] نیز جهت اعمال سیاست‌های امنیتی در متن برنامه‌ها وجود دارد، ولی مشکل این است که این مکانیزم‌ها، سیاست‌های امنیتی را در قالب فایل‌هایی می‌پذیرند، که بایستی مستقیماً ویرایش شوند و در نتیجه امکان مدلسازی سطح بالا را نمی‌دهند. برخی از زبان‌های توصیف سیاست‌های امنیتی و کنترل دسترسی نیز همانند [11] وجود دارند، ولی اینها نیز صرفاً امکان توصیف متنی سیاست‌های کنترل دسترسی را به طراحان سیاست‌های امنیتی می‌دهند، نه محیط مدلسازی سطح بالایی ارائه می‌دهند، و نه در مورد اعمال آن سیاست‌های توصیف شده به متن برنامه‌ها حرفی به میان می‌آورند.

۴- نتیجه

این مقاله راهکاری تازه برای تبدیل خودکار سیاست‌های کنترل دسترسی بر مبنای دیدگاه به کدهای مناسب و سپس اعمال آنها در داخل کدهای برنامه پیشنهاد می‌کند. این سیاست‌ها که به شکل UML مدل شده‌اند در ابتدا بصورت فایل‌های XMI از یک محیط مدلسازی دریافت می‌شوند. سپس محیط توسعه داده شده در این مقاله این فایل‌های XMI را به کدهای VPL تبدیل می‌کند. این کد سپس به دستورالعمل‌های زبان جاوا تبدیل شده و در نهایت در داخل برنامه‌هایی به زبان جاوا که قرار است امن شوند، قرار می‌گیرد.

مشخصاً با حذف خودکار شکاف موجود بین مدلسازی و اعمال سیاست‌های کنترل دسترسی، می‌توان مطمئن شد که کدها همانطور که در مرحله مدلسازی مد نظر بوده‌اند، امن شده‌اند. همچنین استفاده از UML به عنوان محبوب‌ترین زبان مدلسازی

برای تعریف سیاست‌های کنترل دسترسی، مزیت دیگری برای راهکار ارائه شده در این مقاله است.

مراجع

[1] Anderson, R.J., *Security Engineering: A Guide to Building Dependable Distributed Systems*, Wiley, 2001.
[2] Brose, G., *Access Control Management in Distributed Object Systems*, PhD thesis, Freie Universität Berlin, 2001.
[3] Erlingsson, U., *The Inlined Reference Monitor Approach to security policy enforcement*, PhD thesis, Cornell University, January 2004.
[4] Basin, D., Doser, J., Lodderstedt, T., *Model Driven Security: From UML Models to Access Control Infrastructures*, Technical Report, Albert-Ludwigs Universitat Freiburg, June 2003.
[5] Brose, G., Koch, M., Lohr, K. P., *Integrating Security Policy Design into the Software Development Process*, Technical Report, Freie Universität Berlin, Nov. 2001.
[6] Jayaram, K. R., and Mathur, A. P., Software *Engineering for Secure Software-State of the Art: A Survey*. Technical Report, Department of Computer Science Purdue University. September 2005.
[7] Anderson, R. J., *"Why cryptosystems fail"*, Communications of the ACM, 37(11):32–40, 1994.
[8] Baratloo, A., Singh, N., and Tsai, T., *"Transparent runtime defense against stack smashing attacks"*, In Proceedings of the USENIX Annual Technical Conference, pages 251–262, June 2000.
[9] Brose. G., *"A view–based access model for CORBA"*, *Secure Internet Programming: Security Issues for Mobile and Distributed Objects*, LNCS 1603, Springer 1999.
[10] Chess B., and McGraw, G., *"Software security"*, IEEE Security and Privacy Magazine, (2):53–84, 2004.
[11] Damianou, N., Dulay, N., Lupu, E., and Sloman, M., *"The Ponder policy specification language"*, In Proceedings of International Workshop on Policies for Distributed Systems and Networks (POLICY), 2001.
[12] Erlingsson, U., and Schneider, F. B., *"SASI enforcement of security policies: A retrospective"*, In Proceedings of the New Security Paradigms Workshop, Ontario,Canada, Sept. 1999.
[13] Hardy, N., *"The confused deputy"*, ACM Operating Systems Review, 22(4):36-38, October 1998.
[14] Howard, M. and Lipner, S., *"Inside the windows security push"*, IEEE Security and Privacy, 1(1):57–61, 2003.
[15] Jürjens, J, *"Formal Development and Verification of Security-Critical Systems with UML"*, AVoCS'01 – Workshop on Automated Verification of Critical Systems, Oxford, April 19–20 2001.
[16] Jürjens, J., *"Secure Java Development with UML"*, I-NetSec 01 – First International IFIP TC-11 WG 11.4 Working Conference on Network Security, Leuven ,Belgium, November 26–27, 2001.
[17] Jürjens, J., *"Towards Secure Systems Development with UMLsec"*, International Conference on Fundamental Approaches to Software Engineering (FASE/ETAPS) LNCS, Springer-Verlag, Genoa, Italy, April 2001.
[18] Jürjens, J., *"UMLsec: Extending UML for Secure Systems Development"*, UML, LNCS, Springer, Dresden, 2002.
[19] Jürjens, J., *"Using UMLsec and Goal-Trees for Secure Systems Development"*, ACM Symposium of Applied Computing (SAC), Madrid, March 10–14, 2002.
[20] Koch, M., and Presicce, F.P., *"Access Control Policy Specification in UML"*, Springer-Verlag, 2003.
[21] Koch, M. Mancini, L., and Presicce, F.P., *"Conflict detection and resolution in access control specifications"*, In proceedings of foundations of software science and computation structures (FoSSoCS), LNCS, Springer,2002.
[22] Koch, M., Mancini, L., and Presicce, F.P., *"Foundations for a graph-based approach to the specification of access control policies"*, In Proceedings of FoSSoCS, LNCS, Springer, March 2001.
[23] Lodderstedt, T., Basin, D.A., and Doser, J., *"SecureUML: A UML-based modeling language for model-driven security"*, In UML, pages 426–441, 2002.
[24] Rudys, A., Wallach, D.S., *"Enforcing Java Runtime Properties Using Bytecode Rewriting"*, In Proceedings of International Symposium on Software Security ,Tokyo, Japan, November 2002.
[25] Schneider, F.B., Morrisett, G., and Harper, R., *"A Language-Based Approach to Security"*. Informatics—10 Years Back, 10 Years Ahead. LNCS, Springer, 2000.
[26] Shabalin, P. and Jürjens, J., *"Towards Tool Support for UMLsec"*, ISTA 2003, Kharkiv, June 19–21, 2003.
[27] Shin, M.E., and Ahn, G.J., *"UML-based representation of the role-based access control"*, In Proceedings of the IEEE, 2000.
[28] Wahbe, R., Lucco, S., Anderson, T.E., and Graham, S.L., *"Efficient software-based fault isolation"*, Operating System Review, 1993.
[29] Borland Together, September 2006, www.borland.com.
[30] MDR: Meta Data Repositiry, September 2006, mdr.netbeans.org.
[31] Poseidon, September 2006, www.gentleware.com.
[32] XMI: XML Metadata Interchange, September 2006, http://www.omg.org.

زیرنویس‌ها

[1] Secure Development Lifecycle (SDL)
[2] View-Based Access Control (VBAC)
[3] View
[4] View Policy Language (VPL)
[5] Interface
[6] Schema