Recent Research in Secure Software
Yee, George

For the publisher's version, please access the DOI link below./ Pour consulter la version de l'éditeur, utilisez le lien DOI ci-dessous.

https://doi.org/10.4224/8914119

National Research Council Canada
Institute for Information Technology

Conseil national de recherches Canada
Institut de technologie de l'information

NRC · CNRC

*Recent Research in Secure Software **

Yee, G.
March 2006

* published as NRC/ERB-1134. 8 pages. March 2006. NRC 48478.

Canada

National Research Council Canada
Institute for Information Technology

Conseil national de recherches Canada
Institut de technologie de l'information

ERB-1134

NRC · CNRC

Recent Research in Secure Software

Yee, G.
March 2006

Canada

NRC 48478

Recent Research in Secure Software[1]

George Yee
Institute for Information Technology
National Research Council Canada
george.yee@nrc-cnrc.gc.ca

Abstract

The rapid propagation of software systems into nearly every aspect of modern life together with the ever growing number of threats against these systems have given rise to one of the greatest challenges in information technology today. This is the challenge of obtaining software systems that are secure from threats. These threats range from exploitations of buffer overflows and unprotected critical memory locations to reverse engineering in order to find vulnerabilities. Researchers have risen to this challenge by proposing solutions that touch all aspects of software development and operation. Yet, an overall view of this research, showing how seemingly diverse research efforts fit together, does not appear to exist. Such an organized view may help the secure software research community understand where recent research has occurred and direct new research to interesting and promising areas. In addition, newcomers to this field will quickly see what secure software is all about. This paper provides this view and suggests a way to identify new research topics in secure software.

1. Introduction

Today, software touches almost every aspect of our lives. Almost everything that we do depends on software that runs computers and computer networks. Software runs computers and networks that control and manage manufacturing processes, water supplies, electric power generation and distribution, air traffic control systems, stock market trading systems, and many other engines of the modern economy. The Internet has become indispensable for governments, companies, universities, and financial institutions. Yet, despite the important roles that software plays in modern life, it is full of vulnerabilities that put our collective and individual well being at risk. A recent report (September 2005) suggests that computer crimes are skyrocketing [1]. In studies on trends in Internet threats, prepared for CSI (Computer Security Institute), IBM, and McAfee, the following points were made [1]: i) there is increasing risks to individuals due to the growth of identity theft schemes and the growing level of financial damage due to theft of sensitive company data, ii) computer virus attacks continue to be the leading source of financial loss, but unauthorized access is a close second, responsible for almost one quarter of financial losses, iii) threats now originate from professional criminals exploiting the Internet, with about 300 malicious threats per month two years ago to 2000 such threats a month in 2005, iv) types of cyber crimes include extortion, damage to reputation, fraud, phishing, service disruption, information theft, and money laundering, v) there were more than 237 million security attacks in the first half of 2005, mostly targeted at the US government, followed by manufacturing, financial services, and health care, and vi) the incidence of security threats contained in email rose from one in every 52 messages in December 2004 to one in every 35 in January 2005 to one in every 28 in June 2005. In the face of such threats, building software systems that are resistant to these threats is one of the greatest challenges of modern times.

The above statistics are alarming but the computer security problem has existed for more than twenty years as evidenced by the following quote from a 1981 paper: "Efforts to build "secure" computer systems have now been underway for more than a decade" [2]. It's just that recently the security problems have grown many times worst. Researchers have risen to the above challenge by proposing different varied solutions with the purpose of making software more secure. Solutions range from integrating security requirements with software functional requirements, to specific dangers to watch for during design, to code obfuscation to resist reverse engineering, to protecting critical memory locations at run time and many others. However, no overview of recent research in secure software appears to exist. Such an overview would provide at least the following benefits: i) help students, established researchers, and new comers to the secure software field know what approaches have been taken (especially useful for new comers from related fields such as security), ii) help students and researchers see the "big picture" of where the different approaches fit, and iii) identify new opportunities for research based on where the research coverage has been sparse or how the varied approaches interrelate. The objective of this work is to provide this overview.

Before proceeding further, it is useful to define the meaning of "secure software". The field of secure software is made up of two subfields: software security and application security. McGraw [3] defines these terms nicely: "*Software security* is about building secure software. Issues critical to this subfield include software risk management, programming languages and platforms, software audits, designs for security, security flaws, and security tests. Software security is mostly concerned with designing software to be secure, making sure that software is secure, and educating software developers, architects, and users." "*Application security* is about protecting software and the systems that the software runs after development is complete. Issues critical to this subfield include sandboxing code, protecting against malicious code, locking down executables, monitoring programs (especially their input) as they run, enforcing software use policy with technology, and dealing with extensible systems." This work reviews research in both software security and application security.

In the literature, there are two works that "summarize" work in software security. Apart from the fact that they are not as recent as this work, they also differ from this work in the following ways. Wang & Wang [26] present a taxonomy of security considerations as they relate to software quality. They show how different types of security risks affect software quality. In addition they indicate the effectiveness of various security technologies in dealing with security threats and risks. Wang & Wang [26] differs from this work in that they do not look at recent research approaches to securing software nor are the security technologies they discuss oriented towards building secure software. Devanbu & Stubblebine [22] discuss a roadmap for incorporating security into software engineering. They examine the interplay between software engineering and security engineering roughly along the lines of the waterfall model and discuss a number of security challenges along the way. Devanbu & Stubblebine [22] differs from this work in that their focus is to highlight challenges in integrating security into software engineering rather than examine recent research approaches for securing software. However, their work is closer to this work than Wang & Wang [26] in that they provide references to research that appear promising at dealing with the challenges.

The rest of this paper is organized as follows. Section 2 surveys recent research in secure software. Section 3 discusses the coverage of research in secure software and proposes how to find new areas for research. Section 4 gives conclusions.

2. Recent Research in Secure Software

The research papers examined here were retrieved from two databases, the ACM Digital Library and IEEE Xplore using the search expression "secure software" on September 9, 2005. The number of papers retrieved form these databases were 200 and 43 respectively. The actual papers used (numbering 64) were selected from all those retrieved using the following criteria:

- Must be about research, so education type papers were excluded,
- Must be about software security or application security,
- Must be of wide applicability, hence niche papers (e.g. features of a specific language) were excluded,
- Must have been published within the last 10 years (1996-2005) (most papers are within the last 5 years).

As a result, this work has the following limitations: i) the research coverage is incomplete, and ii) paper selection for this work is imprecise. Nevertheless, the results of this work should be a good approximation to the actual situation.

2.1. Classification Method

The retrieved papers were classified according to their subjects. Software security and application security papers were classified separately in two tables. Software security papers were classified according to "requirements", "processes and methods", "coding methods", "vulnerabilities identification", "usability", "testing", "tools", and "other". These categories refer to software or software development. Application security papers were classified under: "threat identification", "protection from tampering", "protection from copying", "other protection", "integrity verification", and "challenges". Application security categories refer to the software in execution. For all category headings, the number next to each heading is the number of papers under that heading.

2.2. Recent Research in Secure Software

Table 1 shows the papers retrieved for software security. The paper reference appears in the left column with a summary of each subject in the right column.

Table 1. Research in software security

	REQUIREMENTS (9)
Del Grosso et al [23]	Proposes generating tests for buffer overflows using static analysis, program slicing, and data dependency analysis.
Haley et al	Describes how representing threats as

[30]	crosscutting concerns can determine and incorporate security requirements with functional requirements.
Koch & Parisi-Presicce [37]	Investigates how access control security requirements may be integrated into the analysis phase of software development using a model-driven approach.
Kienzle & Wulf [42]	Presents a new approach to assess the degree to which software meets its security requirements.
Vetterling et al [44]	Shows how to integrate security aspects into the software development process using the Common Criteria.
Doan et al [51]	Incorporates mandatory access control (MAC) into UML elements to allow UML to express security requirements.
Pauli & Xu [58]	Presents an approach to architectural design and analysis of secure software systems based on system requirements in the form of use cases and misuse cases.
Alghathbar & Wijesekera [60]	On a high-level approach for analyzing information flow requirements and ensuring enforcement of flow control policies; improves security by detecting unsafe flows early in the life cycle.
Hauf et al [61]	Presents an approach to add role based security to CORBA; security settings are expressed using a XML-based description language.
	PROCESSES AND METHODS (15)
Davis et al [4]	Discusses and makes recommendations on processes for producing secure software.
Yu et al [5]	Proposes a formal approach for designing secure software architectures.
Beznosov & Kruchten [13]	Examines mismatches between security assurance techniques and agile development methods and proposes resolutions.
Kocher et al [14]	Introduces the challenges involved in designing secure embedded systems; surveys solutions to challenges.
Ravi et al [17]	Introduces the challenges involved in designing secure embedded systems, discusses recent advances in solutions, and identifies opportunities for future research. (more detailed version of Kocher et al (2004)).
Zdancewic et al [19]	On secure program partitioning, a language-based approach for protecting confidential data during computation in distributed systems with mutually untrusted hosts.
Jürjens [21]	Proposes an approach for developing secure software using an extension of UML called UMLsec.
Devanbu & Stubblebine [22]	Lists a number of research challenges in integrating security with software engineering and suggests solutions to the challenges.
Flechais et al [31]	Presents AEGIS, a secure software engineering method that integrates asset identification, risk and threat analysis, and context of use.
Deubler et al [34]	Proposes an approach for facilitating the development of security-critical service based-software using a tool called AutoFocus, based on the formal method Focus.
Sharma & Trevedi [43]	Proposes an architecture based unified hierarchical model for predicting software reliability, performance, security, and cache behaviour.
Viega et al [57]	Considers and explores trust assumptions during every stage of software development.
Yu et al [59]	Proposes a formal aspect-oriented approach to designing secure software architectures.
Moriconi et al [62]	Describes an approach to secure software design in which the software architecture is described formally and desired security properties proven for it.
Harrison & Hook [63]	For constructing secure software, advocates controlling information flow and maintaining integrity using monadic encapsulation of effects.
	CODING METHODS (2)
Peine [12]	Outline of a tutorial on rules of thumb for coding secure software; the rules are listed.
Chinchani et al [66]	Observes that software vulnerabilities may arise due to the syntax and grammar of a programming language.
	VULNERABILITIES IDENTIFICATION (5)
Tevis & Hamilton [8]	On software vulnerabilities, code vulnerability auditing tools, and functional programming as possibly better at ensuring security.
Kemmerer [11]	Identifies known threats and analyzes protection techniques for countering the threats, also mentions principles for designing secure software.
Hangal & Lam [18]	Describes DIDUCE, a tool for detecting complex program errors and identifying their root causes, using program instrumentation.
Viega et al [32]	Describes ITS4, a tool for statically scanning C and C++ code for security vulnerabilities.
Salter et al [38]	Presents a method for enumerating the vulnerabilities of a system and determining what countermeasures can best close those vulnerabilities.
	USABILITY (2)
Zurko & Simon [45]	Discusses the need for user-friendly security and develops three categories for work in this area.
Smetters & Grinter [52]	Proposes the need to design usable and useful systems as opposed to just improving usability.
	TESTING (3)
Thompson et al [15]	Proposes the necessity of testing for security failures in hostile environments together with a black box approach for such testing.
Ray [24]	Presents "Security Check", a model level technique that exercises small units of a system and then model checks them. This avoids the complexity of the whole system.

Jiwnani & Zelkowitz [65]	Proposes a test strategy based on a classification of vulnerabilities that allows prioritization of testing effort based on the impact the vulnerabilities have on the system.
	TOOLS (2)
Viega et al [32]	Describes ITS4, a tool for statically scanning C and C++ code for security vulnerabilities.
Gilliam et al [67]	Discusses a set of tools that offers a formal approach for engineering network security into software systems and applications throughout development and maintenance.
	OTHER (4)
Devanbu & Stubblebine [22]	Lists a number of research challenges in integrating security with software engineering and suggests solutions to the challenges.
Wang & Wang [26]	Presents a taxonomy of security considerations as they relate to software quality; considers the effectiveness of various security technologies.
Blakley [40]	Argues that the traditional model of computer security is no longer viable and that new definitions of the security problem are needed.
Shah & Kesan [41]	Argues that an important source of values in software is the institution in which it is developed; this impacts software security among other qualities.

Table 2 shows the papers retrieved for application security. The format of Table 2 is the same as Table 1.

Table 2. Research in application security

	THREAT IDENTIFICATION (2)
Kemmerer [11]	Identifies known threats and analyzes protection techniques for countering the threats.
Salter et al [38]	Presents a method for enumerating the vulnerabilities of a system and determining what countermeasures can best close those vulnerabilities.
	PROTECTION FROM TAMPERING (8)
Colberg & Thomborson [7]	Considers the use of tamper-proofing, and obfuscation to protect software from a malicious host.
Zambreno et al [9]	Protection against software tampering using hardware/software co-design techniques via a FPGA.
Zhang et al [10]	Proposes a secret sharing based compiler solution to protect critical program data and achieve intrusion tolerance.
Huang et al [16]	Describes protecting web applications using a combination of static analysis and runtime guards – describes a tool for achieving the protection.
Zambreno et al [20]	Protection against software tampering using hardware/software co-design techniques via a FPGA (more detailed version of Zambreno et al [9]).
Zhuang et al [29]	Presents a hardware assisted obfuscation technique that can dynamically obfuscate control flow information.
Monden et al [49]	Proposes a framework for obfuscating the program interpretation instead of the program itself.
Platte & Naroska [55]	Presents a combined hardware/software architecture to provide a secure and tamper resistant computing environment.
	PROTECTION FROM COPYING (3)
Colberg & Thomborson [7]	Considers the use of watermarking to protect software from a malicious host.
Zhang & Gupta [36]	Describes an approach for preventing the creation of unauthorized copies of software by splitting modules into open and hidden components.
Curran et al [54]	Investigates a new software watermarking scheme for securing Java from software pirating.
	OTHER PROTECTION (8)
Stytz [6]	Advocates a defense-in-depth strategy to protect applications from threats.
Castro & Liskov [25]	Describes the BFT algorithm for building highly available systems that tolerate Byzantine faults.
Kihlstrom et al [27]	Describes the SecureRing message delivery protocol that can be used for secure, reliable communication in distributed systems.
Zhang et al [28]	Proposes a new mechanism for protecting user privacy on trusted processors.
Covington et al [33]	Proposes the use of environment roles to capture security relevant context for access control.
Devenbu et al [35]	Proposes the use of trusted hardware in combination with a key management infrastructure for trusted hosting of applications.
Cowan & Pu [56]	Presents a categorization scheme for security bug tolerance techniques and populates it with techniques from the authors and the literature.
Kojima et al [64]	Describes a mechanism that prevents abuse of trusted Java applets.
	INTEGRITY VERIFICATION (6)
Spinellis [39]	Addresses software integrity verification; proposes the use of reflection, whereby the software examines its own operation in conjunction with cryptographic hashes.
Kirovski et al [46]	Presents SPEF, a combination of architectural and compilation techniques that ensures software integrity at runtime (prevent execution of unauthorized code).
Sadeghi & Stüble [47]	Points out the deficiencies of platform integrity verification and qualities binding as proposed in the existing specification of the Trusted Computing Group and proposes a new approach.
Fong [48]	Describes link-time bytecode verification as a pluggable service for the JVM.

Sekar et al [50]	Presents an approach called "model-carrying code" for safe execution of untrusted code (the model is a concise high-level representation of the code's security behavior).
Arora et al [53]	Presents an architecture for hardware-assisted runtime monitoring to enforce permissible program behavior.
	CHALLENGES (2)
Devanbu & Stubblebine [22]	Lists a number of research challenges in integrating security with software engineering and suggests solutions to the challenges.
Blakley [40]	Argues that the traditional model of computer security is no longer viable and that new definitions of the security problem are needed.

3. Discussion of Recent Research

The above results show that for software security, the categories with number of papers from high to low are in the order: processes and methods (15), requirements (9), vulnerabilities identification (5), other (4), testing (3), coding methods (2), usability (2), and tools (2). In other words, researches have worked mostly on processes and methods for building secure software, followed by expressing security as requirements, followed by techniques for identifying vulnerabilities. Areas such as testing, coding methods, usability, and tools appear relatively under-represented. Figure 1 shows this graphically.

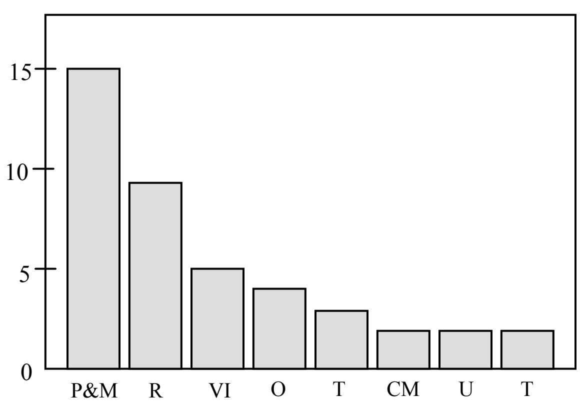

Figure 1. Distribution of research in software security (first "T" is "Testing")

For application security, the categories with number of papers from high to low are in the order: protection from tampering (8), other protection (8), integrity verification (6), protection from copying (3), threat identification (2), and challenges (2). In other words, researchers have worked mostly on protection from tampering and other protection, followed by integrity verification, with protection from copying and threat identification relatively under-represented. Figure 2 shows this graphically.

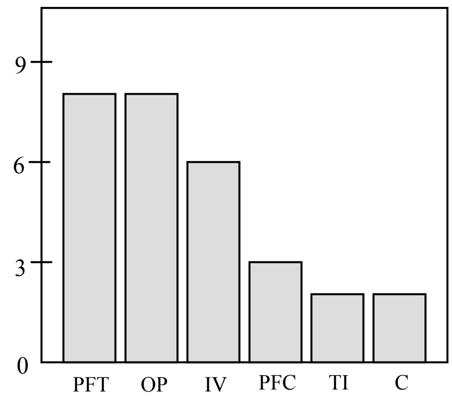

Figure 2. Distribution of research in application security

New researchers to the secure software field can make use of these results to select a research topic within either software security or application security. Assuming that the priorities of selection from most important to least important are: personal interest, utility from research, and relatively unexplored topic, the new researcher can peruse the summaries in Table 1 or Table 2 looking for areas of interest and then think of how this interest can be transformed to one of high research utility located in a relatively unexplored area and possibly in a related but new category (since the categories mentioned are not exhaustive) not mentioned here (which would be very unexplored).

4. Conclusions

This paper has provided an overview of recent research in the field of secure software, specifically in the subfields of software security and application security. Readers of this paper can benefit by: i) seeing a quick picture of what research has been carried out in the last ten years, ii) getting an introduction to what secure software is all about, iii) using the results to zero in on a potential secure software research topic for investigation.

Although the above benefits are put forward, it must be noted that they are tempered by the limitations of this work as mentioned in Section 2. Further, there is the assumption that papers found in the stated ACM and IEEE databases are representative of research in secure software throughout the world. Finally, the categories used to classify the papers were based on the papers' subjects.

Therefore, there could be other categories not mentioned above, with no matching papers. Thus it is important when zeroing in on a research topic, not only to think of the above categories, but to also try to think of other areas outside the above categories.

5. References

[1] J. Millar, "Computer Crime Skyrocketing", The London Free Press, available as of Sept. 11, 2005 at: http://www.canoe.ca/NewsStand/LondonFreePress/Business/2005/09/08/1206625-sun.html

[2] C.E. Landwehr, "Formal Models of Computer Security", ACM Computing Surveys, Vol. 13, No. 3, September 1981.

[3] G. McGraw, "Building secure software: better than protecting bad software", IEEE Software, Volume 19, Issue 6, Nov.-Dec. 2002.

[4] N. Davis, W. Humphrey, S.T. Redwine Jr., G. Zibulski, G. McGraw, "Processes for producing secure software", IEEE Security & Privacy Magazine, Vol. 2 , Issue 3 , May-June 2004.

[5] H. Yu, X. He, Y. Deng, L. Mo, "A formal approach to designing secure software architectures", Proceedings, Eighth IEEE International Symposium on High Assurance Systems Engineering, 25-26 March 2004.

[6] M.R. Stytz, "Considering defense in depth for software applications", IEEE Security & Privacy Magazine, Vol. 2 , Issue 1, Jan.-Feb. 2004.

[7] C.S. Collberg, C. Thomborson, "Watermarking, Tamper-Proofing, and Obfuscation – Tools for Software Protection", IEEE Transactions on Software Engineering, Vol. 28, No. 6, June 2002.

[8] J. J. Tevis, J.A. Hamilton, "Methods for the prevention, detection and removal of software security vulnerabilities", Proceedings of the 42nd Annual Southeast Regional Conference, April 2004.

[9] J. Zambreno, A. Choudhary, R. Simha, B. Narahari, "Flexible Software Protection Using Hardware/Software Codesign Techniques", Proceedings of the Conference on Design, Automation and Test in Europe - Volume 1, February 2004.

[10] T. Zhang, X. Zhuang, S. Pande, "Building Intrusion-Tolerant Secure Software", Proceedings of the International Symposium on Code Generation and Optimization (CGO '05), March 2005.

[11] R. A. Kemmerer, "Cybersecurity", Proceedings of the 25th International Conference on Software Engineering, May 2003.

[12] H. Peine, "Rules of Thumb for Secure Software Engineering", Proceedings of the 27th International Conference on Software Engineering, May 2005.

[13] K. Beznosov, P. Kruchten, "Towards agile security assurance", Proceedings of the 2004 Workshop on New Security Paradigms, September 2004.

[14] P. Kocher, R. Lee, G. McGraw, A. Raghunathan, and S. Ravi, "Security as a new dimension in embedded system design", Proceedings of the 41st Annual Conference on Design Automation, June 2004.

[15] H. H. Thompson, J. A. Whittaker, F. E. Mottay, "Software security vulnerability testing in hostile environments", Proceedings of the 2002 ACM Symposium on Applied Computing, March 2002.

[16] Y. Huang, F. Yu, C. Hang, C. Tsai, D. Lee, S. Kuo, "Securing web application code by static analysis and runtime protection", Proceedings of the 13th International Conference on World Wide Web, May 2004.

[17] S. Ravi, A. Raghunathan, P. Kocher, S. Hattangady, "Security in embedded systems: Design challenges", ACM Transactions on Embedded Computing Systems (TECS), Volume 3, Issue 3, August 2004.

[18] S. Hangal, M. S. Lam, "Dynamic program analysis: Tracking down software bugs using automatic anomaly detection", Proceedings of the 24th International Conference on Software Engineering, May 2002.

[19] S. Zdancewic, L. Zheng, N. Nystrom, A. C. Myers, "Secure program partitioning", ACM Transactions on Computer Systems (TOCS), Volume 20, Issue 3, August 2002.

[20] J. Zambreno, A. Choudhary, R. Simha, B. Narahari, N. Memon, "SAFE-OPS: An approach to embedded software security", ACM Transactions on Embedded Computing Systems (TECS), Volume 4, Issue 1, February 2005.

[21] J. Jürjens, "Using UMLsec and goal trees for secure systems development", Proceedings of the 2002 ACM Symposium on Applied Computing, March 2002.

[22] P. T. Devanbu, S. Stubblebine, "Software engineering for security: a roadmap", Proceedings of the Conference on The Future of Software Engineering, May 2000.

[23] C. Del Grosso, G. Antoniol, M. Di Penta, P. Galinier, E. Merlo, "Improving network applications security: a new heuristic to generate stress testing data", Proceedings of the 2005 Conference on Genetic and Evolutionary Computation (GECCO '05), June 2005.

[24] A. Ray, "Security check: a formal yet practical framework for secure software architecture", Proceedings of the 2003 Workshop on New Security Paradigms, August 2003.

[25] M. Castro, B. Liskov, "Practical byzantine fault tolerance and proactive recovery", ACM Transactions on Computer Systems (TOCS), Volume 20, Issue 4, November 2002.

[26] H.Wang, C. Wang, "Taxonomy of security considerations and software quality", Communications of the ACM, Volume 46, Issue 6, June 2003.

[27] K. P. Kihlstrom, L. E. Moser, P. M. Melliar-Smith, “The SecureRing group communication system”, ACM Transactions on Information and System Security (TISSEC), Volume 4, November 2001.

[28] Y. Zhang, J. Yang, Y. Lin, L. Gao, “Architectural support for protecting user privacy on trusted processors”, ACM SIGARCH Computer Architecture News, Volume 33, Issue 1, March 2005.

[29] X. Zhuang, T. Zhang, H. S. Lee, S. Pande, “Hardware assisted control flow obfuscation for embedded processors”, Proceedings of the 2004 International Conference on Compilers, Architecture, and Synthesis for Embedded Systems, September 2004.

[30] C. B. Haley, R. C. Laney, B. Nuseibeh, “Deriving security requirements from crosscutting threat descriptions”, Proceedings of the 3rd International Conference on Aspect-oriented Software Development, March 2004.

[31] Flechais, M. A. Sasse, S. M. V. Hailes, “Bringing security home: a process for developing secure and usable systems”, Proceedings of the 2003 Workshop on New Security Paradigms, August 2003.

[32] J. Viega, J. T. Bloch, T. Kohno, G. McGraw, “Token-based scanning of source code for security problems”, ACM Transactions on Information and System Security (TISSEC), Volume 5, Issue 3, August 2002.

[33] M. J. Covington, W. Long, S. Srinivasan, A. K. Dev, M. Ahamad, G. D. Abowd, “Securing context-aware applications using environment roles”, Proceedings of the Sixth ACM Symposium on Access Control Models and Technologies”, May 2001.

[34] Deubler, J. Grünbauer, J. Jürjens, G. Wimmel, “Sound development of secure service-based systems”, Proceedings of the 2nd international conference on Service oriented computing”, November 2004.

[35] P. T. Devanbu, P. W-L Fong, S. G. Stubblebine, “Techniques for trusted software engineering”, Proceedings of the 20th International Conference on Software Engineering, April 1998.

[36] X. Zhang, R. Gupta, “Hiding program slices for software security”, Proceedings of the International Symposium on Code Generation and Optimization: Feedback-directed and Runtime Optimization (CGO '03), March 2003.

[37] M. Koch, F. Parisi-Presicce, “Formal access control analysis in the software development process”, Proceedings of the 2003 ACM workshop on Formal methods in security engineering, October 2003.

[38] C. Salter, O. S. Saydjari, B. Schneier, J. Wallner, “Toward a secure system engineering methodolgy”, Proceedings of the 1998 Workshop on New Security Paradigms, January 1998.

[39] D. Spinellis, “Reflection as a mechanism for software integrity verification”, ACM Transactions on Information and System Security (TISSEC), Volume 3, Issue 1, February 2000.

[40] B. Blakley, “The Emperor's old armor”, Proceedings of the 1996 Workshop on New Security Paradigms, September 1996.

[41] R. C. Shah, J. P. Kesan, “Nurturing software”, Communications of the ACM, Volume 48, Issue 9, September 2005.

[42] D. M. Kienzle, W. A. Wulf, “A practical approach to security assessment”, Proceedings of the 1997 Workshop on New Security Paradigms, January 1998.

[43] V. S. Sharma, K. S. Trivedi, “Architecture based analysis of performance, reliability and security of software systems”, Proceedings of the 5th International Workshop on Software and Performance (WOSP '05), July 2005.

[44] M. Vetterling, G. Wimmel, A. Wisspeintner, “Requirements analysis: Secure systems development based on the common criteria: the PalME project”, ACM SIGSOFT Software Engineering Notes, Volume 27, Issue 6, November 2002.

[45] M. E. Zurko, R. T. Simon, “User-centered security”, Proceedings of the 1996 Workshop on New Security Paradigms, September 1996.

[46] D. Kirovski, M. Drinić, M. Potkonjak, “Enabling trusted software integrity”, Proceedings of the 10th International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 37 , 30 , 36, Issue 10 , 5 , 5, October 2002.

[47] A. Sadeghi, C. Stüble, “Property-based attestation for computing platforms: caring about properties, not mechanisms”, Proceedings of the 2004 Workshop on New Security Paradigms, September 2004.

[48] P. W. L. Fong, “Pluggable verification modules: an extensible protection mechanism for the JVM”, Proceedings of the 19th annual ACM SIGPLAN Conference on Object-oriented programming, systems, languages, and applications, Volume 39 Issue 10, October 2004.

[49] A. Monden, A. Monsifrot, C. Thomborson, “A framework for obfuscated interpretation”, Proceedings of the Second Workshop on Australasian Information Security, Data Mining and Web Intelligence, and Software Internationalisation - Volume 32 CRPIT '04, January 2004.

[50] R. Sekar, V.N. Venkatakrishnan, S. Basu, S. Bhatkar, D. C. DuVarney, “Model-carrying code: a practical approach for safe execution of untrusted applications”, Proceedings of the Nineteenth ACM Symposium on Operating systems Principles, October 2003.

[51] T. Doan, S. Demurjian, T. C. Ting, A. Ketterl, “MAC and UML for secure software design”, Proceedings of the 2004 ACM Workshop on Formal Methods in Security Engineering, October 2004.

[52] D. K. Smetters, R. E. Grinter, “Moving from the design of usable security technologies to the design of useful secure applications”, Proceedings of the 2002 Workshop on New Security Paradigms, September 2002.

[53] D. Arora, S. Ravi, A. Raghunathan, N. K. Jha, “Secure Embedded Processing through Hardware-Assisted Run-Time Monitoring”, Proceedings of the Conference on Design, Automation and Test in Europe - Volume 1, March 2005.

[54] D. Curran, N. J. Hurley, M. Ó Cinnéide, “Securing Java through software watermarking”, Proceedings of the 2nd International Conference on Principles and Practice of Programming in Java PPPJ '03, June 2003.

[55] J. Platte, E. Naroska, “A combined hardware and software architecture for secure computing”, Proceedings of the 2nd Conference on Computing Frontiers, May 2005.

[56] C. Cowan, C. Pu, “Death, taxes, and imperfect software: surviving the inevitable”, Proceedings of the 1998 Workshop on New Security Paradigms, January 1998.

[57] J. Viega, T. Kohno, B. Potter, “Trust (and mistrust) in secure applications”, Communications of the ACM, Volume 44, Issue 2, February 2001.

[58] J.J. Pauli, D. Xu,, “Misuse case-based design and analysis of secure software architecture”, Proceedings, International Conference on Information Technology: Coding and Computing (ITCC 2005), Volume 2, April 2005.

[59] H. Yu, D. Liu, X. He, L. Yang, S. Gao, “Secure Software Architectures Design by Aspect Orientation”, Proceedings,
10th IEEE International Conference on Engineering of Complex Computer Systems (ICECCS 2005), June 2005.

[60] K. Alghathbar, D. Wijesekera, “Analyzing information flow control policies in requirements engineering”, Proceedings, Fifth IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY 2004), June 2004.

[61] M. Hauf, J. Schwarz, A. Polze, “Role-based security for configurable distributed control systems”, Proceedings, Sixth International Workshop on Object-Oriented Real-Time Dependable Systems, January 2001.

[62] M. Moriconi, X. Qian, R.A. Riemenschneider, L. Gong, “Secure software architectures”, Proceedings, IEEE Symposium on Security and Privacy, May 1997.

[63] W.L. Harrison, J. Hook, “Achieving Information Flow Security through Precise Control of Effects”, Proceedings, 18th IEEE Workshop on Computer Security Foundations (CSFW-18 2005), June 2005.

[64] H. Kojima, I. Morikawa, Y. Nakayama, Y. Yamaoka, “Cozilet: transparent encapsulation to prevent abuse of trusted applets”, Proceedings, 20th Annual Computer Security Applications Conference, Dec. 2004.

[65] K. Jiwnani, M. Zelkowitz, “Maintaining software with a security perspective”, Proceedings, International Conference on Software Maintenance, Oct. 2002.

[66] R. Chinchani, A. Iyer, B. Jayaraman, S. Upadhyaya, “Insecure programming: how culpable is a language's syntax?”, Proceedings, IEEE Systems, Man and Cybernetics Society Information Assurance Workshop, June 2003.

[67] D.P. Gilliam, J.C. Kelly, J.D. Powell, M. Bishop, “Development of a software security assessment instrument to reduce software security risk”, Proceedings, Tenth IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET ICE 2001), June 2001.

[1] NRC Paper No.: NRC 48478; ERB-1134