Linköping Studies in Science and Technology
Dissertation No. 1075

Usable Security Policies for Runtime Environments

by

Almut Herzog

Linköping University
INSTITUTE OF TECHNOLOGY

Department of Computer and Information Science
Linköpings universitet
SE-581 83 Linköping, Sweden

Linköping 2007

ISSN 0345-7524
ISBN 978-91-85715-65-7

Printed by LiU-Tryck, Linköping 2007

“Humans are incapable of securely storing high-quality cryptographic keys, and they have unacceptably slow speed and accuracy when performing cryptographic operations. (They are also large, expensive to maintain, difficult to manage, and they pollute the environment. It is astonishing that these devices continue to be manufactured and deployed. But they are sufficiently pervasive that we must design our protocols around their limitations.)”

Charlie Kaufman, Radia Perlman, Mike Speciner. *Network Security: Private Communications in a Public World*. 2nd Edition. Prentice-Hall. 2002. p. 237.

Abstract

The runtime environments provided by application-level virtual machines such as the Java Virtual Machine or the .NET Common Language Runtime are attractive for Internet application providers because the applications can be deployed on any platform that supports the target virtual machine. With Internet applications, organisations as well as end users face the risk of viruses, trojans, and denial of service attacks. Virtual machine providers are aware of these Internet security risks and provide, for example, runtime monitoring of untrusted code and access control to sensitive resources.

Our work addresses two important security issues in runtime environments. The first issue concerns resource or release control. While many virtual machines provide runtime access control to resources, they do not provide any means of limiting the use of a resource once access is granted; they do not provide so-called resource control. We have addressed the issue of resource control in the example of the Java Virtual Machine. In contrast to others' work, our solution builds on an enhancement to the existing security architecture. We demonstrate that resource control permissions for Java-mediated resources can be integrated into the regular Java security architecture, thus leading to a clean design and a single external security policy.

The second issue that we address is the usability and security of the set-up of security policies for runtime environments. Access control decisions are based on external configuration files, the security policy, which must be set up by the end user. This set-up is security-critical but also complicated and error-prone for a lay end user and supportive, usable tools are so far missing. After one of our usability studies signalled that offline editing of the configuration file is inefficient and difficult for end users, we conducted a usability study of personal firewalls to identify usable ways of setting up a security policy at runtime. An analysis of general user help techniques together with the results from the two previous studies resulted in a proposal of design guidelines for applications that need to set up a security policy. Our guidelines have been used for the design and implementation of the tool JPerM that sets the Java security policy at runtime. JPerM evaluated positively in a usability study and supports the validity of our design guidelines.

Acknowledgements

Many people have contributed to this thesis in many different ways and I am grateful to all of them. First and foremost, I gladly acknowledge my debt to my supervisor Professor Nahid Shahmehri, who has supported me during my graduate studies with her advice, inspiration, valuable discussions and patience. Thanks also to my co-supervisor and critical reader of my publications, Doctor Germano Caronni, for providing helpful and clarifying input. I would also like to thank Brittany Shahmehri for her prompt and thorough proof-reading. Thanks to all study users for their interesting comments.

This work would not have taken shape without the support and time of all fantastic present and former graduate students and staff at the department of computer and information science. Many thanks to Claudiu, Patrick, Lena, Lin, Cécile, Johan, Juha, Nanu, Eduard, Dennis, Slava, Ross, Vaida, Martin, Pawel, He, David, Shanai, Christoph, Tommy E. and evening guest Jakob. It has been a long journey and you have all helped me proceed. Thanks to Ola, Tommy P. and the Latex community at IDA for answering all my Latex questions. A big hug to the department technical staff, who have saved many a file from backup, answered peculiar helpdesk questions and have always assisted me in a timely and most helpful fashion with database installations, cable lending, Linux and Vista access, software tips, real-world insights and much more. And thanks to family and friends who help me in the background at all times.

The financial support of my research by ECSEL (Excellence Center in Computer Science and Systems Engineering in Linköping) and VINNOVA (Swedish Agency for Innovation Systems) is gratefully acknowledged.

Almut Herzog
Linköping, April 2007

List of Enclosed Papers

This thesis contains revised versions of the following papers:

1. Almut Herzog and Nahid Shahmehri. An evaluation of Java application containers according to security requirements. In *Proceedings of the 14th IEEE International Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises (WETICE'05)*, pages 178–183. IEEE, June 2005.

2. Almut Herzog and Nahid Shahmehri. Performance of the Java Security Manager. *Computers and Security*, 24(3):192–207, May 2005.

3. Almut Herzog and Nahid Shahmehri. Using the Java sandbox for resource control. In *Proceedings of the 7th Nordic Workshop on Secure IT Systems (NordSec'02)*, pages 135–147. November 2002.

4. Almut Herzog and Nahid Shahmehri. A usability study of security policy management. In Simone Fischer-Hübner, Kai Rannenberg, Louise Yngström and Stefan Lindskog, editors, *Security and Privacy in Dynamic Environments, Proceedings of the 21st International Information Security Conference (IFIP TC-11) (SEC'06)*, pages 296–306. Springer-Verlag, May 2006.

5. Almut Herzog and Nahid Shahmehri. User help techniques for usable security. In *Proceedings of the 1st Symposium on Computer Human Interaction for Management of Information Technology (CHIMIT'07)*. ACM Press, March 2007.

6. Almut Herzog and Nahid Shahmehri. Usability and security of personal firewalls. Accepted for publication in the *Proceedings of the 22nd International Information Security Conference (IFIP TC-11) (SEC'07)*. Springer-Verlag, May 2007.

7. Almut Herzog and Nahid Shahmehri. Usable set-up of runtime security policies. Accepted for publication in the *Proceedings of the International Symposium on Human Aspects of Information Security and Assurance*. July 2007.

8. Almut Herzog, Nahid Shahmehri and Claudiu Duma. An ontology of information security. Accepted for publication in the *International Journal of Information Security and Privacy*. 2007.

In addition to the publications enclosed in this thesis, the following work has been published.

- Claudiu Duma, Almut Herzog and Nahid Shahmehri. Privacy in the semantic Web: What policy languages have to offer. Accepted for publication in the *Proceedings of the International Workshop on Policies for Distributed Systems and Networks (Policy'07)*. IEEE. June 2007.

- Almut Herzog and Nahid Shahmehri. Problems running untrusted services as Java threads. In *Proceedings of the 2nd IFIP TC-11 Second International Workshop on Certification and Security in Inter-Organizational E-Services (CSES'04), World Computer Congress (WCC'04)*, pages 19–32. Springer-Verlag. August 2004.

- Kerstin Roback and Almut Herzog. Home informatics in healthcare: Assessment guidelines to keep up quality of care and avoid adverse effects. *Technology and Health Care* 11(3):195–206. 2003.

- Almut Herzog, Leili Lind. Network solutions for home health care applications. *Technology and Health Care* 11(2):77–87. 2003.

- Almut Herzog. *Secure execution environment for Java electronic services*. Licentiate Thesis No. 991. Dept. of Computer and Information Science, Linköpings universitet. December 2002.

- Almut Herzog and Nahid Shahmehri. Towards secure e-services: risk analysis of a home automation service. In *Proceedings of the 6th Nordic Workshop on Secure IT Systems (NordSec'01)*, pages 18–26. November 2001.

- Claudiu Duma, Almut Herzog and Nahid Shahmehri. Towards secure e-services. In *Proceedings of the 9th International Workshop on Enterprise Security (WETICE'00)*, pages: 221–222. IEEE. 2000.

Contents

1 Motivation **15**

2 Problem Description **17**

3 Research questions, objectives, methodology **19**

4 Contributions **20**

5 Paper Summaries **23**

6 Related Work **24**
6.1 Resource control 25
6.2 Usable security 26
6.2.1 User assistance for setting up a security policy 28

7 Limitations **30**

8 Future directions **31**

9 Concluding remarks **32**

Introduction

1 Motivation

Application-level virtual machines provide a runtime environment for programs written in an intermediate code representation. The virtual machine takes the intermediate code representation as input and executes that code on a native platform. The idea of virtual machines for applications dates back to the 1960's and the OCODE machine for the language BCPL [17]. But with the advent of the World Wide Web these high-level runtime environments and their associated programming languages—residing between compiled and interpreted languages—have received a boost.

One of the things that makes high-level virtual machines interesting is the ease of porting a program to different platforms. This is even more true today than it was when virtual machines were originally developed [100]. Web service providers create cross-platform code that can be downloaded by web site visitors and run in any number of client environments to offer advanced, interactive content. In the web scenario, programs must be able to execute on as many client machines as possible with as little development and deployment cost as possible. Even though web clients running applets were the snowball that started the avalanche of web services [113, p. 243], end user machines are not the only target platform for virtual machines. Glue software, using virtual machine technology, is used on all web server platforms to connect web interfaces with business logic. For providers of such software, platform-independence is important in order to support many different architectures, operating systems, web servers and web applications—hence the need for cross-platform languages and their virtual machine architecture. In recent years, many virtual machines have been specifically developed for use in an Internet setting. Virtual machines run Java applets or Java Web Start applications, ASP.NET web applications written in C# or glue applications written in Python or Perl to connect html-code to e.g. back-end databases.

In these Internet scenarios, security requirements play an important role [38]. With applets and Web Start applications that execute on the browser machine, security is of paramount importance to protect security-unaware end users. CGI (common gateway interface) scripts and web applications that execute solely on the web server must also be secure so as not to create a vulnerability that a hacker can exploit to get access to back-end data, to subvert client connections or to disrupt the web server. In addition, some web servers—notably web hotels but also application servers—run web applications from multiple, possibly untrusted parties, and therefore require isolation for these web applications.

All of the current virtual machines have responded to some extent to the security needs of web applications. Python used to have a restricted execution and bastion mode [68, 71], where the untrusted code executed in an environment customized—i.e. programmed by subclassing a standard sandbox—by the code deployer. Because this execution mode was easy to circumvent with the introduction of new style classes it was deprecated.

Perl 6 and the Parrot Virtual Machine [89] support input security through ‘taint mode’, which must be switched on for regular programs and is automatically set for privileged setgid/setuid programs. Tainted data may not be used directly or indirectly in any command that invokes a sub-shell, nor in any command that modifies files, directories, or processes (with certain exceptions). Perl 5 already supports Safe Mode, where a low-level opcode mask defines which operator names are accessible to the module in safe mode. By default, all access to the system is denied in safe mode, but variables can be passed in explicitly.

The Java and .NET virtual machines share the same view on runtime security and have both achieved the same level of control, which is much more advanced than that of Python, Perl or other application-level virtual machines. Java provides a so-called sandbox [41, 75, 84], a reference monitor called the ‘Security Manager’, which provides a contained environment customized through a text-based policy file that denotes the permissions with which a piece of code is allowed to run. This sandbox was initially designed only to contain applets but, since Java 1.2, it can also be used to restrict local Java applications [40]. The runtime environment of the .NET framework, inspired by the security features of the Java Virtual Machine [8, 9, 31, 87], supports similar so-called Code Access Security which controls access to sensitive resources based on code properties and the security policy in effect.

The advanced runtime environments of Java and .NET require end users to configure the security policy that holds the allowed permissions for resource access. If this configuration is performed by a dedicated system administrator, it is the administrator’s job to do it properly and to check that the result is secure. However, if this security-critical configuration is meant to be done by security-unaware end users—as in the case with applets or Web Start applications—a prerequisite for a truly secure environment is that the security policy set-up is usable and effective. This is not a trivial task as end users are notoriously bad at making security decisions [21, 44, 45]: A security decision often requires the user to decide between performing a primary task insecurely or not at all [115]. Regardless of whether the task is work- or leisure-related, users tend to choose the primary task over security [92, 94]. When lay end users are confronted with a security warning or the need to interact with security features, they often see these security events as obstacles [101] and experience frustration, even resignation and futility [23]. These negative feelings are caused by the fact that end users want to get their primary task done, that they do not understand and have no patience for security jargon, and that the impact of an erroneous security decision can be so high [23, 66, 108]. This duality of security and usability is well-recognised and further explained in the related work section 6.2. One interesting approach to tackling usability and security requirements has been suggested by e.g. Smetters and Grinter [99] who ask “if you put usability first, how much security can you get?”

It is in this interesting area of efficient and usable runtime environments and runtime security policies that our work makes its contribution.

2 Problem Description

From the current virtual machines that provide security features, we chose to investigate one in greater detail and to use it as a platform for our implementations: the Java Virtual Machine. The Java Virtual Machine is ubiquitous, well-documented, open source and has reached a certain stability and maturity. The Java runtime environment is installed with every web browser, and the J2EE web application server has a market share of 35–40% in the web application business [79]. It is thus a much-used but also open platform that lends itself easily to extensions.

Our focus within the Java Virtual Machine runtime environment is the Java sandbox with its Security Manager. The Security Manager checks whether code is allowed to e.g. access a file or other sensitive resources. Our ontology of information security (paper 8, p. 201ff.) describes the Java sandbox as a countermeasure that provides access control and ensures policy compliance. The Java sandbox prevents the violation of the integrity and confidentiality of technical assets. Other countermeasures that provide access control are e.g. anti-virus software, firewalls or e-mail filters. Countermeasures that protect similar goals and assets are e.g. intrusion detection systems. The sandbox is configured through a policy file, a plain text file (see middle of figure 1), which contains the permissions granted to a given piece of code.

After a literature study and hands-on experience it became clear that the sandbox with the Security Manager is underspecified in two important ways:

Resource control The security architecture, by way of the Java Security Manager, provides *access control* for resources. This protects the confidentiality and integrity of resources. However, the Security Manager does not support

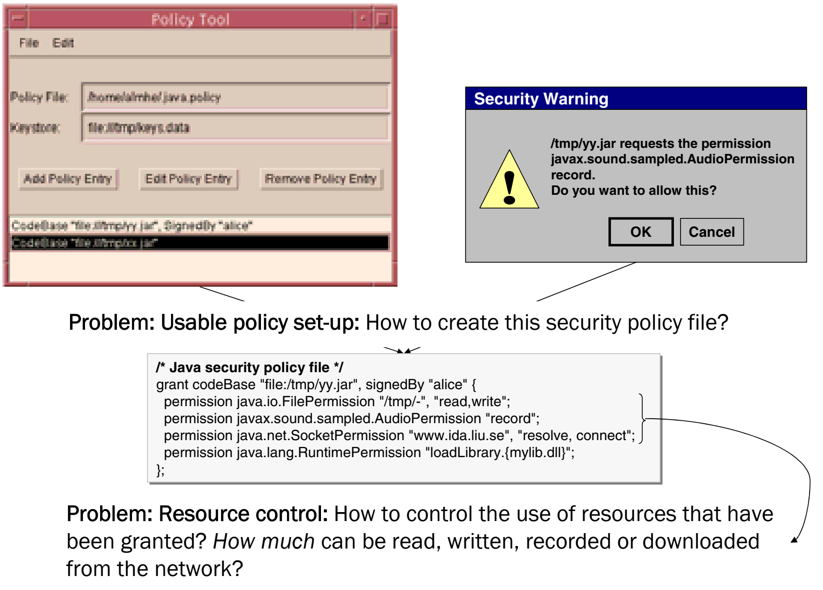

Figure 1: Two problematic areas with runtime security. How can an end user create a runtime security policy? Simple syntax support tools do not help the lay user. Can the runtime security policy be set-up such that it not only allows access control but also resource control? Resource control limits the extent to which a granted resource can be used.

release control or *resource control*[1]. The availability of resources remains unprotected: Once the Security Manager has concluded that a piece of code has access to a resource, that piece of code has complete access. There is no control of the extent to which the code is allowed to use a resource. Thus, while the Security Manager can enforce that an untrusted application can only write to a specific directory (see policy file in figure 1), there is no way to restrict how many files the untrusted application can create or how many bytes it can write to a file or how long it can keep a file open for writing. This shortcoming can result in denial-of-service (DoS) attacks that can make the virtual (and physical) execution environment unresponsive to legitimate requests and eventually

[1]For the remainder of the thesis we use the term *resource control* to denote access control *including* release control or quota management. The terms *resource control* and *release control* are used interchangeably. We use the term *access control* if only the initial access to a resource is controlled.

force a restart or other user intervention.

Usable policy set-up While the security policy is configurable through a so-called policy file, there is no user-friendly tool that enables an end user to set up such a policy. The existing policytool (see the upper left-hand side of figure 1) allows only offline set-up of a security policy. Offline setting is generally not possible, because the end user cannot know beforehand which permissions a piece of code may need. A simple runtime tool as suggested on the upper right-hand side of figure 1 is not suitable for a security lay user either: It is too technical and does not reflect that not all permissions are equally dangerous. A consequence of this lack of a usable tool is that the Security Manager is not used at all and that the runtime security features of Java are often switched off. This means that the sandbox is effectively disabled and the execution platform is open to all kinds of malware.

The same two issues also hold for the .NET Code Access Security. While our contribution in the area of resource control is specific to Java, our findings for usable policy set-up can also be applied to the Common Language Runtime.

3 Research questions, objectives, methodology

It is the aim of our research to investigate two specific security needs in runtime environments and to contribute with solutions for these. We address two major research questions:

1. How can resource control be achieved in a runtime environment?

2. How can lay end users be enabled to set up a runtime security policy?

The connection between these two seemingly unconnected objectives is that enhanced security features for runtime security (i.e. allowing resource control) must also be made usable to become a secure feature. If the security feature is not used it does not actually enhance runtime security.

Our goal is to answer these questions in the context of the Java Virtual Machine, but answers shall be as general as possible so that a solution for Java can also be applied to other application-level virtual machines.

One objective of our research is to achieve runtime resource control for the Java Virtual Machine. As related work in section 6.1 shows, previous research has addressed such resource control but not in the context of the existing runtime security architecture. We aim at a general solution that should fit seamlessly into the existing security architecture. Preferably the runtime monitor that manages access control should also be used for resource control. This would lead to a clean design with one monitoring component and one policy file for access and resource control.

The method of this first part is exploratory, in order to gain technical insights into the runtime environment of the Java virtual machine. One insight concerns performance. We must ensure that the proposed solution can achieve reasonable performance. First, the current runtime monitor shall be checked for performance penalties. If the penalty is low enough, an implementation of runtime resource control within the Security Manager shall be done.

The second objective is to make the set-up of runtime security policies so usable that not only security administrators but also lay end users can achieve high security in the runtime environment of, for example, their web browsers. The method for this second objective is to propose the hypothesis that existing tools are not sufficient. If this proves true, we use the method of usability engineering [28] with user studies to arrive at a better tool and verified guidelines for tools that set up a security policy.

Consequently, we set out to identify shortcomings in the existing tool for policy setup by means of a usability study. In order to gain further experience and insights in usability issues for setting up a security policy, we evaluate the usability and security of personal firewalls. Personal firewalls usually set up their rules (security policies) at runtime, with user interaction. The evaluation of their usability should supply indications as to which user interface techniques would be most appropriate for security tasks. In a parallel and more analytical approach, we survey literature on user help techniques in order to identify those techniques that could assist end users in their security decisions. Given this background we shall recommend guidelines and present an implementation of a tool that guides lay users through the set-up of a runtime security policy. The tool shall be evaluated by user studies in several stages: during the design phase as a paper prototype, during the implementation phase with early user feedback and with a final user evaluation. The early evaluations shall provide input for enhancements in the design and implementation process. The final evaluation shall show the extent to which our recommendations hold and to which we could obtain a practical answer for our second research question.

4 Contributions

Part I of this thesis introduces and specifies security in runtime environments and addresses the problem of missing resource control. This part consists of papers 1–3 and paper 8 as a reference and general overview over the area of information security. In the first paper, we set the scene and identify requirements for a secure runtime system for executing untrusted code by way of a risk analysis of runtime environments. A performance study of the Security Manager provides evidence that the performance penalty is low enough to build further on runtime security. Consequently, in an exploratory approach, we implement resource control in an integrated fashion by enhancing the Java Security Manager.

Part II, consisting of papers 4–7, deals with the problem of usable set-up of security policies for runtime environments. The connection to the first part is that part I proposes a solution for resource control that extends the syntax for Java policies with constructs that allow resource control policies. But how can users be enabled to actually set up such advanced security policies? This question dominates the second part of the thesis.

We start from the hypothesis that offline policy editing is neither suitable nor usable. In order to prove the hypothesis, we conducted a usability study of the existing tool for setting up Java permissions and received strong indications that offline editing was not enough for end users and that integrated runtime editing would be favoured (paper 4). This evaluation—together with our work on personal firewalls (paper 6) and general user help techniques (paper 5)—has led to the definition of design guidelines for applications that must set a security policy. The tool JPerM, the Java permission manager, for setting up Java permissions at runtime has been designed and implemented according to these guidelines. In a final user study (paper 7), JPerM proved to be superior in terms of efficiency, effectiveness and user satisfaction and thus supported the validity of our guidelines.

In a concise overview, the contributions of this thesis are

- an analysis of the security features of the Java runtime environment with regard to its effective security and with regard to its security effect as a general countermeasure to information security threats,
  - We derive security requirements for Java container software from a risk analysis and evaluate to which extent the requirements are fulfilled in a number of Java containers (paper 1, p. 47ff.).
  - The Security Manager, the policy enforcer, is not used often, which is often blamed on its alleged slowness. We show that the Security Manager need not impair performance by more than a few percent and provide a checklist on how to improve performance (paper 2, p. 63ff.).
  - Security of runtime environments, and specifically the Java sandbox, is only a small part of the domain of information security. We

provide a structured overview of the domain of information security by way of an ontology that places our research area in a greater context (paper 8, p. 201ff.).

- an extension of the Java runtime environment that provides resource control,

  - Release or resource control is often solved outside of the existing security mechanisms (see related work on resource control, section 6.1). We show that resource control for Java-mediated resources can be done within the Security Manager, which results in a cleaner security design with only one security policy configuration file (paper 3, p. 97ff.).

- an extension of the Java runtime environment that provides interactive or automated set-up of permissions for the code that is currently running (paper 7, p. 181ff.),

- an analysis and one solution for how to present security decisions to end users in a usable fashion.

  - The Java development toolkit from Sun contains a small graphical user interface tool for setting permissions. We evaluate the usability of this tool (paper 4, p. 117ff.).
  - We evaluate the usefulness of different help systems in a security context (paper 5, p. 139ff.).
  - We present a template that can help designers identify those help features that are suitable for their security application (paper 5).
  - Editing of a security policy is tedious but security-critical work. We show how personal firewalls have addressed this problem of letting individual users set up a security policy and draw general conclusions for successful policy set-up (paper 6, p. 163ff.).
  - We present and evaluate JPerM, an interactive application monitor that helps the user set a least-privilege security policy (paper 7, p. 181ff.).

5 Paper Summaries

This section provides short summaries of the papers that are included in this thesis and explains their connections to the research area and to each other.

Paper 1: An evaluation of Java application containers according to security requirements [52] There are a number of Java application containers—web browsers, web servers, Java application servers and OSGi frameworks—that load and execute Java code. By way of a risk analysis, we identify security requirements that the containers should meet and show the extent to which these requirements are fulfilled. In the evaluation, the container for Enterprise Java Beans scores best. It addresses the requirements partially through technical solutions but also by providing a written contract that disallows certain behaviour for the components it runs.

Paper 2: Performance of the Java Security Manager [51] The Java Security Manager, responsible for the Java runtime protection, is often not used because of its alleged performance penalty. If this was true, building solutions on top of the Security Manager would be difficult to promote. However, our measurements show that the performance penalty is often negligible and that there are certain measures that can be taken by developers and policy writers to make that penalty even lower.

Paper 3: Using the Java sandbox for resource control [50] One criticism of the Java sandbox is that it only addresses access control but not release control, i.e. once granted, access to a resource is unlimited. We show an implementation where resource control can be integrated into the standard Java sandbox. Due to an implementation on the Java level (rather than using native code), the performance penalty of our solution is high, but as a proof of concept the solution is interesting and shows the potential of the Java sandbox.

Paper 4: A usability study of security policy management [53] The previous two papers have shown the technical potential of the Java sandbox. However, users have a hard time using the sandbox because they do not receive adequate help in their task of setting up a security policy. We have evaluated the graphical user interface tool that is meant to assist the user and found a great number of usability problems that effectively prevent the user from setting up a tight security policy.

Paper 5: User help techniques for usable security [54] If one wants to assist the user in setting up a policy, be it for the Java sandbox or any other security application that relies on user-defined rules such as personal firewalls or intrusion detection systems, one can choose from the tool box of user help techniques. This tool box contains techniques such as online help, context-sensitive help, wizards, and even some specific help techniques for security applications. We compare these techniques according to which user questions they may answer and how well they can address criteria that make security applications usable. While our comparison is general, we provide a template that is meant to be instantiated by system designers for their specific applications.

Paper 6: Security and usability of personal firewalls [55] While the previous paper describes user help on a general level, this work evaluates 13 free and commercial personal firewalls according to their security and usability. With a personal firewall, it is the end user that has to make the security decision whether or not to allow an application to access the local host or the network. Here we show which user help features are used and which are relevant and helpful.

Paper 7: Usable set-up of runtime security policies [56] After having identified existing shortcomings and possible remedies, we have developed JPerM, a tool that aids the user in setting up a security policy for a Java application. JPerM prompts users for security decisions using meaningful, non-technical wording. JPerM uses colour-coding as a visual aid and assists the user with editing a security policy for the application. User tests show that it is superior to the existing policytool. It does not force users to understand Java policy syntax or security implications, and may even increase user awareness of security issues.

Paper 8: An ontology of information security [57] We describe the area of general information security by means of an ontology. The ontology focuses on the classic concepts of risk analysis—assets, threats, vulnerabilities and countermeasures—and their relations. The ontology is implemented in OWL (Web Ontology Language) and supports querying and searching. It contains general concepts such as 'access control' as well as specific, technical concepts such as 'Java sandbox', 'SSH' or 'Blowfish'. With the ontology at hand, the reader can find the area—'Java sandbox'—that our research addresses and see how it relates to other areas of information security. However, the ontology addresses neither resource control nor usability issues. This paper serves as a reference and is an independent piece of work.

6 Related Work

Java security is an active area of research. Especially in the early years of Java, with the more rigid sandbox of Java 1.1 suggestions abounded on how to ensure Java code security by containing applets [15, 30, 47, 72, 73], on providing more expressive policies [16, 27, 76, 85, 105], or by providing alternative sandbox solutions [13, 25, 67, 88, 102], often by using bytecode rewriting (runtime modification of Java system classes). In recent years, sandbox research has focused more on static analysis rather than the runtime security features [14, 70, 78, 98].

6.1 Resource control

It is well known that the Java security architecture provides access control but not release or resource control (e.g. [75, p. 43], [6, 42]). Naturally, this has led to a number of suggestions and implementations that attempt to remedy this shortcoming. This section gives the background for resource control solutions for Java.

A start was made by researchers within Sun [18] as early as 1998 by introducing the resource-accounting interface JRes. JRes resides outside of the existing security architecture and policies are written as Java code. In fact, it was this early work that led to the proposal of a multi-tasking virtual machine [20]. Eventually, this approach was the most accepted one. It evolved into the Isolate API (application programming interface) [63], a specification for isolating Java applications in the same Java environment from each other. In 2001, the approach was promoted to a Java Community Process, a process that aims at integrating a new API into the Java Development Kit. But although 'JSR 121: Application Isolation API Specification' [2] has been finalised and was meant to be integrated into Java 1.5 and then Java 1.6, it is still unclear if it will be part of Java 1.7.

Since 1998, many other publications have addressed resource control in the Java virtual machine. In 1998, Hashii et al. [46] provide an early overview of resource control issues and how they can be addressed in mobile code environments. Back et al. [2] achieve resource control by using processes, not threads, and operating system resource control for processes. Naccio [26] allows compile-time definition of resource control and other safety policies (as Java code). Q-JVM [86] provides resource control through a modified thread scheduler. The Aroma virtual machine [104] reimplements the Java specification and enhances it with support for resource control using low-level routines. J-SEAL2 [7] defines a new resource-aware Java kernel. In 2002, a workshop on resource control in safe languages attracted a number of publications [4, 11, 19, 43] for providing Java resource control, among them also the group from Sun, presenting their then early results from the Isolate API. The other approaches use bytecode rewriting [4, 11] and a changed virtual machine [43]. The paper by Calderon and Binder [11] evolved into the only current contender

[2] http://jcp.org/en/jsr/detail?id=121 (visited 15-Feb-2007)

of the Isolate API: Binder and Hulaas [5] use rewriting for resource accounting on the thread level.

In contrast to our approach, all of the above cited solutions solve resource control outside of the existing Java security architecture with the Security Manager. All of them introduce new hooks for resource control into Java code and do not consider use and augmentation of the existing policy enforcer, the Security Manager. While disregarding the Security Manager is reasonable for low-level resources (CPU, memory etc.) that are not represented as Java objects, the Security Manager should be the natural place to provide resource control for Java-managed objects such as files, network connections or the screen.

In addition, it is notable that none of the publications pay attention to the usability of the actual policies at deployment time or even to thorough descriptions of policy APIs or policy syntax specifications.

6.2 Usable security

The second part of this thesis deals with usability aspects of security. Assessing usability is usually done by user studies. Methods for such studies are described by Nielsen [81] and Nielsen and Mack [82]. Comparisons of different methods are provided in [48, 61, 77, 96], with Hornbæk and Frøkjær [58] providing an insight in problems that arise when usability studies lead to redesign proposals.

General and more specific guidelines exist for the design of usable applications, as shown in table 1 and further described in the section 'Guidelines for usable set-up of security policies' of Paper 7: Usable Set-up of Runtime Security Policies (p. 181ff.).

Early influential work on the importance of usability in security applications is provided by Whitten and Tygar [108], Zurko and Simon [114] and Smetters and Grinter [99]. Zurko and Simon as well as Smetters and Grinter raise and argue the issue. Much like Smetters and Grinter, Sandhu [91] argues for 'good-enough security' that balances security and usability. Whitten and Tygar have conducted the well-known 'Johnny'-study that shows how difficult lay users find the notion of public-key encryption in e-mail clients.

In the last few years usability evaluations of security applications or security features of general-purpose applications abound: Gerd tom Markotten [39] repeated the Johnny-study with similar results. Internet banking and the usability and security of user authentication is often evaluated (e.g. [49, 83]). Furnell and others have studied the usability of security features of Internet Explorer [32], Outlook Express [35] and Microsoft Word [33] as well as Wireless LAN set-up [34] and have found usability rather low. Our contribution is the eval-

Table 1: Overview of influential general and security-specific usability guidelines

General usability guidelines	Security-specific usability guidelines
• Nielsen [80]: 10 design slogans, • the ISO standard 9241 (parts 10–17) provides high-level as well as concrete suggestions and advice on usability issues, • Shneiderman and Plaisant [97]: eight golden rules of interface design.	• Johnston et al. [62]: six criteria for successful human-computer interaction in security applications, • Leveson [69]: 60 guidelines for safe human machine interaction design, • Yee [110]: 10 principles, • Garfinkel [36]: six principles.

uation of the Java policytool for setting up Java permissions (paper 4) and the evaluation of personal firewalls (paper 6). Lately, attention has shifted to the usability and security evaluation of countermeasures against phishing schemes [59, 60, 93, 109].

Apart from the guidelines shown above and heuristics that should be considered during application design, a number of methods have been proposed that could enhance usability in security applications. Fléchais [29] proposes a software design method that is based on a risk analysis of the software and is meant to mediate security risks (also through low usability) at an early stage. Yoder and Barcalow [112] and Schumacher and Roedig [95] have identified patterns that can enhance the security and usability of applications. Yee [111] advocates built-in security—security by designation—that mirrors the user's intent in a secure way. Safe staging [107] and social navigation [22] are two concrete design methods for making security issues more usable; they are further explained in paper 5 (p. 139ff.).

Attempts have been made to create usable security applications. Balfanz et al. [3] have simplified the set-up of certificate-based Wireless LAN configuration. Two approaches show that using encrypted e-mail can be made much easier [37, 90]. The work that is closest to ours is where researchers have tried to simplify security policy set-up. That literature is reviewed in the following section.

6.2.1 User assistance for setting up a security policy

A technical solution that inspired our work is JSEF [64], a Java framework that is primarily intended to allow more expressive, hierarchical, positive and negative XML-based permissions. As a by-product, JSEF can also negotiate, i.e. prompt for Java permissions at runtime. This part of JSEF is not designed with usability in mind. It presents every permission with a technical text and cannot be used if no display is associated with the Java server. However, it is an inspiring starting point.

Polaris (Principle of Least Authority for Real Internet Security) [103], an add-on to Windows XP developed by Hewlett Packard, is an operating-system sandbox for compartmentalising applications and thus defeating virus attacks. Each application, such as Web browsers or text editors, can be polarized. The polarized application, called pet, runs from an unprivileged user account and can only access its local copy of configuration files and files opened by a trusted dialog box. If the same application, e.g. a browser, is used for both trusted and untrusted resources, for example, Intranet vs. other web sites, at least two pets should be created. Thus the pet for the Intranet can save passwords, to which the pet for the external web sites has no access. The security policy in Polaris is set up explicitly at application installation time and implicitly by user designation when opening files or accessing web sites. Polaris has been evaluated in a usability study [21] which shows that users often forget to create a pet, do not realise that they should have created yet another pet or assume that Polaris would create a pet automatically when launching an application. It demonstrates again that security-by-default is a better solution than relying on end users to remember security issues (their secondary task) when they want to run an application (their primary task).

Brostoff et al. [10] have worked on the usability of PERMIS access control in an e-science setting where access control is used for sharing equipment, data and software among geographically distributed scientists. Their first attempt at a user interface for setting the access control policy failed, mostly due to technical jargon (‘SOA’, ‘RBAC’) in the user interface. Their second attempt was more successful but was only tested with a very small number of users. While the authors do not provide constructive advice to those facing similar usability problems, we present the findings from the JPerM studies in the form of general guidelines.

Maxion and Reeder [74] compare the user interface for file permission set-up of Windows XP with a tool, Salmon, that they have developed. In their design of Salmon they use a method from cognitive science called external subgoal support (ESS). ESS ensures that users make correct goal assumptions for the task they want/need to complete, and that it is straightforward to define and fulfil subgoals. However, the user must still show interest and commitment in defining and solving the subgoal. Such interest or commitment cannot be taken for granted when security is not a user's primary task.

In initial work, Cao and Iverson [12] have improved the setting of an access control policy for a web authoring system. The existing ACL (access control list) editor was too general. In one study task it gave the users a false sense of security: The users thought they had successfully solved the task and improved security, while this was not true. The proposed solution is implemented as a wizard that in a more task-oriented way guides the user to setting an ACL. Their early findings of important design principles have been incorporated into JPerM: (1) The user has access to essential information for making the decision (see also Whitten and Tygar [108]), (2) the system should be responsible for predicting and presenting such information when it can.

Our work builds on the user's ability to make an informed decision at the time when an untrusted application requests access to a sensitive resource. How we provide input to this informed decision is influenced by the following publications.

Hardee et al. [45] have put up guidelines for designing such dialogs. They advocate explicitly to stress the negative effects of granting an action. The user should be warned of the loss of time, money and privacy. The guidelines emerge from their previous study which has shown that users tend to underestimate the risks in a computer setting. Acquisti and Grossklags [1] present a study on factors that influence the security decision making process of users. Specifically, users will easily trade privacy and security for convenience. Whalen et al. [106] report early results from a user interview study about user habits for setting access control to files that shall be shared within an organisation. They conclude that access control set-up must be integrated into people's workflow to be used, that it must be convenient and must fit into social settings.

Our design guidelines for applications that must set a security policy—and the design of JPerM—have been influenced by these specific studies, implementations and their findings but also by more general usability publications as cited previously.

7 Limitations

This section shows ways in which our work could be improved to even better fulfil the stated objectives.

One of our contributions (paper 3) is to present a proof-of-concept implementation, using the Security Manager for resource control. However, the work is not a full-blown solution for the resource control problem. The syntax that we use for denoting resource-controlling policies such as "allow only the writing of 2MB to the file /tmp/a" is crude. A *better suitable syntax for resource control* should be chosen. One could, for example, evaluate, and, if possible, select one of the many policy languages of the Semantic Web. If none of the languages support resource control as such, one could possibly identify a language that can be extended with a resource control syntax. It is important that the chosen language is external—and not Java code—so that it can be set up by an end user. From our work on the comparison of policy languages [24] we suggest that Protune or XML-based Ponder might be suitable candidates.

Both our performance study of the regular Java Security Manager (paper 2) and our work on resource control (paper 3) show that the Java permission check is a certain (regular permissions) or even high (resource control permissions) performance bottleneck. *Caching of Java permissions* in a suitable structure could make a permission check faster than stack inspection and reduce the bottleneck. While caching may not improve performance for regular permissions much, it is important for resource control permissions because these are invoked more frequently than access control permission. Potentially, a file resource permission could be invoked for every byte written to the file.

For now, JPerM, the usable permission set-up manager, only handles regular Java permissions. It should be extended to *support even resource control permissions*. The additional twist with resource control permissions is twofold: (1) The user must make the rather technical decision of limiting resource access. This means that the user must be supported in gaining an understanding for existing resources and reasonable limits. (2) An overzealous user may choose resource control limitations that degrade performance. If the user wants to control every read-operation in a file access, every byte written to disk must be approved by the Security Manager, leading to a noticeable performance overhead. A usable JPerM would recognise this problem and try to mediate it.

It could be worthwhile to verify with proof-of-concept implementations that *resource control in the Common Language Runtime of the .NET architecture* can be achieved with a similar approach as taken with Java. That work would have to be preceded by a performance evaluation of the runtime access control of the Common Language Runtime.

8 Future directions

Our research has led to the identification of the following additional research problems.

The thesis shows that setting up a security policy for a runtime environment or for an application is a difficult task, especially for a security lay user. Many advanced policy languages for access control, privacy or trust exist and with the advent of the Semantic Web, their importance is increasing. While we have addressed usability in security decisions, further work is needed for also making *advanced policy languages* more usable through usable set-up routines and thus available to lay users. In this context, interaction with the policy enforcer becomes even more important and challenging for the designer. Kapadia et al. [65] show one such challenge, namely the importance of providing useful and security-insensitive feedback as to why an access control failed and how the current state could be changed so that the access control mechanism would allow access. Such a component would need to be integrated into a tool that helps set up advanced policy languages. Further challenges are likely to be identfied and need to be addressed.

JPerM is mainly intended for end-user interaction. However, it can also be configured to silently log, not warn, users of security-critical events. Thus in an organisational context, JPerM could be used as a data collection or auditing tool that can be connected to an intrusion detection system, which has centrally managed rules for detecting unwanted actions. The security-critical events that JPerM captures are on a higher level than more network-centric intrusion detection system. One should investigate whether such *high-level events provide a more reliable footprint of an attack* than lower-level events.

The principle of user designation is enticing. It suggests allowing security-critical actions automatically, if the user has indicated by e.g. double-clicking a file or entering a URL, what she wants to do. If the principle of *user designation* is used for setting up security policies, how much security does it achieve? For investigating user designation, learning mode in the Norton firewall could be an interesting starting point. Learning mode sets up a rule according to what an application (not necessarily the user) wants to do. Is user designation always obvious? Can a monitoring application reliably distinguish between user designation and the acting of a malicious application? These are general but important security usability questions to consider.

9 Concluding remarks

Our work has shown that the Java Security Manager is an underused security component of the Java Virtual Machine security architecture. Its alleged slowness could not be proven and it can be easily extended to not only support access control but also release control for sensitive resources.

However, before the Security Manager or any other policy-guided reference monitor or runtime checker can be used, users must have means of setting up the access and, if possible, resource control policy. This is addressed in the second part of the thesis. As end users do not have much patience with security applications—these rarely constitute a primary task for the user—policy set-up must be convenient and interfere as little as possible with a user's primary task. We have shown this by implementing a more user-friendly tool for setting up Java security policies. JPerM allows e.g. the set-up of policies at runtime, much like personal firewalls, and clearly classifies user alerts by severity to help users in their security decisions.

References

[1] Alessandro Acquisti and Jens Grossklags. Privacy and rationality in individual decision making. *IEEE Security and Privacy*, 3(1):26–33, January 2005.

[2] Godmar Back, Wilson C. Hsieh, and Jay Lepreau. Processes in KaffeOS: Isolation, resource management, and sharing in Java. In *Proceedings of the 4th Symposium on Operating System Design & Implementation (OSDI'00)*. Usenix, October 2000.

[3] Dirk Balfanz, Glenn Durfee, Diana K. Smetters, and Rebecca E. Grinter. In search of usable security: Five lessons from the field. *IEEE Security and Privacy*, 2(5):19–24, September 2004.

[4] Walter Binder and Vladimir Calderon. Creating a resource-aware JDK. In *Proceedings of the Workshop on Resource Management for Safe Languages, 16th European Conference on Object-Oriented Programming (ECOOP'02)*. http://www.ovmj.org/workshops/resman (visited 20-Feb-2003), June 2002.

[5] Walter Binder and Jarle G. Hulaas. Extending standard Java runtime systems for resource management. In Thomas Gschwind and Cecilia

Mascolo, editors, *Software Engineering and Middleware*, LNCS 3437, pages 154–169. Springer-Verlag, March 2005.

[6] Walter Binder and Volker Roth. Secure mobile agent systems using Java: Where are we heading? In *Proceedings of the 17th Symposium on Applied Computing (SAC'02)*, pages 115–119. ACM Press, March 2002.

[7] Walter Binder, Jarle G. Hulaas, and Alex Villazón. Portable resource control in Java: The J-SEAL2 approach. In *Proceedings of the ACM Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA'01)*, pages 139–155. ACM Press, October 2001.

[8] Judith Bishop, R. Nigel Horspool, and Basil Worrall. Experience in integrating Java with C# and .NET. *Concurrency and Computation: Practice and Experience*, 15(5–6):663–680, April 2005.

[9] Lars Arne Brekken. Securing mobile code: An examination of code access security in the .NET framework. Technical report, Norwegian University of Science and Technology, Dept. of Telematics, December 2005.

[10] Sacha Brostoff, Martina Angela Sasse, David Chadwick, James Cunningham, Uche Mbanaso, and Sassa Otenko. 'R-What?' Development of a role-based access control policy-writing tool for e-scientists. *Software—Practice and Experience*, 35:835–856, 2005.

[11] Vladimir Calderon and Walter Binder. JRAF—the Java resource accounting facility. In *Proceedings of the Workshop on Resource Management for Safe Languages, 16th European Conference on Object-Oriented Programming (ECOOP'02)*. http://www.ovmj.org/workshops/resman/ (visited 22-Apr-2004), June 2002.

[12] Xiang Cao and Lee Iverson. Intentional access management: Making access control usable for end-users. In *Proceedings of the Symposium on usable privacy and security (SOUPS'06)*, pages 20–31. ACM Press, July 2006.

[13] Ajay Chander, John C. Mitchell, and Insik Shin. Mobile code security by Java bytecode instrumentation. In *Proceedings of the DARPA Information Survivability Conference & Exposition (DISCEX'01)*, pages 27–40. IEEE, June 2001.

[14] Byeong-Mo Chang. Static check analysis for Java stack inspection. *ACM SIGPLAN Notices*, 41(3):40–48, March 2006.

[15] Tzi-cker Chiueh, Harish Sankaran, and Anindya Neogi. Spout: A transparent distributed execution engine for Java applets. In *Proceedings of the IEEE International Conference on Distributed Computing Systems (ICDCS'00)*, pages 394–401. IEEE, April 2000.

[16] Antonio Corradi, Rebecca Montanari, Emil Lupu, Morris Sloman, and Cesare Stefanelli. A flexible access control service for Java mobile code. In *Proceedings of the 16th Annual Computer Security Applications Conference (ACSAC'00)*, pages 356–365. IEEE, December 2000.

[17] Iain D. Craig. *Virtual Machines*. Springer-Verlag, August 2005.

[18] Grzegorz Czajkowski and Thorsten von Eicken. JRes: A resource accounting interface for Java. In *Proceedings of the ACM Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA'98)*, pages 21–35. ACM Press, October 1998.

[19] Grzegorz Czajkowski, Stephen Hahn, Glenn Skinner, and Pete Soper. Resource consumption interfaces for Java application programming—a proposal. In *Proceedings of the Workshop on Resource Management for Safe Languages, 16th European Conference on Object-Oriented Programming (ECOOP'02)*. http://www.ovmj.org/workshops/resman/ (visited 15-Feb-2007), June 2002.

[20] Grzegorz Czajkowski, Stephen Hahn, Glenn Skinner, Pete Soper, and Ciarán Bryce. A resource management interface for the Java platform. *Software—Practice and Experience*, 35(2):123–157, February 2005.

[21] Alexander J. DeWitt and Jasna Kulijs. Is usable security an oxymoron? *interactions*, 13(3):41–44, May 2006.

[22] Paul DiGioia and Paul Dourish. Social navigation as a model for usable security. In *Proceedings of the Symposium on usable privacy and security (SOUPS'05)*, pages 101–108. ACM Press, July 2005.

[23] Paul Dourish, Rebecca E. Grinter, Jessica Delgado de la Flor, and Melissa Joseph. Security in the wild: user strategies for managing security as an everyday, practical problem. *Personal Ubiquitous Computing*, 8(6):391–401, November 2004.

[24] Claudiu Duma, Almut Herzog, and Nahid Shahmehri. Privacy in the semantic web: What policy languages have to offer. In *Proceedings of the International Workshop on Policies for Distributed Systems and Networks (Policy'07)*. IEEE, June 2007.

[25] Úlfar Erlingsson and Fred B. Schneider. IRM enforcement of Java stack inspection. In *Proceedings of the IEEE Symposium on Security and Privacy (S&P'00)*, pages 246–255. IEEE, May 2000.

[26] David Evans. *Policy-Directed Code Safety*. PhD thesis, Dept. of Electrical Engineering and Computer Science. Massachusetts Institute of Technology, October 1999.

[27] David Evans and Andrew Twyman. Flexible policy-directed code safety. In *Proceedings of the IEEE Symposium on Security and Privacy (S&P'99)*, pages 32–45. IEEE, May 1999.

[28] Xristine Faulkner. *Usability Engineering*. Macmillan Press, 2000.

[29] Ivan Fléchais. *Designing Secure and Usable Systems*. PhD thesis, University College London, February 2005.

[30] Brett D. Fleisch and Yougang Song. Rico: a security proxy for mobile code. *Computers & Security*, 23(4):338–351, June 2004.

[31] Adam Freeman and Allen Jones. *Programming .NET Security*. O'Reilly, June 2003.

[32] Steven M. Furnell. Using security: easier said than done. *Computer Fraud & Security*, 2004(4):6–10, April 2004.

[33] Steven M. Furnell. Why users cannot use security. *Computers & Security*, 24(4):274–279, June 2005.

[34] Steven M. Furnell and Bogdan Ghita. Usability pitfalls in wireless LAN security. *Network Security*, 2006(3):4–8, March 2006.

[35] Steven M. Furnell, Adila Jusoh, and Dimitris Katsabas. The challenges of understanding and using security: A survey of end users. *Computers & Security*, 25(1):27–35, 2006.

[36] Simson L. Garfinkel. *Design Principles and Patterns for Computer Systems That Are Simultaneously Secure and Usable*. PhD thesis, Massachusetts Institute of Technology, May 2005.

[37] Simson L. Garfinkel and Robert C. Miller. Johnny 2: a user test of key continuity management with S/MIME and Outlook Express. In *Proceedings of the Symposium on usable privacy and security (SOUPS'05)*, pages 13–24. ACM Press, July 2005.

[38] Simson L. Garfinkel and Eugene E. Spafford. *Web security, privacy, and commerce*. O'Reilly, 2002.

[39] Daniela Gerd tom Markotten. *Benutzbare Sicherheit in informationstechnischen Systemen*. Rhombos Verlag, Berlin, 2004. ISBN 3-937231-06-4.

[40] Li Gong, Marianne Mueller, Hemma Prafullchandra, and Roland Schemers. Going beyond the sandbox: An overview of the new security architecture in the Java DK 1.2. In *Proceedings of the USENIX Symposium on Internet Technologies and Systems*. Usenix, December 1997.

[41] Li Gong, Gary Ellison, and Mary Dageforde. *Inside Java 2 Platform Security: Architecture, API Design, and Implementation*. Addison Wesley, 2nd edition, 2003.

[42] Stefanos Gritzalis and J. Iliadis. Addressing security issues in programming languages for mobile code. In *Proceedings of the 9th International Workshop on Database and Expert Systems Applications*, pages 288–293. IEEE, August 1998.

[43] Frédéric Guidec and Nicolas LeSommer. Towards resource consumption accounting and control in Java: a practical experience. In *Proceedings of the Workshop on Resource Management for Safe Languages, 16th European Conference on Object-Oriented Programming (ECOOP'02)*. http://www.ovmj.org/workshops/resman/ (visited 22-Apr-2004), June 2002.

[44] Peter Gutmann and Ian Grigg. Security usability. *IEEE Security and Privacy*, 3(4):56–58, July 2005.

[45] Jefferson B. Hardee, Ryan West, and Christopher B. Mayhorn. To download or not to download: an examination of computer security decision making. *interactions*, 13(3):32–37, May 2006.

[46] Brant Hashii, Manoj Lal, Raju Pandey, and Steven Samorodin. Securing systems against external programs. *IEEE Internet Computing*, 2(6):35–45, December 1998.

[47] Guy Helmer, Johnny Wong, and Subhasri Madaka. Anomalous intrusion detection system for hostile Java applets. *Journal of Systems and Software*, 55(3):273–286, January 2001.

[48] Morten Hertzum and Niels Ebbe Jacobsen. The evaluator effect: A chilling fact about usability evaluation methods. *International Journal of Human-Computer Interaction*, 15(1):183–204, 2003.

[49] Morten Hertzum, Niels Jørgensen, and Mie Nørgaard. Usable security and e-banking: Ease of use vis-à-vis security. In *Proceedings of the Annual Conference of CHISIG (OZCHI'04)*. http://webhotel.ruc.dk/nielsj/research/papers/eBanking-ajis.pdf (visited 3-Aug-2005), November 2004.

[50] Almut Herzog and Nahid Shahmehri. Using the Java sandbox for resource control. In *Proceedings of the 7th Nordic Workshop on Secure IT Systems (NordSec'02)*, pages 135–147. Karlstad University, November 2002.

[51] Almut Herzog and Nahid Shahmehri. Performance of the Java security manager. *Computers & Security*, 24(3):192–207, May 2005.

[52] Almut Herzog and Nahid Shahmehri. An evaluation of Java application containers according to security requirements. In *Proceedings of the 14th IEEE International Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises (WETICE'05)*, pages 178–183. IEEE, June 2005.

[53] Almut Herzog and Nahid Shahmehri. A usability study of security policy management. In Simone Fischer-Hübner, Kai Rannenberg, Louise Yngström, and Stefan Lindskog, editors, *Security and Privacy in Dynamic Environments, Proceedings of the 21st International Information Security Conference (IFIP TC-11) (SEC'06)*, pages 296–306. Springer-Verlag, May 2006.

[54] Almut Herzog and Nahid Shahmehri. User help techniques for usable security. In *Proceedings of the 1st Symposium on Computer Human Interaction for Management of Information Technology (CHIMIT'07)*. ACM Press, March 2007.

[55] Almut Herzog and Nahid Shahmehri. Usability and security of personal firewalls. In *Proceedings of the International Information Security Conference (IFIP TC-11) (SEC'07)*. Springer-Verlag, May 2007.

[56] Almut Herzog and Nahid Shahmehri. Usable set-up of runtime security policies. In *Proceedings of the International Symposium on Human Aspects of Information Security and Insurance (HAISA'06)*. Emerald Group Publishing, July 2006.

[57] Almut Herzog, Nahid Shahmehri, and Claudiu Duma. An ontology of information security. *International Journal of Information Security and Privacy*, 2007. To appear.

[58] Kasper Hornbæk and Erik Frøkjær. Methods & usability: Comparing usability problems and redesign proposals as input to practical systems development. In *Proceedings of the Conference on Human Factors in Computing Systems (CHI'05)*, pages 391–400. ACM Press, April 2005.

[59] Collin Jackson, Daniel R. Simon, Desney S. Tan, and Adam Barth. An evaluation of extended validation and picture-in-picture phishing attacks. In *Proceedings of the Workshop on Usable Security (USEC'07)*. http://www.usablesecurity.org/papers/jackson.pdf (visited 26-Feb-2007), February 2007.

[60] Markus Jakobsson and Sid Stamm. Invasive browser sniffing and countermeasures. In *Proceedings of the 15th International World Wide Web Conference (WWW'06)*, pages 523–532. ACM Press, May 2006.

[61] Robin Jeffries, James R. Miller, Cathleen Wharton, and Kathy M. Uyeda. User interface evaluation in the real world: A comparison of four techniques. In *Proceedings of the Conference on Human Factors in Computing Systems (CHI'91)*, pages 119–124. ACM Press, March 1991.

[62] J. Johnston, Jan Harm Petrus Eloff, and L. Labuschagne. Security and human computer interfaces. *Computers & Security*, 22(8):675–684, December 2003.

[63] Mick Jordan, Laurent Daynès, Marcin Jarzab, Ciarán Bryce, and Grzegorz Czajkowski. Scaling J2EE application servers with the multitasking virtual machine. *Software—Practice and Experience*, 36(6): 557–580, May 2006.

[64] Haruhiko Kaiya and Kenji Kaijiri. *Security Policy Checker and Generator for Java Mobile Codes*. Kluwer Academic Publishers, 2002.

[65] Apu Kapadia, Geetanjali Sampemane, and Roy H. Campbell. KNOW why your access was denied: Regulating feedback for usable security. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS'04)*, pages 52–61. ACM Press, 2004.

[66] Clare-Marie Karat, John Karat, and Carolyn Brodie. Why HCI research in privacy and security is critical now. *International Journal of Human-Computer Studies*, 63(1–2):1–4, July 2005. Editorial.

[67] Larry Koved, Marco Pistoia, and Aaron Kershenbaum. Access rights analysis for Java. In *Proceedings of the 17th ACM Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA'02)*, pages 359–372. ACM Press, November 2002.

[68] A.M. Kuchling. What's new in Python 2.3? New, improved and deprecated modules. Python 2.3 Documentation, http://www.python.org/doc/2.3/whatsnew/node18.html (visited 14-Feb-2007), July 2003.

[69] Nancy Leveson. *Safeware: System Safety and Computers*. Addison Wesley, 1995.

[70] V. Benjamin Livshits and Monica S. Lam. Finding security vulnerabilities in Java applications with static analysis. In *Proceedings of the 14th USENIX Security Symposium (Security'05)*. Usenix, July 2005.

[71] Mark Lutz. *Programming Python*. O'Reilly, 2nd edition, 2001.

[72] Dahlia Malkhi, Michael K. Reiter, and Aviel D. Rubin. Secure execution of Java applets using a remote playground. In *Proceedings of the IEEE Symposium on Security and Privacy (S&P'98)*, pages 40–51. IEEE, May 1998.

[73] David Martin, Sivaramakrishnan Rajagopalan, and Aviel D. Rubin. Blocking Java applets at the firewall. In *Proceedings of the Internet Society Symposium on Network and Distributed System Security (NDSS'97)*, pages 1–11. IEEE, 1997.

[74] Roy A. Maxion and Robert W. Reeder. Improving user-interface dependability through mitigation of human error. *International Journal of Human-Computer Studies*, 63(1–2):25–50, July 2005.

[75] Gary McGraw and Edward W. Felten. *Securing Java: Getting Down to Business with Mobile Code*. Wiley & Sons, 1999.

[76] Nimisha V. Mehta and Karen R. Sollins. Expanding and extending the security features of Java. In *Proceedings of the 7th USENIX Security Symposium (Security'98)*. http://www.usenix.org/publications/library/proceedings/sec98/full_papers/mehta/mehta.pdf (visited 13-Feb-2002), January 1998.

[77] Rolf Molich, Meghan R. Ede, Klaus Kaasgaard, and Barbara Karyukin. Comparative usability evaluation. *Behaviour & Information Technology*, 23(1):65–74, January 2004.

[78] Gleb Naumovich. A conservative algorithm for computing the flow of permissions in Java programs. In *Proceedings of the International Symposium on Software Testing and Analysis (ISSTA'02)*, pages 33–43. ACM Press, May 2002.

[79] Ted Neward. .NET and Java: A study in interoperability. June 2004. http://www.theserverside.net/tt/articles/showarticle.tss?id=Interoperability (visited 5-Mar-2007).

[80] Jakob Nielsen. *Usability Engineering*. Morgan Kaufmann Publishers, 1993.

[81] Jakob Nielsen. Usability inspection methods. In *Proceedings of the Conference on Human Factors in Computing Systems (CHI'94)*, pages 413–414. ACM Press, April 1994.

[82] Jakob Nielsen and Robert L. Mack, editors. *Usability Inspection Methods*. Wiley & Sons, 1994.

[83] Maria Nilsson, Anne Adams, and Simon Herd. Building security and trust in online banking. In *Proceedings of the Conference on Human Factors in Computing Systems (CHI'05)*, pages 1701–1704. ACM Press, April 2005.

[84] Scott Oaks. *Java Security*. O'Reilly, 2nd edition, 2001.

[85] Raju Pandey and Brant Hashii. Providing fine-grained access control for Java programs. In Rachid Guerraoui, editor, *Proceedings of the 13th European Conference for Object-Oriented Programming (ECOOP'99)*, LNCS 1628, pages 449–473. Springer-Verlag, June 1999.

[86] James C. Pang, Gholamali C. Shoja, and Eric G. Manning. Providing soft real-time QoS guarantees for Java threads. In *Proceedings of the ACM Conference on Java Grande*, pages 39–47. ACM Press, 2001.

[87] Nathanael Paul and David Evans. .NET security: Lessons learned and missed from Java. In *Proceedings of the Annual Computer Security Applications Conference (ACSAC'04)*, pages 272–281. IEEE, December 2004.

[88] David S. Peterson, Matt Bishop, and Raju Pandey. A flexible containment mechanism for executing untrusted code. In *Proceedings of the 11th USENIX Security Symposium (Security'02)*. Usenix, August 2002.

[89] Allison Randal, Dan Sugalski, and Leopold Tötsch. *Perl 6 and Parrot Essentials*. O'Reilly, 2nd edition, June 2004.

[90] Volker Roth, Tobias Straub, and Kai Richter. Security and usability engineering with particular attention to electronic mail. *International Journal of Human-Computer Studies*, 63(1–2):51–73, July 2005.

[91] Ravi S. Sandhu. Good-enough security. *IEEE Internet Computing*, 7(1): 66–68, January 2003.

[92] Martina Angela Sasse, Sacha Brostoff, and Dirk Weirich. Transforming the weakest link—a human/computer interaction approach to usable and effective security. *BT Technology Journal*, 19(3):122–131, July 2003.

[93] Stuart E. Schechter, Rachna Dhamija, Andy Ozment, and Ian Fischer. The emperor's new security indicators—an evaluation of website authentication and the effect of role playing on usability studies. In *Proceedings of the IEEE Symposium on Security and Privacy (S&P'07)*. IEEE, May 2007.

[94] Bruce Schneier. *Secrets and Lies: Digital Security in a Networked World*. Wiley & Sons, January 2004.

[95] Markus Schumacher and Utz Roedig. Security engineering with patterns. In *Proceedings of the Conference on Pattern Languages of Programs (PLoP'01)*. http://www.ito.tu-darmstadt.de/publs/index_en_html/pdf/plop2001.pdf (visited 14-Feb-2007), 2001.

[96] Andrew Sears. Heuristic walkthroughs: Finding the problems without the noise. *International Journal of Human-Computer Interaction*, 9(3): 213–234, 1997.

[97] Ben Shneiderman and Catherine Plaisant. *Designing the User Interface*. Addison Wesley, 4th edition, 2004.

[98] Jan Smans, Bart Jacobs, and Frank Piessens. Static verification of code access security policy compliance of .NET applications. *Journal of Object Technology*, 5(3):35–58, May 2006.

[99] Diana K. Smetters and Rebecca E. Grinter. Moving from the design of usable security technologies to the design of useful secure applications. In *Proceedings of the New Security Paradigms Workshop (NSPW'02)*, pages 82–89. ACM Press, September 2002.

[100] James E. Smith and Ravi Nair. *Virtual machines: versatile platforms for systems and processes*. Morgan Kaufmann Publishers, June 2005.

[101] Sean W. Smith. Humans in the loop: Human-computer interaction and security. *IEEE Security and Privacy*, pages 75–79, May 2003.

[102] Sunil Soman, Chandra Krintz, and Giovanni Vigna. Detecting malicious Java code using virtual machine auditing. In *Proceedings of the 12th USENIX Security Symposium (Security'03)*. Usenix, 2003.

[103] Marc Stiegler, Alan H. Karp, Ka-Ping Yee, and Mark Miller. Polaris: Virus safe computing for Windows XP. External HPL-2004-221, HP Labs, December 2004.

[104] Niranjan Suri, Jeffrey M. Bradshaw, Maggie R. Breedy, Kenneth M. Ford, Paul T. Groth, Gregory A. Hill, and Raul Saavedra. State capture and resource control for Java: The design and implementation of the Aroma virtual machine. In *Proceedings of the Java Virtual Machine Research and Technology Symposium*. Usenix, April 2001.

[105] V.N. Venkatakrishnan, Ram Peri, and R. Sekar. Empowering mobile code using expressive security policies. In *Proceedings of the New Security Paradigms Workshop (NSPW'02)*, pages 61–68. ACM Press, September 2002.

[106] Tara Whalen, Diana K. Smetters, and Elizabeth F. Churchill. User experiences with sharing and access control. In *Proceedings of the Conference on Human Factors in Computing Systems (CHI'06)*, pages 1517–1522. ACM Press, April 2006.

[107] Alma Whitten and J.D. Tygar. Safe staging for computer security. In *Proceedings of the CHI2003 Workshop on Human-Computer Interaction and Security Systems*. http://www.andrewpatrick.ca/CHI2003/HCISEC/hcisec-workshop-whitten.pdf (visited 21-Jul-2005), April 2003.

[108] Alma Whitten and J.D. Tygar. Why Johnny can't encrypt: A usability evaluation of PGP 5.0. In *Proceedings of the 8th USENIX Security Symposium (Security'99)*. Usenix, August 1999.

[109] Min Wu, Robert C. Miller, and Simson L. Garfinkel. Do security toolbars actually prevent phishing attacks? In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 601–610. ACM Press, April 2006.

[110] Ka-Ping Yee. User interaction design for secure systems. In *Proceedings of the International Conference on Information and Communications Security (ICICS'02)*, pages 278–290. Springer-Verlag, December 2002.

[111] Ka-Ping Yee. Aligning security and usability. *IEEE Security and Privacy*, 2(5):48–55, September 2004.

[112] Joseph Yoder and Jeffrey Barcalow. Architectural patterns for enabling application security. In Neil Harrison, Brian Foote, and Hans Rohnert, editors, *Pattern Languages of Program Design, vol. 4, Software Pattern Series*. Addison Wesley, 1999. Chapter 15.

[113] Edward Yourdon. *Rise and Resurrection of the American Programmer*. Prentice-Hall, 1996.

[114] Mary Ellen Zurko and Richard T. Simon. User-centered security. In *Proceedings of the New Security Paradigms Workshop (NSPW'96)*, pages 27–33. ACM Press, 1996.

[115] Mary Ellen Zurko, Charlie Kaufman, Katherine Spanbauer, and Chuck Bassett. Did you ever have to make up your mind? What Notes users do when faced with a security decision. In *Proceedings of the Annual Computer Security Applications Conference (ACSAC'02)*, pages 371–381. IEEE, 2002.