EMERGING FRAMEWORK FOR THE EVALUATION OF OPEN SOURCE SECURITY TOOLS

E. Biermann & JC Mentz

Tshwane University of Technology

biermanne@tut.ac.za
012 382 4743
F'SATIE, Private Bag X680, Pretoria, 0001,
mentzjc@tut.ac.za
012 382 4312
Department Enterprise Applications, Private Bag X680, Pretoria , 0001

ABSTRACT

The drive from the South African Government towards the adoption of open source software across all platforms, incurred a number of research and development questions. The open source domain provides especially SMME's with options to implement high quality software that are financially viable. Although software costs is a major factor within providing proper working environments, specific security issues pertaining to open source needs to be addressed. With the opening of networks as well as the availability of information, companies need not only implement security policies, but also constantly upgrade implementations. The study of open source security issues as well as the actual evaluation of tools therefore becomes essential.

The purpose of this paper is to study the security issues within the open source environment and looking specifically at the use of security software originating from the open source domain. We provide details and results of surveys conducted around the adoption of security tools within South

African companies. The study leads to us proposing a emerging framework for the evaluation of open source security tools.

KEY WORDS

Open source software, security, framework, evaluation, tools.

EMERGING FRAMEWORK FOR THE EVALUATION OF OPEN SOURCE SECURITY TOOLS

1 INTRODUCTION

Interest in open source software (OSS) has grown significantly within South Africa (SA) during the last couple of years. This intensification is partly due to the drive from the SA government towards the adoption of OSS within both the Government and the private sector (FOSS, 2006).

Hoepman & Jacobs (2007) defines OSS as "*software for which the corresponding source is available for inspection, use, modification and redistribution by the user"*. Dimaio (2007) states that the change to OSS is beneficial in terms of cost implications; fast implementation time; tailored applications as well as providing a shortcut to technological independence. As with any new paradigm, disadvantages or challenges are also a reality. Challenges or problems that are present within the open source domain mainly evolve around security issues (Mookhey, 2004).

Industry and academia are divided into two main outlooks or groups when it comes to the security of open source software. On the one side are those stating that the openness of the code automatically decreases the security of the application or tool. This group states that the openness of the code leads to vulnerabilities being easier recognised and misused by attackers (Williams & Danahy, 2006; Hoepman & Jacobs, 2007). Attackers are also provided with a complete view of the product, including its vulnerabilities (Ford, 2007). Research conducted by Fortify (Chess, Lee & West, 2007) indicates that a poorly designed software built process may allow for an attacker to insert malicious code within the developed product. Any developer may contribute to OSS projects and no skills selection are required which may lead to un-secure code (Lawton, 2002). As no standardized quality control seems to be present within the development of OSS, this may result in the code not developed with security issues in mind

(Hoepman & Jacobs, 2007), or malicious code can be inserted within the developed product (Chess, Lee & West, 2007).

The second group beliefs that the publishing of the source code adds to providing more secured programs or applications. Arguments published include: the availability of source code means that there is complete disclosure on how a specific software or feature or section is implemented (Ford, 2007). Hoepman & Jacobs (2007) state the free distribution of source code allows for the independent evaluation of that specific software by external parties. Williams & Danahy (2006) point out that the first step in assuring whether applications are secure is to study the source code. This leads to the identification of security vulnerabilities, design flaws as well as policy violations. Security flaws are rectified faster as the open source development domain see the fixing of bugs as a major interest rather than developing new features or a new version (Lawton, 2002). The likelihood of patching bugs within the software increases within the OSS domain thus making it easier to repair holes (Hoepman & Jacobs, 2007). In the case that a vulnerability becomes known within the proprietary domain, the client is dependent on the specific vendor to develop and publish suitable patches or solutions. Within the open domain, this is however not the case (Ford, 2007). Finally Hoepman & Jacobs (2007) states that the distribution of the source code forces programmers to produce quality code, especially since it will be evaluated by a world-wide audience.

Arguments from both these camps hold value, and our focus is not in proving either. We rather set our focus on the utilisation and effectiveness of security tools developed within the open source domain. Evaluation results of open source security tools in terms of set standards and procedures seems lacking from the open source environment. Security experts within companies that need to implement security solutions have thus no means of determining which security tools are currently utilised effectively by companies. Also lacking is specific technical test results for open source security tools.

In this paper we set to determine the use of open source security tools within SA (specifically Gauteng) companies. This as well as an intensive study into open source security tools lead us to proposing an evaluation framework for open source security tools. This research is guided by the following question: in the quest for providing secure systems and networks,

what OSS tools are available and how can they be evaluated for the quality of protection they provide?

The paper is organised as follows: Section 2 describe current evaluation methods utilised to evaluate the usefulness of open source security tools; as well as a description of our evaluation process. Section 3 provides a categorization of security tools according to the results from the industry surveys. Section 4 details the evaluation criteria and framework while Section 5 portrays results from our partially implemented framework. The paper concludes within Section 6.

2 SECURITY TOOLS EVALUATION

The evaluation of security tools in both the open source and proprietary domains are done in a number of different ways. For example McGann & Sicker (2005) mentions that such tools need to be evaluated in terms of robustness, ease of use, documentation, usefulness and actual functionality. Actual functionality refers to whether claims made by the developer/s are valid. Wilander & Kamkar (2003) focuses on a specific category of tools and evaluating the category by simulating a range of possible attacks.

In the evaluation of security tools it is vital to determine the specific security category for which the tool is developed. In specifying this, the classified category can then be described in terms of minimum security features which the security tool need to satisfy. The evaluation of tools within a specific category can then be achieved by determining which security feature/s it adheres to. The institute for security and open methodologies (ISECOM) has developed an Open-Source Security Testing Methodology Manual (OSSTMM2 – see http://www.isecom.org/osstmm/) that describes a methodology for conducting security testing for organizations. The dimensions of the OSSTMM security testing process include visibility, access, trust, authentication, non-repudiation, confidentiality, privacy, authorisation, integrity, safety and alarm. In addition to this base list the software should also be tested in terms of its quality for example number of internal errors (Li *et al.*, 2006).

In order to work towards determining whether open source software with all its various security issues can be utilised effectively within the security tools environment, we have to follow a detailed process (see Figure 1).

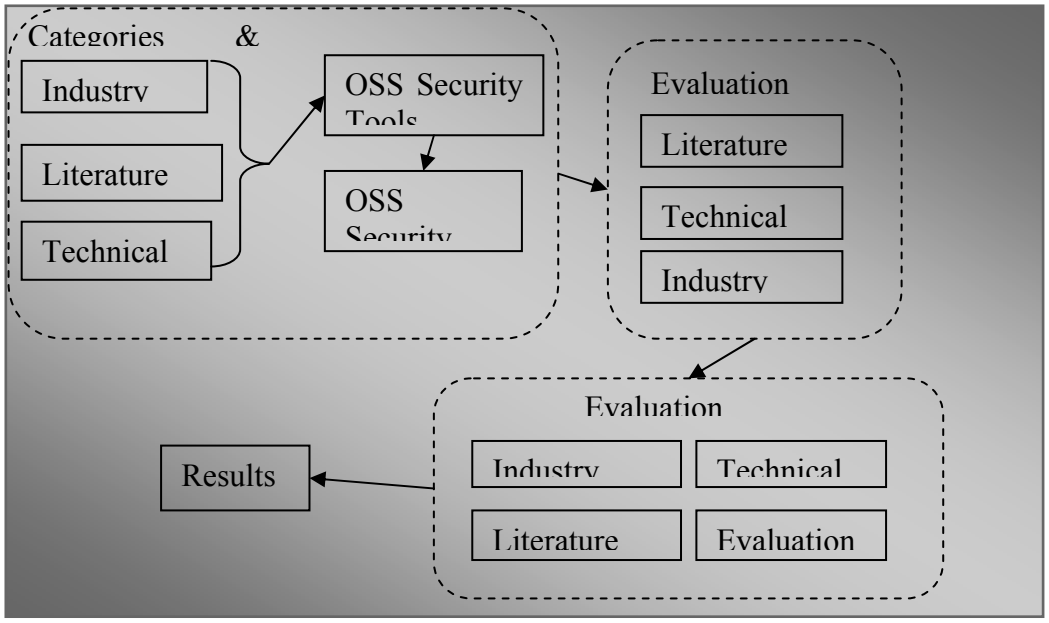

Figure 1: Evaluation Process

The process consists firstly of defining the different categories in which open source software tools can be classified. This is achieved by studying the outputs from industry surveys as well as a detailed literature and technical study. After establishing the categories we set to list the available tools within the different categories. In working towards setting an evaluation framework the aspects necessary to evaluate the different tools need to be defined. These aspects are used to firstly set the evaluation framework and finally evaluate the tools.

3 CATEGORIES & LIST OF TOOLS

Two preliminary surveys were conducted within the Gauteng region (SouthAfrica) during 2006 & 2007 targeting a total of 208 companies. The focus of these surveys was to determine amongst others the adoption of OSS security tools. A total of 192 companies responded and 47% of the 192 respondents indicate that they make use of open source security tools. 44% of these utilize OSS firewalls, 26% utilize OSS network monitoring tools and only 19% utilize OSS anti-malicious tools. A staggering 78% of the respondents within the survey regarded the security of Linux as inherently

adequate. Although this large percentage deems Linux as sufficient, 95% stated that they experience security threats on a daily basis.

Results from the industry surveys as well as a detailed literature and technical study were used to define the following categories of open source security tools within the network domain:

3.1 Category A: Scanning & Monitoring Tools

The aim of a scanning tool is to search and detect systems that have not been configured with security in mind or that have not implemented security patches for specific software as vulnerabilities are exposed (Mookhey, 2004). Monitoring tools refers to software that are utilized to continuously monitor networks in order to detect vulnerabilities in real-time. Tools within this category are further refined to include *port scanners, network vulnerability scanners, web vulnerability scanners, password vulnerability testers* and *network monitoring.*

Port scanners remotely scan a target host and determine open ports. These types of applications are normally lightweight and are a valuable tool in determining unattended server applications (Poole, 2003). Network vulnerability scanners are used to determine incorrect network configurations (Mookhey, 2004), while web vulnerability scanners automatically scan and evaluate web servers and web applications for possible vulnerabilities. Password crackers or password vulnerability testers are programs focused around guessing passwords and comparing them to an illegally obtained password file off-line (Poole, 2003). Network monitoring tools are software tools that are used to scan or track inside the boundaries of a network (Tomsho, Tittel & Johnson, 2003).

3.2 Category B: Analysis Tools

A network analyzer is a tool that enables a person to listen to network traffic that originated from or is destined to the local network. These types of programs are able to intercept and decode all traffic routed along a network as well as display the actual content. The content displayed is for example the IP addresses (source and destination); protocol type and the contents of each of the seven protocol layers (Poole, 2003). The main type of analysis tools is packet sniffers, which collects copies of network packets and analyzes them to provide information that can be used to diagnose and resolve networking issues (Whitman & Mattord, 2004).

3.3 Category C: Intrusion Detection & Prevention Tools

An intrusion detection system operates on the notion of a burglar alarm, activated upon detecting changes or violations within the network configuration. Different types of intrusion detection systems exist, namely network based that are used to protect network information assets as well as host-based to protect server or host information assets. Intrusion detection systems operate on either a signature-based or anomaly-based detection methods. Signature based systems establishes signatures of different attacks and threats; all network traffic is then compared against these signatures for possible attacks. Anomaly-based systems collect data from normal traffic and establish a base-line against which network traffic are compared (Whitman & Mattord, 2004).

3.4 Category D: Firewalls

A firewall is a device (hardware or software) that prevents specific type of information from moving between the un-trusted network (outside the organization) and the trusted network (inside the organization).The advancements in firewall technology has led to defining three different type of firewalls, referred to as generations (Whitman & Mattord, 2004). They are first generation (packet filtering firewalls); second generation (application level firewalls) and third generation (stateful inspection firewalls).

3.5 Category E: Anti-Malicious Tools

Malicious software increased tremendously with the introduction and opening of networks. Security specialists have to guard constantly against malicious code such as viruses, worms and Trojans. Anti-malicious software is the most utilized OSS security software and various tools exist.

3.6 Category F: Cryptography Tools

Cryptography tools refer to specific encryption and decryption tools used to protect data. Tools that can provide cryptographic functions range from example protecting specific communication sessions; encryption of files; encryption of hard disks as well as wireless sessions. It is therefore not feasible within this study to provide a list of specific tools per category of protection due to the vast array of cryptographic possibilities. We rather focused on providing three tools that are used mostly in providing general cryptographic functions.

The surveys as well as the literature and technical study led to an extensive list of OSS tools currently available and used. Due to size restrictions within this paper it is not possible to provide the list (the complete list is available from the authors).

4 EVALUATION FRAMEWORK

This section describes a two level framework to guide the selection of a short list of security tools for further research and evaluation. The first level of the framework deals with aspects related to the accessibility of tools and the second, more detailed level addresses the effectiveness of the tools. Each level consists of a set of criteria that address its intent. The framework is structured in this way to reflect a requirement of widespread use. Before any tool is eligible to be tested for its efficiency at protecting a system it would be necessary to be widely used first. The profile of the average user of open source software for the purpose of this research is therefore not limited to people with technically advanced computer skills. As such the general user is regarded as able to find, download, install and configure on the level of capability similar to general computer literacy.

4.1 Level 1

To fulfil the requirement that a security tool must be widely used, a number of aspects are identified. These include availability, version, platform, interface, download size, available documentation and support.

Availability: this aspect indicates the ease of acquiring the software. This does not only relate to well known web sites but also to the same piece of software being available on multiple download locations. High availability shows that the software is easy to get hold of and by implication an indicator of a large user base.

Version: software change over time and as the developer participation of a particular tools might be large the versions of the program can become confusing. This is seen especially in descriptions such as pre-release and release versions. In addition to this multiple versions with minor variations may exist. The importance of this indicator is that the user will be able to identify which versions are available and to make a choice between a stable or a developing version.

Platform: many different operating systems are available to computer users. A user interested in a tool must be able to identify the platform for

which it is designed. The wide spread use of a tool is also influenced by the same tool available for different platforms.

Interface: with the introduction of Windows 95 the computer user changed to a GUI user. Although a graphical user interface is the standard for program usage there is still the capability to use it from a command line which is predominately text based.

Download size: the size of the program will affect the choice made by the user. A very large file will take longer to download and cost more. At the same time a program that needs to be distributed by any other means than download (for example mailing a digital media such as a compact disk), might make the user think twice before choosing to use it.

Available documentation: this aspect is critical if the user is unfamiliar with the installation and configuration of the software. The more complete and easily available the documentation the wider it will be used.

Support: in conjunction with the documentation, support for a particular product is useful in the event that problems are experienced or additional information are required.

The application of the level 1 criteria resulted in a shortlist of tools that represent those most used for security purposes. The tools on this list has not been evaluated for the quality of security protection that it provides and for that purpose a second level of evaluation are required.

4.2 Level 2

The tool as an application forms an integral part of security and the methodology is instructive towards the compilation of a set of aspects for level 2 testing. The ability of the specific security tool to assist in the creation of a secure network forms the basis of level 2 testing. In specific the following aspects are included:

Functionality: this refers to the actual functionality of the tool in relation to claims made by the developer/s. The documentation from level 1 is analysed and it is determined whether the tool actually include the stated functionalities.

Protection: the protection ability is evaluated according to the category in which the tool is classified. Each category is defined by a set of minimum protection abilities and the security tool is tested according to these defined abilities.

Interoperability: security tools are generally created to only provide protection according to a set category. A requirement for such tools is its

ability to operate successfully with security tools from the same as well as other categories.

Usability: the ease of use as well as the usefulness of the tool. The level of difficulty to install, configure and maintain the tool is evaluated. Also included is to determine the existing need for such a specific tool.

Simulation: a simulated test bed or test environment is required in which current threats and attacks can be simulated. The security tool is then evaluated within this simulated environment for real-time attacks and vulnerabilities.

5 RESULTS

The results from the questionnaires showed a wide variety of tools in use. This is consistent with the milieu of the open source domain but makes the analysis and subsequent answer of the research question challenging. The evaluation of the extensive list of tools according to the first level of the framework was completed utilising a system of weights. For example a weight of 1 was assigned for each platform on which the tool can be implemented and 1 weight was assigned if telephonic support was available. This approach was followed to ensure that the tool that is most widely used in terms of level 1 criteria would make it to the short list of security tools that will be tested in the second level of the framework. Evaluating the extensive list of OSS security tools against the first level of our emerging framework led to the results displayed in Table 1.

Table 1: First level Evaluation Results

Category	Security Tool
A: Port Scanners	*Nmap -Network Mapper (http://insecure.org/nmap/):* Rapidly scan small and large networks. A number of tools also make use of the functionality provided by Nmap, for example XNmap and Nessus
	Angry IP Scanner (*http://www.angryziber.com/*: A very fast scanner that scans IP addresses in any range as well as their ports.
	Unicornscan (http://www.unicornscan.org): This tool provides an interface for introducing small stimuli and measuring the response from TCP/IP enabled devices or networks.
A: Network Vulnerability Scanners	*X-Scan (*http://www.xfocus.org*):* A general network vulnerabilities scanner that can be utilized to scan for network vulnerabilities by using a multi-threading method.
	SARA (http://www.www-arc.com/sara/): The Security Auditor's Research Assistant (SARA) is A third generation Unix-based security analysis tool. This scanner is derived from the famous SATAN (Security Administrators Tool for Analyzing Networks) and features extensive usability and auditing capabilities.
A: Web Vulnerability	*Nikto (http://www.cirt.net):* Performs comprehensive tests against web servers for multiple items, including over 3300 potentially dangerous files/CGIs.

Scanners	*Webscarab*(*http://www.owasp.org/index.php/OWASP_WebScarab_Project*): This tool analyses applications that are communicating via the HTTP and HTTPS protocols.
	Wikto (*http://www.sensepost.com/research/wikto/*): Built for the .NET 2 framework and contains a built-in web spider for directory discovery purposes.
A: Password Vulnerability Testers	*John the Ripper (http://www.openwall.com/john/):* A fast password cracker that is available for different UNIX distributions, Windows, DOS, BeOS and OpenVMS.
	Cain & Abel(http://www.oxid.it)*:* A password recovery tool for MS Windows. It allows easy recovery of various types of passwords by sniffing the network.
	Ophcrack (*http://sourceforge.net/projects/ophcrack/*): A Windows password cracker based on a time-memory trade-off using rainbow tables.
A: Network Monitoring Tools	*Nagios (http://www.nagios.org):* A powerful network monitoring tool that are used for detecting specific network problems. Included in distributions such as Debian, Fedora and Suse.
	EtherApe(*http://sourceforge.net/project/showfiles.php?group_id=2712*): A graphical network monitoring tool, featuring Ethernet, IP, TCP, FDDI and Token Ring modes.
	*WireShark(http://www.wireshark.org):*Network protocol analyzer for UNIX, OS X and Windows. It allows for the examination of data from a live network or from a captured file on disk. WireShark was known as Ethereal up to 2006.
B: Packet Sniffers	*Snort (http://www.snort.org):* Performs real-time traffic analysis and packet logging on IP networks.
	Kismet (*http://www.kismetwireless.net/*): Wireless network detector and sniffer for 802.11 layer 2 wireless networks.
	TCPDump (http://tcpdump.org): An IP sniffer that requires few system resources. It is a specialized sniffer used for detecting network problems.
C: Intrusion Detection & Prevention	*Snare* -System Intrusion Analysis and Reporting Environment (*http://www.intersectalliance.com*): This is a series of log collection agents that facilitate centralized analysis of audit log data.
	OSSEC (http://www.ossec.net): This tool performs functions such as log analysis; integrity checking; time-based alerting as well as active responses.
	Tripwire (http://sourceforge.net/projects/tripwire/): A tool used by security administrators to determine the integrity of files as well as possible modifications or tampering of specific files is the file integrity scanner (Poole, 2003).
D: Firewalls	*Smoothwall Express (www.smoothwall.org):* Smoothwall includes traffic shaping, VPN capability as well as proxy and DHCP server capabilities.
	IPCop (www.IPCop.org): IPCop includes a whole range of services including traffic shaping on outgoing connections and a built-in DHCP and proxy server.
	Netdefender (http://www.codeplex.com/netdefender/): A specialized open source firewall. It operates by blocking all communication on specified ports after the rules have been setup.
E: Anti-Malicious Tools	Clam AV (*www.clamav.org*): : This application is specifically designed for scanning e-mail gateways for malicious code. Virus scans take place from the command line in a terminal window.
	ClamWin (www.clamwin.com): Windows version of the ClamAV engine. It seperates the processes of scanning for viruses on harddisk and scanning for viruses in program memory.
	Winpooch (*http://sourceforge.net/projects/winpooch/*): Windows watchdog that detects and monitors changes in the system and effectively blocks spyware.
F: Cryptography Tools	*GnuPG(http://www.gnupg.org): Gnu Privacy Guard* features a complete implementation of the OpenPGP standard and allows for the encryption and the digital signing of data and communication.

	OpenSSL (http://www.openssl.org): This tool implements the Secure Sockets Layer (SSL v2/v3) and Transport Layer Security (TLS v1) protocols.
	TrueCrypt (http://www.truecrypt.org): Disk encryption software that creates a virtual encrypted disk within a file and mounts it as a real disk. It is able to encrypt entire hard disk partitions or storage devices such as USB flash drives.

It has to be noted that an alarming number of open source tools become proprietary after a few versions of being publicly available (see for example Nessus at www.**nessus**.org). This means that once a company is utilising a specialised tool in providing a secure working environment, the possibility that newer versions will become available at an additional price is increasing.

Results displayed only reflects implementation of Level 1 of our proposed framework. Level 2 aspects is not finalised and the implementation is set to be completed in July 2008.

6 CONCLUSION

This paper addresses the research question by clearly identifying that a number of open source security tools are being used to protect systems. The application of the 1st level of the testing framework resulted in a shortlist of tools that can be identified as the most widely used security tools. This result is significant in that it shows the security awareness of open source developers and it suggests that open source operating systems might not be inherently as secure as is claimed. In addition the results indicate user security awareness.

Further research is in progress as to the evaluation of the quality of the protection that the shortlist of security tools provides to software systems. To that effect this paper highlighted the possible aspects for a second level of security tool testing. Finally the completed security evaluation framework will play a significant role in the protection of systems using open source tools by promoting a high quality of development as well as the development of a standard for the evaluation of such tools.

7 ACKNOWLEDGEMENTS

This research was fully funded by Eskom, Research & Innovation Department. They also fund the research and implementation of Level 2 of the framework.

8 REFERENCES

CHESS, B., LEE, FD. & WEST, J. 2007. Attacking the Build through Cross-Build Injection. [Online] Fortify Software. Available at: http://www.fortifysoftware.com

DIMAIO, A. 2007. Open Source Software in Government: How much open and how much source? Gartner Symposium ItxPO 2007.

FORD, R. 2007. Open vs. Closed: Which is more secure? *ACM Queue Magazine February*.

FOSS. 2006. Policy of Free and Open Source Software use for South African Government. [Online]. Available from: http://www.doc.gov.za/index.php?option=com_docman&task=doc_view&gid=49

HOEPMAN, J. & JACOBS, B. 2007. Increased Security Through Open Source. *Communications of the ACM*, January, Vol. 50, No.1.

LAWTON, G. 2002. Open Source Security: Opportunity or Oxymoron? *IEEE Xplore*, Volume 35, Issue 3, March 2002, Pg 18-21.

LI, Z., TAN, L., WANG, X., LU, S., ZHOU, Y., and ZHAI, C. 2006. Have things changed now?: an empirical study of bug characteristics in modern open source software. In *Proceedings of the 1st Workshop on Architectural and System Support For Improving Software Dependability*.

McGANN, S. & SICKER, D. 2005. An analysis of security threats and tools in SIP-based VoIP systems. *Presented at the 2nd Annual Workshop VoIP Security.,* Washington, DC, June.

MOOKHEY, K.K. 2004. Open Source Tools for Security and Control Assessment. *Information Systems Control Journal* , Volume 1.

POOLE, O. 2003. *Network Security: A practical guide.* Butterworth-Heinemann, Oxford. ISBN 0-7506-50338.

TOMSHO, G., TITTEL, E. & JOHNSON, D. 2003. *Guide to Networking Essentials*. Third Edition. Thomson Course Technology. ISBN 0-619-13087-3.

VIEGA, J. 2004. Open Source Security: Still a Myth. O'Reilly [Online]. Available from: http://www.oreilly.com

WHITMAN, M.E. & MATTORD, H.J. 2004. *Management of Information Security.* Thomson Course Technology. ISBN 0-619-21515-1.

WILANDER, J. & KAMKAR, M. 2003. A comparison of publicly available tools for dynamic buffer overflow prevention. In *Proceedings of the 10th Annual Network and Distributed Systems Security Symposium.*

WILLIAMS, J. & DANAHY, J. 2006. "Opening the black box" A Source Code Security Analysis Case Study. Aspect Security Inc. & Ounce Labs Inc.