Ethics and Information Technology **6:** 93–110, 2004.

Agents of responsibility in software vulnerability processes

Ari Takanen, Petri Vuorijärvi, Marko Laakso and Juha Röning
University of Oulu, Finland

Abstract. Modern software is infested with flaws having information security aspects. Pervasive computing has made us and our society vulnerable. However, software developers do not fully comprehend what is at stake when faulty software is produced and flaws causing security vulnerabilites are discovered. To address this problem, the main actors involved with software vulnerability processes and the relevant roles inside these groups are identified. This categorisation is illustrated through a fictional case study, which is scrutinised in the light of ethical codes of professional software engineers and common principles of responsibility attribution. The focus of our analysis is on the acute handling of discovered vulnerabilities in software, including reporting, correcting and disclosing these vulnerabilities. We recognise a need for guidelines and mechanisms to facilitate further improvement in resolving processes leading to and in handling software vulnerabilities. In the spirit of disclosive ethics we call for further studies of the complex issues involved.

Key words: information security, professional ethics, security evaluation, software development, software testing, software vulnerability

Introduction

To produce robust and trustworthy software, professional and proficient approaches to software development are required. Cost-benefit and risk analyses may be performed to determine the level of expertise required in a software project and to specify the security and quality requirements[1] of the final product. Although it is desirable to strive to build dependable software, in some cases the public can be adequately served with inexpensive, less robust software with no heavy quality assurance procedures. In cases of more demanding customers requiring robustness from the product, professional software engineering methods exist to help software developers provide better quality software.

However, trivial errors that cause security vulnerabilities[2] and safety hazards are frequently found in both consumer software and advanced building blocks of the security-critical information infrastructure.[3] Information security typically focuses on protecting the confidentiality, integrity and availability of information from external threats, whereas the safety issues of computer systems are related to risks to life or property from natural or accidental hazards. Security and safety in software are attributes that cannot be effectively measured or guaranteed, and are thus always based on levels of risk. A system is safe or secure when the level of risk is acceptable. Modern society depends on software products to the extent that when considering flaws in security and robustness, we can often use the terms safety and security interchangeably, or combine them under the term dependability. Unfortunately, often all this may not have been taken into consideration when developing software. One insecure component in a security-critical environment can compromise the reliability of the whole system.

The scope of this paper is narrowed down to issues related to creating and handling of unintentional security-related development flaws, also called software vulnerabilities. Our goal is to describe and

[1] We divide information security roughly into two different aspects. Traditional information security takes the perspective of security requirements, focusing on methods for protecting the confidentiality, integrity and availability with means such as filtering, attack fingerprinting, user authentication and encryption. On the other hand, the software security approach focuses on software quality and reliability, where the issues relate to the actual development flaws with security implications and their solutions.

[2] A vulnerability can be a lacking security requirement (e.g. lack of, or improper authentication, encryption, ...), or a development error in the software (e.g. buffer overflow, race condition, ...).

[3] Peter G. Neumann. *Computer-Related Risks*. ACM Press/Addison Wesley, 1995; The risks digest. Forum on risks to the public in computers and related systems. http://catless.ncl.ac.uk/Risks; Nancy G. Leveson. *Safeware: System Safety and Computers*. Addison-Wesley Publishing Company, 1995.

identify the main ethical issues to be considered when assessing software vulnerability processes, focusing on disclosing the actors and relevant roles involved in a way that enables more thorough analyses to be made of the distribution of responsibility among them. By responsibility we here mean the attribution of burden resulting from realised compromises of computer security, but also safety. The existence of accepted professional guidelines allocates responsibilities to those who claim or are expected to follow them.

We begin with an introduction to disclosive ethics and responsibility, and a short presentation of generally accepted codes of ethics for professional software engineers. After outlining the most distinctive agents and mechanisms involved in software vulnerability handling processes and summarising some viewpoints on the issue, we move to a fictional case study of software vulnerability exploitation. It illustrates the practical meaning of the categories of actors and roles we define as the main agents in software vulnerability processes. Subsequently, we scrutinise the moral and professional responsibilities and practices of the three major role groups involved. The article is concluded by focusing our analysis on the role groups involved in vulnerability handling. By this analysis we aim to provide a relatively neutral starting point explicating the necessary facts and relationships for further normative analyses of this controversial issue.

Disclosing responsibility and professionalism

Disclosive ethics

Philip Brey[4] has suggested a three-level methodology in identifying and analysing previously undisclosed ethical issues in information technology consisting of disclosure, theoretical and application levels. In this classification, emphasis is placed on illuminating the often indistinct normativity inherent in computer systems, but also in their usage and closely related societal issues. In the actual disclosure level of (disclosive) computer ethics Brey recommends a joint effort of computer specialists, philosophers and social scientists in its practise. Basically, these disclosure level studies are descriptions of computer systems and their usage, where the focus on issues deemed ethical is directed by a common sense definition of a selected moral principle. This is done in order not to push a phenomenon into a narrow predetermined category imposed by an elaborate ethical theory, but to preserve the pragmatical complexities for further analysis. The actual recommendations for correct policies concerning the information technology belong to the area of applied ethics, where normative analyses founded on selected moral ideologies are made.

As the writers of this article represent the engineering and social sciences traditions, Brey's recommendation of an interdisciplinary approach is appreciated. We categorise our article as a variant of disclosive ethics at the disclosure level, even though the dilemmas considered here are not altogether previously unknown[5] and have already caused contradictions between different parties involved. However, the ethical implications of the subject at hand are hard to grasp as the vulnerability process is usually not understood as the wide continuum in the social and societal level it is.

Subsequently, our presentation differs somewhat from the notion of disclosure level ethics as meant by Brey, by focusing on the explication of social relations and societal structures surrounding this issue, rather than centering on the functioning of the technology itself. The explicated structures and relations are then preliminarily utilised in fathoming the distribution of responsibility between the agents capable of carrying it (e.g. human agents and compositions thereof) in the event of unforeseen or unwanted functioning of the technology. The categories of roles and actors are defined further in this article to facilitate this analysis, the former with reference to individual human agents and the latter pointing to the compositions (groups, structures, organisations, etc.) of individuals. At this stage we tentatively call our variant of the disclosive ethics approach as doing social disclosure level ethics. It should be noted that we do not attempt to formulate any final normative (concrete) recommendations regarding our subject in this article, but leave them for further normative analysis.

Responsibility

Professional players in information technology are not without responsibility to those whom they serve, be they employers or customers. This is especially so with software with security or safety implications. Issues such as ignorance, irresponsibility or even arrogance have an ethical aspect, especially in pro-

[4] Philip Brey. Method in Computer Ethics: Towards a Multi-level Interdisciplinary Approach. *Ethics and Information Technology*, 2(2): 125–129, 2000.

[5] OUSPG has collected the past discussions on vulnerability handling in a link list. http://www.ee.oulu.fi/research/ouspg/sage/dis_closure-tracking/.

fessional cadres but also outside of them. Often malfunctions in computers are attributed solely to "bugs", for example, or the computer itself is seen as the root cause of the problem, neglecting the fact that the technology has its makers, suppliers and utilisers who all carry their share of moral responsibility.[6] When considering the relationship between moral responsibility and security, the malicious intent which the presumed security is protecting against does create its own share of moral responsibility. When considering safety and moral responsibility, this malicious intent of the abuser is sometimes missing, thus leaving moral responsibility solely on the developers, intermediaries and users of the technological construct. In our article, we prefer to use the term dependability (combining security and safety), because we primarily exclude intentionally malicious attack activity from the analysis of our article.

Anton Vedder[7] describes the conditions of responsibility attribution in his article discussing the accountability of Internet access and service providers. He identifies two aspects of responsibility; prospective and retrospective. By retrospective responsibility the consequences of actions can be attributed to persons, whereas by prospective responsibility he means an obligation or duty to act or not to act in a certain way prescribed by a moral principle, which applies to the person at the time of the event. If a person is to be held retrospectively responsible, a moral principle has to be shown to have applied to him at the time of the action, or omission of action. These principles, apparently, can be found both in formal and informal (meaning common sense) cultural forms. Furthermore, it has to be determined whether the two remaining conditions for attributing retrospective responsibility, being (1) causality and (2) intentionality, are sufficiently in effect. Causality is defined as the contribution of the person's action to the effect or consequences, be the contribution "direct or indirect, substantial or additional". Intentionality means that the person must have had some amount of purpose in his actions and/or their relation to the consequences, and is not totally ignorant of the meaning of his actions and their consequences. The same of course applies for omissions of action.

Vedder also delves into a philosophical debate about the applicability of responsibility as a collective measure in the area of ethics.[8] Sara Baase has clarified this dilemma by pragmatically claiming that responsibility can be attributed to both individuals and organisations.[9] Our application of the concept of responsibility also has similarities with the notion of positive moral responsibility as defined by John Ladd, where emphasis is placed on the distribution of responsibility to multiple actors.[10] This was explicated in opposition to the more traditional method of negative responsibility, where finding a single scapegoat is essential to exonerate the others involved.

Thus in this article, we focus on the moral responsibilities of persons involved in developing and providing software to customers, while including also those individuals participating in later stages of vulnerability handling processes. Many of the issues discussed here are already covered by legal, contractual or professional responsibilities. As we are not experts in the field of law nor in theoretical ethics, our viewpoint is out of necessity pragmatical. Therefore, we opt to take cover behind the shield of the disclosive ethics mandate and primarily use our common sense, not moral philosophy, as regards to this question in our preliminary analysis.

Professional ethics

In an ACM[11] report[12] on their position on considering software engineering as a licensed engineering profession, the ACM concludes that the software engineering field does not need licensing to be a profession and that it cannot support licensing efforts for software engineers, as is promoted and required in many countries in fields offering services directly to the public, such as doctors, lawyers, civil engineers, contractors, day care workers, barbers, and surveyors. The state of knowledge and practice in software engineering is too immature to warrant licensing, and

[6] John Ladd. Computers and Moral Responsibility: A Framework for an Ethical Analysis. In Carol-Gould, editor, *The Information Web*: *Ethical and Social Implications of Computer Networking*, pp. 207–227. Westview Press, Boulder, Colorado, 1989.

[7] Anton Vedder. Accountability of Internet Access and Service Providers – Strict Liability Entering Ethics? *Ethics and Information Technology*, 3(1): 67–74, 2001.

[8] Vedder (2001).

[9] Sara Baase. *A Gift of Fire*: *Social*, *Legal and Ethical Issues in Computing*, p. 342. Prentice Hall, 1997.

[10] Ladd (1989).

[11] Association for Computing Machinery. http://www.acm.org.

[12] In "A Summary of the ACM Position on Software Engineering as a Licensed Engineering Profession" the ACM points out that they believe the problem of reliable and dependable software, especially in critical applications, is the most important problem facing the IT profession. http://www.acm.org/serving/se_policy/.

licensing would not be effective in assuring software quality and reliability.[13]

There are numerous rules and guidelines in software engineering that state how the profession of software engineering should be practised with high moral standards. Organisations such as ACM and IEEE-CS,[14] have started promoting professional approaches among their members, with the help of certification, training and also by the deployment of ethical codes. Several organisations have compiled their codes of ethics[15] which their members should follow to show at least some concern toward the ethical aspect of the business and the responsibilities brought up by these considerations. Ethics related to software dependability issues are also considered in the professional codes of ethics. They point out the moral responsibilities of professional developers in ensuring that suitable protections are in place to avoid loss of human lives or property due to faults in the software.

A short version of the code of ethics[16], which all IEEE members should agree to follow, contains 10 simple rules of ethics. Of these, attention is drawn to four that are closest to the focus of this article:

"1. to accept responsibility in making engineering decisions consistent with the safety, health and welfare of the public, and to promptly disclose factors that might endanger the public or the environment." This clearly points out that both developers and external researchers are responsible and obligated to do their best to bring to public attention any faults in their products and the products of the others that endanger the public or the environment. The more critical the environment, the more important this point becomes.

The responsibility of software engineers to understand the consequences and to inform others about what they know, and to keep on studying, are pointed out in the following two rules. "5. to improve the understanding of technology, its appropriate application, and potential consequences; 6. to maintain and improve our technical competence and to undertake technological tasks for others only if qualified by training or experience, or after a full disclosure of pertinent limitations." These two rules together point out that software engineers should also be able to make dependable software for critical applications and to comprehend the impact these applications may have on human lives. Developers, and perhaps even vendors, should clearly point out if they are unable to meet the security and safety requirements or follow the engineering practices that should result in good quality, and they should inform their co-workers and warn their customers of the possible risks in using the product.

"7. to seek, accept, and offer honest criticism of technical work, to acknowledge and correct errors, and to credit properly the contributions of others." This encourages external researchers to find and resolve security or safety related flaws in the software of others, and obliges developers to fix them the best they can and not shoot the messenger but instead give credit where it is due.

Some work has been done to combine the different codes of ethics into one specially tailored for software engineering. This code[17] has been accepted by the IEEE and the ACM as a standard supporting individual software engineers or software engineering managers, and groups thereof, in their striving for ethically sound professional practices. This code is seen to bind anyone claiming to be or aspiring to be a software engineer. According to the ACM/IEEE-CS code, the first obligation of the engineer is always to the public good. The ACM/IEEE code of ethics is on the principal level quite similar to the IEEE code previously discussed, but it is one step further operationalised towards the practise in software engineering. However, generally accepted explicit guidelines regarding the reporting and handling of software vulnerabilities do not yet exist.

Software vulnerabilities

The proactive means of preventing software security vulnerabilities from emerging can only be achieved by

[13] The history of the joint IEEE Computer Society and ACM Steering Committee for the establishment of software engineering as a profession. http://computer.org/tab/History.htm.

[14] Institute of Electrical and Electronics Engineers – Computer Society. (A member society of IEEE.) http://computer.org.

[15] Online Ethics Center. Codes of Ethics and Conduct. http://www.onlineethics.org/codes/.

[16] IEEE Code of Ethics. A note distributed to all members of the IEEE association. http://www.ieee.org/about/whatis/code.html.

[17] Software Engineering Code of Ethics and Professional Practice. As recommended by the ACM/IEEE-CS Joint Task Force on Software Engineering Ethics and Professional Practices (SEEPP). http://www.acm.org/serving/se/code.htm; Don Gotterbarn, Keith Miller and Simon Rogerson. Computer Society and ACM Approve Software Engineering Code of Ethics. *Computer*, 32(10): 84–88, 1999. http://computer.org/computer/code-of-ethics.pdf.

good software development practises. The state of the art in development of dependable software is based on at least four identifiable aspects: qualified engineers, extensive audits, test coverage and quality assurance.[18] These aspects are complementary and none of them can be ignored. For most vendors, qualified engineers may be difficult to come by, and certifications of secure programming skills are scarce. In the OpenBSD[19] project, most critical source code is thoroughly audited by security-aware developers. The project is also an example of software development where security and quality override a wide set of features in priority. Although testing can never reach full test coverage and can never prove that there are no flaws left, the PROTOS[20] project shows how, by using unexpected but systematic tests, testing can focus on external threats and dependability by concentrating on a wide set of inputs in testing the external interfaces of the software. Security standards[21] enforce quality in the various phases of the software life-cycle, exerting an indirect effect on the development and use of software.

The reactive methods of resolving software security vulnerabilities include, e.g., discovery and reporting processes. Vulnerabilities are discovered both by developers themselves and by external evaluators. There are several approaches with which both professional and non-professional evaluators can resolve or disclose vulnerabilities.[22] In addition to directly contacting the software developers or vendors, public disclosure mailing lists[23] and other channels are used for providing feedback to the vendors about vulnerabilities in their software. It is obvious that an inconsiderate approach to publishing vulnerability details can expose the users of deployed systems to unnecessary risks of intrusion, in addition to causing harmful publicity to the software vendors. Thus, balancing all these considerations in software vulnerability handling processes[24] is a matter to be delicately handled and likely to bring forth controversy regardless of the route opted for.

The rapid creation and active deployment of security fixes[25] is critical in maintaining software system reliability. It can be harmful for the customers if the software vendor decides to fix an emerged vulnerability quietly in later versions without announcing the necessity of patch deployment in software currently in use. When the vulnerability details are known only by the vendors and developers of the software and they do not produce a method of repair, they are making it impossible to protect against intrusions by malicious users that find out about this vulnerability on their own. On the other hand, whenever a fix is published on the Internet, it also provides enough information on analysing the vulnerability details to allow breaking into systems that have not applied this fix to their systems.[26] Currently, public databases[27] of past and present vulnerability exploitation details and exploitation scripts[28] are available for anyone to use for both research and education as well as for questionable purposes.[29]

In conventional risk management concerning information security breaches, the focus of attention is usually on system administration, maintenance and use. Risk management of computer networks traditionally involves the physical, technical and administrative controls and procedures for reducing risks cost-effectively. For risk management in software development, the best solutions would involve training, tools and engineering processes that improve quality. Certification processes are one metric on

[18] See e.g. Leveson (1995, pp. 158–159).

[19] See http://www.openbsd.org.

[20] PROTOS – Security Testing of Protocol Implementations. http://www.ee.oulu.fi/research/ouspg/protos/.

[21] Such as ISO/IEC 15408 with software development aspects, and ISO/IEC 17799 with interest in system maintenance and use.

[22] Marko Laakso, Ari Takanen and Juha Röning. The Vulnerability Process: A Tiger Team Approach to Resolving Vulnerability Cases. In *Proceedings of the 11th FIRST Conference on Computer Security Incident Handling and Response*, Brisbane, 13–18 June 1999.

[23] Best-known is Bugtraq. Mailing lists discussions are archived by e.g. Neohapsis, Inc. http://archives.neohapsis.com/.

[24] The software vulnerability handling process is typically seen to mean the interaction of agents and factors causing an opening of the window of vulnerability in software and the process of closing this window. Opening does not necessarily mean publicity of the vulnerability, as failures due to the errors can happen without active and malicious involvement.

[25] Also called security patches.

[26] Laakso et al. (1999).

[27] SecurityFocus.com claims to provide the Internet's largest and most comprehensive database of security knowledge and resources freely available to the public. http://www.securityfocus.com/.

[28] The automated attack programs or scripts are colloquially called exploits.

[29] Ari Takanen, Marko Laakso, Juhani Eronen, and Juha Röning. Running Malicious Code by Exploiting Buffer Overflows: A Survey of Publicly Available Exploits. In *Proceedings of the 9th Annual EICAR Conference*, Brussels, Belgium, 4–7 March 2000.

which to base risk management. A certain quality factor can be expected from people with a certain certification or from products that carry certification from a trusted evaluator.[30] In risk avoidance, one way is to use licence agreements to transfer the risk to the customer.[31] However, evaluation of software quality may be difficult for customers, and thus the decision to purchase software, even in a security-critical context, is often made according to the marketed features and not according to the actual security impact of using the product. In fact, the actual security impact of the product may not be known even to the software marketers or developers themselves.

Legislation tries to interpret the needs of the modern information society. New laws have been proposed to prevent reverse-engineering of commercial software, but this can also restrain security research by limiting the allowed methods.[32] Although consumer protection laws generally apply to software, additional laws have been proposed to release developers from responsibility and liability for damage caused by programming errors.[33] In some countries, there has been discussion on restricting access to security vulnerability information.[34]

In information security incidents or intrusions where the software failures were abused, the responsibility of software developers of both free and commercial software is rarely discussed in public. In addition to developers, distributors and integrators of software or systems, customers and system administrators are essential parties in software vulnerability processes, but often without clearly stipulated responsibilities. Even external evaluators of security, be they organised tiger-teams or dilettante groups having different motivations for the disclosure of discovered vulnerabilities, may choose to act in ways that others might see as irresponsible[35] acts towards the general public or the software vendor in question.

Vulnerability process – a fictional case-study

In software development, people who usually consider themselves professional developers or programmers create software for the public with responsibilities and obligations to the company they work for, but also to the clients and users of the software. With the adoption of professional roles, their responsibilities in the moral sense can be argued to be enhanced compared to non-professional stances. In security vulnerability testing, a team of external evaluators of security, called a tiger-team, searches for vulnerabilities in either design, implementation or configuration of the software components. If both these major parties claim to be acting professionally, their actions can be viewed from the same perspective of professional ethics. These two groups are not, however, the only ones involved in software security incidents and vulnerability processes. The following case study was invented to point out the most frequently encountered roles of different actors in software vulnerability processes, and also to exemplify the level of complexity often inherent there.

> Emil's house burned down due to a software fault in the heating system controller. This particular software was purchased by his employer, Org Ltd, who uses it to control the heat in their chemical processes, and given to Emil to learn and to practise its operation at home. The software was purchased from a company called Dist Inc., who are the retailers for the developer, Vend Inc. A subcontractor for Vend Inc., HeatSW Ltd, a company comprised of a team of five fellows, produced the vulnerable component for use in the process control software. Fortunately, Break-Team, a professional tiger-team, had reported the problem to Vend Inc. a few months ago, but the vendor-provided security patch was not available at the time of the incident at Emil's house. However, a commercial security product vendor, Snake-Oil Inc., had produced and marketed a product that was supposed to protect against any attack against this vulnerability. Emil

[30] Ana del Amo Calvo. The Liability of Professional Experts Like Risk Managers. In F. Galindo and G. Quirchmayr, editors, *Advances in Electronical Government*, *Pre-Proceedings of the Working Conference of the International Federation of Information Processing WG 8.5 and the Center for Computers and Law*, Zaragoza, Spain, 10–11 February 2000.

[31] *Ibid.*

[32] Kerstetter J. A Reprieve for 'Ethical Hacking'. PC Week Online. July 20, 1998. http://www.zdnet.com/eweek/news/0720/20ewto.html; Lemos R. Security Expert Blasts Shoddy Software. ZDNet News. July 9, 1999. http://www.zdnet.com/zdnn/stories/news/0,4586,2290399,00.html.

[33] Cem Kaner. Software Engineering and UCITA. *Computer & Information Law*, 18(2), 1999; IEEE-USA UCITA Network, http://www.ieeeusa.org/forum/grassroots/ucita.

[34] NTBugtraq mailing list moderator Russ Cooper, mentions in his email titled 'Administrivia #31473: NTBugtraq at the White House' that 'classified channels' are being considered to replace public forums. E-mail on the NTBugtraq mailing list, April 19, 2000. Mailing list archive available at http://www.ntbugtraq.com/.

[35] Vulnerability disclosure publications and discussion are collected at http://www.ee.oulu.fi/research/ouspg/sage/disclosure-tracking.

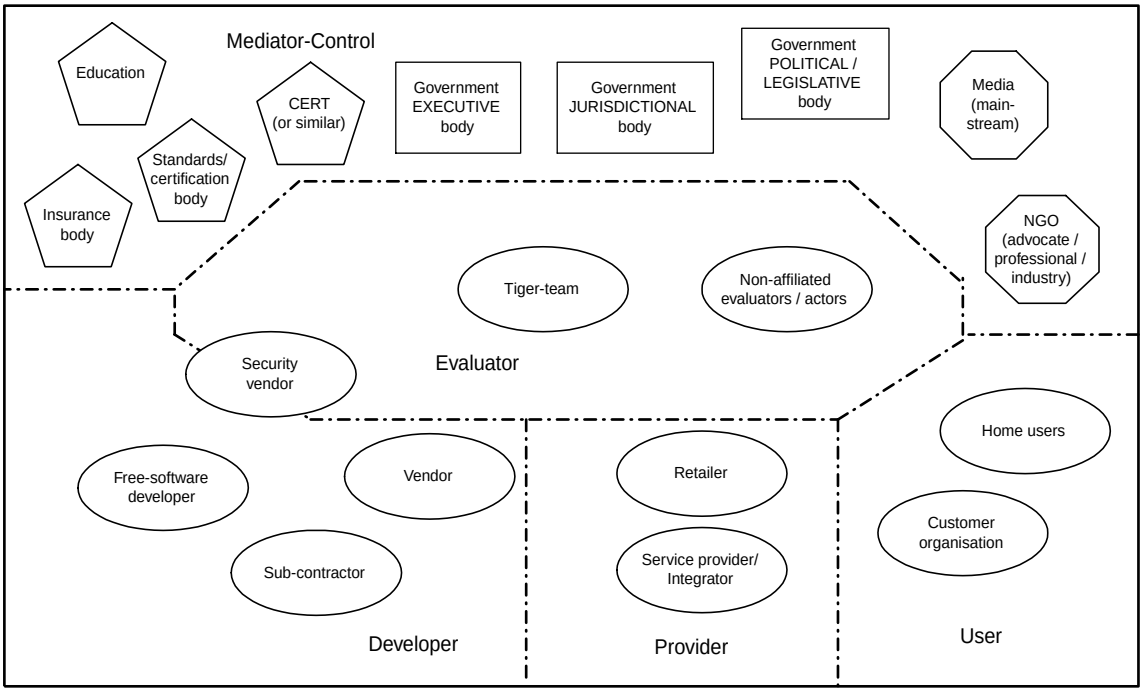

Figure 1. Major actors in typical software vulnerability processes grouped according to typical functions.

had this commercial protection installed at home according to the company check-list for installing the software. Still, a young elementary school student, Kid, found an exploit created by a person called Rogue, who had evaluated the security of the software and published a working exploit against it on a public disclosure mailing list, and Kid used it against Emil's home. The legislation of Govenia does not cover such a case, but the law enforcement caught up Kid, who is now under prosecution, but has no money for covering the damage.

To begin analysing this case study, it is first necessary to figure out the relationships between the different parties in typical vulnerability cases. Figure 1 shows a simplified diagram of the various parties in the vulnerability process, introducing a division into five actor groups: the mediator-control group, the user group, the provider group, the developer group and the evaluator group.

In vulnerability handling,[36] various new parties and problem areas enter the scene. An external evaluator studies the software with the purpose of discovering software vulnerabilities, and reports the results to the vendor, who in part notifies the necessary developers about the discoveries. The discoverer can also directly notify the public of the existence of the vulnerability, thus performing a "public disclosure". If full details are disclosed, then a "full disclosure" takes place. A non-affiliated discoverer of the vulnerability typically follows the best practices as seen on the various information channels in the security community. These usually involve writing a proof-of-concept demonstration script against the software vulnerability and reporting the vulnerability to the relevant software vendors. Depending on the response, if any, from the vendor, he usually notifies the public about the potential risk in information systems. More experienced external evaluators have their own practises that they follow in the reporting process. Organisations such as CERT/CC[37] and AusCERT[38] are able to assist in the reporting process.

When trying to point a finger at the responsible parties, the obvious starting point, as often seen in the public media, is the "malicious hackers", i.e., the attackers, and sometimes the disclosers of the vulnerability details. This obvious but oversimplified

[36] Here, by vulnerability handling we mean the part of the vulnerability process initiating from the discovery of the vulnerability by any agent involved. Typically vulnerability handling consists of the discovery/reaction, correction creation, disclosing and nullifying of the vulnerability.

[37] CERT Coordination Center. http://www.cert.org.

[38] Australian Computer Emergency Response Team. http://www.auscert.org.au.

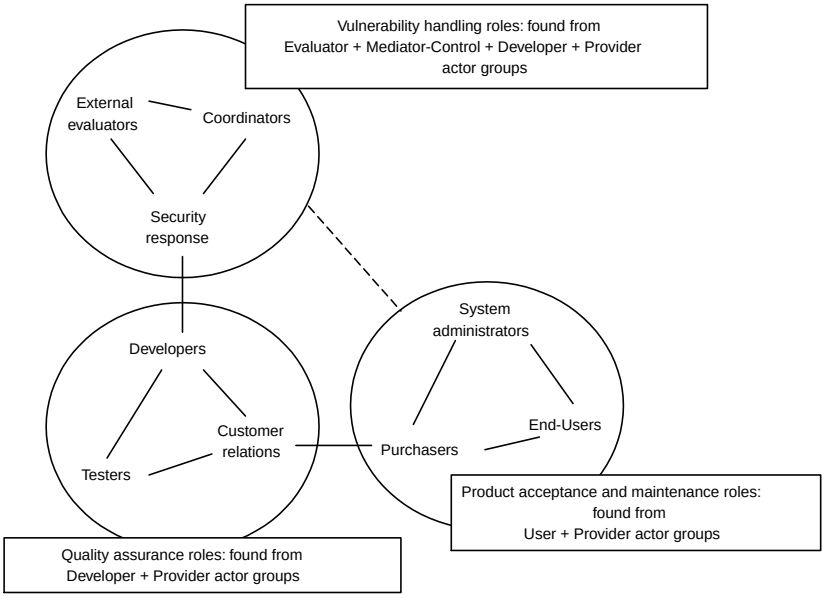

Figure 2. Main roles found from different actor groups distributing responsibilities arising from software vulnerability processes.

view would stop the analysis at these lone scapegoats. The responsibilities of the users, other evaluators, developers and providers should also be examined in the light of the case study.

However, when starting to unwrap the possible attribution of responsibility among the different parties, one soon encounters the problem of different functions and segments inside organisations or groups of individuals, called "actors" in Figure 1. Therefore, we introduce a second model. This model is described in Figure 2, where three role groups are defined by a common denominator of functionality. The roles in a specific role group can be found from different actor groups, but all roles in a role group are not necessarily found from a single actor group in our model. This allocation of certain roles to certain actors is made from the perspective of responsibility attribution in software vulnerability processes, and is neither generalisable nor complete. Furthermore, the role of management or executives is excluded from this modelling because of the unfeasibility of going into details of organisational ethics. Each role is regarded as basically autonomous without explicitly discussing the limitations in hierarchical organisations. However, if there is a necessity to use an actor as a responsible entity in our analysis, we will do so without deliberating the possible decision-making mechanisms found inside this actor. With these reservations, the main roles relevant to the study are presented and the unwrapping supported by these two categorisations of actors and roles can proceed.

User actor group

Three roles can be distinguished from the user actor group: end-users, system administrators and purchasers of software. The user actor group includes customer organisations and the users within those organisations. In the case study, Emil has an employment contract with the customer organisation, Org Ltd., which has purchased software or software solutions from both the retailer and the security vendor. The customer organisations also typically have a group of administrators that install, configure and maintain the systems, in addition to setting various guidelines of operation in the organisation. The user group also includes home users, who are regarded as independent actors comparable to customer organisations. Home users usually are responsible for maintaining their own systems. Both the home users and the customer organisations are usually bound by the licence agreements agreed upon with the purchase of the software. In cases of custom software specially built for the purchaser, these agreements are often formed when the contract for developing the software is signed. In the case of shrink-wrapped[39] software, the agreement is typically presented at the moment of purchase or installation of the software.

[39] Also called commercial off-the-shelf software (COTS).

Developer and provider actor groups

People doing software development as subcontractors typically have a contract with their client, the software vendor, where their business responsibilities and obligations are defined. For the purposes of this case study, we will focus on the moral responsibilities of the vendor packaging or developing the software, towards their customers. Also, considering that the vendor can sometimes be seen as the distributor or retailer, at least in some shrink-wrapped software, we will assume in this case study that the retailer did not acquire any legal responsibility for the quality of the software, but just transferred the agreements between the vendor and the customer organisation.

From the developer actor group, again three relevant roles can be deduced, forming the quality assurance role group. Marketing and sales form what is here called the customer relations role. Customer relations personnel typically give the facts, based on the information they receive from within the software developing company, that the customers base their purchasing decisions on. The developers, including the designers and implementors of the software, are defined as a central role in quality assurance. The role of testers is accordingly in the assessment of products handed over from the developers. Both developers and testers have a conception of the quality of the software, and are chiefly responsible for ensuring that the required quality is met with the best possible effort.

In vulnerability handling the role of security response can be found in the developer and provider actor groups. Security response personnel communicate with the external evaluators and the coordinators facilitating this exchange of information.

Evaluator actor group

Now, attention can be shifted to the evaluator actor group, and to its relationships to the user and developer actor groups. The following actor types of evaluators can be recognised, with varying levels of professionalism in their approaches:

External tiger-team: Professional and benign tiger-teams use legal means to discover faults and to notify the vendor about the discovered vulnerabilities and risks in software products. These teams are often supported by university or government funding.

Security vendor or team: Commercial security evaluators develop security products, or evaluate and certify software produced by other vendors.

Non-affiliated evaluators: Sometimes disregarding licence agreements, some motivated and skilled individuals look for faults in software. Their motivation may vary from benign and legitimate interests to an irresponsible hunt for their 15 min of fame. Their methods of reporting vary accordingly and often include ready-to-use demonstration scripts as proof of the vulnerabilities discovered. Non-affiliated evaluators with below-average technical skills, but with public access and the will to use the tools available from vulnerability databases and mailing lists, are colloquially called "script-kiddies".[40] To impress their friends and group members, these youngsters interested in the security field and hacking culture often try the available exploit scripts without thinking of the consequences.

Internal testing teams of software vendors are regarded as part of the developer actor group for the purposes of this presentation. These professional teams are, in all probability, bound by non-disclosure agreements prohibiting them from any disclosure of their findings outside their commissioning organisation.

Professional evaluator teams have sometimes applied professional codes of ethics and other guidelines within the group. In cases of non-professional evaluators, the risks are extremely high, as the process and communication are typically uncontrollable and a published vulnerability can cause surprising financial losses when caught by the public media. These evaluators, with sometimes malicious intent, can disregard or be outside the jurisdiction of modern legislation against reverse-engineering and thus have a wider choice of tools available than the professional evaluators using legitimate testing methods. The ethical commitment of unprofessional evaluators may vary, resulting in methods of operation ranging from considerate and sound practices to outright malicious actions intended to cause harm to other parties. If and when a security assessment or a security solution is made into a commercial product, the evaluator faces the same responsibilities towards their customers as any other commercial company.

Mediator-control actor group

Although several actors can influence vulnerability handling processes, the relevant actors in the mediator-control actor group in particular are the entities actively participating in the communication process. Although the focus of organisations such as national CERTs has traditionally been on incident handling, they nowadays actively participate in vulnerability

[40] Script-kiddies can be seen as evaluators of system maintenance and security practises, i.e., they typically attack known errors which should have been repaired by the maintainers.

handling as well. With dedicated resources and existing connections to several software developing organisations, including commercial vendors, they can connect the relevant people in a coordinated and safe manner, and can help in the assessment of the related risks and coverage of the problems. All necessary vendors are contacted by means of personal communication, and most system administrators follow the mailing list used to report on the risks, work-arounds or fixes.

Publicly available mailing lists such as Bugtraq and NTBugtraq, typically involve an active group of people participating in the moderation and organisation of the lists. A moderator of a public disclosure mailing list can assist in the process of vulnerability handling or decide to control the disclosure by postponing it to a later time. The moderator can also influence the reporter by discussing about the level of information given and the professionality of the reporting approach. Even with active moderation, these lists can generate from hundreds to thousands of messages per month, and thus are not something the big audience, or even all vendors and administrators, follow.

Thus, the role of the actors in the mediator-control group is to coordinate the communication process and information management in the vulnerability handling process. If requested, other parties, even external evaluators such as commercial security vendors, can also take the role of the coordinator and may use their own dedicated people as coordinators in reporting the security or safety risks they notice.

Responsibility attribution to different roles

To clarify the attribution of moral and professional responsibilities in the five actor groups involved in software vulnerability processes, three different role groups were identified. We will focus on the roles related to actual vulnerability handling, but will also superficially describe the other two role groups: quality assurance, and product acceptance and maintenance.

Disclosure level ethics presentations leave ethical analysis on a basic, usually common sense level in order not to push a phenomenon into a narrow predetermined category imposed by an elaborate ethical theory.[41] However, in order to disclose the relevant factors for further, more thorough normative analysis of responsibility attribution, we have utilised the three previously introduced conditions for retrospective responsibility attribution. These conditions are (1) the existence of prospective responsibility, (2) the intentionality condition and (3) the causality condition.[42]

We will focus mainly on the various, most often professional roles in our preliminary analysis of responsibility attribution in software vulnerability processes, but will also use the category of actors where necessary as dictated by common sense.[43]

Quality assurance roles

In quality assurance, found in the developer and provider actor groups, we identified the roles of developers, testers and customer relations.

The customer relations role entails the marketing, public relations and sales personnel of the aforementioned actor groups. The customer relations personnel depend on the information provided by the technical experts of their organisation and possibly the statements of the previous actors in the marketing chain. In addition to this, concerning the responsibility aspect, publicly available facts also have to be taken into consideration. Customer relations personnel have to seek, receive and convey valid knowledge of the products, and remain truthful to their customers in known security aspects while in their natural pursuit of profit. Here, the effort of the mediator-control actor group, e.g., in the form of various standards or the information flows from different sources, play an increasingly important role. To summarise, the intentionality condition is an essential factor. Customer relations personnel can be seen to be bound by the moral principles dictated by common sense, even if they did not have formal and explicit codes of conduct regarding their profession. The causality condition is often easily demonstrated as the purchasers of software products rely on the given information on product security and quality. In the case of open source and non-profit software, the role and motivation of customer relations blurs, although even if striving only for fame, the burden of responsibility remains.

In the case study presented, the main responsibility within the role group of customer relations falls on the security vendor actor, whose marketing claims turned out to be without credibility. If any express

[41] Brey (2000).

[42] Vedder (2001).

[43] Guidelines and obligations for providers/developers and customers are presented in e.g.: W. Robert Collins, Keith W. Miller, Bethany J. Spielman and Phillip Wherry. How Good is Good Enough? *Communications of the ACM*, 37(1): 81–91, 1994; Also commented on by Baase 1997, pp. 344–346.

warranties were actually made by the security vendor, they naturally are the crucial arguments. The original vendor of the software produced vulnerable bad-quality software, but if the vendor did not make any groundless dependability assurances, it may be argued to be less accountable in comparison with a security vendor that claimed to provide add-on security. However, if the original vendor's customer relations knew about, or even had a reason to suspect the existence of vulnerabilities and still proceeded to market their product without proper disclosed reservations, the moral appraisal obviously changes.

Developers and testers of software have an obvious moral responsibility to ensure to their best ability that their products are safe to use.[44] Secure programming and safety engineering skills are essential for professionals developing dependable software. Abnormal situations need to be tested besides the traditional approach of testing and validating conformance to requirements.[45] However, as previously discussed, a risk analysis has to be made considering the probable use of the product. Does a software malfunction during its use lead to safety-critical situations that threaten human lives/health or are the repercussions limited to economical losses or just to the suffered nuisance? What level of robustness is adequate for the use the product or system is intended for, and how carefully should these proper areas of use be defined by personnel in the roles of quality assurance? The intentionality condition fails if the testers and developers of software cannot sufficiently define the areas of usage for their products, even if the causality condition is in effect and the moral principles of professional ethics clearly apply.

If in the case study the vulnerable software was intended to be specifically used in an industrial process defined by the quality assurance of the software vendor, the customer organisation may have a hard time trying to point out vendor responsibilities in a malfunction occurring in private home use. Instead, the customer organisation's responsibility to its employee may come into consideration. In this case study, however, we assume that the vendor had not strictly specified the product's area of use, but had intended it as a malleable, general-use heat controlling software component.

The implications of professional ethics place heavy responsibilities on the individual software engineer or manager of software engineering, who has to try to balance these ethical requirements with obligations to the commissioners of his work. Withholding information of known security vulnerabilities is clearly unacceptable. However, if applicable standards are lacking, deciding when an adequate level of software robustness has been achieved is more ambiguous. In the case study, it can be argued that all stages of the software development process should have met the qualification recommendations as applied to safety-critical systems, that is, as if the product were made for the most challenging usage environment conceivable.

Roles in product acceptance and maintenance

In product acceptance and maintenance, the roles of the administrator, purchaser and end-user were identified. These roles are found from the user and provider actor groups, but in reality can also be found in other actor groups, as well.

System administrators have the responsibility of upholding the level of security in their systems. Maintenance here differs significantly from less malleable products in that with software, vulnerabilities are constantly surfacing to be reacted upon. However, all customer organisations or individual home users may not be able to devote a sufficient, often excessive, amount of time to track the bad news. The level of system administration is a major contributing factor for intrusions, as some organisations repair their installations rather late, if ever.[46]

In the presented case study, the customer organisation's administrators had included the supposedly adequate protection for the exploited vulnerability in the company check-list and thus are excused from responsibility. The only window for responsibility attribution on them would be a situation where they were commissioned to constantly monitor the same public disclosure mailing list where the exploit was published by Rogue, and either left the protective measures disclosed by Rogue unimplemented or if they had the technical capability to independently protect their systems, left that undone. However, we assume that Rogue did not include any protective measure to counter his demonstration script, and the administrators did not have the skills or resources to nullify the vulnerability on their own.

Customer organisations' purchasers and the home-users in this role have their share of responsibility[47] in assessing the suitability of the software to the function it is used in and in ensuring resources and guidance for the proper usage and maintenance

[44] See e.g. Baase (1997, pp. 346–348).

[45] Leveson (1995, pp. 158–159).

[46] William A. Arbaugh, William L. Fithen and John McHugh. Windows of Vulnerability: A Case Study Analysis. *Computer*, pp. 52–59, December 2000.

[47] See e.g. Collins et al. (1994); and Baase (1997, pp. 344–346).

of the product. In assessing the suitability of the software product, the purchaser's difficulties in ascertaining the security attributes of the product are obvious. Usually the purchaser is very heavily dependant upon the information provided by others and without the possibility of independent security assessment of the product.

The concept of informed consent used in medical ethics has been applied to the decisions to purchase and use software as well.[48] Generally it means that the patient is informed of the probable risks and benefits involved in a procedure, its possible alternatives and is able to make knowledgeable decisions on the matter. When software purchasing decisions are made, from the security perspective these conditions cannot be satisfied if commonly understood information or criteria in assessing the security level or robustness of the product is lacking.

Vulnerability handling

Here, by vulnerability handling we mean the part of the vulnerability process initiating from the discovery of the vulnerability by any agent involved. Various phases and life-cycles of software flaws and vulnerabilities have been formulated.[49] For the purposes of this study, aimed to be a neutral starting point for further analysis, we divide vulnerability handling into the phases of (1) discovery/reaction, (2) correction creation, (3) disclosure and (4) nullification of the software vulnerability. The phases do not necessarily succeed each other in a chronological order and are not all even necessarily present, with the exception of the discovery/reaction phase. We recognise that the creation of a ready-to-use script exploiting the vulnerability can take place in any of the phases, the crucial factor being whether and in what manner it is disclosed or used. Earlier, the roles of external evaluators, security response and coordinators were identified in vulnerability handling. The phase of nullification also includes roles from the product acceptance and maintenance role group.

Main moral principles applied in vulnerability handling

There are some relatively widely accepted formal moral principles and guidelines that can be applied when considering vulnerability handling by the various roles identified. Of course, the clear stipulations of the IEEE and ACM/IEEE-CS ethical codes concerning the responsibility to find, disclose and cooperate in correction of flaws apply, as well as similar clauses for accepting professional criticism and giving due credit where it is appropriate. The core principle of these codes is to promote the well-being and safety of the public.[50] However, these codes do not deliberate on the responsible disclosure of information regarding flaws in the specific conditions of software vulnerability disclosures when the benefit-risk relationship of this action is not clearcut.

Two applicable principles in this matter can nevertheless be found from the Organisation for Economic Co-operation and Development (OECD) Guidelines for the security of information systems. This recommendation introduces the principles of awareness and democracy, which, according to the definitions in the guidelines, can be interpreted as applying to vulnerability handling and processes. The awareness principle, when at first including an objective to promulgate information dissemination of the security of information systems to agents with legitimate interest, clearly states that this should be done in a manner which does not compromise security. Alleviating this restriction, the democracy principle nevertheless concludes that the security interests of various actors must be counterbalanced by the principles of a democratic society concerning the flow and use of information.[51]

The main content of the principles presented above can be condensed to the principles of beneficence and non-maleficence. The common sense definition of beneficence is the providing of good to others, whereas non-maleficence means the avoidance of causing harm to others. In the medical setting, non-maleficence implies that necessarily some amount of pain has to be inflicted upon the patient to attain greater good, but the agent's responsibility is to minimise this harm inflicted.[52] This is readily applicable to software vulnerability handling, where inevitably because of practical limitations, some window of opportunity for malicious activity will manifest itself regardless of the method of vulnerability handling opted for. The intensity by which

[48] Collins et al. (1994).

[49] Laakso et al. (1999) and Arbaugh et al. (2000) See also e.g. Steve Christey and Chris Wysopal. "Responsible Vulnerability Disclosure Process", a currently withdrawn IETF draft dated February 2002.

[50] See e.g. Gotterbarn et al. (1999).

[51] Organisation for Economic Co-operation and Development. Guidelines for the security of information systems. November 1992. http://www1.oecd.org/dsti/sti/it secur/prod/e_secur.htm.

[52] See e.g. Tanya Fusco Johnson. Ethical Issues: In Whose Best Interest? In Tanya Fusco Johnson, editor, *Handbook on Ethical Issues in Aging*, pp. 17–18. Greenwood Press, Westport, Connecticut London, 1999.

these two principles are applicable to agents in vulnerability handling depends on, among other factors, whether they consider themselves as professional or not. For unprofessional agents the demands set by common sense can be argued to be essentially similar to professional ones, but less intense.

Discovery/reaction phase

One role that could be considered to share responsibility for software security incidents is formed by external evaluators in the case of not practicing a professional approach, i.e., not adhering to professional ethics and accepted practises in the disclosure of the vulnerability, and more so in cases of demonstrating the vulnerability in a hostile manner. The worst example of reacting to discovered vulnerabilities is taking advantage of them without any disclosure for evaluation by other related parties.

Although software vulnerabilities are sometimes found during the normal usage of the product, more often deliberate testing is the method leading to the discovery. In testing activities, methods such as reverse-engineering that do not always have a clear legislative status should be avoided, and used only when the legal status has been ascertained or as a last resort measure for compelling reasons. External vulnerability analysis of commercial software can also be done by using functional[53] testing methods, i.e., without knowing or reverse-engineering the internal structure of the software.

Reporting of a vulnerability can quickly follow its discovery, but sometimes a lengthy verification period is needed to show that the vulnerability is valid and that all technical facts are correct before approaching the developer of the system or software. A well prepared bug report can increase the chance of a prompt response.

Vendor policies on handling vulnerability reports cover issues such as the acknowledgement timeframe and methodology. Having a single point of contact for vulnerability reports helps in controlling and handling the reporting process so that all reporters receive a reply in a timely manner. If a vulnerability appears to be low-risk to the system in question, the nature of the response sent to the reporter can make a difference in his reaction to the possible delays in the repair process.

Various coordinators in the software vulnerability handling process promote various guidelines for reporting vulnerabilities to the developers. Disclosure policies of mailing-lists and organisations specify the used timeframe, e.g., the grace period that the vendor is given for preparing the corrective measures and customer notifications or advisories. The availability of contacts and resources is necessary if an organisation takes the responsibility of coordinating the handling of a vulnerability. Networks of coordinating organisations can handle a vulnerability covering several reporters and vendors located in different countries.

Correction creation phase

Discovered vulnerabilities should be fixed in a coordinated and healthy manner without unnecessary publicity and consequent risk of criminal activity. Although this phase mostly involves the developers of software, sometimes urgent attention is required from the evaluators and coordinators.

A lengthy process is sometimes necessary to develop a good means of correcting the vulnerability, and the main responsibility for the external evaluators is in verifying that the final correction closes the vulnerability. This sometimes requires urgent action if the developer does not reveal the schedule for the evaluator to prepare for.

The more open the developer is towards both the evaluator and the coordinator in the actual correction creation phase, the better the cooperation typically can be. Adequate quality assurance methods can decrease the probability of similar errors existing elsewhere in the same product or re-emerging in the future versions. Explaining these reasons to the other involved parties can again increase the chance of successful cooperation.

When a vulnerability case involves more than one company, either several developers, or a combination of providers and developers, the communication process often requires coordination and mediation. A neutral party is often also necessary when several countries are involved, perhaps even with government or military involvement or interest in the vulnerabilities.

Disclosure phase

The window of vulnerability typically starts from the public or limited disclosure.[54] Minimising the window of opportunity for malicious intrusions, while at the same time effecting as wide a security patch-up of the discovered vulnerability as possible, requires systematic approaches. Different models of vulnerability disclosure have been tried out in order to hold the exploitability in check.

[53] Also called black-box testing methods.

[54] Although it can be argued to start from the creation of the fault behind it, as potential mishaps do not always require malicious activity.

Unconstructive publicity can end up in loss and damage to the actors in the developer, provider and user groups, thus also causing the evaluators an unnecessary risk of liability for damages. Publishing the vulnerability details, when the vendor leaves no other option by consistently ignoring or downplaying the vulnerability, may be necessary as a last resort to promote public interest, which means considering the health, safety and welfare of the public.[55] In the case study, the external tiger-team appeared to act professionally in informing the vendor of the vulnerability and in giving the vendor the time it needed to react to the situation, rather than immediately seeking publicity or a rash way out of the situation.

Not all parties in the evaluator-tester group have internalised considerate methods in disclosing security vulnerabilities, but instead disclose them perhaps spontaneously, or according to their more selfish motives.

In the case study the most obvious culprits were the unprofessional evaluators; the script-kiddie and the publisher of the vulnerability details. They, in the ethical sense, carry the weight of responsibility of their actions in proportion to their understanding of the results. The actions of the publisher of the vulnerability details and attack scripts are somewhat defended, even in public mailing lists, by the fact that without the publicity of the security hole, the vulnerability would have been left untouched, unrepaired, and furthermore left to spread even wider to other software systems. However, notifying the vendor in advance, allowing them some time to prepare a fix, is promoted in the full disclosure policies available.[56] In spite of this, a wide consensus as to what amounts to a responsible disclosure policy or enforceable guidelines is yet to be attained between different actor groups.

For the security response role of the actors in the developer and provider actor groups, the main tasks in the disclosure phase include, naturally, the disclosing of relevant information through their own channels and possibly using mainstream media as well. Delivering the security advisories and the available corrections to customers as effectively as possible would be a central goal. Additionally, the security response personnel should offer support and guidance to their customers.

In the case study, the software vendor had not issued a security advisory containing any interim workarounds alleviating the exploitability of the vulnerability, even when several months had passed since the tiger-team had reported the problem. If they had the technical capability to find or knew about the interim workarounds, but decided not to disclose them before their actual security patch was ready and tested, it could be argued that they share their part of the moral responsibility for the security incident. However, as mentioned before, the publication of any corrective measure to mitigate a vulnerability carries with it the risk of drawing malicious interest and evaluation in order to exploit the vulnerability. Even if this corrective measure did thoroughly protect the systems against this vulnerability, there are systems where, for one reason or another, the corrective measure is not taken into use.

The role of security response, in both the developer and provider actor groups, and the role of coordinators of vulnerability handling, typically an organisation such as CERT/CC or a moderator of a mailing-list, is to control the communication process. Failure to interact with the reporters often results in uncontrollable vulnerability handling, confused or angry customers and unnecessary windows of opportunity for malicious actions. As the Figure 3 shows, it is possible to attempt to limit the communication around the evaluator actors by using the developer and mediator-control actor groups' channels to reach the actors in the provider and user groups.

Nullification phase

In the nullification phase the main task falls to the role of system administration of the relevant actors in the user and provider actor groups. In cases where security updates for the software product would have been available, it has been argued that the possible security incidents were due to bad practices of security by the system administrators. Some obligations and responsibility may filter down to the role of end-user. If the end-users clearly violate the correct procedures in software usage, they can be partially responsible for the security violations.

Databases of past vulnerabilities[57] provide means for collecting the details of past vulnerabilities and learning from them. This provides incentive and possibilities for the quality assurance of developer and provider actor groups, but also to evaluators and nonaffiliated actors with various motives.

[55] See Gotterbarn et al. (1999).

[56] See e.g. Full Disclosure Policy (RFPolicy) v2.0. http://www.wiretrip.net/rfp/policy.html by Rain Forest Puppy; The CERT Coordination Center Vulnerability Disclosure Policy. http://www.cert.org/faqvuldisclosurepolicy.html.

[57] See e.g. SecurityFocus http://www.securityfocus.com and the Mitre http://www.mitre.org.

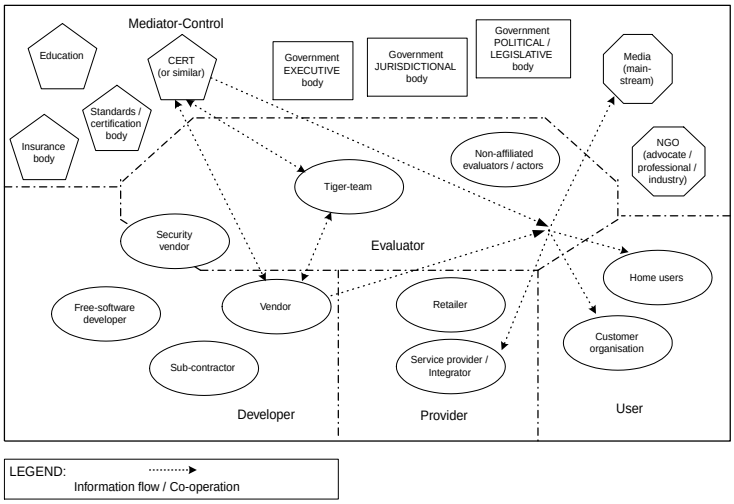

Figure 3. Tiger-team actor's (external evaluator role) triangle co-operation method with the software vendor (security response role) and the CERT organisation (coordinator role) in vulnerability handling.

Realities of software vulnerability processes

The evaluators' reporting of discovered vulnerabilities to the public appears to have little effect in general. The typical vendor approach is to quickly fix the exact notified vulnerability and not the entire poor quality of the software module, thus leaving other vulnerabilities in the software or even reintroducing the problems unfixed in later versions. Improvement of the general quality of software is slow. This may be due to the number of non-professional developers in the field. Even people who have been in software development for decades sometimes miss the whole area of robust software and the dangers that non-robust software brings.[58]

The vendor is responsible, and hopefully sometimes liable, to the whole public, in addition to clients, for bad quality software. However, the current trends in legislation[59] may in effect make an exception in the common (Western) practise of vendor/developer liability when applied to the software industry. These new legislative drafts would in practise move the whole responsibility of software malfunction, including security incidents, to the licence-bound customer. Furthermore, disclosing flaws in software could be restricted only to those cases, where the software vendor gives permission to do so. Vendors would not be required (by law) to disclose even the known flaws in their software. These last two propositions would be in clear violation of the software engineering code of ethics.[60]

From the vendor's perspective, there is a clear rationality behind these aspirations, which possibly have one of their main justifications in "computer malleability", a concept introduced by James Moor.[61] The multitudes of operating environments and possible modifications of software products dwarf the variability of more conventional products, making it harder to foresee the proper quality requirements for software.[62]

If the liability for possible malfunctions in software is transferred to customers, software vendors can release unpolished software to be tested in the field. Nevertheless, when considering society as a whole, this obviously is not acceptable. Developers and vendors of software should be, within reasonable limits, held also legally responsible for the quality of their products and for losses their clients suffer because of their software. In cases of widely deployed software, the moral responsibility is towards the whole society. Additionally, vendors and developers can be held responsible for not acting professionally in repairing the problems they become aware of in their software.

Professional security evaluators of software also have a very thin line that they tread on in order to maintain professional ethical integrity. For example, the coverage of vulnerability testing of software may be biased, in effect providing negative publicity to

[58] Leveson (1995, pp. 156, 233–234).

[59] Kaner (1999).

[60] ALERT: a danger to the Public and a danger to the development of Safe Quality Software in new legislation. A white paper from the Software Engineering Ethics Institute (SEERI) to software professionals. http://www.seeri.etsu.edu/WhitePaper.shtm.

[61] James Moor. What is Computer Ethics. *Metaphilosophy*, 16(4): 266–275, 1985.

[62] See e.g. Baase (1997, p. 347).

some software vendors but leaving other vulnerable products by different vendors untouched. This biasing can even be deliberate, as can of course be the timing of various disclosures. Professional security evaluators/vendors may also be tempted to withhold information of discovered vulnerabilities at their disposal, so as to retain their competitive edge in relation to rivaling companies. The good of the external security evaluator is not necessarily exactly the same as the good of the public, especially when operating on commercial prerogatives.

The possible role of neutral third-party organisations as coordinators and intermediaries in vulnerability handling has been discussed and has sometimes been taken into practise. Including a reliable international organisation as an overseer of information flows in a process, especially when dealing with public channels and media, may lessen the software vendors' apprehensions toward external security evaluators. The roles of various other mediator-control actors are also discussed regarding the whole continuum of software vulnerability processes. One neutral party anticipated to have a strongly increasing role are the insurance companies covering the risks of customer organisations using software applications.[63] Often being the partial bearers of economical losses inflicted in software security incidents, the insurance companies would have a natural interest in keeping these incidents at bay.

Conclusions and future work

In this article the main features of a fictional software security vulnerability process were discussed, from which some aspects were clarified using professional ethical codes, some widely accepted principles for responsibility attribution and foremost of all, common sense as reference points. The presented case study was not constructed to represent a typical software vulnerability process, but rather to exemplify the complexity often inherent there and to introduce the commonly involved agents. However, the main actors and roles were then defined in a way to reflect their typical functions and connections in software vulnerability processes. It is apparent that these processes and the compromises inherent in the current level of software security are not well known to the public at large. It is hoped that this disclosive ethics presentation would prompt a second stage in which more specific normative evaluations of software security issues are conducted.

In the fictional case study presented, five distinctive actor groups were identified: developer, provider, user, evaluator and mediator-control. From these actors three role groups were identified and further analysis was focused on the roles responsible for the actual handling of the discovered software vulnerability. It was noted that the crucial aspects in software vulnerability handling were the approach taken by the external evaluators in the dissemination of vulnerability details, the manner in which the vendor acknowledges and reacts to this information and the inclusion of a neutral third party as a facilitator and coordinator in this process. The overall impact is felt in the user actor group, where the role of system administrators is to react to reasonably well formulated instructions resulting from the vulnerability handling process. In software quality assurance, security maintenance of products and the veracity of information concerning the product offered to purchasers were identified as major aspects to be considered. In all these activities the ethical codes of professional software engineers apply, if the agents under scrutiny claim to take a professional stance. Also the principles included in the Guidelines for the security of information systems adopted by the OECD member countries are applicable in this area. However, the practical concretisation of these recommendations seems to be unrealised in software vulnerability processes.

In developing software products for use where there is a safety issue or even a risk of losing human lives, there traditionally have been high standards and strict control of the quality of the services and products.[64] On the other hand, in the field of software engineering with little experience with safety issues, the quality of software has been noted to be poor from the security perspective. The current situation shows that generically used software engineering processes and quality assurance methods are still immature as far as the security factor is concerned, and professional ethics do not yet give explicit guidelines for the handling of software vulnerabilities. Security and safety problems with software are far too often seen as accidents, without demands for accountability for the losses due to bad quality software.

In software engineering, professional obligations are still not based on accepted standards. Many times the responsibility to find out sound practises in software vulnerability handling processes seems to fall on the individual, the ethically committed software engineer or manager, and groups thereof. At the societal level, software vendor/developer organisa-

[63] See e.g. del Amo Calvo, (2000).

[64] See e.g. Leveson (1995).

tions do not yet seem to have an effective counterpart of equal size and strength forcing them to adopt transparent, generally acceptable practices, especially if legislation bypasses their liabilities. The situation in software development seems too often to be similar to that in vulnerability handling. In the absence of official or organisational guidelines the professionals must refer to their sense of moral responsibility to ensure sufficient dependability and quality of software. Demands for better dependability of software have also been increasingly voiced by significant customer entities. However, the irregularities in the approaches adopted by external evaluators of software are also problematic from the vendor/developer point of view. Regulative and standardisation activities in these areas can safely be predicted to be on the rise, according to the normal maturing of novel technologies and their integration into society.[65]

The risks involved in the ever-growing software industry, and the security compromises they pose for the individual or society, are not yet disclosed to or understood by the greater public. Nor do these risks seem to have quite the same popular appeal as more traditional objects of attention, such as environmental threats caused by other industries. One subject that was not thoroughly discussed, and which often draws the concern of the security community, is the threat that a software application gains too wide a population in society. The great majority of the security community agrees that a lack of diversity can cause much distress, and the security threats it brings are enormous. Because of its overt permeation and implicated major repercussions, a vulnerable application does not even have to be used in critical environments to pose a major threat to infrastructure.

The general public relies on and trusts in the products that traditional software engineering produces. Is the trust placed on software engineers blind and without any questioning? The real problem is that today the responsibilities of software engineers as professionals in the ethical dimension are yet not publicly recognised and enforced, when compared to professions such as lawyers or various vocations in health care. At the same time, enabling technology is making these other professions increasingly dependent on the integrity of software engineers. Still, the responsibility of the software industry to society at large is not scrutinised in the same magnitude or manner as is the case with many other industries or segments of society.

How the ethical problems in software security can be solved is worth some profound research effort on the application (and possibly theoretical) level as defined in Brey's concept of disclosive computer ethics methodology. We have adapted and created some terms for the purposes of this pragmatical study, but they are not intended as authoritative definitions. The various responsibilities and professional obligations of agents in various roles involved in software vulnerability processes should be argued for and specified in more detail. Approaches to effectively improving the quality of security-critical software and widely distributed software packages can also be formulated and promulgated, taking into account the overall impact software vulnerabilities have on the infrastructure of modern society. Finally, these considerations and the agreements formed upon them should be enforced in standardisation, certification programmes and legislation.

[65] See e.g. Baase (1997, pp. 140–141, 345, 347).

Acknowledgements

The authors thank the staffs of the Secure Programming Group at the University of Oulu (OUSPG) and Codenomicon Ltd. for their extensive help and participation. We also offer our gratitude to the AusCERT and CERT/CC for their patient support in resolving the vulnerability cases that led to the development of this article. Special thanks to Lic.-Tech. Rauli Kaksonen, Prof. Gerald Quirchmayr, Prof. Jorma Kajava and Prof. Mikko Siponen for their supportive comments on this issue. This research is part of the PROTOS project.

References

A. del Amo Calvo. The Liability of Professional Experts Like Risk Managers. In F. Galindo and G. Quirchmayr, editors, *Advances in Electronical Government*, *Pre-Proceedings of the Working Conference of the International Federation of Information Processing WG 8.5 and the Center for Computers and Law*, Zaragoza, Spain, 10–11, February 2000.

W.A. Arbaugh, W.L. Fithen and J. McHugh. Windows of Vulnerability: A Case Study Analysis. *Computer*, pp. 52–59, December 2000.

S. Baase. *A Gift of Fire: Social, Legal and Ethical Issues in Computing*. Prentice-Hall Inc., 1997.

P. Brey. Method in Computer Ethics: Towards a Multilevel Interdisciplinary Approach. *Ethics and Information Technology*, 2(2): 125–129, 2000.

W.R. Collins, K.W. Miller, B.J. Spielman and P. Wherry. How Good is Good Enough? *Communications of the ACM*, 37(1): 81–91, 1994.

D. Gotterbarn, K. Miller and S. Rogerson. Computer Society and ACM Approve Software Engineering Code of Ethics. *Computer*, 32(10): 84–88, 1999.

T.F. Johnson. Ethical Issues: In Whose Best Interest. In T. F. Johnson, editor, *Handbook on Ethical Issues in*

Aging, pp. 17–18, Greenwood Press, Westport, Connecticut London, 1999.

C. Kaner. Software Engineering and UCITA. *Computer & Information Law*, 18(2), 1999.

M. Laakso, A. Takanen and J. Röning. The Vulnerability Process: A Tiger team Approach to Resolving Vulnerability Cases. In *Proceedings of the 11th FIRST Conference on Computer Security Incident Handling and Response*, Brisbane, 13–18 June 1999.

J. Ladd. Computers and Moral Responsibility: A Framework for an Ethical Analysis. In C. Gould, editor, *The Information Web: Ethical and Social Implications of Computer Networking*, pp. 207–227, Westview Press, Boulder, Colorado, 1989.

N.G. Leveson. *Safeware: System Safety and Computers*. Addison-Wesley Publishing Company, 1995.

J. Moor. What is Computer Ethics. *Metaphilosophy*, 16(4): 266–275, 1985.

P.G. Neumann. *Computer-Related Risks*. ACM Press/ Addison-Wesley Publishing Company, 1995.

A. Takanen, M. Laakso, J. Eronen and J. Röning. Running Malicious Code by Exploiting Buffer Overflows: A Survey of Publicly Available Exploits. In *Proceedings of the 9th Annual EICAR Conference*, Brussels, Belgium, 4–7 March, 2000.

A. Vedder. Accountability of Internet Access and Service Providers – Strict Liability Entering Ethics? *Ethics and Information Technology*, 3(1): 67–74, 2001.