Introducing constructive vulnerability disclosures

Marko Laakso, Ari Takanen, Juha Röning
University of Oulu, Computer Engineering Laboratory
PL4500, FIN-90014 University of Oulu, Finland
{fenris,art,jjr}@ee.oulu.fi

ABSTRACT

Product flaws that compromise information security emerge constantly, and a vivid debate is taking place on how these vulnerabilities should be handled. A partial disclosure concept, constructive disclosures, was introduced as an alternative to full disclosures and as a safety-net against reoccurring vulnerabilities of a similar kind. The proposed model was executed in a multi-vendor, multi-vulnerability case involving WAP gateway products. A complicated vulnerability case was successfully handled, with positive feedback. This result promotes the seeking of solid engineering practices that will take the vulnerability process beyond an art form.

1. Introduction

Flaws that compromise information security are constantly being discovered in a variety of products, and the minimising or elimination of the potential for damage that these flaws may cause appears to be a complicated issue. Further complexity can be anticipated when handling defects revealed in products that have penetrated our infrastructure, products such as widely spread consumer appliances and the building blocks of telecommunication networks. There has been a lively and sometimes heated discussion and debate over procedures for reporting and managing such vulnerabilities, especially regarding the concept of full disclosure.

Earlier work by our research group [12] illustrates a systematic approach to reporting and resolving vulnerability cases, proposes a life-cycle model with checkpoint-based metrics, identifies actors and their roles in the scene and provides definitions for the related terminology. The vulnerability process has been described and improvements proposed by Ivan Acre from CORE-SDI [1].

Experts in the security field have contributed to the discussion on how to handle vulnerability disclosures. Rik Farrow gives an in-depth historical perspective on disclosing vulnerabilities [7], and Crispin Cowan has voiced a request for a vulnerability escrow by involvement of a neutral 3rd party [6]. Sarah Gordon contrasts the full disclosure approach adopted by many practitioners with the *no-disclosure*, or at best *partial disclosure*, approach practised by the anti-virus community [9]. The discussion re-emerged recently after Marcus Ranum promoted no-disclosure as a way of reducing the number of script kiddies [21]. A strong stance in favour of full disclosure has been taken by contributors such as Weld Pond and Mudge [16] [14], while Bruce Schneier sketches the phases of the vulnerability process and analyses the full disclosure debate [22].

Policies for handling vulnerability disclosures have been put forward by Russ Cooper (the moderator of NTBugtraq) [5], the CERT Coordination Center (CERT-CC) [3], Rain Forest Puppy (RFP) [20] and vendors such as Microsoft [13]. RFP disclosure policy stems from the perspective of an active originator who discovers and discloses vulnerabilities, whereas the NTBugtraq and CERT-CC policies represent the views of coordinators of the vulnerability process, who in this role receive vulnerability details and attempt to oversee and aid the process of resolving the cases. Microsoft policy reflects the company's role as the repairer of vulnerabilities found in its products.

The vulnerability process can be observed in practice on active (full) disclosure mailing lists such as BugTraq [23] and NTBugTraq [15]. These and other mailing lists specialising in any level of disclosure discuss the different types of disclosure and promote certain disclosure policies promoted for use.

Models for partial disclosure, which represent a viable compromise, have not been explored in depth, and the existing models do not address the life of a vulnerability after an advisory or patch release. These later stages in the life-cycle should attempt to address the problem of exactly the same vulnerabilities reappearing in new products, perhaps by considering methods of providing current developers with information on past implementation mistakes. Moreover, more attention could be paid to supporting the use of vulnerability information in customer-driven evaluation or acceptance testing.

Purpose of this work is to introduce an alternative partial disclosure concept that attempts to support all stages in the vulnerability life-cycle. This novel concept of *constructive disclosure* is based on test-suite releases. Special emphasis is placed on the customer perspective.

This paper first introduces the *constructive disclosure* model based on test-suite releases, and then presents the results of applying the concept to a multi-vendor, multi-vulnerability case concerning wireless application protocol (WAP) suite implementations. Thirdly, experiences with the WAP case-study are reflected against the original goals and analysed from the viewpoint of other disclosure policy proposals. Finally, the discussion is presented in a broader perspective and conclusions are drawn.

2. Constructive disclosures

A constructive disclosure manifests itself through a test-suite release process. Traditionally, publicly available test-suites have been used to verify the standard compliance and conformance of a product [8]. In the context of this paper, test-suites aim at providing material to enable both customers and vendors to evaluate software for security-related vulnerabilities and for robustness. A test-suite will wrap up code and data to form a stimulus with a potential for revealing vulnerable behaviour in a product.

2.1. Goals

A test-suite aims to set a baseline that will eliminate a specific subset of errors that could infest implementations of a specific functionality. The test-suite may be used to identify and resolve vulnerabilities in products in use today or for the early elimination of some of the trivial pitfalls that appear during the development process. A test-suite may be adopted as a part of the *regression test-suite* to ensure that the errors covered by it do not emerge during code maintenance.

When customers evaluate a commercial off-the-shelf product, they may use the test-suite as a set of criteria for accepting the product (*acceptance testing*). The same applies to a custom-made product, although a more effective net outcome may be achieved by insisting on an ability to survive the test-suite in the requirements specification phase. A test-suite may be used as a supplementary metric for evaluating several competing products in the process of making purchase decisions (*product evaluation*).

The test-suite should be able to eliminate some of the most trivial vulnerabilities that may otherwise be found only by manual trial and error. This may free some of the resources of the security community and end users to be spent on more fundamental (information security-related) issues.

2.2. Phases of a constructive disclosure

During the *creation phase* the focus area for the test-material is selected, the test-material is created and related code and data are packaged. Internal testing is conducted against available products implementing the functionality covered by the test-cases. This phase maps to the research and verification stages as proposed by us in [12].

During the *pre-release phase* the respective vendors are notified of any vulnerabilities discovered. The test-suite is distributed to the identified vendors implementing the functionality concerned. Patch and advisory coordination and support is conducted. Exploits per se are constructed for specific products if needed to motivate the respective vendor and are delivered privately. Affected vendors will not receive detailed information about the performance of

their competitors. A grace-period is kept before moving to public release of the test-suite. This phase maps to the reporting, advisory and patch stages as proposed by us in [12].

During the *release phase* the test-suite is deployed for public perusal. Vendor-specific details are kept confidential. No exploits are released, only material sufficient to indicate potential vulnerability. This phase is equivalent to public disclosure, but not to full disclosure.

2.3. Deliverables

The final, public deliverable of the constructive disclosure process is the test-suite, which will package individual test-cases and code for delivering the test-cases against the system under test. This material should be placed under a public licence, such as the GNU General Public License (GPL), at no charge. This is done in order to ensure that vendors and their customers can freely utilise the test-material.

2.4. Limitations

Since the test-suite tackles one specific problem area, passing it is in no way a guarantee of a vulnerability-free system. In a typical case, test-suites will be created for a very specific subset of the target functionality and for very specific types of errors. This means that only a portion of the system under test will be examined and only a narrow subset of the overall security of the system will be addressed.

3. Results

The proposed model of disclosure was adopted in the "PROTOS - Security Testing of Protocol Implementations" project [17] [19]. In this project robustness and implementation-level vulnerability test-suites were created based on the principles of syntax testing [11].

3.1 Creation phase

The test-suite was created focusing on WAP gateway implementations. WAP was seen as a possible world-wide standard for providing Internet communications and services on wireless terminals, and it has already been adopted in the infrastructure of some digital mobile phone networks [24]. The focus of the test-material was narrowed down to the initial traffic of the protocol responsible for receiving content from a WAP gateway, namely the WAP-WSP-request.

In the creation phase, the PROTOS/c04-wap-wsp-request test-suite was created and the tests were conducted [18]. Out of the 24 gateways found in the wild, 7 were tested. The test-suite contained 4236 test-cases that consisted of string anomalies, length anomalies and delimiter anomalies and their combinations. All of the implementations under test failed in several categories.

The test-suite material was prepared, and placed under Gnu Public License. At this phase, only the personnel involved in the Protos project had access to the material.

3.2 Pre-Release phase

One-by-one the vulnerabilities were reported, and four of the seven bug-reports sent included an operational buffer-overflow exploit against one of the vulnerabilities found in that particular implementation. The bug-reports included a link to the test-suite pre-release documentation and material, and a private URL for the possible exploit for that particular implementation was also given.

At this phase, in addition to the originators, only the relevant vendors and the coordinators (AusCERT and WAP Forum) had access to the test-suite material, which was on a public web-page but not linked from anywhere but the bug-reports. A grace-period of at least 45 days was given.

3.3 Release phase

After the grace-period, which eventually extended to 51-56 days, depending on the vendor, the test-suite with all the material [18] and accompanying documentation [19] was published on the public project web-page. AusCERT also mentioned the test-suite release in their newsletter [10].

Feedback was collected from vendors and coordinators. Some vendors provided positive feedback, for example indicating that the material had motivated them to do a more extensive code reviews or had led to discoveries of further problems that were not noticed during the creation phase. Web-server logs indicated that all the vendors had indeed accessed the test-suite material, and thus they were left to make their own decisions on how to proceed. All details about vendors, even the names of the products tested were omitted from the public material.

3.4 Metrics

The times elapsing in different phases of the vulnerability process are presented in Table 1. The time of reporting the vulnerabilities for each vendor is marked as the zero day, although the real date of reporting varies. The discovery of the vulnerabilities by our research group and time of verification are shown as negative figures, and represent the numbers of days before reporting. The dates of vendor acknowledgement, patch availability and advisory release are shown as positive figures. The last column shows the delay between reporting and disclosure, which took effect on the day on which the test-suite documentation was linked to the publicly available pages, 11th November, 2000.

Table 1: Delays between phases in the vulnerability process

Case#	Discovery	Verification	Reporting	Acknowledgement	Patch	Advisory	Disclosure
0064	**-27**	**-24**	**0**	**-**	**-**	**-**	**55**
0065	**-30**	**-30**	**0**	**5**	**14**	**-**	**56**
0066	**-28**	**-27**	**0**	**11**	**30**	**-**	**51**
0067	**-22**	**-22**	**0**	**-**	**-**	**-**	**56**
0068	**-27**	**-20**	**0**	**1**	**46**	**-**	**55**
0069	**-32**	**-32**	**0**	**15**	**15**	**15**	**54**
0070	**-9**	**-8**	**0**	**11**	**-**	**-**	**54**

4. Analysis

This chapter analyses the method used in the case-study and the experiences gained from it against the goals of our proposal. Moreover, the method and case-study are reflected against RFP policy (the originator perspective), CERT-CC and NTBugtraq policy (the coordinator perspective) and Microsoft policy (the repairer perspective).

The method of constructive disclosure in the form of a test-suite represents a way of testing several independently developed implementations. The test-suite found several trivial vulnerabilities, mostly buffer overflows, in all the products tested.

This test-suite, although limited in scope, can be seen to provide a baseline, or at least a guideline, on the robustness of the software. It is hoped that the developers of WAP software will see this as a good addition to their conformance and regression testing. When using the test-suite as a method of acceptance testing or product evaluation, it is important to be aware of its limitations. A product that had several vulnerabilities initially but has

been patched against this test-suite might still contain several vulnerabilities not covered in the test-suite. In contrast, a product which appeared to be robust to start with will give a better impression on its estimated quality, although the coverage of testing is not comprehensive. This can also promote the idea of evaluating the previous versions to get an overview of the quality before and after the test-suite release.

Comparison of the timeline and disclosure methods of this case-study with available policies points to many differences and a few similarities. The aim was to minimise communication, which thus became scarce, as opposed to the RFP (originator) policy, which promoted constant communication and forced the vendor to make some contact every 5 days, with the threat of full disclosure otherwise. Contrary to the NTBugtraq (coordinator) policy, this case involved no prior risk analysis of the vulnerability disclosure effect as a means of promoting full disclosure, but the products available were simply tested and reported internally. The promoted timeline of 14 days was seen to be inadequate, and the timeline of 45 days as proposed by CERT-CC was adopted as the minimum to be given to each vendor whose products were tested. Most vendors seemed to act almost according to the policy adopted by Microsoft (repairer), i.e., almost no communication took place during the vulnerability process. The initial feedback from vendors was typically the only information provided before their possible software update came out.

During the pre-release phase an attempt was made to give access to the test-suite material to vendors whose products were not being tested. However when avoiding a full-disclosure, a limited channel was sought for. In this case a channel existed, the WAP Forum, but they had neither experience nor interest in channelling information such as this to their members. Thus, the most conspicuous omission in this disclosure was penetration of the information to other vendors of WAP-gateways. The impact of the whole test-suite was not visible, and everything happened quietly, so that repaired software items were announced without much ado. The actual partial disclosure was executed not through full disclosure email-lists but through an AusCERT security bulletin and by linking the material to the research web-site.

5. Discussion

A Request For Comments (RFC) has been proposed by Christey and Pease [4] to be drafted and adopted in order to establish a guideline, or standard practices, for all the participants in the vulnerability disclosure process. This chapter briefly notes and discusses the more generic aspects of the vulnerability disclosure process.

The most critical and necessary component in multi-vendor vulnerability process is the efficiency of the neutral third party as a coordinator. Several organisations such as the CERT can prove a valuable help in these issues, as can the support provided in adjunction with some full disclosure mailing lists, but formalisation of the roles of these resources is necessary, with a known set of rules.

In rapidly progressing disclosures the release timing has to take account of the effect of weekends and holidays. Releasing the material on a Friday, or just before a public holiday can cause distress to the vendor, in the same way as an ill-timed advisory affects the customers.

Especially in multi-vendor vulnerability processes, the first vendor that releases the vulnerability details to the public puts the other vendors and the users of the products, at risk of compromise. An ethical and professional approach to the disclosure process can help, and thus codes of ethics for related parties should be promoted into use.

The effect of legislation, especially the liabilities involved in malicious usage of the publicly provided test-material such as the PROTOS/c04-wap-wsp-request test-suite, should be taken into account, and the legislation against reverse-engineering should be studied when creating such test-suites so that violations can be avoided. Black-box testing methods seem to be the safest solutions for testing other people’s implementations. The liability of the actual vendor producing poor-quality software should be considered, as opposite to shooting the messenger.

Since advisory (and disclosure) copyright and licensing issues can limit distribution of the information, these should be left as open as possible in constructive disclosures and test-suites, to promote their usage without hindering the distribution. Downplaying of the vulnerability details is substantially hindered, as the same test-suite material is

provided to several vendors at the same time. Credit disputes can also be limited by supplying the test-material to the public for later verification.

Products with no active vendors or support behind them, typically beta-versions or end-of-line products, are still often in use. The vulnerability process will grind to a halt in such case, as there will be nobody willing to repair them. Use of such software in security or safety-critical environments is risky, not to mention dependence on it.

5.1. Future work

The drafting of clearly spelled-out disclosure policies should be encouraged. This will take us beyond unwritten and possibly hazy etiquette, towards a better defined form that is open to practical and ethical debate. Involvement of a neutral 3rd party should be encouraged, and actors willing to assume that role should be identified. Solid, systematic engineering efforts should be invested in creating test-suites against trivial vulnerability types, and an attempt should be made to have these adopted by standardisation organisations alongside conformance test-suites and the like.

6. Conclusions

This paper set out to introduce the concept of constructive disclosure as an alternative to full disclosure. Special emphasis was placed on the often neglected customer perspective, in this case by providing the needed software evaluation methods to the public perusal. The constructive disclosure model was coupled with a systematic approach which sought to create vulnerability test-material with coverage beyond one isolated case. This material was designed as a safety-net that, once published, would eliminate some of the vulnerabilities that recur and reappear in the same form in products of the same type.

The proposed approach was successfully adopted for testing protocol implementations and handling the resulting vulnerability process. A complicated multi-vendor, multi-vulnerability case, the PROTOS/c04-wap-wsp-request test-suite, was successfully handled in a systematic way, with positive feedback. Constructive disclosures were analysed to contribute a new perspective to the disclosure scene.

The case-study was limited to one test-suite release in a very specific subset of vulnerabilities, namely implementation-level errors. This area may have been more suitable for packaging the test-material, a prerequisite for reusability, than potentially more complicated vulnerability types. The test-material was aimed against a standardised functionality (open protocol), and the covering of a proprietary set of functionalities is not considered. No proof is provided for the suitability of the concept in a different context.

The experiences described here indicate that new models of partial disclosures can be introduced without any significant trade-offs with the full disclosure approach. Furthermore, the adopting of any systematic approach to the vulnerability process may be of considerable aid in handling cases that exhibit growing complexity. The seeking of solid engineering practices that will take the vulnerability process beyond an art form is promoted, and the concept of vulnerability escrows executed via neutral 3rd parties should be further developed. A call should be voiced for the involvement of standardisation organisations in ensuring the robustness of products already in use.

7. Acknowledgements

This work was carried out within the "PROTOS - Security Testing of Protocol Implementations" project [17], a government-funded research partnership between the University of Oulu and VTT Electronics, supported by two partner companies from the telecommunication industry.

We wish to express our gratitude to the individual vendors who worked with us to resolve the vulnerabilities related to the PROTOS/c04-wap-wsp-request test-suite. We are grateful to AusCERT [2] for their patient help, advice and active role during the vulnerability process.

8. References

[1]
Ivan Acre. (2000). "Vulnerability Reporting: Bugs in the bug reporting process". Column in TISC Insight, Volume 3, Issue 3. "http://tisc.corecom.com/newsletters/33.html"

[2]
AusCERT. (2000). "Australian Computer Emergency Response Team". "http://www.auscert.org.au/"

[3]
CERT-CC. (2000). "The CERT. Coordination Center Vulnerability Disclosure Policy". "http://www.cert.org/faq/vuldisclosurepolicy.html"

[4]
Steve Christey, Barbara Pease. (2001). "An informal analysis of vendor acknowledgement of vulnerabilities". Published on Bugtraq on March 11, 2001.

[5]
Russ Cooper. (1999). "NTBugtraq Disclosure Policy". "http://www.ntbugtraq.com/default.asp?sid=1&pid=47&aid=48"

[6]
Crispin Cowan. (1999). "Vulnerability Escrow (was: Extreme Hacking)". Firewall Wizards mailing list. Jul 07 1999. "http://www.securityportal.com/list-archive/firewall-wizards/1999/Jul/0096.html"

[7]
Rik Farrow. (2000). "The Pros and Cons of Posting Vulnerabilities". Network Magazine, 10/05/00. "http://www.networkmagazine.com/article/NMG20001003S0001"

[8]
Federal Geographic Data Committee. "Directive #10 - Conformance" "http://fgdc.er.usgs.gov/standards/directives/dir10.html"

[9]
Sarah Gordon, Richard Ford. (2000). "When Worlds Collide: Information Sharing for the Security and Anti-virus Communities". EICAR 2000 Best Paper Proceedings pp.1-20.

[10]
Joel Hatton. (2001). "Improving the Security of WAP Gateways". Article in AusCERT Security Services Newsletter, Volume 4, Number 4. "http://www.auscert.org.au/newsletter/news20010103/"

[11]
Rauli Kaksonen, Marko Laakso, Ari Takanen. (2000). "Vulnerability Analysis of Software through Syntax Testing" White-paper. "http://www.ee.oulu.fi/research/ouspg/protos/analysis/WP2000-robustness/"

[12]
Marko Laakso, Ari Takanen, Juha Röning. (1999). "The Vulnerability Process: a tiger team approach to resolving vulnerability cases". In the proceedings of the 11th FIRST Conference on Computer Security Incident Handling and Response. Brisbane. 13-18 June, 1999. "http://www.ee.oulu.fi/research/ouspg/protos/sota/FIRST1999-process/"

[13]
Microsoft Corp. (2000). "Microsoft Corporation Product and Service Security Policy". "http://www.microsoft.com/technet/security/policy.asp"

[14]
Mudge. (2000). In an interview for ZDNet Inter@ctive Week, September 28, 2000. "http://www.zdnet.com/intweek/stories/columns/0,4164,2634819,00.html"

[15]
NTBugtraq. (2000). "Welcome to NTBugtraq!". "http://www.ntbugtraq.com"

[16]
Weld Pond. (2000). "Do security holes demand full disclosure?". ZDNet, August 16, 2000. "http://www.zdnet.com/zdnn/stories/comment/0,5859,2615973,00.html"

[17]
Protos. (2000). "PROTOS - Security Testing of Protocol Implementations". Project homepage. "http://www.ee.oulu.fi/research/ouspg/protos/"

[18]
Protos. (2000). "PROTOS Test-Suite: c04-wap-wsp-request". "http://www.ee.oulu.fi/research/ouspg/protos/testing/c04/wap-wsp-request/"

[19]
Protos. (2000). "Test-suite releases in Theory and Practice". "http://www.ee.oulu.fi/research/ouspg/protos/testing/test-suites/"

[20]
Rain Forest Puppy. (2000). "Full Disclosure Policy (RFPolicy) v2.0". "http://www.wiretrip.net/rfp/policy.html"

[21]
Markus Ranum. (2000). "Full Disclosure and Open Source". Key-note speaker at Black Hat Briefings, July 26th, 2000. "http://www.blackhat.com/html/bh-usa-00/bh-usa-00-speakers.html"

[22]
Bruce Schneier. (2000). "Full Disclosure and the Window of Exposure". Crypto-Gram, September 15, 2000. "http://www.counterpane.com/crypto-gram-0009.html"

[23]
SecurityFocus. (2000). "What is BugTraq?". "http://www.securityfocus.com/frames/?content=/forums/bugtraq/"

[24]
WAP Forum. "http://www.wapforum.org/"