

# Developing Expertise in Software Security: An Outsider's Perspective\*

Gary McGraw & Anup K. Ghosh  
Reliable Software Technologies Corporation  
21515 Ridgetop Circle, Suite 250  
Sterling, VA 20166  
(703) 404-9293  
<http://www.rstcorp.com>  
[gem@rstcorp.com](mailto:gem@rstcorp.com)    [anup@rstcorp.com](mailto:anup@rstcorp.com)

May 16, 1996

## Abstract

This document is presented as a preliminary position paper meant to help foster discussion at the Invitational Workshop on Computer Vulnerability Data Sharing. We argue for the development of a centralized database of security vulnerabilities *including exploitation information* for use by legitimate researchers in computer security.

---

\*This work was supported in part by a grant number F30602-95-C-0282 from the U.S. Department of Defense Advanced Research Projects Agency.

# 1 A legitimate need for vulnerability data

Information system security is gaining more and more prominence as computer networking becomes ubiquitous. Sites with Internet-networked computers must pay special attention to security concerns since the Internet, by its very nature, makes attacking sites easier than ever before. The Internet is growing at an incredible rate. As a result, the demand for expertise in security is likewise growing. In fact, the need for security expertise seems to be so great that there is no alternative to co-opting “outsiders” such as myself into the discipline. One question we would like to address at the Invitational Workshop on Computer Vulnerability Data Sharing meeting is: “how can an expert researcher from outside the security community become an expert in security and vulnerability issues?”

We come to the security research arena as researchers experienced in other areas of computer science (for McGraw, artificial intelligence, cognitive science, and software engineering; and for Ghosh, computer engineering and software reliability). Our pursuit of knowledge about security has been to some extent hampered by the lack of a centralized repository of vulnerability data that can be used in experiments and analyses. This position paper includes a few observations that we have made as “outsiders” learning about computer security.

In general, we have been unpleasantly surprised by the lack of resources in the security research community. Although there are some general books and a couple of reasonable journals, the software security discipline sorely lacks a common repository of vulnerabilities. The CERT alerts, for example, are purposefully vague and usually amount to patch-distribution documents. Finding out exactly *what* a security vulnerability is and *how* it can actually be exploited is not a straightforward task. Subscribing to some of the many “underground” mailing lists and newsgroups is both a time consuming and a frustrating way of coming up to speed. This brings up the question as to what a reasonable level of abstraction for vulnerability definition might be.

The answer to this question is unfortunately complicated by the fact that broadcasting exploitation scripts and spreading the word about non-patched vulnerabilities is simply not feasible. Obviously, care must be taken not to spread “cracking” expertise far and wide (throwing kudzu seeds to the wind, so to speak). But there seems to be a severe lack of any *legitimate* path to knowledge of computer security, especially with regard to actual exploitation

scripts. As security takes on a more prominent role in computer science, this lack will need to be addressed.

From a software engineering point of view, or more specifically a software assessment point of view, the more vulnerability knowledge we software engineers have, the better tools and assessment techniques we can develop. As a concrete example, consider that the ARPA-sponsored research project that we are currently involved in has the goal of developing a security assessment prototype tool based on fault-injection [2,3]. The more that RST researchers can find out about actual vulnerabilities (and not at some abstract level, but at the down-and-dirty machine level), the better our fault-injection and intrusion detection methodology will be.

This brings up an interesting fact about the amount of overlap between software engineering needs and intrusion detection needs. We would argue that intrusion detection and software engineering require the same level of access to security data for use in experimentation. How these common needs overlap with incident handling needs is another story (one that we are not qualified to address).

## **2 Creating a shared database for research**

We are concerned about the lack of a vulnerability database — something that exists as a glaring “hole” in the security research community. Our security work at RST can certainly be counted as one example of how and why such a database might be put to “appropriate” use. Another example is the vulnerability detection work of Stephanie Forrest at the University New Mexico [1]. Forrest also came to the security community as an outsider and has injected new ideas into the intrusion detection arena.

Vulnerability data in such a database should probably be structured by several different classes of information including: operating system, date of discovery, some assessment of possible damage (including a severity rating), whether or not a patch has been created (and if so patch distribution information), cross-references to CERT documents, and a technical point of contact. It should be possible to search the database along any of these dimensions. It is our opinion that the database should include explicit and un-censored exploitation scripts as well as a detailed technical account of the vulnerability.

This information should be compiled in a central repository by an organization dedicated to improving security through information dissemination (such as CERT). Other possibilities include academic research groups, or possibly an agency of the U.S. Government (e.g., the NSA or NIST). In our opinion it would be a bad idea to have the database administered by a private corporation or individual as the potential of abuse through exclusive distribution may be too tempting. (Of course, the same possibility of “information hoarding” exists for academic groups as well, though the prospects are not as great.) The benefits of centralized information versus information distributed across individuals and organizations include: (1) the ease by which researchers can obtain security vulnerability data from a single source, (2) a consistent validation scheme with regards to the accuracy of vulnerability descriptions, (3) a central locus for the dissemination of *necessary* patch information to the Internet-at-large, and (4) centralized control over dissemination of methods of vulnerability exploitation. We would imagine that CERT has to some extent dealt with most of these issues. It would be interesting to get their input.

## 2.1 Exploitation scripts

We feel strongly that detailed descriptions of vulnerabilities *including exploitation scripts* and technical descriptions should be made available to legitimate researchers in security. Of course, this point begs the question as to how to define just what a “legitimate” security researcher is. That point aside, without the means of re-creating actual vulnerabilities in a research setting, the development of tools to *prevent* future exploitation of software vulnerabilities will clearly be thwarted. I believe that the benefits of making this information available to researchers and developers of commercial and/or public-domain security tools far outweigh the potential hazards of keeping too tight a “lid” on vulnerability data. In any case, some sort of cost/benefit analysis approach is warranted in this situation since the decision as to vulnerability information release has both positive aspects (in terms of advancing research agendas) and negative aspects (in terms of arming the enemy).

Perhaps the most important question arising out of this discussion is how to establish a “trusted user base” of security researchers and developers to whom the vulnerability database can safely be made available. Researcher

legitimacy can probably be propagated on the basis of trust (as is the case with some kinds of public key validation [4]). That is, if some trusted researcher (or two) vouches for a potential new security researcher or research group, then the new researcher should then be trusted. This sort of validation scheme would also help to foster a better sense of community between security researchers. A potential drawback to this approach is that it might result in too much of a “good old boy” network arising. It might not be a bad idea to implement some sort of minimal background check as part of the approval process as well. Once again, though, it seems clear to us that erring on the side of releasing too much information is far better than trying to keep everything secret and stifling progress.

## 2.2 Unpatched software

As a related point, the release of unpatched (and thus vulnerable) versions of software should also be made available (though be restricted to the security research/development community) so that effective tools can be created based on known intrusions. These buggy programs would be used exclusively in a laboratory setting.

## 3 Conclusion

From our point of view as a software engineers and researchers who are fairly new to security research, it seems obvious that the security community needs to address these issues head on. Though the issues are complex, it is clear that the lack of a central repository for detailed and specific vulnerability information is a hindrance to future progress in the field.

## REFERENCES

1. Forrest, Stephanie, Stephen Hofmeyr, Anil Somayaji, & Thomas Longstaff. (1996) A Sense of Self for Unix Processes. In *Proceedings of the 1996 IEEE Symposium on Security and Privacy*, May 1996.
2. Voas, Jeff, Gary McGraw, Anup Ghosh, Frank Charron & Kieth Miller. (1996) Defining an adaptive software security metric from a dynamic

software failure tolerance measure. To appear in the *Proceedings of the Ninth Annual Conference on Computer Assurance*, June 1996.

3. Voas, Jeff, Gary McGraw, and Anup Ghosh. Defining an Adaptive Software Security Metric from a Dynamic Software Failure Tolerance Measure. Reliable Software Technologies Technical Report. March 28, 1996. Sterling, VA.
4. Zimmerman, P. PGP User's Guide, Volume I: Essential Topics. October 1994.

### **Disclaimer**

THE VIEWS AND CONCLUSIONS CONTAINED IN THIS DOCUMENT ARE THOSE OF THE AUTHORS AND SHOULD NOT BE INTERPRETED AS REPRESENTING THE OFFICIAL POLICIES, EITHER EXPRESSED OR IMPLIED, OF THE DEFENSE ADVANCED RESEARCH PROJECTS AGENCY OR THE U.S. GOVERNMENT.