Editors: Matt Bishop, bishop@cs.ucdavis.edu
Deborah A. Frincke, deborah.frincke@pnl.gov

I Am a Scientist, Not a Philosopher!

Sean Peisert
University of California, San Diego

Matt Bishop
University of California, Davis

We no longer live in the era of Aristotelian philosophers or alchemists attempting to turn lead into gold. Yet, you might be forgiven for thinking we were, after observing many computer security researchers' claims—even in papers published in peer-reviewed journals and conference proceedings. Computer security is both an art and a science,[1] but researchers frequently fail to follow the scientific method to support the claims they make in scientific, peer-reviewed papers.

Some computer security research is highly mathematical and can be proven formally without experimentation. But formal proofs depend on correct implementation of theory and also assume that the foundational work that they treat as axiomatic has also been proven correct. This is often an unsafe assumption. To ultimately demonstrate a useful, scientific contribution, most of the work being done today (including much of the mathematical and theoretical work) must therefore prove to work in practice. To evaluate anything we can't prove by using pure mathematics or logical syllogism, we must test hypotheses by performing controlled experiments to generate measurable, empirical data. But today's computer security researchers often claim "proof" without following this approach.

Failure to follow the scientific method rigorously can create problems. It has become common practice to make claims about a researcher's technique, develop software, and sell products, so that the lay public quickly buys into a solution that's never been scientifically justified. When a security breach eventually occurs, it calls the entire field of computer security into question because the public can't distinguish between valid methods and those that have yet to be proven.

In previous work, we discussed a process for applying the classical scientific method to computer security experiments.[2] The key qualities we discussed were the ability to falsify a hypothesis (*falsification*), measure and observe data (*measurability*), repeat controlled experiments, and reproduce results (*reproducibility*). This article presents a method for scientific experimentation when others aren't appropriate or can't be readily applied. Our goal is to further motivate researchers to apply science to experiments and, in concert with our earlier work, offer a new technique for doing so.

One reason people don't follow the scientific method

One of the principal challenges in security experiments is the limited quality and availability of data sets against which to test. New solutions sometimes have access to well-defined, highly regarded data sets they can use to compare current and previous results. More often, though, researchers must not only develop their own techniques but also their own data, procedures to follow, and metrics to measure.

One reason that existing data sets are sometimes poorly designed is that the conditions they test aren't well defined. Claims that arise from resulting experiments are therefore imprecise or overstated. For example, if we run an experiment using a particular tool on a particular data set and make a claim about the results and the tool's effectiveness or efficiency in comparison to existing tools, several questions arise: How well does that particular data set reflect the scenarios the new tool will face when deployed in practice? How general is the data, and thus, how broad can our claims be and still be considered valid and appropriate? Even with a good data set, the question is whether claims are based on the results of all experiments or on the results of favorably chosen examples that are actually a small, unrepresentative subset of all results (sometimes known as the Potemkin village model or, as we prefer, the Rock Ridge model[3]). Such situations frequently arise, suggesting more ambiguous conclusions. Indeed, a particular technique might actually be better than others under certain conditions, and that knowledge could be scientifically and practically useful. Choosing favorable experiments and data just makes it harder to identify the conditions in which the technique is truly useful because the difficulty in replicating the experiments inhibits other researchers' ability to extend and enhance the techniques.

Consider the problem of measuring effectiveness in intrusion detection. A large number of papers have been published—and continue to be published—that use the controversial Lincoln Labs network intrusion detection data sets (www.ll.mit.edu/IST/ideval/data/data_index.html) in evaluating their own techniques, despite the problems known in doing so.[4] (For more information on Lincoln Labs' intrusion-detection data, see Basic Training on p. 65.) Unfortunately, creating new data sets is challenging, and getting them widely adopted (to facilitate direct comparisons among methods) is perhaps even harder. Furthermore, even researchers who create their own data sets often run their experiments under variable conditions, such that even if they properly captured the data set, no one could reproduce the experiment to validate it.

Of course, conducting falsifiable, controlled, and reproducible experiments by applying the classical scientific method to computer security isn't always possible. We now discuss a new approach for experiments.

Our idea for designing experiments

In our own work with computer forensics, we've discovered that there isn't even a forensic equivalent to the Lincoln Labs data sets to compare our techniques against. Of course, even if there were, who would want to use it? In forensics—which attempts to answer such questions as how an intrusion occurred and what happened during it—a measure of effectiveness would probably be based largely on experiments with humans, and, as far as we're aware, no such previous, standardized experiments against which to compare have been published.

In some cases, there simply isn't a good way to use syllogism, or comparisons with existing data sets, and there's no practical way to conduct exhaustive experiments using human subjects. An alternative method is needed.

We've previously used a method[5,6] in which we chose a set of experiments because they covered all classes of two flaw domains, as enumerated in the seminal Research into Secure Operating Systems (RISOS) and Protection Analysis (PA) reports,[7,8] which are generally accepted as complete. (Note: when referring to the classes enumerated in the PA report, we use the revised hierarchy described by Peter Neumann.[9]) In forensics, a large enough collection of examples with overlapping coverage of the flaw domains might be sufficient to let investigators analyze any attack in the same flaw domains as the specified attacks. We suspect that this technique will be generalizable to fields in computer security other than forensics. Therefore, experiments based on well-accepted flaw classifications, used to evaluate model implementations, should be effective for most of the situations these flaw domains cover. Of course, our own assertion must be validated! To test this method, we must observe how it works in reality.

Consider the following list of RISOS flaw domains:

1. Incomplete parameter validation
2. Inconsistent parameter validation
3. Implicit sharing of privileged or confidential data
4. Asynchronous validation or inadequate serialization
5. Inadequate identification, authorization, or authentication
6. Violable prohibition/limit
7. Exploitable logic error

Now, consider the following list of PA flaw domains:

- 1a. Improper choice of initial protection domain
- 1b. Improper isolation of implementation detail (exposed representations)
- 1c. Improper change (data consistency over time)
- 1d. Improper naming
- 1e. Improper deallocation or deletion (residuals)
- 2. Improper validation (of operands and queue-management dependencies)
- 3a/b. Improper synchronization (indivisibility and sequencing)
- 4. Improper choice of operand or operation (critical operation selection errors)

Now consider the 1988 Internet worm, which exploited a buffer-overflow vulnerability in `fingerd` and several problems with other Unix programs, including `sendmail`, to break into systems. The attack didn't damage the software on the systems, nor did the worm attempt to gain root access, but it did cause denial-of-service attacks by attempting to propagate to as many machines as possible from those it had infected. The worm's ultimate goal was to spread.[10]

Because it involved several distinct attacks, the worm also crossed multiple flaw classifications:

- cracking encrypted passwords from `/etc/passwd` by running significantly enhanced version of crypt (violates RISOS 3, RISOS 7, PA 1a, and PA 1b);
- bug allowing a buffer overflow of

In some cases, there simply isn't a good way to use syllogism, or comparisons with existing data sets, and no practical way to conduct exhaustive experiments using human subjects.

Table 1. Exploits and corresponding flaw domains.

EXPLOIT	PROTECTION ANALYSIS (PA) FLAW DOMAINS								RESEARCH INTO SECURE OPERATING SYSTEMS (RISOS) FLAW DOMAINS						
	1A	1B	1C	1D	1E	2	3A/B	4	1	2	3	4	5	6	7
Buffer overflow		x	x			x	x		x					x	
Spyware	x										x				
Ignoring permissions					x		x					x	x		
Authentication						x		x					x		
Trojan horse				x											x
Bypassing interfaces	x	x	x								x				
Parameter validation						x				x					
Land attack						x		x			x		x		
Shared memory injection	x	x	x			x	x					x			
1988 Internet worm	x	x	x	x	x	x		x	x	x	x		x	x	x
Christma exec worm	x			x							x				
Network file system (NFS) exploits	x	x				x		x			x		x	x	

the `fingerd` daemon (RISOS 1, RISOS 6, and PA 2);
- bug involving improper checking of arguments in `sendmail` (RISOS 2 and PA 4); and
- vulnerability improper trust of hosts `rsh` and accounts `rexec` (RISOS 5, PA 1a, and PA 2).

Furthermore, the worm used several techniques, such as renaming itself and removing its command-line arguments after execution, which were defenses for itself rather than explicit attacks. We can categorize these actions under RISOS 7 (exploitable logic error), PA 1c (improper change), PA 1d (improper naming), and PA 1e (improper deallocation or deletion).

After this analysis and categorization, we can compare the results against another attack that exploits the same flaw class. In our forensic analysis work, we discovered this to be a highly useful technique. Table 1 lists the set of exploits we used to test our techniques[6] along with the flaw domains that each covers.

Using this grid and the experimental results, we could forensically observe and analyze attacks that exploited a subset of the flaws (for example, the simple buffer overflow) from another attack (for example, the 1988 Internet worm). To do so, we recorded the forensic information demanded by the attack that exploits more flaws. Our experience also extended to all other equivalent flaws and subsets of other flaws in the attack examples that we analyzed.

This method doesn't obviate the need to apply the scientific method, nor does it allow researchers to bypass good scientific practice. Many more tests are needed to verify this method of scientific experimentation. Furthermore, the method's success might also vary highly with the flaw classes' quality, the number of experiments performed, and the flaw-class coverage achieved. In the interim, this method appears valuable and deserves further experimentation and analysis.

Computer security experiments must ultimately use the scientific method, including well-defined, highly regarded data sets, to compare new and existing results. But our initial results suggest that computer security experimentation based on sufficient coverage of classes of flaw classifications shows promise. Ultimately, of course, our methods must also prove to be effective using one of the two classical methods—experimental validation with well-accepted data sets or formal proof—and we intend to perform such experiments in the future. The result could help more computer security researchers produce and present scientifically valid data. It might even help to bring computer security research on par with animating the dead, and thus satisfy the rantings of a certain fictional scientist, who protested: "I am a scientist, not a philosopher!"[11] ▢

References

1. M. Bishop, *Computer Security: Art and Science*, Addison-Wesley Professional, 2002.
2. S. Peisert and M. Bishop, "How to Design Computer Security Experiments," *Proc. 5th World Conf. Information Security Education* (WISE), Springer, 2007, pp. 141–148.
3. M. Brooks, *Blazing Saddles*, Warner Brothers, 1974.
4. J. McHugh, "Testing Intrusion Detection Systems: A Critique of the 1998 and 1999 DARPA Intrusion Detection System Evaluations as Performed by the Lincoln Laboratory," *ACM Trans. Information and System Security* (TISSEC), vol. 3, no. 4, 2000, pp. 262–294.
5. S. Peisert et al., "Analysis of Computer Intrusions Using Sequences

of Function Calls," *IEEE Trans. Dependable and Secure Computing* (TDSC), vol. 4, no. 2, 2007, pp. 137–150.
6. S.P. Peisert, *A Model of Forensic Analysis Using Goal-Oriented Logging*, PhD dissertation, Univ. of California, San Diego, Dept. of Computer Science and Eng., Mar. 2007.
7. R.P. Abbott et al., *Security Analysis and Enhancements of Computer Operating Systems (RISOS)*, tech. report, Lawrence Livermore Laboratory, Apr. 1976.
8. R. Bisbey and D. Hollingworth, *Protection Analysis: Final Report (PA)*, tech. report, Information Sciences Inst., May 1978.
9. P. Neumann, "Computer Security Evaluation," *Proc. 1978 Nat'l Computer Conf.*, Am. Federation of Information Processing Societies, vol. 47, 1978, pp. 1087–1095.
10. D. Seeley, "A Tour of the Worm," *Proc. Winter 1989 Usenix Conf.*, Usenix Assoc., 1989, pp. 287–304.
11. M. Brooks, *Young Frankenstein*, Crossbow Productions, 1974.

***Sean Peisert** is a postdoctoral scholar in the Department of Computer Science and Engineering at the University of California, San Diego (UCSD). His research interests include computer forensic analysis, intrusion detection, vulnerabilities analysis, security policy modeling, and electronic voting. Peisert has a PhD in computer science from UCSD. He is a fellow of the San Diego Supercomputer Center. Contact him at peisert@cs.ucsd.edu.*

***Matt Bishop** is a professor in the Department of Computer Science at the University of California, Davis, and a codirector of the Computer Security Laboratory there. His research interests include vulnerabilities analysis, the design of secure systems and software, network security, formal models of access control, and intrusion detection. He is the author of* Computer Security: Art and Science *(Addison-Wesley, 2002). Contact him at bishop@cs.ucdavis.edu.*

Interested in writing for this department? Please contact editors Matt Bishop (bishop@cs.ucdavis.edu) and Deborah A. Frincke (deborah.frincke@pnl.gov).