Research Statement

Jedidiah R. Crandall

5 February 2007

I am fascinated by the way that a system's architecture can affect the security of that system, so while my interests are in architectural security mechanisms and history, my research is more in the systems security area, including malware detection and analysis. Applying systems skills to security problems as a graduate student has been an intriguing experience. Systems research has a history of evaluating ideas very thoroughly, whereas the adversarial aspect of computer security gives any application many layers of hidden complexity. My advisor, our colleagues, and myself have had many discussions on how to evaluate a security idea in great detail without losing sight of the larger picture. I hope that this reflects in our past research, and in future research I intend to continue to foster theory that bears fruit because it is firmly grounded in practice. In my various projects I have always found that running experiments in as realistic of an environment as possible and doing a thorough literature survey never fail to yield compelling insights.

1 Minos

Minos was developed as an architecture to stop control data attacks. The basic idea is to tag data from untrusted sources such as the network, and raise an alert whenever untrusted data is loaded into the program counter for a control flow transfer. More details about how to do virtual memory swapping with tag bits, handle low integrity data in the filesystem, and deal with information flow problems are in the conference paper [3]. Minos is surprisingly effective for such a simple mechanism. Virtual-machine-based implementations of Minos for both Linux and Windows have been defending themselves against daily remote attacks for almost two years now with no false positives.

The project taught us a lot about the tradeoffs between security and compatibility, and a virtual machine implementation of Minos running on off-campus honeypots has detected actual attacks for ten different vulnerabilities. In total, Minos has detected attacks for 27 real vulnerabilities. Both through finding interesting past work, such as the ideas of Boris Babayan from the Elbrus line of computers, and from several attacks that worked in unexpected ways, we have discovered and argued in a MICRO-37 paper [3] (MICRO is a top computer architecture conference, acceptance rates are typically under 20%) and an upcoming TACO paper [4] (TACO is a well-respected journal with a rigorous review process) that memory corruption attacks are a much more fundamental problem that stems from the coercion of pointers and integers, and can even be traced back to Von Neumann architecture. While Minos has the potential of being an important component in a substantially more secure system, any future work to totally ameliorate memory corruption attacks will have to address this fundamental problem.

There are a number of ongoing projects at various institutions that improve on the basic mechanism behind Minos, and we were pleased to see Minos' security deconstructed in two WDDD papers this year [10, 8]. The Minos project also generated a great deal of interest in the security community because of the ability of virtual-machine-based Minos honeypots to catch most zero-day worms. I was invited to give a talk at the Workshop on Rapid Malcode (WORM 2004), a yearly workshop that brings together the top researchers interested in the Internet worm problem.

A theme that I have been interested in since the Minos project, and hope to explore with specific projects in the future, is the problems with various addressing methods, such as capabilities and the difficulty of ensuring that they are properly revoked. For example, page tables are essentially capabilities and the complexity of virtual memory management in both Linux and Windows has led to many vulnerabilities where a user-space page table entry (a capability) to a physical page is not revoked when the physical page is reused as a kernel page.

2 DACODA

DACODA is a full-system implementation of symbolic execution for analyzing worm exploits caught by Minos. We analyzed the exploit vectors of fourteen real exploits, seven of them actual attacks or worms on Internet honeypots, and developed a theory of polymorphic and metamorphic worm attacks that we feel is deeply rooted in practice. Based on Cohen's insight more than twenty years ago [2] that "information only has meaning in that it is subject to interpretation," we gave empirical evidence that none of the proposed methods in the literature at that time (2005) for generating worm signatures to block worms in the network would have been effective against polymorphic/metamorphic variants of past worms if the attack had been designed

specifically to avert that particular defense mechanism. In two papers [5, 7] (one of which was published in CCS, one of the top two conferences in computer security that has been historically difficult to publish intrusion detection papers in because of the community's high standards for evaluation methodology) we challenged many of the implicit assumptions in the intrusion detection literature, such as the necessity of NOP sleds, constraints on bogus control data, or that the exploit vector of typical attacks would trace through a single, user-space process. Along with the Minos paper, the DACODA work is now cited in many worm detection and signature generation studies. Our work was the first to study a large enough corpus of real worms and remote exploits to provide a quantitative, empirical measure of the invariant content that could be used in string-based network signature schemes.

Now that we have developed a model of polymorphic and metamorphic worms that reflects real systems and real vulnerabilities, there are many directions we hope to explore with the aim of mitigating the threat of Internet worms. Based on the structure of exploits for memory corruption vulnerabilities and the predicates that DACODA discovers (*i.e.* heap corruption exploits typically involve network data used as pointers without predicates to bound what memory can be addressed, integer overflows typically involve network data used as pointers with a signed comparison predicate, etc.), we plan to explore architectural improvements, effective signature generation techniques, and testing methods that more efficiently discover vulnerabilities. Some specific ideas are described in a recently funded NSF Cyber Trust grant [11]. More recently I have become interested in latent semantic analysis [9] as a way to understand the semantics of worm exploits, which I will say more about in Section 6.

3 Behavior-Based Malware Analysis

Temporal search is a behavior-based analysis technique using virtual machines where it is possible to discover that a piece of malware is counting down to some event in the future (when it might, for example, delete all of your files or download new instructions from a public web server) without waiting for the event to occur. It is based on slight time perturbations, symbolic execution, predicate inversion, and then a weakest precondition analysis to account for quirks in the Gregorian calendar (leap years, number of days in each month, etc.). Our analysis of real malware has revealed a complex relationship between behavior and time that will require future work, and also demonstrated the need for more control-flow-sensitive symbolic execution and program analysis techniques in the domain of processing dates and times. We have also learned a great deal about behavior-based analysis and evasive malware (both of which are starting to attract much attention in the security community) which we outlined in a recent ASPLOS paper [6] (ASPLOS brings together the best researchers in architecture, systems, and programming languages, and is probably the best outlet for the kind of work I do). Most of what we learned was the result of applying the technique to actual malware on a real system (Windows XP); our conclusions could not have been drawn from microbenchmarks or a pencil-and-paper theory.

4 Replay-based Enforcement of Non-Interference

Enforcing information flow policies across a full system is a long-standing grand challenge of computer security. One way to enforce these polices is non-interference, which basically states that all low-security output should be independent of high-security inputs. While non-interference has been extensively studied in language-based information flow research, a full-system implementation was abandoned for various reasons, the most challenging of which were that: 1) architectural nondeterminism in the system confounds the problem of matching inputs to outputs; 2) a full-system implementation traditionally entailed a highly structured, and thoroughly specified and validated operating system with an exhaustive covert channel analysis; and 3) sometimes it is more useful for a policy to bound information leakage rather than disallow it altogether.

In a paper that is currently under submission we have proposed logging all nondeterministic events during a transaction and then using repeated deterministic replays to enforce non-interference. This allows for a direct measurement of the amount of entropy between inputs and outputs and has the property that all covert channels must involve modulating at least one event that appears in the log file, enabling a systematic, exhaustive covert channel analysis.

In addition to this, we see many directions to go with this research, possibly including analyzing malware that uses cryptovirological techniques or enumerating all inference channels in an architecture that heavily shares resources (such as a mulithreaded or multi-core processor). What excites me about this area is that it may be possible to develop general techniques, for example that address all inference channels rather than finding a single one. While I believe that "thinking like an attacker" and discovering individual vulnerabilities is important, I am personally more interested in more comprehensive defense techniques. It is one thing to think outside the box, and an entirely different matter to draw a better box.

5 Great Firewall of China

As a side project inspired by recent work on the Great Firewall of China [1] (GFC), we are trying to model and understand keyword-based Internet censorship. Rather than attempt to find a gold nugget, such as a vulnerability in the GFC's operation,

we seek to refine a quantity of ore and probe a broad cross section of the Chinese Internet to understand the implementation and application of this important censorship mechanism. In a paper under submission we propose and describe the architecture of Doppler, a "weather tracker" for Internet censorship. My contribution to this project has been to model keyword-based censorship using latent semantic analysis. Just as an understanding of the mixing of gases preceded effective weather reporting, understanding the relationship between keywords and concepts is essential for efficiently tracking Internet censorship.

Also, related to my main line of research (Internet worms), the non-technical aspects in terms of cost and development of how the GFC went from a concept to a nationwide filtering mechanism, though not yet achieving full coverage, can teach us some lessons about what it would take to build an infrastructure to protect our own networks from Internet worms.

6 Future Work

Many of the projects I have described raised as many questions as they did answers. In addition to developing an end-to-end implementation of temporal search and exploring new applications of replay-based entropy flow measurement, and continuing to be involved in side projects related to censorship, there are a few novel directions I would like to explore including possibly a promising development in my main line of research which is the capture and analysis of Internet worms based on their exploit vector.

For the past year, after the DACODA project gave us an idea of how difficult it is to do robust worm signature generation, we have struggled with finding a way to build on the DACODA work and come up with an effective way to analyze a worm's exploit vector to automatically generate protection such as a signature or a patch. Then it occurred to me that if latent semantic analysis enables me to summarize the semantics of Chinese, a language I know nothing about, it should be able to help us in the arduous task of understanding the semantics of exploits for various types of vulnerabilities based only on traces of predicates that DACODA discovers. We have only just begun to talk about this idea, but I think it holds a lot of promise because it is so simple. The DACODA project showed us that worm exploit analysis must be done on the host and that even typical worms can require full-system analysis, including the kernel, multiple processes, and multithreading. Building a control-flow graph and doing standard program analysis in this environment is too complicated to be robust. The kind of *"Wormipedia"* that we are exploring may allow us to identify all of the interesting predicates and their relationship to the vulnerability based purely on empirical data. Latent semantic analysis is a more organic and principled technique than many machine learning algorithms, in my opinion, because you start with a large corpus and distill it down to the underlying concepts. The challenge will be building a database of worm exploits that is of encyclopedic proportions.

Coming from an architecture and systems background, I am convinced that there is no purely technical solution to any of the major security problems that we face: rapid Internet worms, botnets, evasive malware, covert channels, etc. These problems are all congenital to the way information systems have been architected. Thus I believe that my role as a Computer Science researcher is to support nontechnical solutions to these problems by implementing tools that are useful, even if not universally applicable, and always seeking a deeper understanding of the technical issues involved so that, even if particular proposed techniques are not adopted, at least their scientific insights can guide whatever solutions are adopted. This mindset will guide my future research.

References

[1] R. Clayton, S. J. Murdoch, and R. N. M. Watson. Ignoring the great firewall of china. In *6th Workshop on Privacy Enhancing Technologies*, June 2006.

[2] F. Cohen. Computer viruses: Theory and experiments. In *7th DoD/NBS Computer Security Conference Proceedings*, pages 240–263, September 1984.

[3] J. R. Crandall and F. T. Chong. Minos: Control data attack prevention orthogonal to memory model. In *Proceedings of the 37th International Symposium on Microarchitecture (MICRO)*, December 2004.

[4] J. R. Crandall, F. T. Chong, and S. F. Wu. Minos: Architectural Support for Protecting Control Data. *To appear in ACM Transactions on Architecture and Code Optimization*, 2006.

[5] J. R. Crandall, Z. Su, S. F. Wu, and F. T. Chong. On Deriving Unknown Vulnerabilities from Zero-Day Polymorphic and Metamorphic Worm Exploits. *12th ACM Conference on Computer and Communications Security (CCS)*, 2005.

[6] J. R. Crandall, G. Wassermann, D. A. S. de Oliveira, Z. Su, S. F. Wu, and F. T. Chong. Temporal search: Detecting hidden malware timebombs with virtual machines. In *Proceedings of ASPLOS-XII*, Oct. 2006.

[7] J. R. Crandall, S. F. Wu, and F. T. Chong. Experiences using Minos as a tool for capturing and analyzing novel worms for unknown vulnerabilities. In *DIMVA*, 2005.

[8] M. Dalton, H. Kannan, and C. Kozyrakis. Deconstructing hardware architectures for security. In *Fifth Annual Workshop on Duplicating, Deconstructing, and Debunking*, June 2006.

[9] T. K. Landauer, P. W. Foltz, and D. Laham. Introduction to latent semantic analysis. In *Discourse Processes*, 1998.

[10] K. Piromsopa and R. J. Enbody. Defeating buffer-overflow prevention hardware. In *Fifth Annual Workshop on Duplicating, Deconstructing, and Debunking*, June 2006.

[11] Z. Su, S. F. Wu, and F. T. Chong. A Vertical Systems Framework for Effective Defense against Memory-based Attacks (recently funded by the NSF Cyber Trust program).