Discussion Topic Proposal

For NSPW 2005, we are proposing a panel discussion on the use of diversity as a computer defense mechanism. The panel will comprise individuals knowledgeable about diversity that can lead the audience in a discussion on key issues related to diversity for computer security. We anticipate that the following items will be addressed:

- Critical evaluation of diversity research conducted by the computer security community
- The application of important diversity research from other disciplines relevant to computer security
- Definition of future research directions and possible topics in order to further the usefulness of diversity

Topic Justification

So, why is Diversity a good topic for NSPW? The purpose of NSPW, as stated in the call for papers, is to explore new paradigms, propose innovative solutions to existing problems or address controversial topics. Diversity for computer security qualifies based on several NSPW criteria.

While diversity for computer defense is not a new paradigm it is a paradigm in need of further definition. Currently, not enough is known about diversity to make it useful for computer security and yet it continues to surface as a proposed solution. There is no quantitative information on the costs associated with implementing diversity. Thus diversity may be prohibitively expensive compared to other security strategies. Another unknown is the strength of protection offered by diversity. The typical way diversity is used in computer security is to create some type of code or system obfuscation in order to increase attacker effort. Yet, the quantity of effort needed is again undefined. The general relationship between diversity and typical attacks has not been determined for even the average case so the amount of diversity required to thwart a specific attack is unknown.

Diversity also qualifies as an NSPW topic on the basis of its status as a controversial topic. A recent 2003 Computer & Communications Industry Association (CCIA) report asserted that the US is at risk from computer insecurities because of the overwhelming dominance of Microsoft Windows as the Operating System (OS) of choice [8]. The report authored by seven "security experts" contends that Microsoft dominance has created a monoculture totally lacking in diversitywhich is one of the primary reasons for our susceptibility to cyber attacks. The report did not present any evidence that diversifying the OS market with the existing choice of OS's would increase system security. This report was the subject of a lively debate at the 2004 Usenix Security Conference between Scott Charney of Microsoft and Dan Geer, one of the report's authors[1] [13]. A common misconception held by the security community is that more diversity is always better which is not necessarily true. In order for diversity to improve security the component

[1] There was no clear winner of that debate!

being diversified must not be susceptible to the same types of attacks or create even more vulnerabilities in combination with the original system. Discussion at NSPW should help clarify these concepts for everyone involved in secure system development and reduce some of the confusion and controversy associated with diversity as a security strategy.

The goal for this panel will be to encourage discussion by NSPW participants that can then be documented and more widely disseminated through the NSPW proceedings. The hope is that the topics discussed at NSPW will generate interest within the security community, which might lead to further research into diversity for computer security.

Proposed Panelists

The proposed panelists include along with their background qualifications include:

Bev Littlewood

Bev Littlewood is a Professor of Software Engineering at City University London. Bev has worked for many years on problems associated with the modeling and evaluation of dependability of software-based systems. He is a member of the UK Nuclear Safety Advisory Committee, of IFIP Working Group 10.4 on Reliable Computing and Fault Tolerance, and of the BCS Safety-Critical Systems Task Force. He is a Fellow of the Royal Statistical Society. In particular, Dr. Littlewood has studied probability associated with N-Version program design. He has developed probability models related to forced N-version programs where diversity is deliberately induced between the various programming teams. More recently, Dr. Littlewood has studied diversity for computer security. Dr. Littlewood's knowledge of the use of diversity to increase reliability for fault tolerance is extensive and his interest in applying that knowledge to computer security makes him an ideal panelist.

Roy Maxion

Dr. Maxion is a professor in computer science at Carnegie Mellon University. Dr. Maxion's research covers several areas of computer science, including development and evaluation of highly reliable systems, machine-based concept learning, and human-computer interfaces. He is developing dependable systems for automated detection, diagnosis and remediation of faulty or unanticipated events in many domains -- international banking, telecommunications networks, vendor help systems, semiconductor fabrication, information warfare and others.

John McHugh

John McHugh is a senior member of the technical staff at the CERT® Coordination Center, part of the Software Engineering Institute (SEI) at Carnegie Mellon University where he does research in survivability, network security, and intrusion detection. Prior to joining CERT®, Dr. McHugh was a professor and chairman of the Computer Science

Department at Portland State University in Portland, Oregon where he held a Tektronix Professorship. He has been a member of the research faculty at the University of North Carolina and has taught at UNC and at Duke University. For a number of years, Dr. McHugh was a Vice President of Computational Logic, Inc. (CLI), a contract research company formed to further the application of formal methods of software design and analysis in support of security and safety critical systems.

Carol Taylor

Carol Taylor recently finished her Ph.D. in Computer Science in May, 2004 and continues to work as a Post Doctorate Fellow at the University of Idaho. She has several degrees in biology in addition to her CS background and has worked as an ecologist in a previous position. Dr. Taylor has a strong interest in the application of biology to computer security including diversity. She has studied the way diversity is currently applied in much of the security research and believes that stronger, more usable results could be achieved with better quantification and more experimentation. A discussion about the limitations of biological diversity for computer security would hopefully produce more realistic, usable results.

Background Material

The accompanying background material provides the motivation behind the proposed panel topic.

Diversity for Greater Reliability

Software that operates in safety critical applications must be highly reliable in order to avoid catastrophic consequences such as loss of lives or huge financial loss. Yet, how do you improve software reliability knowing it is nearly impossible to eliminate all faults that could potentially cause system failure?

The fault tolerant community addresses this problem through redundancy, running several identical components and by diversity, using a number of different components. Voting is then typically done to determine differences between components which could signify component failure. For hardware, failure is typically caused by random faults so duplicating components provides added insurance since the assumption is that failures are independent. However, software failures are generally due to design faults created by developers. Consequently, faults are embedded within the software and every copy of that software behaves identically with respect to a given input. In this case, having multiple copies of the software doesn't help with reliability since each one will fail identically.

In an effort to increase the failure independence between software versions, N-version programming was proposed back in the 70's [1] as a technique for increasing overall reliability. Diversity is introduced by having different programming teams produce versions of the same program. It was hypothesized that diversifying the software

producers should result in programs with the same functionality but not the same faults. Research on the outcomes of N-version programming showed that the assumption of total independence of failures was false [10]. Programmers tended to make similar mistakes over more difficult parts of a problem. These experiments [10], plus others reported in [12], led to doubts as to the actual benefits of N-version software diversity. However, other researchers suggested that failure independence did not provide the complete picture of software reliability [11, 12].

Coincident failures between software versions was studied by Eckhardt and Lee as reported in [12]. Their research showed that failures were not independent between versions developed by different development teams. Independent teams were more likely to err over the same difficult parts of the problem. Research conducted by Littlewood and Miller [11] who studied forced[2] diversity in N-version programming, built a model of the probability of version failure over sub-domains of the program space. They showed that problems difficult for one sub-domain might be easy for a different method and averaging the results could prove favorable. Summarizing the results of theirs and others work with software diversity includes the following findings:

- Benefits of software design diversity are difficult to measure
- Software diversity has been used in real safety critical applications
- Diversity seems to help with reliability, but there is not enough data to say that diversity absolutely helped with overall reliability
- The same level of reliability might have been achieved by some other means

Even after many years of study, there are unanswered questions relating to design diversity. Yet, some of the knowledge gleaned about failure mechanisms of diverse software could possibly be applied to the area of computer security.

Diversity for Biology

Diversity in the biological world appears to function by maintaining species and ecosystems. Within a species, diversity is credited with assisting species survival by varying the genetic make-up since all members won't be equally susceptible to environmental threats such as predators or disease. At the ecosystem level, higher species diversity is considered correlated with ecosystem stability. The main concept is that there should be enough species diversity so that substitution can occur among functionally equivalent species in the case of species extinction. This is the primary idea behind the often cited statement that monocultures are bad since one disease event could potentially wipe out an ecosystem of one species if the one species is susceptible to the disease.

As stated in a previous section, there are many misconceptions related to biological diversity that are spread throughout the non-biological world. Clarification of concepts related to diversity would benefit researchers that try to apply biological models as solutions to computer based problems.

[2] Forced diversity is where diversity is deliberately introduced by requiring different languages, tools, testing suites or some other required differences between development teams

Diversity in Computer Security

Diversity has been studied as a technique for increasing system security. Some relevant research results will be highlighted in order to provide a picture of the current state of diversity research in computer security. While the goal of diversity in fault tolerance is to promote failure independence between program versions, the purpose of diversity in computer security is to increase the attacker's effort to compromise a system. Independence of failure is assumed but not measured in most of the security diversity research.

Past research examined the feasibility of obscuring programs and OS components plus looked at various levels where diversity could help defend computers from attacks. Forrest, Somayaji and Ackley examined potential sources of diversity within the operating system [7]. Their research discussed ways to introduce obfuscation such as changing memory layout, reordering code, adding padding to stack frames and changing names of important system files [7]. Cowan et al [5] evaluated and compared restrictiveness techniques, defined as methods that restrict certain behavior, to obscuring strategies, strategies that hide some system aspect from would-be attackers. The study discussed the relative merits of the two approaches and found that in most cases obscuring techniques are more difficult to implement plus introduce complexity which is less likely to be implemented correctly. Another study examined several well known attacks and questioned if diversity could have stopped the attacks from occurring [2]. The authors point out that in theory diversity could have helped resist the attack for some of the attacks examined. Yet, diversity for most OS and network versions is not very large and would likely not have presented much difficulty for the attackers. Another study by Deswarte [6] approached diversity for security from a fault tolerant viewpoint. He described the various faults that could affect systems and noted where diversity could assist with masking each fault. Deswarte considered diversity at five levels including, operator, user interface, OS level, N-version and execution level and believes that diversity can help ameliorate both design and intrusion (malicious) faults [6].

Current diversity research looks at very specific vulnerabilities that can be defended against through obfuscation. Instruction set diversity was examined in two separate studies [3, 9]. One used a binary translation technique [3] while the other performed a kernel modification to achieve instruction set variance [9]. Both techniques were guarding against code injection attacks. Neither technique is affective against all code injection attacks. Another study involved using randomization for address obfuscation [4]. They tried a number of ways to randomize the location of code and data. Their address randomization methodology will cover some but not all memory exploits. The assumption behind all three studies is that the obfuscation will result in greater effort by the attacker to compromise each randomized version.
Panel discussion of these and possibly other studies should help validate the diversity results achieved to date by the security community and perhaps suggest ways to quantify the effort for using diversity in system security.

References

[1] Avizienis, A. and L. Chen. On the implementation of N-version programming for software fault tolerance during execution. In *Proc. First IEEE-CS Int. Computer Software and Applications Conf. (COMPSAC 77),* Chicago, Nov., 1977.

[2] Bain, C., D. Faatz, A. Fayad, D. Williams. Diversity as a defense strategy for information systems. Mitre Corp., 2000.

[3] Barrantes, E. G. et al. Random instruction set emulation to disrupt binary code injection attacks. In *Proc. CCS'03, Oct. 27-31, Wash., D.C.,* 2003.

[4] Bhatkar, S., D.C. DuVarney and R. Sekar. Address obfuscation: an efficient approach to combat a broad range of memory error exploits. In *Proc. 12th Usenix Sec. Symp., Wash., D.C.,* 2003.

[5] Cowan, C., H. Hinton, C. Pu, J. Walpole. Cracker patch choice: an analysis of post hoc security techniques, 23rd NISSC, Baltimore, MD, 2000.

[6] Deswarte, Y., K. Kanoun, J. Laprie. Diversity against accidental and deliberate faults. In **Computer Security, Dependability and Assurance: From Needs to Solutions,** 1998, York, England.

[7] Forrest, S., A. Somayaji and D.H. Ackley. Building diverse computer systems. In *Proc. 6th Workshop Hot Topics in Operating Systems*, 1997, pp. 67-72.

[8] Geer, D. et al. Cyber Insecurity: the cost of monopoly, how the dominance of Microsoft products poses a risk to security. Computer & Communications Industry Association Report, 2003.

[9] Kc, G. S., A. D. Keromytis, V. Prevelakis. Countering code-injection attacks with instruction set randomization. *CCS'03, Wash., D.C.*, 2003.

[10] Knight, J. and N.G. Leveson. An experimental evaluation of the assumption of independence in multi-version programming. *IEEE Trans. Soft. Eng., SE-12, 1.*

[11] Littlewood, B. and D. R. Miller. Conceptual model of coincident failures in multi-version software. *IEEE Trans. Soft. Eng. SE-15,12*. 1989.

[12] Littlewood, B., P. Popov, and L. Strigini. Modeling software design diversity – A Review. *ACM Computing Surveys, Vol. 33, No. 2*, June 2001.

[13] Usenix Security Symposium, www.usenix.org, 2004.