MANAGING COMPLEXITY AND CHANGE!
INCOSE 2004 - 14th Annual International Symposium Proceedings

Systems Engineering Approach To Information Assurance

Kim H. Taylor
University of Maryland University College
3501 University Blvd. East
Adelphi, MD 20783
kimtaylor@att.net

Crystal D. Sloan
EagleRidge Technologies, Inc.
114 Ledgerwood Lane
Rockwood, TN 37854 USA
onetahiti@alum.mit.edu

Abstract. Today's security solutions for protecting telecommunications and information infrastructures are primarily defensive. They are not commensurate with the increasingly sophisticated nature of the global cyberthreat, and likely ineffective in countering *new, dormant* threats that could be a strategic surprise to organizational security. Defensive measures such as firewall and antivirus software can only protect computer networks and systems against *known signatures* of malicious code. To engineer pre-emptive capabilities and processes that discourage and repel attacks would require integration of key aspects of security domain engineering into systems engineering practices. An integrated approach would promote the implementation of security controls and processes as built-in features of future human/computer systems and networks, rather than as separate solutions, thus enabling a more holistic and robust information assurance.

Introduction

Networked computer systems have become the nerve centers that control most of the world's critical infrastructures. Developed countries depend on this technology for virtually every aspect of their livelihood, to include: banking and finance, energy, transportation, government, defense industrial base, information, and telecommunications. This makes such countries especially vulnerable to computer network attacks designed to disrupt its critical infrastructures (Clinton Administration 1998). To effectively counter this threat, one must devise pre-emptive capabilities that thwart and discourage malicious attacks. A closer coupling of the security and systems engineering activities at the change control process level is proposed, to develop a more robust defense of networked information systems.

Security is the System Engineer's Business

Security and systems engineering. Traditionally, security is not a systems engineering topic, unlike human factors, safety, supportability, maintainability, or reliability. No chapter on security appears in some standard texts (Sage and Rouse 1999, Blanchard and Fabrycky 1998), and some classic papers (Sheard 1996, McCauley 1992) on the extent of systems engineering do not even mention the word "security," although security concerns fit well in at least three of the SE roles identified by (Sheard 1996): system designer, system analyst, and logistics/ops. As governments and industries use Internet technologies more and more to conduct businesses, security engineering of computer, network, and information systems has emerged as a separate multidisciplinary domain, closely linked to software engineering. Systems security, however, need not and should not be limited to computer technology and software code. Current security wisdom holds that security needs to be more than mere products and measures that can be instituted or installed; security should be a continuous process (Schneier 2000). Systems security risks and solutions exist in the realms of technology, people, and physical organizations and facilities. Indeed, (Sheard and Moini 2003) state:

> “Security engineering is a multidisciplinary field encompassing many areas of expertise, ranging from the traditional computer security, and software engineering to knowledge of business process analysis and engineering, applied psychology, organizational method, audits, and the law.”

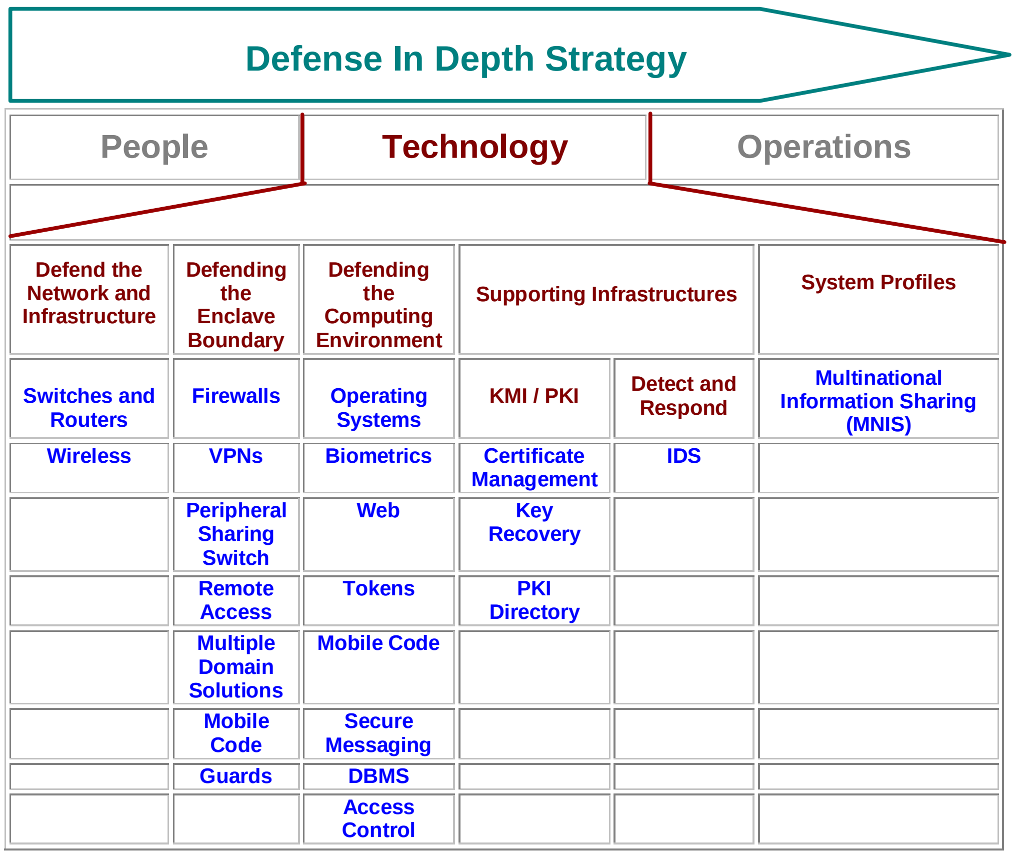
Of all those areas, computer security warrants a closer coupling, if not full integration, with systems engineering because of software dominance in almost all new complex systems. Moreover, virtually all complex system developments depend on software design and management tools. Therefore, it is important that system engineers understand and become involved in the challenge faced by software security engineers who work on systems development, implementation, and maintenance projects.

The asymmetric threat. Security engineers are fighting a hit-and-run cyber war, in which the attackers have been defining the battlefield from the beginning, ultimately inflicting two types of damage: disruption of service and information theft. Serious *capability gaps* exist between the defenders and attackers. Tactics, techniques, and tools to achieve the attackers' objectives abound, and are freely available to Internet users worldwide. In contrast, no countermeasures exist today that can effectively push back the threat. Thus far, the defense in depth strategy—using three extensive layers of security measures involving people, technology, and operations—has not deterred the attackers.

Facing the increasingly fortified layers of network defense such as firewalls, proxy servers and public key infrastructures, the attackers still control and define the battlefield in three ways. Firstly, attackers could easily use the distributed computing technology to harness the immense power of idle CPU cycles on the Internet, and launch a coordinated attack, such as the well-known distributed denial of service, without the users of compromised computers even knowing their PCs were used as attack agents (Cisco Systems 2003). Secondly, with the openness of the Internet, attackers could control the systems of passive and uninformed home PC users who do not implement security measures such as using the latest firewall and antivirus software and signatures. It is likely that the majority of home PCs are vulnerable and easy targets of malicious exploitation. Thirdly, the increasing global presence of governments, industries, and commerce on the Internet opens up windows of opportunities for attackers. Using simple, public tools such as the web browser they could quietly penetrate organizations and commit crimes such as theft of sensitive data (credit card information, etc.) with a very low risk of getting caught (Ghosh and Voas 1999).

“Defense In Depth” Strategy

Why it does not work? The technology component of the defense in depth strategy considers only the IT infrastructure and peripheral networking layers. See figure 1. The strategy evaluates and responds to intrusions or attacks only as they occur, and therefore is not effective in countering emerging and unknown threats. According to a (CERT/CC 2002) report on attack trends, the increasing sophistication of attack methods and techniques makes them more difficult to detect through signature-based systems such as the antivirus software and intrusion detection systems being used today. Analysis, lab testing, and reverse engineering of attack tools are becoming more challenging and time-consuming. Moreover, in the aftermath of the Love Bug virus, (Sullivan 2000) reported that a group of security developers demonstrated the potential for creating a far more sophisticated virus. The working model showed that a deadly, *silent* virus that would "work across platforms, do its work with stealth, and disappear before it could be stopped."



Defend the Network and Infrastructure	Defending the Enclave Boundary	Defending the Computing Environment	Supporting Infrastructures		System Profiles
Switches and Routers	Firewalls	Operating Systems	KMI / PKI	Detect and Respond	Multinational Information Sharing (MNIS)
Wireless	VPNs	Biometrics	Certificate Management	IDS	
	Peripheral Sharing Switch	Web	Key Recovery		
	Remote Access	Tokens	PKI Directory		
	Multiple Domain Solutions	Mobile Code			
	Mobile Code	Secure Messaging			
	Guards	DBMS			
		Access Control			

Figure 1. Current defense in-depth strategy on information systems security being adopted by the Information Assurance Technical Framework (IATF) forum of government, industry, and academia. The tools and methods listed in the lower table are used for the technology component and address only the defense of the network periphery and IT infrastructure. Defense mechanisms of web-enabled applications that manipulate organization's critical information structures are not featured in the strategy. Source: (IATFF, n.d.).

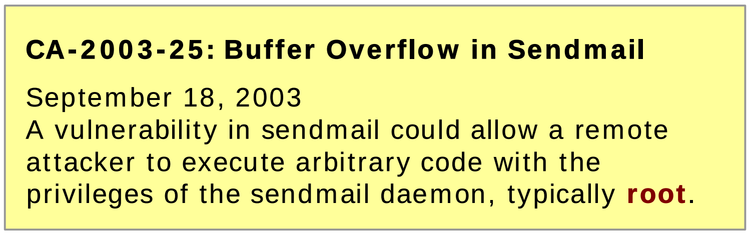
Attack methods and venues. Technically, there is nothing deadly, sophisticated, or complex about the *stealth* malicious code. The silent virus that (Sullivan 2000) described likely existed in one form or another since the birth of the digital information age. The attacker needs to do only two things. Firstly, he needs to identify one basic flaw in the software running on the target system(s) that can be exploited to achieve his objective. Secondly, he packages the code to look like billions of other packets on the Internet, so it can slip through the fortified network defense layers of intrusion detection, firewall, proxy server, and anti-virus software. How does anyone learn about exploitable flaws of complex software designed and written by industry experts? Go to the University of Carnegie Mellon Computer Emergency Response Team Coordination Center (CERT/CC) Web page. There, one can find current updates on the latest computer and network software design or implementation flaws, known as vulnerabilities.

Once the vulnerability is known, it is a matter of racing against time between four stakeholders. The attackers develop the malicious code to exploit the flaw and launch the attack. The vendors strive to correct the flaws and dispatch the fix to the public as rapidly as possible. The system and network administrators evaluate the impact of deploying the patch for their organizations, and decide whether or not to apply the patch. The home PC users may or may not know about the vulnerability, and may or may not implement the countermeasures. It is more than just a race. It is almost a mental game in cyber warfare, where all can see the array of targets, opportunities, and capabilities. For the attacker, the rule of engagement is to leverage commonly available methods and venues, exploit well-known vulnerabilities, and use the opponents' own tools and capabilities against their weaknesses. The objective is not only to inflict damage, but also to escape detection through Trojan horse techniques.

A case in point: buffer overflow. A "buffer overflow" condition occurs

> "when more data is put into a buffer or holding area, then the buffer can handle. This is due to a mismatch in processing rates between the producing and consuming processes. This can result in system crashes or the creation of a back door leading to system access" (FOLDOC 1996).

From the systems and security engineering point of view, buffer overflow is a well-known vulnerability that originates as a software architecture design at the input/output processing level. It could be introduced during the design, coding and implementation phases. If the buffer overflow vulnerability was overlooked during the testing phase, active scanning and monitoring could still reveal its presence and neutralize its potential impact during the deployment and maintenance phase. Yet, it remains one of the most prevalent vulnerabilities reported by CERT/CC since January 1997.



CA-2003-25: Buffer Overflow in Sendmail

September 18, 2003
A vulnerability in sendmail could allow a remote attacker to execute arbitrary code with the privileges of the sendmail daemon, typically **root**.

Figure 2. Recent CERT Advisory on Sendmail vulnerability. Sendmail is a function used by mail applications. As of September 29, 2003, seventeen vendors and software groups (such as FreeBSD) have released software patches (fixes) for their affected products. Another buffer overflow advisory for Windows™ machines soon followed. *Sources*: (CERT/CC 2003b, CERT/CC 2003c).

Indeed this vulnerability existed in commonly used UNIX library functions such as Sendmail, Windows™ library functions such as Windows™ Workstation Service (WKSSVC.DLL), and networking protocols such as the International Telecommunications Union standard protocol H.323 used in IP telephony and other multimedia products. See figure 2 for the description of Sendmail buffer overflow vulnerability reported by

(CERT/CC 2003a). WKSSVC.DLL and H.323 vulnerabilities were reported by (CERT/CC 2003a, NISCC 2004).

Of the 30 plus "new" vulnerabilities reported by CERT/CC in 2003, at least 15 were related to buffer overflow; 15 of 35 advisories issued in 2002 were based on buffer overflow sighting; and 16 of 41 advisories in 2001 met the same condition. In 1999, Ghosh and Voas observed that buffer overflow accounted for almost half of the major security bugs reported by CERT/CC. Buffer overflow could cause network performance problems, denial of service, and allow access to privileged network/system operations and sensitive data. In fact, in the reported case in figure 2, with *root* privileges the attacker will have total control of the target network and computer resources. As recently as March 17, 2004, the U.S. Computer Emergency Readiness Team identified "Phatbot" Trojan as malware that may exploit numerous existing buffer overflow vulnerabilities, such as the previously stated Windows™ Workstation Service. In 2001, Code Red worm successfully exploited this vulnerability causing defacement of target web sites, creating packet-flooding denial of service, and halting the forwarding of packets by certain CISCO routers (CERT/CC 2002). The clean-up cost was estimated at $2.62 billion (Computer Economics 2002). One would think that after more than six years, a vulnerability as deadly as *buffer overflow* would be eradicated by now. Yet, that is not the case.

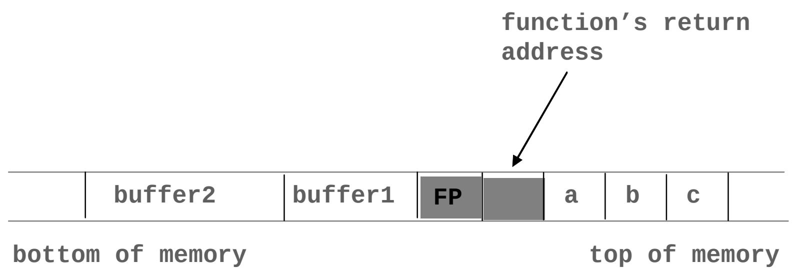


Figure 3. Memory stack region layout. The return address –or address of the next instruction to be executed after a function completes its task—is always stored next to the frame pointer (FP). By sending a calculated amount of data that exceeds the size of buffer 1, the attacker could alter the return address field to point to malicious code instructions. *Source*: (Grover 2003).

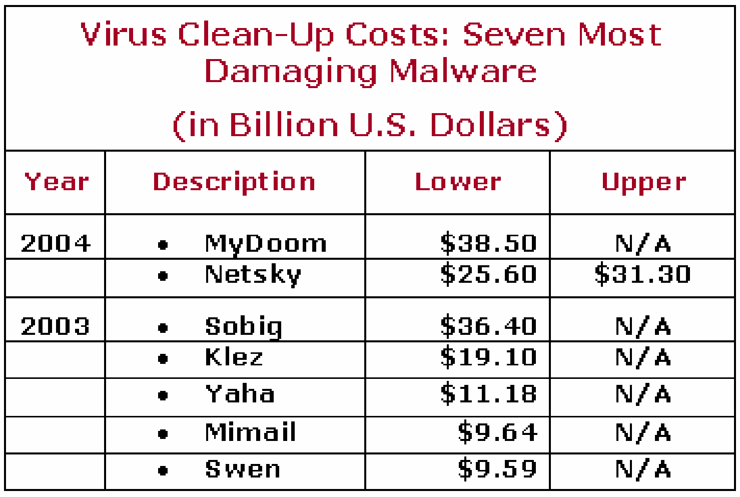
To understand why countermeasures such as the defense in depth strategy have been ineffective in protecting information structures from long running vulnerabilities, one needs to examine how a *potential* buffer overflow condition may be exploited. The vulnerability has little to do with network backbone, infrastructure, and entry/exit points. It has much to do with the coding and internal logic of software programs, and standard stack addressing techniques of modern operating systems. As shown in figure 3, the coding technique does not check the length of received data sent to a buffer of predefined size in memory. The layered "defense in depth strategy" and current practices of separating systems and security engineering activities do not facilitate the early discovery and prevention of this type of coding vulnerability. As (Schneier 2000) notes, "Security cannot be functionality-tested." This paper proposes a "deeper," more *active* defense in depth of networked information systems.

Active "Defense In Depth"

Neutralize the attacks at the application layer. There are two ways to neutralize the attackers in this asymmetric warfare: 1) eliminating tools and capabilities, or 2) eliminating one's weaknesses. Eliminating global internetworking tools and capabilities effectively removes the

Internet from existence, therefore is not a viable option. The second option—eliminating vulnerabilities—is feasible, desirable, and should be a common goal of both systems and security engineers. According to (Arbaugh, Finthen, and McHugh 2000), “anecdotal evidence suggests that known and patchable vulnerabilities cause the majority of system intrusions.” It should come as no surprise that the “defense in depth” strategy seeks to eliminate vulnerabilities at several levels: global network backbone, entry/exit points of an organization's intranet, and computing infrastructure such as operating systems and middleware.

An *active* defense in depth would further seek to root out vulnerabilities at their birthplace or application level. To accomplish this goal, an *active* defense in depth must be a *departure from* the traditional way of deferring all security controls planning, development, implementing, and testing until after the system is deployed. This deferred security mode is still prevalent today as accreditation and certification are considered only because they are mandated at deployment time, before the system can be used in operations. With this approach, an interim approval to operate is normally granted until full accreditation of the system can be obtained, based on successful functional and security testing. This engenders today's norm of “vulnerability patch” mode of operation for information system security. It characterizes the widespread defensive and reactive responses to deadly software viruses since the 1990s, resulting in costly network and software cleanups after the fact, and other economic impacts such as loss of business and


Virus Clean-Up Costs: Seven Most Damaging Malware

(in Billion U.S. Dollars)

Year	Description	Lower	Upper
2004	• MyDoom	$38.50	N/A
	• Netsky	$25.60	$31.30
2003	• Sobig	$36.40	N/A
	• Klez	$19.10	N/A
	• Yaha	$11.18	N/A
	• Mimail	$9.64	N/A
	• Swen	$9.59	N/A

Figure 4. Major viruses and worms have incurred the most significant economic damage since 2003. Note that 2004 data is through February only. Economic damage was measured in terms of overtime payments, contingency outsourcing, loss of business, bandwidth clogging, productivity erosion, management time reallocation, cost of recovery, and software upgrades. Source: (mi2g™ Limited 2004).

productivity, as shown in figures 4 and 5 below.

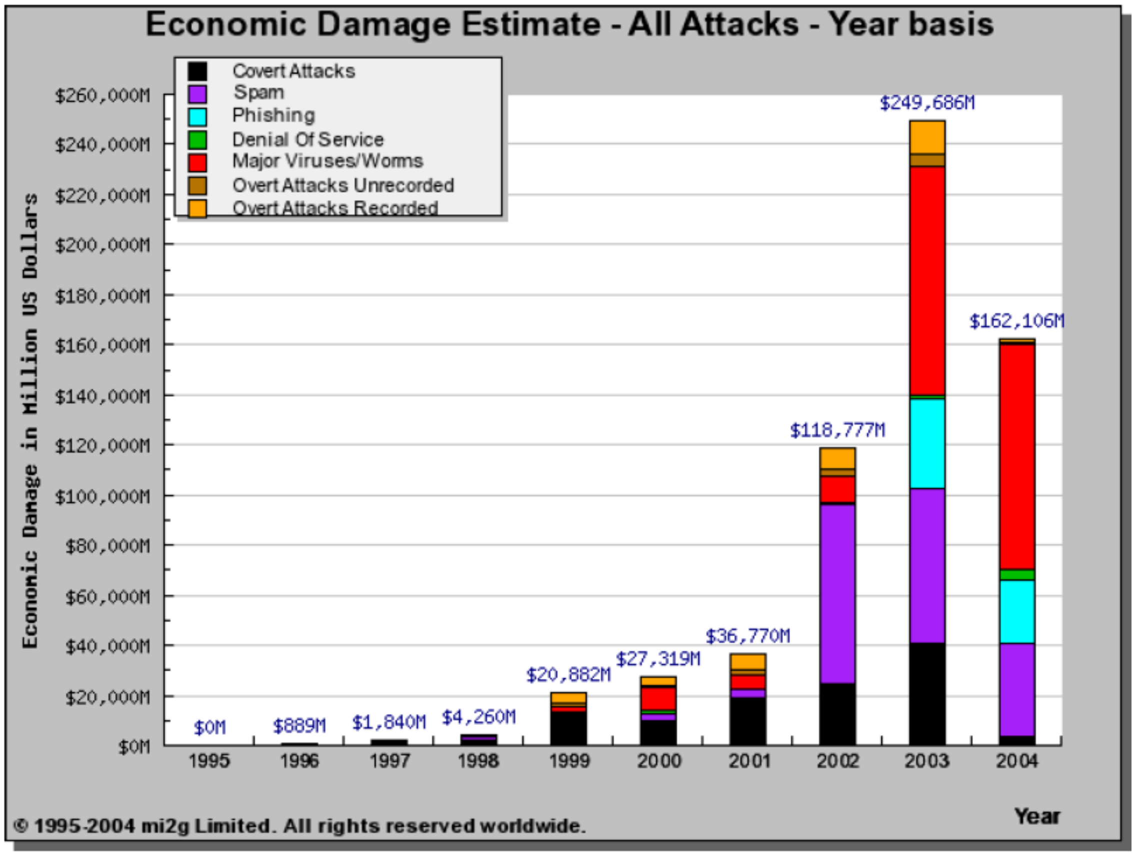


Figure 5. Note that 2004 data is through February only. See FAQ on SIPS and EVEDA at `http://www.mi2g.net/cgi/mi2g/` for the description of the algorithm used to estimate the global economic damage of malware. Source: (mi2g™ Limited 2004).

To counter the threat more effectively, Sheard and Moini suggested a more closely coupled relationship between systems and security engineering as follows:

> "Security is related to a system's complexity, connectivity, and commonality and thus touches all aspects of engineering. . . Good security begins with an awareness of security requirements and implementation of secure features into the architecture of the system. The pros and cons of various security strategies include understanding the business value of the systems, analyzing the threat environment, and identifying and migrating risks."

One can see this goal of *active* defense in depth being embraced recently by the industry (Sima 2002) and mandated by ISO standard 15288, Life Cycle Management – System Life Cycle Processes (ISO 2002). An active defense in depth is also reflected in the National Institute of Standards and Technology (NIST) *Guide for the Security Certification and Accreditation of Federal Information Systems* (Ross and Swanson 2003). NIST guide suggests ten information security program activities associated with the development of information systems, which can be mapped into the system life cycle model as shown in figure 6.

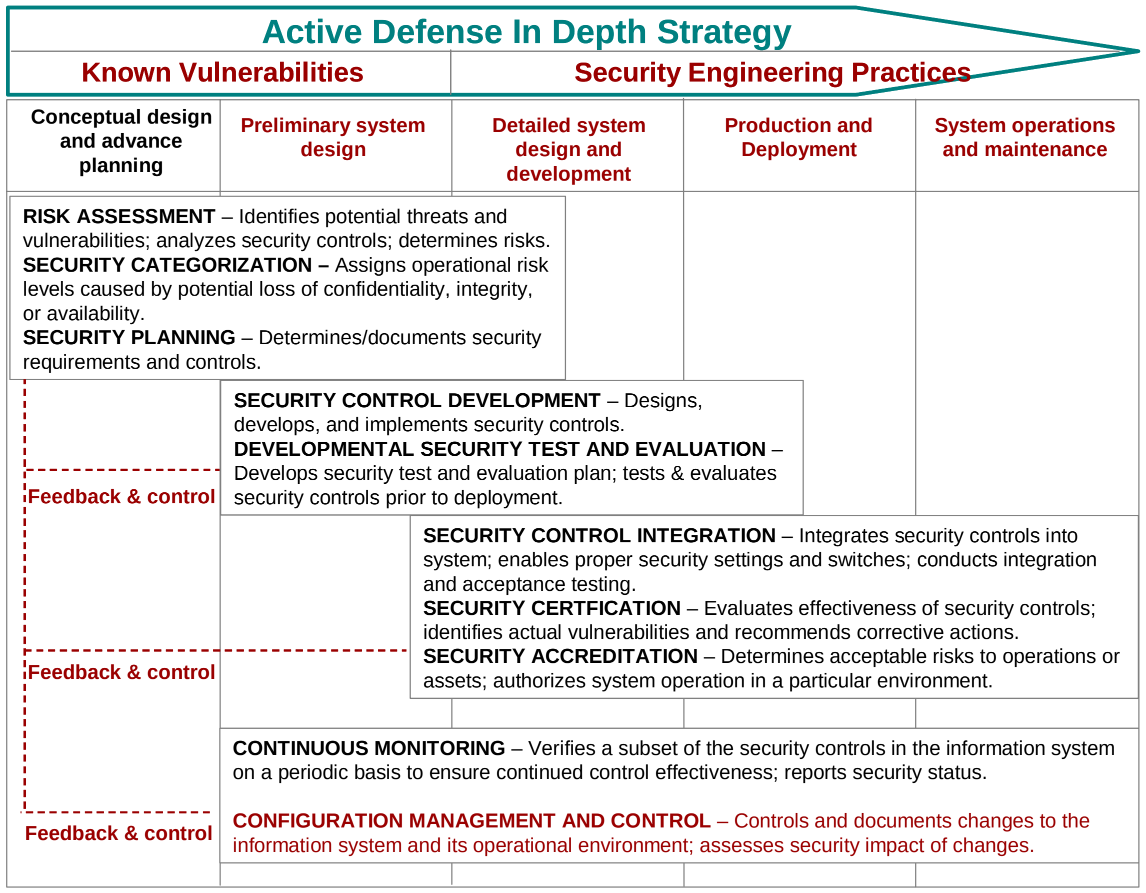

Active Defense In Depth Strategy
Known Vulnerabilities
Security Engineering Practices
Conceptual design and advance planning
Preliminary system design
Detailed system design and development
Production and Deployment
System operations and maintenance
RISK ASSESSMENT – Identifies potential threats and vulnerabilities; analyzes security controls; determines risks.
SECURITY CATEGORIZATION – Assigns operational risk levels caused by potential loss of confidentiality, integrity, or availability.
SECURITY PLANNING – Determines/documents security requirements and controls.
SECURITY CONTROL DEVELOPMENT – Designs, develops, and implements security controls.
DEVELOPMENTAL SECURITY TEST AND EVALUATION – Develops security test and evaluation plan; tests & evaluates security controls prior to deployment.
Feedback & control
SECURITY CONTROL INTEGRATION – Integrates security controls into system; enables proper security settings and switches; conducts integration and acceptance testing.
SECURITY CERTFICATION – Evaluates effectiveness of security controls; identifies actual vulnerabilities and recommends corrective actions.
SECURITY ACCREDITATION – Determines acceptable risks to operations or assets; authorizes system operation in a particular environment.
Feedback & control
CONTINUOUS MONITORING – Verifies a subset of the security controls in the information system on a periodic basis to ensure continued control effectiveness; reports security status.
CONFIGURATION MANAGEMENT AND CONTROL – Controls and documents changes to the information system and its operational environment; assesses security impact of changes.
Feedback & control

Figure 6. Integrated system and security engineering framework. The highlighted phases (in reddish brown) represent key points of the life cycle where security vulnerabilities can be introduced. *Sources*: Adapted from (Blanchard & Fabrycky 1998) and (Ross & Swanson 2003).

Risk assessment, security categorization, and security planning. These three activities should be performed at the start of the life cycle, during the conceptual design / advance planning and extending into the preliminary phase. Most organizations have their own security policy and procedures that may or may not be commensurate with the level of threat they may be facing. Also, traditional organizations are likely to manage information-related assets (information and systems) more from the financial and capability investment point of view than from the security point of view. It is important that system and security engineers coordinate their efforts in assessing risk areas of the system development efforts. Because of the complexity of systems security, there will likely be several security engineers assessing, categorizing and planning different aspects of security throughout the life cycle: physical security, hardware security, data security, information security, software security, applications security, and network security. Sheard and Moini acknowledged that "the current state of the art for security of systems larger than software has many unknowns." This makes it even more imperative that system engineers become involved in security issues right from the start of the life cycle.

Security control development, and developmental security test and evaluation. It is important that these two activities take place during the preliminary design and detailed system design/development. Using the *buffer overflow* vulnerability as an example, design criteria and code development standards could be established during the design/development phase to control the use of C library functions that cause the vulnerability: `strcpy (), strcat (), sprintf (), and vsprintf ()`. Commonly used functions such as `gets ()` and `scanf ()` may also cause buffer overflows (Grover 2003). What those functions have in common is the way they accept input data strings until encountering a designated character –such as null or EOF symbol—signifying the end of the data string. By accepting data strings, those functions preclude the need of defining elaborate data record formats and structures for simple input and output. This built-in simplicity of I/O functions makes them vulnerable to programmer, user and data errors when the end of data string marker is "lost" or omitted, resulting in buffer overflow (Kelley and Pohl 1995).

Stamping out buffer overflow type of vulnerability may involve management policy and control, requiring close coordination and full cooperation between the system, software and security engineers. Indeed, a more robust and cost-effective defense would be to mandate the use of high level and *type-safe* programming languages such as COBOL, Java, etc. that ensure operations are applied only to values of the appropriate type (McGraw and Morrisett 2000). Where C-programming is required, built-in security code should be mandated to prevent out-of-bound memory addressing by potential malicious code. See (Yong and Horwitz 2003) for details on some of the security-enforcement coding techniques for C programs. In addition to conducting thorough testing of *known* vulnerabilities during the design/development phase, identifying security controls and testing procedures for potential *new* vulnerabilities in the system/network architecture, functions and code should also take place.

Security control integration, security certification, and security accreditation. The system engineer must ensure that the integration of security controls, and security certification and accreditation planning activities begin at the start of the detailed design/development phase vice deferring them to security engineers until after system deployment. During this phase, both the system and security engineers should review and validate security controls in the context of the identified threats, vulnerabilities and overall system security management approach that were defined during the conceptual phase. Security certification and accreditation are part of a continuing and iterative process throughout the system life cycle. As changes are planned and new capabilities implemented, the entire system or affected components will need to be re-certified and reaccredited by security experts. Started during the detailed system design and development phase, the integration of security controls and system certification/accreditation testing become an integral part of the deployment, operations and maintenance of the system.

Continuous monitoring, and configuration management and control. Throughout the *design/development*, *deployment*, and *maintenance* phases, information systems are typically in a "constant state of migration" (Ross and Swanson 2003). Design changes; upgrades to hardware, software or firmware; vulnerability patches; changes to physical facilities, power and other environmental factors, etc. are not uncommon during the life cycle of a system. Any of those changes can lead to errors that have a security impact on the information system. An *active* defense in depth strategy would strive to root out those vulnerabilities at all their three possible birthplaces –design/development, deployment, and maintenance phases (Arbaugh, et. al. 2000)—instead of continuing the current way of scanning deployed systems for vulnerabilities after the fact. An active defense in depth would also seek to identify and document security requirements and controls as part of the system requirements definition and analysis. Moreover, the full feedback and control flow between the different groups of security activities would ensure a comprehensive and coordinated approach to configuration management, and control of dynamic and fast changing technology developments. A good mechanism to catch hard-to-find vulnerabilities –whether known or newly introduced—and initiate timely corrective action is the change control process of the life cycle, as it spans all the three phases where vulnerabilities could be introduced. This would require full integration of the systems and security engineering in the change control process, as depicted in figure 7.

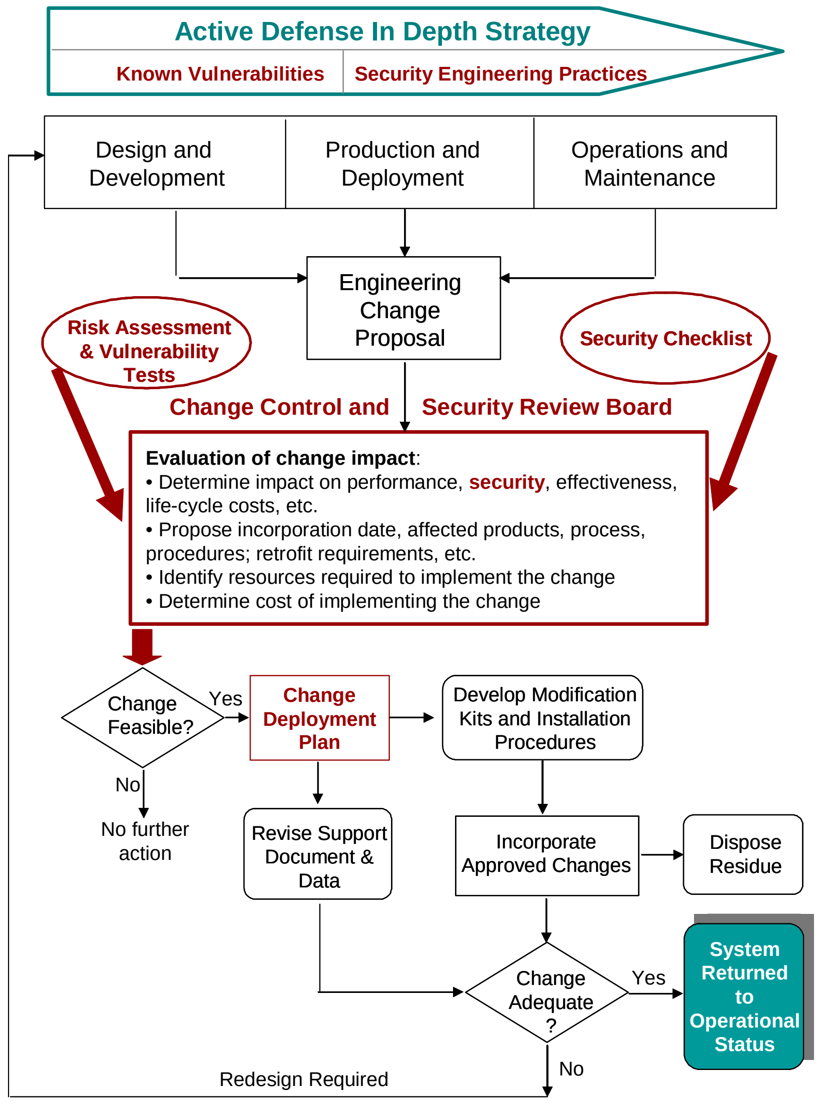

Active Defense In Depth Strategy
Known Vulnerabilities
Security Engineering Practices
Design and Development
Production and Deployment
Operations and Maintenance
Engineering Change Proposal
Risk Assessment & Vulnerability Tests
Security Checklist
Change Control and
Security Review Board
Evaluation of change impact:
• Determine impact on performance, security, effectiveness, life-cycle costs, etc.
• Propose incorporation date, affected products, process, procedures; retrofit requirements, etc.
• Identify resources required to implement the change
• Determine cost of implementing the change
Change Feasible?
Yes
No
No further action
Change Deployment Plan
Develop Modification Kits and Installation Procedures
Revise Support Document & Data
Incorporate Approved Changes
Dispose Residue
Change Adequate ?
Yes
No
System Returned to Operational Status
Redesign Required

Figure 7. System Change Control Process is modified to integrate the change control and security review activities into a single control board. The change control and security review board is an effective mechanism to intercept security vulnerabilities that could be introduced at three key points of the system development cycle: design, deployment and maintenance. *Source*: Adapted from (Blanchard and Fabrycky 1998).

Integrated engineering change control and security review board (IECC/SRB). A main focus of this integrated control (or management) board should be to ensure a rigorous evaluation of the proposed change impact on the design/performance criteria, reliability, maintainability, supportability, and *security* of the system. This would enable the consideration of security challenges based on the whole system concept, as an integral part of the three key operational elements: reliability, maintainability and supportability. The risk analysis and management must consider applicable global security threats, in addition to technology and other program risks. The *security checklist* of best practices, policy and controls should be weighed against the threats and other engineering issues to make informed decisions during the change approval process. Thorough risk assessments and vulnerability tests should be conducted for both known and potential security problems that may arise as part of the proposed change deployment. The increasing complexity of the security engineering domain would require security engineers from different disciplines to participate and play an active role on the IECC/SRB. The integrated board would enable engineering of security functions and protection mechanism as inherent features of information system products versus as separate layers of defense around the products.

Conclusion

The proposed integration of systems and security engineering throughout the life cycle ensures a more robust defense in depth strategy to counter cyberthreats. It is a departure from today's practice of deferring complex and interwoven system, software and security issues until after the system deployment phase. The integrated approach will allow systems and security engineering to build from a single system baseline. Furthermore, security controls will be implemented as inherent features of the system, vice as separate solutions and vulnerability patches. It will help developers comply with the security certification and accreditation guidelines for public information technology systems. Most of all it will drastically cut the overall system life cycle costs by preventing billion dollars in clean-up expenses of reactive and defensive security countermeasures.

References

Arbaugh, W. A., Finthen, W. L., and McHugh, J., “Windows of vulnerability: a case study analysis,” *IEEE Computer*. Institute of Electrical and Electronic Engineers (IEEE), December 2000.

Blanchard, B. S. and Fabrycky, W. J., *Systems Engineering and Analysis, Third Edition*. Prentice Hall, Upper Saddle River, New Jersey, 1998.

CERT, *Survivable Systems Engineering*. CERT, June 17, 2003. Retrieved September 21, 2003 from `http://www.cert.org/sna/`

(CERT/CC 2003a), *CERT/CC Statistics 1988-2003*. CERT/CC, October 17, 2003. Retrieved October 19, 2003 from the CERT Coordination Center Web site: `http://www.cert.org/stats/`

(CERT/CC 2003b), *CERT® Advisory CA-2003-25 Buffer Overflow in Sendmail*. CERT/CC, September 29, 2003. Retrieved March 21, 2004 from the CERT Coordination Center Web site: `http://www.cert.org/advisories/CA-2003-25.html`

(CERT/CC 2003c), *CERT® Advisory CA-2003-28 Buffer Overflow in Windows Workstation Service*. CERT/CC, November 20, 2003. Retrieved March 21, 2004 from the CERT Coordination Center Web site: `http://www.cert.org/advisories/CA-2003-28.html`

(CERT/CC 2002a). CERT/CC, *Overview of Attack Trends*. CERT/CC, 2002. Retrieved September 26, 2003 from the CERT Coordination Center Web site: `http://www.cert.org/archive/pdf /attack_trends.pdf`

(CERT/CC 2002b). CERT/CC. (2002, January 17). *CERT® Advisory CA-2001-23 Continued Threat of the "Code Red" Worm*. Retrieved October 26, 2003 from the CERT Coordination Center Web site: `http://www.cert.org/advisories/CA-2001-23.html`

(CERT/CC 2002c). CERT/CC. (2002, January 17). *CERT® Advisory CA-2001-19 "Code Red" Worm Exploiting Buffer Overflow in IIS Indexing Service DLL*. Retrieved October 26, 2003 from the CERT Coordination Center Web site: `http://www.cert.org/advisories /CA-2001-19.html`

Cisco Systems, Inc., “*Strategies to Protect Against Distributed Denial of Service (DDoS) Attacks*.” Cisco Systems, Inc., April 29, 2003. Retrieved November 11, 2003 from Cisco Web site: `http://www.cisco.com/warp/public/707/newsflash.pdf`

Clinton Administration, “*The Clinton Administration's Policy on Critical Infrastructure*

Protection: Presidential Decision Directive 63." Critical Infrastructure Assurance Office, May 1998. Retrieved September 24, 2003 from `http://www.ciao.gov/publicaffairs /pdd63printer.html`

Computer Economics, Inc., *2001 Economic Impact of Malicious Code Attacks*. CyberSecure, Inc., Fredericton, NB, Canada. January 2, 2002. Retrieved October 18, 2003 from the World Wide Web: `http://www.cybersecure.ca/q2.htm`

FOLDOC,"Buffer Overflow," *Free Online Dictionary of Computing (FOLDOC)*. FOLDOC, 1996. Retrieved October 25, 2003 from `www.tsl.state.tx.us/ld/pubs/compsecurity/glossary.html`

Ghosh, A. K. and Voas, J. M., "Inoculating software to survivability," *Communications of the ACM*, 38-44. ACM, July 1999.

Grover, S., "Buffer overflow attacks and their countermeasures," *Linux Journal*. Specialized Systems Consultants (SSC), Inc., Seattle, WA, March 10, 2003. Retrieved October 25, 2003 from `http://www.linuxjournal.com/article.php?sid=6701.`

IATF, *Chapter 2: Defense in Depth*. Information Assurance Technical Framework (IATF), September 2002. Retrieved October 12, 2003 from `http://www.iatf.net/framework_docs/version-3_1/file_serve.cfm?chapter =ch02.pdf.`

IATFF, *Protection Profiles*, Information Assurance Technical Framework Forum (IATFF), (n.d.). Retrieved March 21, 2004 from `http://www.iatf.net/protection_profiles/profiles.cfm`.

ISO/IEC 15288, *Life Cycle Management – System Life Cycle Processes*, 2002.

Kelley, A. and Pohl, Ira, *A Book on C* (3rd Ed.). The Benjamin Cummings Publishing Company, Inc., New York, NY, 1995.

McAuley, J., "What Should Systems Engineers Know and When Should They Know It," *Proceedings of INCOSE*, INCOSE, 1992.

McGraw, G. and Morrisett, G., "Attacking malicious code: a report to the Infosec Research Council," *IEEE Software*, 33-41. IEEE, September/October 2000.

(mi2g™ Limited 2004a, "Yearly Economic Damage Estimate – All Attacks (from 1995)," SIPS Report – February 2004. mi2g™ Limited, London.

(mi2g™ Limited 2004b), "MyDoom becomes most damaging malware as SCO is paralysed," *News Alert*, February 1, 2004. Retrieved from mi2g Web Site: `http://www.mi2g.net/`

mi2g™ Limited, "Predictions for 2003 – How Accurate Was mi2g?" *News Release*, .December 8, 2003. Retrieved February 25, 2004 from mi2g Web Site: `http://www.mi2g.net/`

NISCC Vulnerability Advisory 006489/H323, *Vulnerability Issues in Implementations of the H.323 Protocol*, March 8, 2004. Retrieved March 21, 2004 from `http://www.uniras.gov.uk/vuls/2004/006489/h323.htm`

Ross, R. and Swanson, M., *Information Security: Guide for the Security Certification and Accreditation of Federal Information Systems*, Second Public Draft. National Institute of Standards and Technology (NIST), Washington, DC, June 2003. Retrieved September 23, 2003 from `http://csrc.nist.gov/publications/drafts/sp800-37-Draftver2.pdf`

Sage, A. and Rouse, W. (eds.), *Handbook of Systems Engineering and Management*. Wiley, 1999.

Schneier, Bruce, "The Process of Security," *Information Security Magazine*. Information Security, Needham, MA, April 2000. Retrieved Nov. 21, 2003 from `http://`

infosecuritymag.techtarget.com/articles/april00/columns_cryptorhythms.shtml

Sheard, S. A., "Twelve Systems Engineering Roles," *Proceedings of the International Council on Systems Engineering (INCOSE) 6th International Symposium.* INCOSE, Boston, 1996.

Sheard, S. A. and Moini, A., "Security engineering awareness for systems engineers," *Proceedings of the International Council on Systems Engineering (INCOSE) 2003 Symposium.* INCOSE, July 2003.

Sima, C., "*Security at the next level: are your web applications vulnerable?*" SPI Dynamics, Inc., 2002. Retrieved October 24, 2003 from: http://www.spidynamics.com/whitepapers/webappwhitepaper.pdf

Sullivan, E., "Next virus will be silent killers," *HPCwire*. Tabor Communications, Inc., San Diego, CA, May 19, 2000. Retrieved Nov. 21, 2003 from http://www.csa.ru/CSA/News/msg00850.htmr

US Computer Emergency Readiness Team (US-CERT), *Phatbot Trojan*, March 18, 2004. Retrieved March 21, 2004 from US-CERT Current Activity Web site: http://www.us-cert.gov/current/

Yong, S. H. and Horwitz, S., "Protecting C programs from attacks via invalid pointer dereferences," *ACM ESEC/FSE '03 Conference Proceedings*. ACM, September 2003. Retrieved October 26, 2003 from ACM digital database.

Biographies

Kim Taylor is Director of Information Systems and Technology, U.S. Pacific Command. She has held other lead technical and management positions with both the private industry and government, to include systems engineer with IBM World Trade Corporation (Tehran, Iran), lead systems programmer and project manager with Pacific Air Forces (Hawaii), systems analyst and program manager with Headquarters United States European Command (Stuttgart, Germany), and chief of networks division, Wright-Patterson Air Force Base, Ohio. Kim is currently pursuing the MS in Information Technology with University of Maryland Graduate School of Management and Technology.

Crystal D. Sloan is a principal of EagleRidge Technologies, Inc. (ertin.com), a Rockwood, Tennessee consulting and e-commerce firm, contributor to a number of books on computing, and a recipient of the Microsoft® MVP award. From 1999 until his death in July 2003, Ms. Sloan also worked with the late Dr. William H. McCumber on the development and teaching of Web-based graduate classes at the University of Maryland University College (UMUC); she continues to assist with classes there. Past projects include development of systems for the space shuttle, medical applications, and radiation monitoring of nuclear plants, many medical laboratory instruments, and numerous business applications. Ms. Sloan holds a S.B. in Mathematics from M.I.T.