mintaka™
TECHNOLOGY GROUP

DO YOU *REALLY* KNOW WHAT YOUR PROGRAMMERS ARE DOING?

Firewalls, Gateways And Intrusion Monitoring Software Have The Functional Equivalence Of Flimsy Cockpit Doors Against Malicious Coding Activities From Attacks By Insiders

Mintaka Technology Group
9903 Santa Monica Blvd #516
Beverly Hills, CA 90212
Phone 323 793-5224
Fax 323 464-5327
www.mintaka.com

White Paper

Do You Really Know What Your Programmers Are Doing?

ABSTRACT

Firewalls, gateways and intrusion monitoring software have the functional equivalence of flimsy cockpit doors against malicious coding activities from attacks by "insiders." Currently, U.S. companies have limited guarantees that the integrity of their software assets for mission critical applications has not been compromised by any malicious programmer activity.

Table of Contents

0. Overview

Information Technology is emerging as one of the key battlefields of the future - the rising number of security threats to source code and intellectual property demonstrates a worrying trend of targeted western corporations. While the Internet is a focus for action against cyber terrorism, securing networks with firewalls, gateways and intrusion monitoring software are appropriate countermeasures. However, those efforts have the functional equivalence of flimsy cockpit doors against malicious coding activities to the source code by attacks from programmer "insiders."

Furthermore, the globalization of software development and the localization of service delivery have combined to enable the dramatic growth of the offshore development industry. According to IDC, spending on offshore development by U.S. companies will increase from $5.5 billion in 2000 to more than $17.6 billion in 2005. Forrester Research predicts that 3.3 million jobs and $136 billion in wages are expected to move offshore by 2015. Gartner projects that by 2004 over 80% of U.S. companies will consider outsourcing critical IT services to countries such as India, Pakistan, Russia and China.

The increasing reliance on IT outsourcing raises serious concerns about the theft of intellectual property as well as the very integrity of the source code being produced. It is ironic that the countries with a history of intellectual property theft and those which companies trust the least with binary code are the places where software code development is being sent. Of greater concern is that many of these countries have known terrorist networks and there is no way to ascertain the security risk of the workers used to produce software components for mission critical applications.

Additionally, Meta Group predicts that as many as 50% of all U.S. IT workers could swing to contract labor by 2007. A highly mobile workforce coupled with insufficient programmer accountability increases the opportunity for IT workers to steal valuable intellectual property and maliciously sabotage software projects with almost total impunity. Corporate and government mission critical systems dwarf the Internet in terms of data and investment. Computer code assets are the exposed underbelly of our nation's digital infrastructure, wide open to abuse from offsite and onsite "insiders".

Application security vulnerabilities are the most significant categories of security problems challenging senior IT Executives today. Software is ubiquitous and bad software is at the heart of all computer security problems. Companies currently have inadequate safeguards in place to deter "insider" programmers who abuse the trust and privileges granted to them, from intentionally harming source code with almost total impunity. The threat to the source code has never been greater - the current level of awareness by organizations is not commensurate to that threat. It is the critical priority for companies today to expand security measures from simply addressing system operational weaknesses to include unprotected software developmental vulnerabilities.

> *"We're not talking about a cyber Pearl Harbor, but something more insidious and harmful such as terrorists integrating a digital component into traditional warfare. It's only a matter of time before they recognize that as a weapon. This is where we are only as strong as our weakest link."*
>
> Casey Dunlevy in "New Department may help craft cyber security strategy."
> National Journal's Technology Daily, June 7, 2002

This White Paper discusses the existence of vulnerabilities in the software development environment and the inherent risks to organizations from the threat of malicious coding activities by "insider" programmers, especially in light of the current unpredictable global security climate. It also addresses critical business issues to mitigate that threat with qualified risk management techniques, team collaboration tools and, critically, a verification and audit process that increases programmer accountability resulting in greater visibility of the software development process.

1. The Definition of Computer Security

Computer security means to protect information against unauthorized access, modification or destruction. It deals with the prevention and detection of illicit acts by users. This definition implies an understanding of the value of the information in order to develop protective measures[1].

A rough classification of protective measures in computer security is as follows:

- Prevention: Measures to prevent illicit damage, change or theft of information;
- Detection: Measures to detect who, when and the how of an information attack;
- Reaction: Measures to allow recovery of original information.

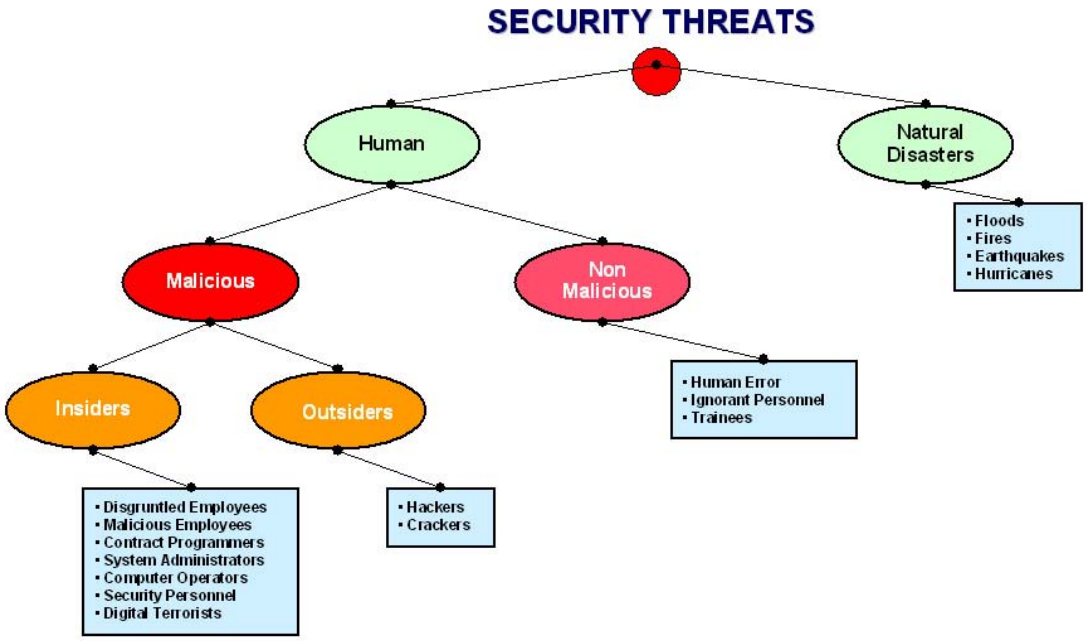
Strong measures can only be created from an understanding of how they can be compromised[2]:

- Confidentiality: Preventing unauthorized disclosure;
- Integrity: Preventing erroneous or malicious modifications;
- Availability: Preventing unauthorized withholding of information and resources;
- Authentification: Verification of user identity;
- Authorization: Validation of authorized access to sensitive systems.

The object of computer security is to protect valuable and sensitive organizational information from attacks while making it readily available to authorized users. While the information revolution opened new avenues for IT, it also opened new possibilities for crime.

2. The Nature of Security Threats

The greatest threat to computer systems and information comes from humans, through actions that are either malicious or ignorant[3]. Attackers, trying to do harm, exploit vulnerabilities in a system or security policy employing various methods and tools to achieve their aims. Attackers usually have a motive to disrupt normal business operations or to steal information.



The diagram is a schematic of the types of security threats that exist. Only those by malicious “insiders” will be discussed, although unintended coding errors can have severe consequences.

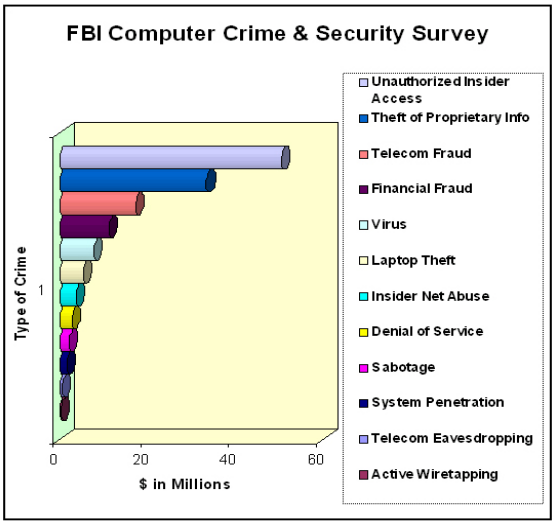
[1] “Computer Security,” Dieter Gollman, 1999
[2] “Department of Defence Trusted Computer System Evaluation Criteria,” (Orange Book) NCSC, 1983
[3] “Security Threats.” www.microsoft.comtechnet/security

The greatest threat of attacks against computer systems are from “insiders” who know the codes and security measures that are in place[4&5]. With very specific objectives, an insider attack can affect all components of security. As employees with legitimate access to systems, they are familiar with an organization’s computer systems and applications. They are likely to know what actions cause the most damage and how to get away with it undetected. Considered "members of the family," they are often above suspicion and the last to be considered when systems malfunction or fail.

Disgruntled employees create mischief and sabotage against systems. Organizational downsizing in both public and private sectors has created a group of individuals with significant knowledge and capabilities for malicious activities[6] and revenge. Contract professionals and foreign nationals either brought into the U.S. on work visas to meet labor shortages or from offshore outsourcing projects are also included in this category of knowledgeable insiders.

Common cases of computer-related employee sabotage include: changing data; deleting data; destroying data or programs with logic bombs; crashing systems; holding data hostage; destroying hardware or facilities; entering data incorrectly, exposing sensitive and embarrassing proprietary data to public view such as the salaries of top executives. Insiders can plant viruses, Trojan horses or worms, browse through file systems or program malicious code with little chance of detection and with almost total impunity.

A 1998 FBI Survey[7] investigating computer crime found that of the 520 companies consulted, 64% had reported security breaches for a total quantifiable financial loss of $136 millions. (See chart)



The survey also found that the largest number of breaches were by unauthorized insider access and concluded that these figures were very conservative as most companies were unaware of malicious activities or reluctant to report breaches for fear of negative press.

The survey reported that the average cost of an attack by an outsider (hacker) at $56,000, while the average insider attack cost a company in excess $2.7 million.

It found that hidden costs associated with the loss in staff hours, legal liability, loss of proprietary information, decrease in productivity and the potential loss of credibility were impossible to quantify accurately.

Employees who have caused damage have used their knowledge and access to information resources for a range of motives, including greed, revenge for perceived grievances, ego gratification, resolution of personal or professional problems, to protect or advance their careers, to challenge their skill, express anger, impress others, or some combination of these concerns. These case examples[8] serve to illustrate real cases of the “insider” threat:

Example 1: A senior MIS specialist at a large international energy firm regularly created outages at Company sites around the world so that he could spend time abroad while gaining attention for his technical expertise.

Example 2: A 31-year old programmer working for a sensitive missile program project reportedly felt unappreciated for his programming work on a parts-tracking system. He planted a "logic bomb" in the system designed to erase critical data after he resigned, in anticipation of returning to rescue the company as a highly paid and valued consultant.

Example 3: A regional PC manager for a large supermarket chain and two computer programmers were charged in an intricate computer fraud that cost the supermarket in excess of $2 million over two years using their position to both commit and cover up the fraud, emphasizing the vulnerability of organizations to trusted employees.

[4] “Threat Assessment of Malicious Code and Human Threats,” NIST Computer Security Division, 1994
[5] Conference sponsored by the Defense Personnel Security Research Center - Computer Crime: A Peopleware Problem, 1993
[6] “Corporate layoffs Create security Havoc for IT Pros,” www.itmanagement.earthware.com, July 2002.
[7] Computer Security Institute/FBI Computer Crime & Security Survey, 1998
[8] “The Insider Threat to Information Systems”, Eric D. Shaw, Kevin G. Ruby, Jerrold M. Post

Example 4: A major international energy company accidentally discovered a logic bomb in software created by a contracted employee, installed as "job insurance" in anticipation of using it as leverage against his employer in case his criminal record was discovered, illustrating the vulnerability of organizations outsourcing their IT functions.

Example 5: A Chinese computer programmer working as a subcontractor for a large system integrator illegally accessed sensitive Air Force information on combat readiness. He copied and posted on the Internet passwords that allowed users to create, change or delete any file on the network, demonstrating the espionage threat posed by contractors.

Example 6: A senior computer programmer for a Fort Worth securities trading firm, designed a virus to erase portions of the company's mainframe after being reprimanded for storing personal letters on his company computer. After being fired, he used a duplicate set of keys to return to the facility, employing an unauthorized backdoor password to re-enter the system and execute the virus, illustrating the case of ex-employees who return to extract vengeance on former employers.

Example 7: On the programming staff of a Colorado software firm working on advanced distributive computing software, a Chinese national transferred, via the Internet, the firm's entire proprietary source code to another Chinese national working in the Denver area. The software was then transferred to a Chinese company. The U.S. Company was subsequently driven to bankruptcy by foreign competition directly attributed to the loss of it proprietary source code.

As these case summaries from the files of military and corporate security investigators demonstrate, growing reliance on information technology increases dependence on, and vulnerability to, those tasked with the design, maintenance and operation of these systems. These information technology specialists - operators, programmers, networking engineers, and systems administrators - hold positions of unprecedented importance and trust.

These cases also demonstrate several points about the insider threat to the critical infrastructure. First, it is clear that insider problems already exist within the critical infrastructure, including the military, telecommunications, and energy sectors. Second, it appears that both inside and outside of our critical infrastructure, there is a tendency for managers to settle these problems quickly and quietly, avoiding adverse personal and organizational impacts and publicity. We do not really know how widespread the problems are[9]. What is reported appears to be only the tip of the iceberg. Overall, case investigators report that the number of computer-related offenses committed by insiders is rising rapidly each year[10].

This last point gives cause for serious concern. The rising trend of American corporations seeking to reduce operational costs by outsourcing their IT development work offshore at a time of acute global insecurity exposes them to a significant operational and infrastructure risk. It is ironic that the countries with a history of intellectual property theft and which companies trust the least with binary code (India, China, Pakistan and Russia) are the places where software code development is being sent. Of greater concern is that many of these countries are known to harbor terrorist networks. Currently, there is no way for U.S. organizations to ascertain the security risk of the workers used to produce software components for their mission critical applications, nor are there methods in place to guarantee that the integrity of the manufactured source code has not been compromised by any malicious activity.

Finally, it is important to note that the efforts of "outside" groups (including foreign interests) could be aided significantly by the assistance of parties within the organization with access to, and knowledge of, critical information systems. For certain secure, self-contained systems, the insider's access proves indispensable and collaboration is a tremendous force multiplier[11]. The potential damage an insider can now commit has also been increased within the last decade by two related trends in information systems -- consolidation and, for all intents and purposes, the elimination of the need-to-know principle. These changes, designed to improve information sharing, have removed obstacles to hostile detection. The hostile, sophisticated information technology professional now has many more opportunities to enter and damage larger systems. These vulnerabilities led one government information technology specialist, who focuses on system security, to refer to many allegedly secure government databases as "single point of failure systems."

[9] United Nations Commission on Crime and Criminal Justice Survey 1998
[10] Security Awareness Bulletin No. 2-98, published by Department of Defense Security Institute, September 1998.
[11] "Department of Defense: Insider Threat Mitigation", Final Report of the Insider Threat Mitigation Process team, April 24th 2002

3. The Need for Security

With attacks on the computer components of the nation's infrastructure becoming frequent, persistent and increasingly malevolent[12], creating a strong security policy is critically necessary. However, severely restricting both users and attackers is time consuming and costly. Heavy security policies create onerous work conditions for no discernable reasons, causing bad politics, lowering morale and employee efficiency within an organization. A common attitude among users is that if no "secret" work is being performed, why bother implementing security?

Organizations need to determine the price they are willing to pay in order to protect data and software code assets. This cost must be weighed against the costs of losing information assets and disrupting services. There is a price to pay when a half-hearted security plan is put into action. It can result in unexpected disaster. Minimizing risk and choosing security measures that allow for flexibility and growth provides a proper balance for an organization's security strategy.

4. A Brief Introduction to the Software Development Environment

The past half-century has produced an unbelievable amount of software code. With over 200 billion lines of code in the federal, state and local government infrastructure alone[13], much of it is developed, maintained and upgraded using ad-hoc techniques with little control of the process. It is astonishing to learn that the development of vital computer software assets has been in a state of chaos[14], ever since it was first discussed back in 1972[15] and that the same errors continue to be repeated in the development of new systems and programs[16&17].

There is now a huge amount of complex software to maintain, upgrade and replace, with too few people and too little time to do the work correctly. More and more software pours out every day. Alfred Spector, VP at IBM Research admits that software complexity has been a serious problem for over 40 years and that it "cannot go on."[18]

Software is increasingly difficult to understand and preserve, with maintenance costs spiraling out of control. Well-structured, intuitive code assumes the organizational characteristics of spaghetti: tangled, slippery and hard to grasp[19]. Pressed by deadlines and unable to keep track of what is being done, programmers "throw code" out as fast as they can.

This leads to software development and maintenance projects qualified as "chaotic" activities often characterized by the phrase "hack code and fix"[20]. Time and cost constraints means that most of this immense code base, old and new, is under-documented, if it is documented at all. Any actual knowledge that exists - how the code worked, how to fix it, how to improve it - is in the minds of the programmers and designers who originally made it. Software projects often become "monsters" of missed schedules, blown budgets and flawed products[21].

Furthermore, with the rise in the number of contract programmers, the significant influx of foreign workers on H1b work visas since 1996[22] and the projected boom in offshore outsourcing, the software knowledge accumulated in the minds of programming staff "walks out the door" once projects are completed, further undermining an already tenuous understanding of programs and computer systems. Entrenched programs became increasingly difficult to maintain, with control over systems and personnel becoming ever more elusive.

Despite a variety of government standards such as the Orange Book and the Common Criteria as well as various panaceas - high-level languages, formal methods, n-version programming, structured programming, code walk-throughs and more - the software development process remains ad-hoc and out of control. There has certainly been progress, but the discipline is still far from the perfection required of computer science. This chaotic state of affairs is absorbed as standard operating procedure; a cycle almost impossible to break and an accepted part of the culture of the way that things are done. This "culture of acceptance" is of concern in these insecure times and need for increased vigilance.

---

[12] "Feeling Insecure," *Interactive week*, October 2001
[13] The year 2000 software Problem: Quantifying the Costs and assessing the Consequences," Capers Jones
[14] "The Chaos Report," *The Standish Group,* 1999 http://www.standishgroup.com/chaos.html
[15] "The Humble Programmer," Dijkstra, E., *Communications of the ACM*, Vol. 15, No. 10, October 1972.
[16] Web crisis.com Inability to Maintain," *Software Magazine*, September 1999.
[17] " Why Software is So Bad, and What To Do About It, MIT Technology Review, June 2002
[18] Cochran, S., "The Rising Cost of Software Complexity", *Dr. Dobb's Journal*, April 2001, pg. S14.
[19] Boes, B, "Complexity Begets Complexity: Development Information Systems - A New Paradigm for Software Development," 2001, www.esj.com
[20] http://www.martinfowler.com/articles/newMethodology.html, 2001.
[21] "No silver Bullets: Essence and Accidents of Software Engineering," Frank P. Brooks, 1987
[22] 47% of the H1B visa quota went to foreign computer programmers, *U.S. Labor Department,* 1998.

5. Understanding the Presence of Developmental Vulnerabilities

Traditionally, it is assumed that software is developed in a "safe" way and executes functionality as programmed. Security problems were understood to be external attacks upon a properly functioning computer system. Likewise, computer security assumes that software developed by "insiders" is considered trustworthy and "outsiders" with malicious intent perpetrated attacks.

This view of the world is fundamentally flawed. The greatest threat of attacks (direct or in error) comes from insiders and most security problems are caused by buggy software. Insiders with legitimate access to critical business systems accounted for more than 80% of the cyber attacks against companies last year[23]. A report by the National Institute of Standards and Technology[24] also found that buggy commercial software costs users and vendors $60 billion a year[25]. Programmers are notorious for innocently injecting bugs into complex computer systems[26]. Software quality has become so bad, some engineers argue, that the only solution is litigation and legislation[27].

It is common practice for programmers to write their programs, compile, test and place them into operational status without anyone else ever seeing the code. Many organizations have little or no source code control or configuration management practices, making it incredibly easy for a malicious programmer to wreak havoc with total impunity.

Programmers know that their managers don't read code. They refuse any form of accountability and jealously protect their independence. Formal, disciplined software development environments do include peer-group "inspections" – a required practice to reach level 3 of the Capability Maturity Model (CMM) - but these are often superficial in nature and are the first practices to be abandoned when under pressure to deliver as they are costly and very time consuming.

A recent report by the Software Engineering Institute[28] (SEI) indicates that only a small minority of IT organizations inside or outside of the government was at level 3 or above. (See table)

The report shows that between 56.7% and 81.3% of government software applications are developed without programmer activities monitored by "peer-group" inspections (one of the prerequisites to achieve a level-3 rating) and 73.1% within the corporate industry.

The report also implies that, by SEI standards, less than 7% of software applications developed in commercial and government sectors meet with the highest level of certification.

SEI Level	Commercial/ Inhouse	DOD/Federal Contractor	Military/Federal Organizations
1	34.90%	21.20%	52.50%
2	38.20%	35.50%	28.80%
3	18.50%	33.90%	11.30%
4	5.50%	5.20%	6.30%
5	2.90%	2.40%	1.30%

The findings extrapolated from this report are significant. In short, there is inadequate surveillance of programmer activities in the majority of commercial and government software projects. Lack of programmer accountability, a culture of acceptance of the chaotic developmental environment is a source of concern in today's climate of global insecurity.

6. Is the Threat of Digital Sabotage and Intellectual Property Theft Real?

Information technology is emerging as one of the key battlefields in the war on terrorism. The recent attacks against the United States signal a new age that redefines both how an enemy will fight wars and how we must defend against them. As such, we must consider all scenarios for possible attacks on our infrastructure, including combinations of physical and cyber-attacks. The US military is aware of hostile nations and stateless terrorists with entire armies of programmers to probe, attack and exploit not only US military computer systems, but also the entire "critical infrastructure" of the country[29]. As the fight against terrorism intensifies, terrorists will become more sophisticated and harder to identify. Attacks are cheap, almost impossible to prevent and they will learn from their failures, making future attacks more insidious.

[23] "Byte Wars: the impact of September 11 on information technology," Edward Yourdon, March 2002.
[24] "The Economic Impacts of inadequate Infrastructure for Software Testing," May 2002 – NIST
[25] Costs of custom-built government applications are not included in this report.
[26] On average 100 - 150 errors per thousand lines of code, study of 13,000 programs by Watts S. Humphrey, Carnegie Mellon University's SEI
[27] "Why Software is so bad and what's being done to fix it," MSNBC Technology Review, June 2002.
[28] "Process Maturity Profiles of The Software Industry," Software Engineering Institute, August 2000
[29] "Byte Wars. The Impact of September 11 on Information Technology," Edward Yourdon, March 2002.

The need for protecting databases and networks is well known; yet few organizations have taken any steps to protect the source code of programs that embody the business rules, policies, and strategies on which business and government are able to function. Programmers' lack of accountability coupled with the absence of documentation and subsequent lack of operational knowledge, make protecting that source code increasingly necessary.

In the past, organizations have turned a blind eye toward foreign nationals given access to sensitive systems without proper background checks[30]. While the overwhelming majority of programmers are bona fide patriots, the unique and unfamiliar nature of computer sabotage makes it impossible to fully anticipate all of the risks. With little accountability, source code is put into production on the "good faith" of a programmer's character. Current security measures mean that malicious code can be introduced. Security experts agree that the "insider" threat is real[31].

We have seen numerous examples of disgruntled programmers coding "time bombs" into computer systems for which they are responsible or introducing "rogue code" to transfer funds from dormant bank accounts. It is surprisingly easy to accomplish this kind of surreptitious change. At dotcom companies and high tech firms throughout the country, as well as across most of America's Fortune 2000 companies, it is an accepted fact that nobody ever sees the computer code that gets installed in systems[32]. We have also seen how easy it is to steal proprietary intellectual property.

Malicious programmers may wait patiently, inactive for years to gain the prerequisite trust and invisibility to acquire access to sensitive software assets, programming "rogue code" into a critical application that might not execute for years or timed to coincide with scheduled physical attacks. The possible presence of small, well-organized groups of digital terrorists with deadly motives and longer time frames cannot be under-estimated.

Today, we need to focus hard on the possibility of security being compromised or circumvented during the development of critical systems, especially when outsourcing overseas, the existence of which may not be known for months or years after the system has been placed in operational status. It is clear that insider problems already exist within the critical infrastructure, including the military, telecommunications, and energy sectors. We do not really know how widespread the problems are[33] for what is reported appears to be only the tip of the iceberg[34]. As a consequence, when defining a security strategy there needs to be a subtle shift in thinking from simply addressing the operational weaknesses of software systems to a more all-inclusive strategy that embraces software developmental vulnerabilities.

7. Understanding the Impact of Digital Sabotage

American organizations have been working with computers, accumulating vast stores of legacy programs and building up enormous databases of information far longer than most industries. Hundreds of billions of dollars have been spent to acquire computers that can count, tabulate, compute and perform a myriad collection of bureaucratic functions. The combined military, federal, state and local governments, for example, total more than 8,000,000 computer applications, run from almost 2,000 sites, created and maintained by some 366,000 programmers[35].

Clearly, information contained in computer systems is more valuable than the hardware itself. Hardware can be replaced if destroyed; if data is lost, it can be recovered but only at great effort and expense. Protecting hardware is vital, but computer data is far more vulnerable than a "box". It can be copied, altered, erased, and corrupted remotely without the visible aspects of a physical attack. Protecting the integrity of software assets is increasingly important, as the impact of malicious code increases to match.

Familiar, readily preventable coding errors have wrecked a European satellite launch, delayed the opening of the hugely expensive Denver airport for a year, destroyed a NASA Mars mission, killed four marines in a helicopter crash, induced a U.S. Navy ship to destroy a civilian airliner, and shut down ambulance systems in London, leading to as many as 30 deaths. These events were the result of unintended programming errors. Maliciously tampered programs will have a significantly higher magnitude of disruption and financial cost than any physical sabotage, with an amplification of consequences that terrorists would aim for when planning and executing attacks.

[30] "DOD adjusts its plans on hiring foreign workers," Government Computer News, May 2002
[31] "Analysts: Insiders may pose security threat," *Computerworld*, October, 2001
[32] "Byte Wars. The Impact of September 11 on Information Technology," Edward Yourdon, March 2002.
[33] United Nations Commission on Crime and Criminal Justice Survey 1998
[34] Security Awareness Bulletin No. 2-98, published by Department of Defense Security Institute, September 1998.
[35] "The year 2000 software Problem: Quantifying the Costs and assessing the Consequences," Capers Jones

While we associate a terrorist act with a singular, physical event, an act of digital sabotage can be far more insidious. Bringing "down" a mission critical computer system may be damaging in itself, but it still remains a singular event, acting as a focus for emergency computer repairs to take place. Of greater concern must be "rogue" code that is introduced into programs that do not "crash" applications per se but rather create erroneous data that compromises the integrity of the information on which intelligence is gathered and decisions are made – however small.

Information sabotage is as Machiavellian a weapon as digital sabotage. Small, incremental changes to a program can have a considerable impact on the result of data output. To illustrate this in simple terms, altering one digit of program code by changing addition to multiplication in this mathematical formula 3.14 + 3.14 drastically alters the resulting value by a very significant 48% differential. Once implemented into a system, it becomes extremely difficult to find the change amid the billions of lines of code that currently comprise mission critical systems – especially if the "rogue" code has been surreptitiously introduced, disguised as something else ("Trojan Horse"). The task of identifying malicious code is further complicated as erroneous date interacts with systems many times removed from the sabotaged application.

While this task of identifying sabotaged code is hard, the effort of fixing sabotaged code will be a massive and costly job. Roughly $100 billion was spent fixing the widespread Y2K bug, which in computing terms was a relatively simple problem. The threat of digital sabotage to an organization's economic survival presents a very different challenge. John Koskinen, former Head of the Government's Y2K Task Force36, stated after 9/11 that the terrorist threat to IT is undefined, the response is difficult and there is no timeframe - unlike the Y2K phenomenon. It is easier – and cheaper – to prevent malicious coding activities at the source by protecting an organization's software portfolio that it is to repair it.

8. What Is Needed to Increase Surveillance Without Impacting Productivity?

Sophisticated software development techniques and improved global bandwidth and communications are making it possible for companies to have various pieces of IT development or integration projects conducted overseas, with the final assembly completed in the U.S. The critical business issues to mitigate the threat posed by malicious "insiders" require risk management, team collaboration and a verification and audit process[37] that increases programmer accountability.

The lure of inexpensive, abundant, and experienced software development resources is drawing a significant number of companies into outsourcing arrangements in which much of the work occurs either overseas or on the perennial use of contract professionals. With onsite and offsite development, the critical business issue is risk management. Enterprises require the ability to respond quickly to changing business needs, network security issues and diverse cultural differences. As a result, outsourcing should be viewed less as a supplier-customer relationship and more as a collaboration.

The goal of collaboration - people working together at peak efficiency - is to gain optimum workflow and risk management processes within and beyond the enterprise. Achieving new levels of collaboration allows companies to successfully differentiate in today's increasingly competitive marketplace. Organizations maximize cross enterprise interactions by significantly improving the capture and reuse of intellectual assets and know-how. They also gain global visibility throughout the entire extended enterprise - directly from their familiar desktop environments.

Collaborative software development provides visibility into the development process. It improves quality, maximizes resource utilization in an environment of continuous process improvement and optimized project management. It facilitates communication with secured Instant Messaging, the sharing of documentation, co-development on source-code among the virtual community members, bringing them together quickly to get a job done. The result is a dramatically reduced time-to-market, lower operating costs, retained critical business knowledge and intellectual capital that used to walk out the door with departing IT contractors. Collaboration is the single most important advance in e-business today.

However, security is a matter of degree. Good security is based on the concept of least privilege, where people are given only the privileges needed to perform their assigned tasks - and no more. Collaboration enables anyone to freely exchange information with anyone else. Implementing a system that incorporates extremely stringent security measures with higher security barriers carries a higher cost, not only to potential intruders but also for authorized users. Extending an existing development infrastructure into a secure collaborative environment that resides beyond a company's firewall creates a quarantined, development "clean room" that creates a first line of defense to protect companies from the threat of malicious programming activities without onerous security policies that erode productivity.

[36] "Business Eyes Y2K Effort As Model For Terrorism Fight," Computerworld, October 2001
[37] "Department of Defense: Insider Threat Mitigation", Final Report of the Insider Threat Mitigation Process team, April 24th 2002

Nonetheless, using a physical metaphor to add context to the virtual world, to simply create a secure virtual building where users can meet to collaborate and communicate does not address the need to ascertain the security risk of the workers used to produce software components, nor does it give any guarantees that the integrity of the manufactured source code has not been compromised by any malicious activity. Policing the extended enterprise becomes the priority.

To increase surveillance in the developmental environment, the collaboration environment has to be extended with security measures that include the protection of the integrity of the software code that is being produced. Yet, heavy security policies increase workflow in an environment that is unable to accept any additional pressures. Therein lies a first challenge for authorities charged with managing software projects and security:

How to successfully implement surveillance technologies without increasing workflow and reducing productivity?

For a surveillance technique to be successful in the current developmental environment it must be non-intrusive; it must exist within the framework that is already established, supplementing and extending that framework rather than subverting it. A successful technique must deal with an environment that cannot be disturbed especially in light of the chaotic state of affairs and the "current culture of acceptance" that is prevalent.

Using the speed and power of the computer to monitor the software itself can be exploited to increase the visibility of the software that is being produced. Given the immense code base in existence and in constant production, to yield the highest return on investment, it is best to spend available effort on the aspects of the process that change, rather than on the process as a whole. Tracking all programmer interactions with code assets increases visibility of the software produced and creates a method to verify and audit programmer accountability at the granular level - the building block for surveillance of all coding activities. The creation, at the source, of available, reliable, non-refutable records of programmer actions mitigates the insider threat by aggressively reinforcing programmer accountability.

However, if the tracking is done surreptitiously, the programmers will resist, as it is perceived as an implicit lack of trust by an organization towards an individual[38]. Herein lies a second challenge for a surveillance technique to be successful:

How to successfully implement tracking technologies to a programmer community that fiercely defends its independence?

In light of the current state of affairs, the solution is surprisingly obvious. For a tracking technique to be successful, it must focus on helping programmers to become better programmers. Unable to keep track of what they are doing, enabling programmers to track their own work as part of their day-to-day operations, offered as a memory aide and personalized knowledge management resource, nullifies concerns about being tracked. If such tools are perceived as personal benefits that empower them as professionals, the benefits are embraces at the expense of concerns over increased surveillance.

The majority of programmers take pride in the effort they put into their work and the quality of the code that they produce. A technique that empowers them to objectively demonstrate to peers and superiors increased quality, with a reduction of effort to achieve it, will be well received. The increased visibility from tracking engenders recognition for their efforts and skills that inevitably leads to higher financial reward– a powerful motivator for acceptance. By default, those who resist tracking as a technique have something to hide; whether it is a malicious activity, poor programming skills or low productivity - they merit increased scrutiny.

The aggregation of all programmer activities into a centralized database for analysis creates a powerful developmental intelligence capability not previously available. Wrapping exception based surveillance around tracking and a productivity tool acts as a very potent deterrent against malicious coding activities; the knowledge of a higher level of vigilance deters the malevolent as well as the disgruntled programmer from producing "rogue" code.

Organizations require real-time vigilance to deter, recognize, and respond to malicious coding activities by insiders. The publicized presence of capabilities to detect malicious activity and consequences imposed by organizations upon those who misuse, abuse or perpetrate malicious activity provides the greatest deterrent to malicious insider activity[39].

[38] Debates on privacy issues from tracking user movements on the Internet engender fierce resistance, notwithstanding that it produces valuable intelligence successes in the fight against terrorism. Programmers view the programs they produce as their individual "property" until the work is finished and given to the employer. This attitude may seem misguided, but it is pervasive – a part of the developmental culture since the early 1960s.
[39] Department of Defense: Insider Threat Mitigation", Final Report of the Insider Threat Mitigation Process team, April 24th 2002

9. Introducing Mintaka Technology Group

Mintaka's mission is to protect organizations against the threat of malicious "insiders" with tools that increase awareness, prevention and deterrence. Mintaka's provides a secure collaboration environment with a proactive security posture that increases real-time vigilance of programmer activities, mitigating the threat of application security vulnerabilities and reducing an organizations' operational risk.

Mintaka's Secure Software Environment™ Portal

The SSE™ Portal is a secure web-based environment that enables collaboration between onsite-offsite programmers with unparalleled security. It integrates into existing development infrastructures, extending a broad-based community to the web to enable effective planning of project scope and schedules. It creates a first line of defense against threats from external hackers and internal malicious programmers to allow organizations to leverage cost-effective outsourcing options while minimizing operational risk. It brings simplicity and high security to online collaboration.

The SSE™ Portal's collaborative techniques create higher visibility of data used to make project control decisions. It helps onsite managers to maintain control ensuring an increased breadth of available critical project and historical data while optimizing operations and communications in real time. The participative nature of the collaborative development philosophy enables a corporate-wide buy-in of common practices. It is an integral part of a virtual team's collaboration toolset, improving the efficiency of complex project execution and building trust through better onsite-offsite partnership. This results in immediate cost savings through reduced development time, with higher levels of assurance and ultimately, the knowledge that a company's software infrastructure is secure from real world vulnerabilities and attacks.

Mintaka's Secure Software Intelligence™ Portals and Portlets

The SSI™ Portals serve as a powerful deterrent against "insider" threats to your source code with a proactive security posture that increases real time vigilance of programmer activities. They provide superior business intelligence capabilities, protecting and supporting development exercises, allowing for the prerequisite flexibility for a balanced security strategy. The non-intrusive tracking, gathering, and centralizing of each programmer's activities into a data warehouse are the building blocks for the SSI™ Portals.

The SSI™ Surveillance Portal protects your source code with a proactive security posture. It is a proactive, exception-based monitoring system that alerts to suspect coding events as they occur and before they become problems. It provides organizations with superior business intelligence of EVERY programmer's interaction with its code assets offering query, reporting and analysis capabilities, statistical functionality, sophisticated transaction-level analyses, and reporting. The tracking of programmer activities aggregated into a centralized database creates a powerful intelligence capability that increases programmer accountability and visibility of development projects. The knowledge of a higher level of vigilance across the extended development environment acts as a potent deterrent against malicious coding activities to deter the malevolent as well as the disgruntled programmer from producing or introducing "rogue" code.

The SSI™ Productivity Portal provides superior analysis of software processes. It offers a valuable measurement tool for best practices, collecting data non-intrusively to help organizations understand current capabilities, develop achievable plans, detect trends, anticipate potential problems and analyze improvements in productivity to ensure the business objectives are achieved in the planned time and budget. It quantifies and maximizes the output of your workforce by tracking programmer performance, increases accountability, analyzes productivity in real time and allows managers to optimize and calibrate development efforts as the work is being done.

The SSI™ Programmer Portlets offers developers an unparalleled knowledge management and support resource. As part of their work domain, programmers use the customized Portlets as a valuable memory aide in their day-to-day operations. The knowledge captured from tracking their own work at the granular level coupled with the built-in query tools allows them to easily reference previous work and review summaries of their work activities while executing programming tasks with higher levels of quality.

Secure, protect, and police the extended development environment in real-time