ISSUES AND REQUIREMENTS FOR CYBERSECURITY IN NETWORK CENTRIC WARFARE

Martin R. Stytz, Ph.D.
Air Force Research Laboratory
Wright-Patterson AFB, OH
(937) 426-2959
mstytz@att.net

Sheila B. Banks, Ph.D.
Calculated Insight
Orlando, Fl 32733
(407) 353-0566
sbanks@calculated-insight.com

ABSTRACT

The transition to network centric warfare brings with it great promise for the effectiveness of future military operations. This promise arises from the capability for network centric warfare to empower individuals at all levels with vast amounts of relevant information and thereby lift the "fog of war." By achieving the promise, commanders will be able to effectively and efficiently employ their resources to achieve objectives; in addition, individuals can exploit information in real-time to increase their effectiveness in mission accomplishment and to capitalize upon transient opportunities in the battlespace. However, a central, but generally unspoken, tenet of network centric warfare is that the information received is actionable; i.e., that the information is timely and correct. However, the increasing sophistication of computer and network attack tools and technologies coupled with the increasing technical sophistication of potential adversaries calls this central tenet into question and raises the question of how to secure the network and software against the threat of attack and subversion. Clearly, network and application security, or cybersecurity, is a broad topic, but is of pressing importance if network centric warfare is to fulfill its potential and become a key component of the future battlespace. There are a variety of attacks upon network and software that must be addressed in order to achieve cybersecurity; some attacks are as simple as denial of service attacks and some as complex as attacks that exploit cyber vulnerabilities in order to alter key networks and software and thereby subvert the information without the user being aware that the subversion has occurred. Our objective is to examine the need for network and software security in light of the network centric warfare paradigm. In view of the need and threat, we present a new strategy for cyber defense, one that builds upon but enhances the proven concept of defense in depth. This new strategy exploits the defensive advantages offered in cyberspace while also minimizing the opportunities for the attacker as well as making an attack more difficult.

This paper discusses the following topics. The first section contains the motivation for our research as well as a discussion of the challenges that must be addressed to provide cybersecurity in the cyber battlespace in support of network centric warfare. Next, the paper presets a discussion of background material necessary to understand this research area. The focus of the paper is in the third section, which contains a discussion of the scope of the threat and of cyber security issues and requirements in the cyber battlespace. The concluding section contains a brief summary and suggestions for further research.

1. INTRODUCTION

"One of the benefits of reverse engineering is that you can gain an understanding of a program in terms of its

binary code. As you become acclimated to the process and gain some experience, you begin to notice and recognize certain data structures and subroutines simply by how they look in a hex[adecimal] editor. This may sound weird, but you might be scrolling through a binary file at a later date and find yourself saying 'oh, there's a jump table' or 'huh, this is probably the prolog to a subroutine.' This is a natural ability that evolves as you learn to understand machine code…The feeling of ***power*** (emphasis added) associated with this skill is very rewarding.[*]" These words, written by two of the "good guys" in the ongoing war in cyberspace, should serve as a warning to us all concerning the future of network centric warfare. People have, to date, been willing to invest untold hours of their own time just to break into computer systems simply for the feeling of power that accompanies the successful attack. What types of attacks will be faced when this powerful psychological incentive for the attacker is coupled with the resources of a well-funded and determined adversary, be it a terrorist group or a nation-state? At this point, we can only speculate, but the skill and effectiveness of the attack must surely increase and, in anticipation of this state, our defenses must start to be improved if we are to be prepared once the cyberspace war escalates in scope and power.

The threat posed by the incentivized cyber attacker of the future is increased because the transition to network centric warfare promises to increase the effectiveness of future military operations. Allow us to elaborate. The promise of increased military effectiveness arises from the capability for network centric warfare to increase the effective combat power of military organizations. The increase occurs as a result of the provision of timely and relevant information organized and presented to facilitate situation awareness, decision-making, and response to enemy activity, friendly activity, and other circumstances. Network centric warfare can substantially reduce the fog and friction of war and thereby reduce the most serious impediments to optimal, effective action in the battlespace. As a result of the empowerment that results from improved, efficient information flows, commanders at all levels will be able to effectively and efficiently employ and coordinate their resources and actions to achieve objectives and capitalize upon transient opportunities in the battlespace to further increase their effectiveness and combat power.

However, a central, but generally unspoken, condition for successful network centric warfare is that the information received is actionable; i.e., that the information is both timely and correct. However, the increasing sophistication of computer and network attack technologies and tools coupled with the increasing technical sophistication of potential adversaries calls this implicit central tenet into question and makes the question of how to secure the network and software against the threat of attack and subversion all the more urgent and important. Hence, our conclusion that the threat posed by cyber attack will increase as the transition to network centric warfare procedes. Clearly, software and network protection is a broad topic [2-14]; nevertheless, cyber seurity must be achieved across the entire cyberbattlespace if the potential of network centric warfare is to be fulfilled in the future battlespace. There are a variety of attacks upon the network and software that must be addressed; some as simple as denial of

[*] From Hoglund and McGraw, *Exploiting Software – How to Break Code*, Addison-Wesley, 2004, page 395.

service attacks and some as complex as attacks that exploit network vulnerabilities in order to alter key software or subvert software into presenting incorrect information without the user being aware that the subversion has occurred. In this paper, we will examine the need for network and software security, or cyber security, in light of the network centric warfare paradigm shift. To support and enable this shift, the discipline and practice of network centric warfare cyber security must be fostered, in this paper we hope to initiate the process of defining the bounds and objectives for this emerging discipline.

Our vision for network centric warfare cyber security can briefly be described as calling for a seamless web of protection technologies for all levels and all portions of the network and software. The protection capabilities (and needs) range from data to network to software with all components being imbued with inherent capabilities to verify their own correct and secure operation as well as the correct and secure operation of the other interacting components of the cyber battlespace. This vision calls for the recognition that the division between software and network security is only a pedagogical difference, and one that should have little impact upon achieving a secure network-centric environment. Upon examination, it is clear that a sound axiom is that all network attacks are, except for the actual transmission of packets across a physical medium outside of a computer, attacks upon software. Conversely, all but the most trivial and elementary software attacks are enabled and conducted via a network connection. In brief, all network attacks are software attacks and almost all software attacks rely upon a network connection. This axiom gains even more validity within a network centric warfare environment.

The remainder of this paper is organized as follows. Section Two contains a brief discussion of background material. In Section Three, we discuss issues and requirements for cyber security in the cyber battlespace. Section Four contains a brief summary and suggestions for further research in this field.

2. BACKGROUND

Before addressing cyber security needs for the network centric warfare environment, we will define important terms and discuss a few of the important tools for cyber defense. Due to space limitations this background is, of necessity, limited and we urge the eader to turn to the references for further information and detailed descriptions of the material that we introduce here.

Cyber secuirity has been mostly limited to network and operating system security activities. Traditionally, information assurance and the security of a computation and its data have been provided by the network defensive systems and the authentication mechanisms in the host operating system[1-14]. Despite intense and ongoing efforts to strengthen these two types of cyber defensive systems, they continue to fail to assure the security of software and data on the host computer. As a result, users continue to place their application software and data at risk whenever they use a computer, especially one that is connected to a network. Recently, the concept of information assurance has broadened from the traditional dyad of network and operating system defensive systems to a triad; a triad that includes defensive technologies embedded in the application software. The technological

components of application software defense, also called *software protection*, are a mix of techniques whose objective is to deny the pirate or intruder the capability to misuse, reverse engineer, tamper with, or steal application software or data. Software protection is the last ring of defense for application software and data; with the first two defensive rings being the protection technologies residing in the network resources and the other being the protection technologies residing in the operating system.

We now define a few cyber security terms. A secure application is one that does what it is supposed to do/designed to do and no more. A bug is a software problem and exists only in code. A flaw is a problem introduced into an application at the design level that propagates down to the implementation. A software vulnerability[†] (or vulnerability) is a bug or flaw that can be exploited by a hacker. Risk is the probability that a bug or flaw will be exploited coupled with the severity of damage that an attacker can inflict through the bug or flaw. An injection vector describes the format of an attack[‡]. The payload is the portion of the injection vector that accomplishes the purposes of the attacker[§]. A Trojan, or Trojan horse, is a computer program containing a hidden nefarious function. A virus is a program that copies itself (and its malicious capabilities) to other applications on a computer host. A worm is a computer program that invades computers on a network, and is usually embedded within a virus. Malware is software and is comprised of Trojans, viruses, and worms. A rootkit is a program that allows access to and manipulation of the functionality of the target computer. Rootkits allow an attacker to take control of the system kernel, write EEPROM memory for the computer (motherboard basic input/output system (BIOS)) and peripherals, modify jump and call back tables, and patch around application security software[**]. Call hooking, also called trampolining, is the name for an attack technique that alters the address jumped **to** when a function or subroutine is called[††]. Regression is the software engineering term employed to signify that the number of application bugs has increased, or regressed, as a result of fixing a bug. Regression is one of the chief and most important side effects of inserting cyber security protection into an application after it is completed. Additional standard terminology can be found in RFC 2828, which can be accessed at http://www.ietf.org.

At the developer's level, there are a number of techniques and practices that have been developed that provide a degree

[†] Or, more formally, a security vulnerability is a flaw in software that makes it infeasible, even when using the software properly and with security in mind, to prevent an attacker from usurping privileges, regulating its operation, compromising data, or gaining ungranted trust.

[‡] An injection vector can be more extensively defined as either 1) a structural anomaly or weakness that allows executable code to be transferred from one computing domain to another or 2) a data structure that contains and transfers code from one computing domain to another. We employ both meanings in this report.

[§] While it is possible to craft multi-platform payloads, size restrictions limit their nefarious scope of activity to doing something minor, such as throwing an interrupt that shutsdown the machine.

[**] Not all rootkits are nefarious, a rootkit can also be used by a computer user to monitor the kernel and activity in the machine from within the kernel in a nearly undetectable manner in order to try to detect if the computer has been penetrated or is under attack. It can be a powerful defensive tool.

[††] In order to remain undetected, normally when a function is hooked the behavior of the original function is duplicated in the nefarious function or else the original, intended function is called when the nefarious function has completed its work. During a hook, interrupts are also disabled in order to prevent a collision with another process that is not being hooked.

of cyber security. As noted by Howard[7], these principles should be employed during software development and can improve the security of the application. In brief, these principles are the following: 1) use a security process, 2) define the cyber security goals, 3) treat cyber security as a critical feature, 4) use least priviledge throughout, 5) employ defense in depth, 6) failsafe to a secure mode, 7) use secure defaults, and 8) do not depend upon achieving cyber security through obscurity of information, techniques, or source code. Unfortunately, these guidelines are often ignored and as a result cyber security remains an ever-increasing problem.

An important adjunct to these principles is security threat modeling. The Unified Modeling Language (UML)[15-18] is the means of choice for portraying and modeling cyber threats in a manner that is clear and standard as well as extensible. An accurate model of the threats is an essential first step in determining the defenses that are needed.

Because of their newness, we will briefly describe the main software protection techniques and refer the reader who is unfamiliar with network and operating system protection techniques to the references for further information in these two cyber security sub-fields. There are three popular techniques for application security that address the protection of high-value assets in a networked, threat-rich environment. These three forms of software protection are the following: 1) obfuscation, 2) watermaring, and 3) application performance degradation. These software protection techniques are used to perform three main functions: 1) detection of attempts to pirate, misuse, or tamper with software; 2) protection of software against attempts to pirate, misuse, or tamper with it; and 3) enable self-modification of the software so that its functionality degrades in an undetectable manner. These three defensive technologies can be applied in computing platforms ranging from single processors to small computer clusters to traditional supercomputers to even wide-area distributed computing. We will briefly describe these three techniques in the following paragraphs.

To prevent or hinder reverse engineering of critical portions of code, obfuscation[26] of program logic and/or data at the source or object code level is used. Software obfuscation typically employs counter-intuitive programming logic, such as branch conditions that have no meaning in the actual implemented algorithm. The use of operations that were not present in the high-level source software, such as architecture-specific assembly instructions is another popular obfuscation technique. The goal of software obfuscation is to mislead the reverse engineer and to prevent the determination of the intended effect and operation of the application being attacked.

To obfuscate a piece of software requires subjecting it to a series of semantics-preserving transformations. Semantics preserving transformations are similar to the optimizing transformations used in compilers; however, instead of making the executable *smaller* or *faster* as in the compiler case, obfuscation transformations make the application more difficult to understand and reverse engineer. For example, within a program loop, an obfuscation transform would do the following: 1) choose a part of the loop to obfuscate; 2) choose an obfuscation algorithm; 3) apply the algorithm to the software in the loop, and 4) decide whether to continue the process by determining if the obfuscation has achieved the desired level of effectiveness. The obfuscation transformation should

also be undetectable; that is, the transform should not be detectable upon examination or especially when compared to other versions of the same software.

The main benefit of obfuscation techniques is that they can reduce the likelihood that an attacker can use automated tools to successfully attack software. Even for experienced reverse engineers, obfuscation can delay understanding the inserted protection, and thus can delay bypassing the obfuscation and so protect the software from attack. The main drawback of code obfuscation techniques is that they increase the development and maintenance costs associated with the software.

To effectively obfuscate a program, several difficulties must be overcome. First, in order to choose the right part of the application to obfuscate, the developer must know which parts of the software are security sensitive (and hence need a high degree of obfuscation) and which are performance sensitive (and hence may only be able to tolerate a lower degree of obfuscation). A second challenge is that, in order to choose an obfuscation algorithm, the developer needs to know the cyber security impact that the algorithm will have on a typical piece of code. In general, these metrics are not available and so performance data must be obtained on a case-by-case basis. Acquiring the required performance and security data is costly and time consuming, which tends to inhibit the use of obfuscation protection techniques. A third challenge is that, in order to decide whether to continue the obfuscation process, the developer must determine the level of security that has been achieved and the performance penalty that has been incurred. Once again, there are no metrics to employ, which increases software development time and tends to inhibit the use of obfuscation.

The software watermarking operation[19-24] can be described as follows: The task is to embed a structure "W" (the watermark) into a program "P" such that 1) W can be reliably located and extracted from P; 2) W is large; 3) embedding W into P does not adversely affect the performance of P (the embedding is computationally *cheap*); 4) embedding W into P does not change any statistical properties of P (the embedding is *stealthy*); and 5) W has a mathematical property that demonstrates that its presence in P is the result of deliberate actions. There are two types of software watermarks: *static* and *dynamic*. Static watermarks are permanently embedded in the application executable; whereas, dynamic watermarks are constructed at runtime and are stored in the dynamic state of the program. Static watermarking techniques are easier to develop and control and permit better estimates of their impact on performance than dynamic watermarks. Dynamic techniques are more resilient to attack and detection but their impact upon performance is difficult to predict.

Application performance degradation is the newest of the three techniques for software protection. The strategy for this type of protection is to leave the pirate with software that is useless while also insuring that the pirate cannot detect that the software has modified itself. Performance degradation depends upon the existence of one or more markers in an application that permit the application to determine if it is under attack. The markers are generally a combination of watermarks and performance metrics embedded in the software and internal test data and software that use this test data. Stealthy authentication between

components of an application can be used as well as authentication with an external device to allow an application to determine if it is under attack or has been subverted. If an attack or subversion is detected, the protected software responds by degrading its performance in such a manner that the degradation response, whether it is the application's precision, speed, and/or memory, is difficult to detect and gradual as well as being irreversible.

There are other protection technologies in addition to these main three; however, they are not as commonly used. One cyber security technique is the use of hardware keys. In the hardware key approach, an authentication or decryption key is stored in an external hardware device such as a smartcard. In general, these systems operate by allowing the software application to exchange cryptographically protected authentication messages with the external device one or more times during the course of execution of the protected program. This approach has some difficulties and weaknesses. Its use is easily detected and the security transmissions are subject to interception and playback to spoof the software, but hardware protection does complement software based protection techniques.

Because of the inherent weaknesses of local authentication/security schemes, some approaches perform verification on a remote server. The weakness of network-based protection is that the developer must take the responsibility of insuring that the program can verify that the server performing the authentication is the correct server. When using this protection approach, it is important to have the server provide critical and unpredictable data to the program so that the program can verify that it is using the correct server.

Another approach to protecting software and networks is through the use of virtual computing machines within the computer. This approache to securityis one of the ones used by the Java programming language. The virtual machine approach to cyber protection requires the identification of the crucial portions of the application and then encoding the data and instructions into a custom bytecode format. Virtual machines can be an effective protection technique, but this approach can be costly in terms of the performance of the program and in an increase in development time.

One key to improved cyber security is a better understanding of the threat and of the vectors used by the attacker to circumvent cyber defenses. One approach to achieving an improved understanding is through the use of the Unified Modeling Language. The Unified Modeling Language (UML) is a standardized graphical-based language that can be used to develop and compose blueprints (architecture specifications) of software systems[15-18]. The UML documents the conceptual and physical representations of a system and permits modeling and visualization of a system and potential attacks upon it and the vectors for those attacks from a variety of viewpoints by using diagrams. UML provides a complete language for capturing the knowledge about a subject and for expressing that knowledge. UML contains a large and useful set of predefined modeling and documentation constructs and supports custom representations of information through its inherent mechanisms for extensibility. The UML also provides constructs for specifying and documenting the building blocks and components of a system. By using its defined types of diagrams, UML provides the means for viewing and analyzing an architecture from five points of view. These five points of view are the *design*

view, the *use case* view, the *process* view, the *implementation* view, and the *deployment* view. However, while the capability for capturing knowledge and insight about attack vectors is contained within the UML, this capability has to date been rarely exploited for cyber defense.

In the next section, we discuss the threat and needs of the cybersecurity field in light of the current dearth and poor performance of defensive techniques and the plethora of effective attacker technologies.

3. ISSUES AND REQUIREMENTS

In this section we cover address two issues related to cyber security in the network centric warfare environment. We open this section with a discussion of the scope of the cyber threat that must be addressed. The section concludes with a discussion of the cyber security objectives and a new strategy for cyber security for the network centric warfare environment.

3.1 The Cyber Threat

In this subsection, we briefly examine the different forms of cyber attack to illustrate the scope of the threat. After reviewing the literature in this area[1-14], there is clearly no commonly accepted classification of attacks or the underlying strategies that are used to execute the attacks. So, in order to assist in understanding the scope of the threat and the techniques used to accomplish an attack, we developed our own classification of the types of cyber attacks and the strategies that they use. To insure that we captured all of the types of attacks and strategies, we used a successive refinement approach to distinguish and classify attacks and strategies. The classification was validated by a regular and thorough re-review of the literature to insure that the attacks and strategies that we identified captured all of the attacks and approaches taken to accomplish a cyber exploit.

As a by-product of our analysis of the cyber security attack literature, we identified three basic attack strategies, and we will open our discussion of the scope of the threat by presenting these strategies. These basic strategies illustrate that the scope and basic approaches for cyber attack have not changed over the years and are relatively straightforward. While the strategies have not changed, there has been a change in the form of an increase in the sophistication and expertise employed to execute the strategy when performing an attack. The three basic attack strategies are the following: 1) to inject faults via the application's runtime environment, 2) to inject faults through the application's source code, or 3) to inject errors to induce a fault. These strategies can be executed using a variety of techniques and tactics, but some of the techniques required to execute them are quite complex. While there are three basic strategies, the literature indicates that these strategies can be further refined and specialized. This refinement is useful for illuminating the scope of the cyber threat, as there are many types of cyber attack, as Table 1 shows. In Table 1, we summarize the different types of attacks and provide a brief description of the strategy underlying each attack (exploit). There are a number of strategies identified in the table but they are all variants of the three strategies identified above.

Table 1 illustrates that there are a wide variety of attacks and strategies used in the attack. In addition to the evident broad scope of attacks, the literature, and the news, indicates that in spite of the effort put forth in recent years we have not been able to redress the weaknesses in

cyber security, thwart any of the basic forms of attack, or develop a strategy for defense that counters the strategies employed by attackers. These problems must be resolved in order for network centric warfare to realize its potential. As a first step toward addressing these problems, we identified a set of objectives for cyber defense and devised a new strategy for cyber defense. We discuss these developments in the next sub-section.

ATTACK NAME	ATTACK STRATEGY
Block Access to Libraries	Attack via environment.
Redirect Access to Libraries	Attack via environment. (Works by altering execution flow into attacker's code instead of allowing the library's function to execute.)
Manipulate application registry values	Attack via environment.
Force the application to use corrupt files or databases	Attack via environment.
Manipulate and replace files that the application creates, reads, writes, or executes	Attack via environment.
Force the application to operate in low memory, disk-space, and network-availability conditions	Attack via environment.
Overflow input buffers	Attack through the user interface or other input vector.
Attack through application switches and options	Attack through the user interface or other input vector.
Use escape characters, different character sets, and commands to get malformed input	Attack through the user interface or other input vector.
Try common default and test names and passwords	Attack through design flaws. Design flaws can leave user accounts and/or passwords that were active and useful during development available after the application is shipped.
Look for and test unprotected application APIs	Attack through design flaws. The design flaw is open test harness APIs.
Connect to all ports	Attack through design flaws. Look for design flaws that have left network ports open in the application that were used for development and testing.
Fake the data source	Attack through design flaws. Look for design flaws that result in misplaced in trust in data sources.
Create loop conditions in an application that reads script, code or other user supplied macros or logic	Attack through design flaws. Look for design flaws that allow flawed loops in scripts used by the application to prevent the application from executing or result in deadlock.
Look for and use alternative execution routes through an application to accomplish its task(s)	Attack through design flaws. Look for design flaws that permit a privileged command to execute in spite of lacking the privilege.
Force the application to reset its values	Attack through default values. Force the application to use default values whenever it asks for an input value from any input source.
Get between time of check of a value and time of use of a value	Interposition attack.
Create fake files with the same name as protected files	Attack through privilege. Exploit the special privileges given to files with certain names or in certain locations to attack an application.
Force all error messages	Attack through privilege. Exploit improper or incorrect error handling to crash or hijack an application.

ATTACK NAME	ATTACK STRATEGY
Look for temporary files for an application and examine their contents for sensitive or exploitable information	Attack through files. Attack the implementation by examining its temporary files to determine if they contain sensitive data or if one of them can be rewritten.
Force invalid outputs to be generated	Attack through files. Some applications process inputs based upon context, at times this reliance on context causes erroneous output that can be exploited.
Attack through shared data	Attack through files. Generate data values in one component of an application that exceed the allowed values in another component that uses the data.

Table 1: Cyber Attacks and Their Strategies

3.2 Objectives and a New Strategy

To lay the foundation for the discussion in this subsection, we present what we consider to be the chief objectives for cybersecurity. The objectives are the following: 1) preserve the integrity/functionality of the network and system; 2) control the use of the system; 3) prevent extraction of software subsets; 4) protect system data; 5) protect network access, prevent unauthorized access; 6) insure correct and accurate execution (unchanged processes that might still produce correct answers or incorrect answers); and 7) insure that computations are correct and accurate

Unfortunately, in spite of numerous efforts undertaken to develop processes and technologies to enhance cyber security, we are far from being able to reliably achieve the goals listed above. Furthermore, no silver bullet solution to the problem of cyber security has been found and none appear to be on the horizon. As a result, we would argue that researchers and developers should re-examine the application of the idea of defense in depth and determine how it can provide better security for an application than a single defense or defensive layer. This straightforward idea appeals to our common sense and is also supported by hundreds of years of security experience in a variety of situations; ranging from national defense to military fort construction. However, most, if not all, of these systems were and are serial (or sequential) in nature. In other words, breaking one system opened the way to the next system, but until the first system was breached the second layer did not come into play. In the physical world, this approach to defense in depth is logical and effective. The nature of the physical world makes a sequential defensive system effective since the attacker cannot begin to devise an attack upon the inner defenses until the outer defenses are breached. However, the cyberworld is different and that reconsideration of how defense in depth should be applied is warranted.

The cyberworld is different from the real world in almost all regards. The concept of interior and exterior as applied in the physical world has no counterpart in the cyberworld; when cyber defenses are arrayed independently, the result is to make them arguably weak in the whole. Consider, almost any cyber defensive measure can be attacked first (so it would be the outer layer of the defense) and any other can be attacked last (making it the inner layer). Additionally, different attack profiles can be used to defeat independent defenses in different sequences, so the concept of an outer ring of defense and inner ring of defense, as traditionally

applied when discussing defense in depth, has no real meaning when the same defensive technique can be attacked first or last at the whim of the attacker. Hence, in the cyberworld there is no advantage to be gained by using defenses that are arrayed independently and sequentially. Each defense can be defeated independently, unless one or more defenses are used as a trip-wire to signal other defenses that an attack is occurring (which is of marginal utility). However, it is known that defense in depth is a good defensive strategy, one that has stood the test of time and proven itself in a variety of circumstances. So, how can the proven defensive strategy of defense in depth be applied in an environment where the concept of physical distance has no meaning?

Consider that the basic strategic motivation for defense in depth is that no single defense should be relied upon to protect important items and instead multiple defenses should be employed. As a result, an attacker cannot defeat one defense and thereby gain access to the items being protected. Instead, all defenses must be defeated and, when properly arrayed, the attacker cannot gain insight into one defense while attacking another and, just as importantly, there is a degree of mutual support but not interdependence between defensive defenses. Therefore, mutual support combined with independence should be the objectives and guiding lights for achieving an effective defense in depth in the cyberworld. As regards the cyberworld, our problem is to interweave all of the defenses into one layer that would consist of mutually reinforcing but independent cyber defensive measures designed to keep a malicious event from occurring. Given the difficulty that the human mind has in maintaining an accurate mental conception of an environment or set of circumstances when seven or more items are in play simultaneously, the more defensive challenges that must be mastered simultaneously by an attacker the stronger the defense should be and more the difficult it is to compromise.

However, the strategy for cyber defense outlined above is not easy to achieve. While we have a sound objective for cyber defense we currently would have a difficult set of challenges to overcome to achieve the goal. Given the state of protection technology; it is difficult to identify which technologies are needed and how to structure cyber defenses so that multiple challenges always confront an attacker. As a first step toward achieving this type of cyber defense in depth, we must first develop an understanding of what it means to have mutually reinforcing but independent cyber defenses. If we have mutually reinforcing but independent cyber defenses they must make it very difficult for an attacker to understand the protected item(s), whether the examination is conducted locally or remotely. The interwoven cyber defenses should make it extremely difficult to perform any form of control-flow analysis, data-flow analysis, network tracing, or program slicing. In addition, the executable must be so complex and variable that no technique could be applied that would allow an attacker to understand what the program does, how it does it, or the data that it uses.

Several other capabilities are needed by the cyber defense. Firstly, the cyber defense should also have the capability to verify that the software it protects was invoked by an authorized entity and that it still protects its intended target. Secondly, the cyber defenses should be capable of determining if they have been changed. Thirdly, and most currently used, the

cyber defense should be capable of determining if it is being examined by a debugger, has been placed within a virtual machine, or has been hijacked to another computer system.

Unfortunately, at this time we generally do not know how to achieve these characteristics for cyber defenses, how to compile source code with these characteristics into an executable binary with the same characteristics, or how to measure the degree to which these characteristics are attained. Indeed, given the current state of technology, if we could develop cyber defenses with the desired characteristics they would be impossible to maintain or update. Additionally, we believe that cyber defense in depth should begin with requirements development and architectural definition and continue on through to design and implementation. Cyber defense should be integral and not a feature inserted after development (both because of cost and because of the likely high degree of regression).

To achieve the type of cyber defense in depth described, we must develop a science of cyber protection. Cyber protection practitioners must be able to measure the effectiveness of a protection technique, measure the effectiveness of multiple protection techniques, and measure the effectiveness of variations of single or multiple techniques. Only with this knowledge can it be determined how to weave together cyber protection techniques in an effective manner and insure that the selected techniques reinforce instead of subvert each other. And, we must develop and deploy more effective cyber defensive techniques. Clearly, in order to be effective a cyber defense technique must be relatively easy to implement, extremely difficult to detect or isolate, and extremely difficult to understand or remove. However, significant research is needed before we can attach metrics and figures of merit to individual and combinations of cyber protection techniques. In addition, given the complexity desired the cyber security team needs tools that can help the team to implement cyber defenses so that it executes efficiently and accurately. In addition, tools are needed to test the cyber protection techniques, to determine the efficiency and effectiveness of the protection mechanisms, the degree of protection attained, and determine if the protection techniques achieve their specified goals. Furthermore, tools that can test for functional errors and security errors are required and it appears that each tool will require intelligent agent support. The intelligent agent support would be reuired monitor the cyber defenses to insure that the security bounds specified are maintained.

Because of the complexity of the issues involved, the cyber defense community requires the means for describing the security needs for an application as well as for an environment. We believe that the UML can address this need through the development of threat cases. The UML-based threat cases will be particularly useful in the development of techniques for testing and assessing cyber defense methodologies. UML-based threat cases would also describe how the cyber defense should respond to attack.

A further cyber defense need is for development of improved technologies that protect integer and floating point data. Protection of the data in the aggregate is the simpler problem since it can be partially addressed using strong encryption techniques, but this technique only insures that the data can not be corrupted or violated during storage and does not provide protection for the data while it is being used. In our view, run-time data

protection capabilities must be able to dynamically vary the precision and accuracy of the individual data items in response to an attack. The variation must be accomplished in a manner that preserves the statistical properties of the data, is not readily detectable, and yields reasonable but incorrect answers. To achieve these objectives, data protection techniques must be developed that preserve the statistical properties of the data before and during the protection process. In addition, techniques for determining if the data is under attack or being used in an unauthorized manner are needed. Preserving the statistical properties of the data is important because one of the easiest means for detecting if protection is present in the data is by determining if its statistical properties (such as mean, median, mode, distribution, variance, etc.) do not correlate well with the expected or known statistical properties for data of that given type. Hence, the dynamic variation of data in response to attack cannot be a random variation but must be controlled and intelligent in its application.

4. SUMMARY AND FUTURE WORK

In this paper we explored how the transition to network centric warfare brings with it the need to improve cyber defenses to insure that information is timely and correct. The increasing sophistication of computer and network attack tools coupled with the increasing technical sophistication of potential adversaries is driving this need for vastly improved cyber defenses for network and software against the threat of attack and subversion. As we have argued, cyber security is of pressing importance if network centric warfare is to fulfill its potential and become a key component of the future battlespace. This paper outlined the challenges and what, in our view, must be done to address these threats. As a foundation for our discussion, we presented a brief description of background material related to the threat, types of attacks, and defensive technologies. We then went on to address cyber security requirements and present a new strategy for cyber defense, one that builds upon and enhances the proven concept of defense in depth. This new strategy exploits the defensive advantages offered in cyberspace while also minimizing the opportunities for the attacker as well as making an attack more difficult. We also discussed needed technological developments if cyber security is to be improved. However, this paper is by no means conclusive and much research remains to be done.

Several research needs are obvious. One need is for the development of standard test suites that can be used for evaluation of cyber protection methodologies in a scientific manner. Standard testing should be coupled with the development of data protection technology assessment standards, standard data sets, and methodologies for evaluating data protection technologies. A further need is for the development of international standards to promote interoperability and insure that systems operate at equivalent levels cyber security.

There are other important research needs as well. One of these needs is a methodology that can be used to determine, in a standard manner, the degree of cyber protection required. A further need is for the development of standard attack profiles that are coupled with intelligent agents in order to enable rapid, autonomous evaluation of the effectiveness of a defense.

In conclusion, as we move toward network-centric warfare, the network and software applications will become ever more tempting and profitable targets for attack by an enemy. Our current capabilities for cyber defense in the face of a concerted attack are not up to the challenge and leave our systems dangerously exposed. Therefore, we argue that a new strategy for defense in depth is needed and that a new approach to cyber security is required. The time has arrived to replace the former dyad of protection techniques (network and operating system) by a triad of cyber protection capabilities, one that includes application security. In addition, in order to achieve a robust degree of cyber protection, the defenses must be interwoven so that an attacker is faced with a highly complex challenge if an attack is attempted. We should act now to be prepared for the more capable cyber attacker of the future.

REFERENCES

[1] Alexander, I. (2003) "Misuse Cases: Use Cases with Hostile Intent," *IEEE Software*, vol. 20, no. 1, January, pp. 58-66.

[2] Amoroso, E.G. (1994) *Fundamentals of Computer Security Technology*. Prentice Hall: Englewood Cliffs, NJ.

[3] Collberg, C.; Thomborson, C.; and Low, D. (1998) "Manufacturing Cheap, Resilient, and Stealthy Opaque Constructs," *Principles of Programming Languages 1998, POPL'98*, San Diego, CA, January.

[4] Denning, D.E. 1999) *Information Warfare and Security*, Addison-Wesley: Reading, MA.

[5] Garfinkel, S. and Spafford, G. (1991) *Practical Unix Security*. O'Reilly & Associates: Sebastopol, CA.

[6] Gollmann, D. (1999*) Computer Security*. Wiley: Mew York.

[7] Howard, M. and LeBlanc, D. (2002) *Writing Secure Code*. Microsoft Press: Redmond, Washington.

[8] Jalal, F. and Williams, P. (1999) *Digital Certificates*: *Applied Internet Security*. Addison-Wesley: Reading, MA.

[9] National Security Council. (1999) *Trust in Cyberspace*. National Academy Press: Washington, DC.

[10] Schneer, B. (1996) *Applied Cryptography*, John Wiley and Sons: New York.

[11] Stallings, W. (1999) *Cryptography and Network Security: Principles and Practice*. Prentice Hall: Upper Saddle River, NJ.

[12] Summers, R. (1997) *Secure Computing: Threats and Safeguards*. McGraw Hill: New York.

[13] Shrobe, H. (2002) "Computational Vulnerability Analysis for Information Survivability," *AI Magazine*, vol. 23, no., 4, Winter, pp. 81-91.

[14] Waltz, E. (1998) *Information Warfare: Principles and Operations*. Artech House: Norwood: MA.

UML

[15] Albir, S.S. (1998) *UML in a Nutshell*, O'Reilly Press, Sebastopol, CA.

[16] Booch, G. (1998) *The Unified Modeling Language User Guide*. Addison Wesley, Reading, MA.

[17] Booch, G.; Rumbaugh, J.; and Jacobson, I. (1999) *The Unified Modeling Language User Guide*, Addison Wesley, Reading, MA.

[18] Henderson-Sellers, B. and Unhelkar, B. (2000) *Open Modeling with UML*, Addison-Wesly Reading, MA.

Watermarking

[19] Collberg, C., and Thomborson, C. (1999) "Software watermarking: Models and dynamic embeddings," In *Conference Record of POPL '99: The 26th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, January.

[20] Kahng, A. B.; Lach, J.; Mangione-Smith, W. H.; Mantik, S.; Markov, I.L.; Potkonjak, M.; Tucker, P.; Wang, H. and Wolfe, G. (1999) "Watermarking Techniques for Intellectual Property Protection," *35th ACM/IEEE DAC Design Automation Conference (DAC-98)*, June, pp. 776-781.

[21] Palsberg, J., Krishnaswamy, S., Minseok, K., Ma, D., Shao, Q., and Zhang, Y. (2000) "Experience with Software Watermarking." *Proceedings of the 16th Annual Computer Security Applications Conference, ACSAC '00*, pp. 308-316.

[22] Nagra, J.; Thomborson, C.; and Collberg, C. (2002) "Software Watermarking: Protective Terminology," *Australasian Computer Science Conference*, pp. 177-186.

[23] Ramarathnam V., Vijay V., and Saurabh S. (2001) "A graph theoretic approach to software watermarking," In *4th International Information Hiding Workshop*, Pittsburgh, PA, April.

[24] Palsberg, J. 2000. "Software watermarking with Secret keys," *In CERIAS Annual Research Symposium on "Advancing the State and Practice*

of Information Assurance and Security, " Purdue University, W. Lafayette, IN, April 21.

Obfuscation

[25] Collberg , C.S.; Thomborson, C.; and Low, D. (1997) "A Taxonomy of Obfuscating Transformations, *Technical Report #148*, Department of Computer Science, The University of Auckland, July.