2004-06-00

Subversion as a Threat in Information Warfare

Anderson, Emory A.

Monterey, California. Naval Postgraduate School

As adversaries develop Information Warfare capabilities, the threat of information system subversion presents a significant risk. System subversion will be defined and characterized as a warfare tool. Through recent security incidents, it is shown that means, motive, and opportunity

Subversion as a Threat in Information Warfare

Emory A. Anderson[1], Cynthia E. Irvine[2], and Roger R. Schell[3]

[1] *Space and Naval Warfare Systems Command*
Systems Center Charleston
Charleston, SC, USA,
E-mail: emory.anderson @navy.mil;

[2] *Department of Computer Science*
Naval Postgraduate School, Monterey, California, USA,
E-mail: irvine @nps.navy.mil;

[3]*Aesec Corporation*
Pacific Grove, California, USA,
E-mail: Roger.Schell@aesec.com;

ABSTRACT

As adversaries develop Information Warfare capabilities, the threat of information system subversion presents a significant risk. System subversion will be defined and characterized as a warfare tool. Through recent security incidents, it is shown that means, motive, and opportunity exist for subversion, that this threat is real, and that it represents a significant vulnerability. Mitigation of the subversion threat touches the most fundamental aspect of the security problem: proving the absence of a malicious artifice. A constructive system engineering technique to mitigate the subversion threat is identified.

Keywords: Subversion, Secure Systems, Assurance

INTRODUCTION

The emerging era of ubiquitous computing and large ad hoc networks harnessed for command and control, intelligence gathering and dissemination, mission planning, etc., has resulted in a widespread understanding that these systems need to be protected from attack. Only when systems provide a verifiable resilience to attacks can the military commander be free to focus on the mission without distracting and disturbing questions regarding the trustworthiness of information placed before him. Yet critical military systems are increasingly linked together in ways that progressively increase their exposure to threats from untrusted, hostile networks. To counter these new ‘cyber’ threats, cryptography, intrusion detection systems, virtual private networks, and firewalls have taken the forefront as the must-have tools for protection. The relative newcomer may look at the wide array of solutions and assume that the industry as a whole is comprehensively addressing cyber threats. Despite these safeguards, one, all but forgotten or simply ignored, threat is system subversion. This single threat can render the plethora of “solution” largely irrelevant.

System subversion involves the hiding of a software or hardware artifice in the system that creates a ‘backdoor’ known only to the attacker. Although understood and addressed in some early systems (e.g. Benzel 1984, Schell et al. 1985, Karger et al. 1990, Weissman 1992), today the subversion threat is rarely discussed, much less is it acknowledged that, unless addressed in the underlying operating systems, subversion undermines the new security technologies that many currently rely upon for protection.

This paper describes subversion and contrasts it with other attack methods to show that it is the attack of choice for the professional attacker. It further demonstrates through real-world events and a brief description of two examples of subversion that the subversion threat is not only realistic but also requires a relatively common skill set to mount. Techniques commonly used to demonstrate the security and trustworthiness of systems are shown to provide no protection from subversion. Finally, the only technique known to date that can address subversion is described.

SUBVERSION

In the context of the broad range of activities encompassed by information warfare, attacks on computer systems and networks can be regarded as probable elements of the adversary's arsenal. Although some attacks, such as rapidly spreading worms and viruses (Hansen 2003), are highly visible and easily characterized to the general public, subversion offers a subtler and more insidious form of attack.

System subversion is '... the covert and methodical undermining of internal and external controls over a system lifetime to allow unauthorized or undetected access to system resources and/or information.' (Myers 1980). The planting of the subversive artifice may occur at any phase of the system lifecycle with varying implications on the difficulty, effectiveness, and ability to avoid detection. To understand why subversion has been described as the technique of choice for professional attackers (Schell 1979), it is necessary to examine the professional attacker, describe subversion, and to review alternative attack methods.

The Professional Attacker

An information warfare professional is distinguished from the amateur by objectives, resources, access, and time. The professional is concerned about avoiding detection whereas amateur attackers are often motivated by a desire for notoriety or simple curiosity as much as for gaining access. A professional will be well funded and have adequate resources to research and test the attack in a closed environment to make its execution flawless and therefore less likely to attract attention. The professional attacker understands the system lifecycle (Myers 1980) and may surreptitiously construct a subverted environment by participating in or controlling the efforts of a development team. In a large system with complex dependencies between modules the opportunities for this are abundant. Finally, the professional is willing to invest significant time in both the development of the artifice as well as its use, possibly waiting years before reaping the benefits of the subversion. The subverter (who plants the artifice) may be - in fact, usually will be - a different individual than the attacker. A professional may have paid or persuaded someone else to perform the subversion and will at some point in the future, activate the artifice and attack the system. This may provide the professional attacker with a degree of plausible deniability not possible with typical frontal attacks. For a state-sponsored adversary such deniability is highly desirable.

Note that the professional does not necessarily possess more skill than the amateur attacker. The technical skill level required to plant an artifice might be comparable to that of an intermediate level programmer. This will be illustrated in the two examples provided in a subsequent section.

Meyers (1980) described the system lifecycle phases in which subversion attacks can be successfully mounted.

Design and Implementation Phase Subversion

Design phase subversion has the least chance of being 'patched' away. Although the artifice will be visible at some level in the design documentation, there is ample opportunity for a subverter to propose a seemingly sound design that offers subtle vulnerabilities.

The subverter can insert the artifice during implementation either by infiltrating the development team as an agent or by exploiting weaknesses at the development facility. In a given system, the subverter often has a vast number of options for hiding an artifice. The use of low level languages such as assembly can be used to code the artifice in a way that make understanding its nature more difficult.

Any artifice placed in the system's source code is visible in the source listing and might be discovered by a developer with understanding of the system of sufficient breadth and depth. Thus, the subverter might remove the artifice from the source listing after it has been compiled or subvert the compiler itself (Karger et al. 1974, Thompson 1984). An attack on the Linux source code tree (Lemos 2003) illustrates that adversaries consider implementation-phase subversion attractive.

Subversion During Distribution, Maintenance, and Support

With attacks during the distribution, maintenance and support phases of the system lifecycle, the subverter can choose to affect all delivered systems or focus on a particular target. Following quality assurance testing, a shippable binary has been produced. The subverter then gains access to the distribution system and applies a patch to the binaries or replaces them entirely. Early experiments (Jbtzhm 2002) demonstrate that this attack vector is feasible.

Similar methods may be used in maintenance and support phases. Most vendors have patches and updates available via internet download, update media, or service personnel. The difficult task for the user is to establish that the copy of the system is a legitimate one. Despite the use of MD5 hashes to verify integrity, digital signatures can be compromised (Verisign 2001).

Contrast with Other Attack Methods

The reasons why subversion best meets the goals and objectives of the professional attacker are evident when it is compared with other attack techniques. Myers (1980) classifies attacks into three categories: Inadvertent Disclosure, Penetration, and Subversion. Inadvertent disclosures predominantly result from accidentally occurring states in the system or human error. Penetration is a more deliberate attempt to exploit an *inadvertent, preexisting* flaw in the system to bypass security controls. The penetrator will exploit bugs or other flaws to bypass or disable system security mechanisms. To succeed the penetrator must depend upon the existence of an exploitable flaw. In contrast, subversion requires the subverter to have sufficient access to the system at one or more points during its life cycle to exert influence on its design, implementation, distribution, installation, and/or production in a way that can later be used to bypass the protection mechanisms. Subversion favors a carefully hidden mechanism with a high likelihood of persistence through new product versions and upgrades. These characteristics help to remove uncertainty regarding the success of an attack thus making subversion more attractive than penetration.

Finally, a subversive artifice will typically include the capability of activation and deactivation. It was recognized long ago (Schell 1979) based on experience with information warfare demonstrations that "deliberate flaws are dormant until activated by an attacker. These errors can be placed virtually anywhere and are carefully designed to escape detection." The artifice activation mechanism waits for the presence of some unique trigger. Examples are a particular stack state, an unlikely sequence of system calls or signals, or codes hidden in unused portions of data structures passed into the kernel via system calls. The possibilities are endless. This trigger can be thought of as a key that can be made arbitrarily long from a cryptographic standpoint.

The distinction between a so-called Trojan Horse and an artifice as used in system subversion is important. The Trojan Horse provides a function that entices the victim to use the software as well as a hidden function that carries out the malicious intentions of its designer. With this technique, system security mechanisms are still in place and functioning properly - the attacker's code is executed with and limited by a legitimate user's access permissions. Subversion does not require action by a legitimate user. It bypasses any security mechanisms that the subverter chooses to avoid.

In summary, subversion is a likely choice in information warfare since it may be introduced at many points in the system lifecycle, its execution is controlled by the attacker, and, thus, is more predictable; and, by its very nature, it provides the attacker with a high degree of plausible deniability.

SUBVERSION THREAT MODEL

Given that subversion is a likely mode of cyber attack, it is useful to examine the elements required for the successful use of subversion. These elements are *means, motive* and *opportunity*.

For means, the adversary must have the intellectual capacity and resources required to conceive of and develop a system subversion.

Motive is ever present. Adversaries wish to be victories over their enemies.

Opportunity requires that the adversary have access to the target system at some point during its lifecycle. To successfully implant an artifice, the adversary must not only have access to the system, but must not be detected.

Is The Subversion Threat Real?

Early analyses of subversion proposed that the U.S Air Force (Schell, 1979) and the U.S. Navy (Grant and Richie, 1983) were sowing the seeds of their own destruction through increased reliance on computers and electronics. At the time, however, it was difficult to see evidence of a mature information warfare threat and reliance on technology was not so widespread as to make the impact of such an attack significant. Today, however, reliance on cyber systems is fast approaching a point of no return, and many nations have publicly displayed intent to develop information warfare capabilities (Wang 1997). Yoshihara's (2001) analysis of the potential of Chinese information warfare suggests that war-planning based upon the habits of the enemy can lead to unexpected victory. A military's habit of dependence upon systems that may have already been subverted could be the key to a rapid and decisive strategic victory for an adversary.

The environment for a large proportion of commercial information technology development is conducive to subversion. The sheer size and complexity of today's systems make the insertion and hiding of an artifice much easier than it was at the time of Myers' (1980) work. Additionally, design and development efforts are more divided among many individuals or departments, which makes it possible to divide the artifice into separate modules further obscuring its true nature. As a result of short product cycles, a subverter may stand a good chance of slipping an artifice through quality control.

Those responsible for critical systems have been willing to accept a cycle of bugs and patches to address flaws discovered in fielded systems. Such systems make particularly soft targets for information warfare attacks. This is illustrated by both the Yorktown incident (Neumann 1998) in which a destroyer had to be towed back to port because of a failure associated with commodity software and the U.S. Navy Marine Corps Internet (NMCI) (2004) effort which depends upon similar commodity products.

Recent reports of security-critical problems could just as easily have been the result of subversion as error. In October 2000, an individual gained access to systems at Microsoft and had access to the source code for a future release of the Windows operating system and Office Suite (Keaton 2000). In January 2001, Verisign Corporation (Verisign 2001) erroneously issued Microsoft certificates to individuals who falsely claimed to be Microsoft employees. As a result, these individuals had the opportunity to publish code that would appear to be certified by Microsoft Corporation. On Dec 17, 2001 a member of Al Qaeda captured in Afghanistan claimed that members of a terrorist organization had infiltrated the Microsoft Corporation as programmers for the expressed purpose of subverting the Windows XP operating system (Newsbytes 2001). Despite a lack of corroborating reports, this indicates that subversion is considered as a valid attack technique in conditions of asymmetric warfare.

Probably the most impressive case validating the threat of subversion is that of the "Farewell Dossier" (Weiss 1996). In 1981, President Reagan received intelligence that provided evidence of a KGB program (known as "Line X") to obtain technology from the U.S. According to a recent press release (Safire 2004), the United States covertly injected subverted systems into the Line X efforts to obtain components for an oil pipeline automation system. These systems were designed to pass inspection and testing, but programmed to subsequently allow pumps to exceed pipeline pressure limits. The result was "the most monumental non-nuclear explosion and fire ever seen from space" (Safire 2004). The impact of this event went beyond the pipeline however, as all of the software and equipment that had been stolen over the years was then viewed with suspicion as untrustworthy. Scientists and technicians stopped or delayed work over these concerns. Furthermore, this was concurrent with President Reagan's Strategic Defense Initiative (SDI) or "Star Wars" which intensified the economic burdens that led to the Soviet downfall.

Combined, these cases demonstrate that means, opportunity and motive exist for carrying out subversive attacks. As noted by Weiss (1996), subversion is an information warfare tool that can benefit an adversary in either of two ways: plausible suspicion or successful use.

Threat Model Analysis

Due to the existence of *means, motive,* and *opportunity* to conduct subversive attacks in the current environment, subversion should be viewed as a real threat. Furthermore, if successful, subversion can have an impact far more pervasive than the corruption of a particular system. To counter the threat of subversion, at least one of the three conditions that make it possible must be eliminated.

Obviously, it would be impossible to reduce or eliminate the subverter's *means* to subvert the system. The skills required to carry out this type of attack are far too common today. Graduate students developed convincing subversion demonstrations in a matter of months (Anderson 2002, Lack 2003, Murray 2003, Rogers 2003). Anderson (2002) created a subversion demonstration with an attack on the Network File System in the Mandrake version of Linux. The artifice, consisting of a total of eleven lines of source code, was distributed in several locations within the operating system, and was activated and deactivated by a single corrupted UDP packet. As this was a student project, this effort clearly demonstrated that a subversion artifice could be created in a relatively short time by someone with high, but not, world-class design and programming skills. In the same work, a sketch of a subversion of the Secure Sockets Layer (SSL) illustrates a credible clandestine attack on a highly popular application-level security mechanism.

Eliminating the *motive* is likewise a losing proposition. To do so, it is necessary to either reduce the benefit of mounting the attack or raise the cost of the attack to the point that it becomes prohibitive. Since subversion bypasses the security controls of the system, the attacker gains at least as much benefit from the system as do legitimate users. Therefore, reducing the benefit for the attacker would reduce the benefit to legitimate users as well.

Raising the cost of mounting the attack is unlikely to affect motive since subversion is the choice of a professional attacker, who is willing to incur substantial costs in resources and time. Some professional attackers would have virtually unlimited resources, thus high cost would have little effect.

A defender might also attempt to reduce motive through deterrence. Laws combined with a strong enforcement system discourage bad societal behaviour. However, with system subversion, avoiding detection is the attacker's overriding concern and any hypothetical law enforcement effort would be mired in its inability to detect the activity and establish attribution. Subversion demonstrations in the early 1970's showed that some artifices were so undetectable that the system manufacturer was unable to locate them, even when told of their presence and given details on their operation (Karger et al. 2002). Since system subversion is a type of attack that would be considered for information warfare by a nation state (or state-sponsored organization), organized crime group, or large corporation involved in corporate espionage, laws would likely be ineffective deterrents.

The only choice left is to reduce or eliminate any *opportunity* to subvert the system. At one level, this involves denying the subverter access to the system at all phases in its lifecycle. Physical security of the development environment, background checks for employees and protected distribution and upgrade channels provide some measure of defence. However, these measures alone are insufficient to provide assurance that subversion cannot occur. Espionage cases show that reliance solely on background checks and security clearances is imprudent. To do so would require certainty of the trustworthiness of all individuals with access to a system throughout its entire lifecycle.

In an attempt to determine how to mitigate the subversion threat, engineering techniques to demonstrate the absence of artifices will be analysed.

EVALUATION OF SYSTEM SECURITY IN THE FACE OF ARTIFICES

The professional attacker will most definitely spend significant effort to obfuscate the artifice, thus rendering it essentially impossible to discover. Apart from catching the subverter in the act of subversion, discovering the presence of a particular artifice after it has been created can be made computationally infeasible (Spinellis 2003). Before exploring techniques to assure the absence of artifices, two possible ways of 'finding' an artifice once it has been planted are refuted. They are source code inspection and security test and evaluation.

Source Code Inspection

Access to source code will not give an inspector the ability to ensure the absence of artifices. A well-crafted artifice will be hidden in a way that makes it exceedingly difficult, if not impossible, to identify through source code inspection. In fact, the subverter can plant an artifice so that it never appears in the source listing of the system, by subverting the compiler used in developing the system, or by planting the artifice in the object or binary code directly.

Even without going to the trouble of subverting the compiler, the subverter can hide an artifice in a way that it is unlikely to be found. Today's systems are large and complex. The more complex a system is, the harder it is to understand its overall function. Capitalizing on this, the attacker can make his artifice code obscure. Developers will often leave code alone if they do not understand what it does and it is not known to cause any problems in the system.

Security Test and Evaluation (ST&E)

Security tests and penetration testing are worthless tools for assuring that a system is free of artifices. Any artifice that incorporates a well-designed trigger (one which is unique) will never be found because the testing will occur when the artifice is disabled. Code testing most often involves testing for the documented features to ensure proper function. It does not attempt to discover all possible undocumented or unexpected functions. To do so by testing would require that all possible inputs be sent to all possible interfaces. This exhaustive approach is infeasible if not impossible for all but trivial systems.

An informal argument supporting this assertion can be made by comparing the task of finding an artifice to finding a software bug in a system. Certain software bugs are particularly difficult to track down. These 'Heisenbugs' may appear and disappear as a result of a rare combination of conditions in the system. Finding them is exceedingly difficult. Now imagine that a similar function is built intentionally. The feature manifests itself only when a rare combination of system conditions exists and furthermore, it can be triggered to occur at the will of the attacker.

The following is presented in order to persuade the reader against any notion that testing will address subversion. Dijkstra provided a scenario of testing a 72-bit adder that may be used here to show the futility of using testing to determine whether or not an artifice exists in the system. In this scenario, the tester is given a black box that adds two 72-bit values with carry. The task is to determine if the adder performs correctly for every possible input. It is necessary to state one of two results: (1) the adder performs correctly for every possible input or (2) the adder does not perform correctly for every possible input. If the result is (2), the tester must provide the input conditions (i.e. the two numbers) for which the adder fails. The only way to complete this task under these conditions is to test all possible inputs and validate the output. The number of inputs to be tested is 2^{72} x 2^{72} or 2^{144} possibilities. Consider a 128 bit cryptographic key and the amount of work required to mount a brute force attack against it. This pales in comparison to the work required to test all inputs to the adder. Conducting this test is computationally infeasible.

Suppose the tester or certifier knows how to tell if a given artifice is active as well as how to try guesses of the trigger value (i.e. how to send an arbitrary value into the trigger mechanism is known). This is essentially the same scenario as Dijkstra's 72-bit adder problem; the total number of tests required to reveal the trigger is likely to be intractable. Now, suppose the task is not just to guess the trigger, but is to test whether an unknown artifice is present at all. In this case, the tester does not know how to send in a guess at the activation key. Even if the key were stumbled upon and somehow accidentally activated the artifice, the tester would likely not realize that this had happened because the behaviour of the artifice is not evident. The problem space is now the product of the key space, potential activation mechanisms, and artifice function. The added difficulty of this task is obvious. The same arguments apply to so-called 'active defences' built upon the use of intelligent agents or network monitoring to ensure system self-protection, or to the use of cryptographic techniques for system protection. These are worthless in the face of subversion.

Having shown that discovery of an artifice after the fact through automated mechanisms, source code inspection, and security test and evaluation are infeasible, an alternative is to demonstrate that the system has been constructed in a way that analysis demonstrates the absence of artifices. First techniques that are inadequate to demonstrate the absence of subversion are presented. These are various popular software engineering approaches and the inappropriate use of formal methods. Finally the technique of verifiable secure systems engineering is described as a constructive approach to mitigation of the subversion threat.

Software Engineering, Object-Oriented Programming (OOP), and Developmental Assurance Approaches

Software engineering attempts to provide a sound methodology for careful system decomposition into highly cohesive modules and reduces the amount of reckless programming that occurs within software projects. OOP language constructs enable software reuse as well as data hiding and encapsulation. However, while these measures bring tremendous benefit to system development, they do not guarantee secure systems. Parnas (1972) provides two modularized designs for the same hypothetical system to demonstrate good and bad examples of modularization; however, neither guarantee the absence of subversion.

Developmental assurance approaches such as the System Security Engineering - Capability and Maturity Model (SSE-CMM) (2004) attempt to address the security problem by enhancing the quality of the system that produces the software. As in most quality-driven models, the SSE-CMM cannot reliably address the threat of subversion because the model assumes that all members of the team share the goal of producing secure software. If a subverter is operating covertly on the development team, the model breaks down.

Gratuitous Formal Methods

The application of mathematics and formal methods to security problems does not necessarily mean that a more secure system has been created. Code level proofs have been suggested as a way to verify system security. Such proofs may address code at the module level but provide no insight into the overall security of the system; they say nothing about how the modules compose and interact. In addition, many schemes for code proofs address the wrong problem. A recent example of this is Proof Carrying Code (Necula 1997), which provides evidence of what the code will do, but does not address the true security problem, which would be to give a proof of what the code will not do.

Verifiable Systems

Can systems be built and verified to behave as specified?

Return to the 72-bit adder with carry illustration. In practice, designers would build this device by constructing 72 one-bit adders with carry and then linking them together properly. Then, testing of the entire system amounts to checking each of the 72 individual components for proper handling of the four possible inputs. The system has been reduced to completely testable modules. Consequently, the operation of the device as a whole has become verifiable and it is possible to claim with high assurance that it will properly handle every possible input without actually testing every possible input.

A slightly more complex example helps to show the scalability of this approach. Dijkstra (1968), in the design of the 'THE'-Multiprogramming System, showed that '...it is possible to design a refined multiprogramming system in such a way that its logical soundness can be proved a priori and its implementation can admit exhaustive testing.' The THE system and the 72-bit adder examples are alike in that to test the THE system (or any general purpose computer) at the system interface with no knowledge of how it was constructed would require feeding all possible programs into the system. Dijkstra states that testing must only involve what is relevant and what is relevant can be decided only if the internal structure of the system is known. The small set of test cases that could be exhaustively tested was a result of the modular and layered nature of the design. Although Dijkstra's system did not entail formal verification, it involved a rigorous constructive argument similar to that used for geometric proofs.

Verifiable Protection

Because of current vagaries and hyperbole surrounding the terms '*assurance'*, high assurance must be defined meaningfully. Systems must meet not only sound security criteria, but also be built in such a way that it is possible to verify the protection mechanisms provided. Accordingly, the term *verifiable protection* (DOD 1985) will be used in the context of a system resilient to subversion.

A system offering verifiable protection provides high assurance that its security properties are correct, complete, and allow nothing beyond their specification. Additionally, it will be designed to have no exploitable security flaws and it will be constructed to be subject to third party inspection and analysis to confirm the protections are correct, complete and do nothing more than advertised.

The size and complexity of today's typical large system prohibit attempts to demonstrate that the entire system is verifiable. The properties of the system that must be verifiable should be limited to the interface of a small, complete and non-bypassable component to which enforcement of security policy has been allocated. When this approach was conceived this was termed a "security kernel" (attributed to Schell by Saltzer et al. (1975)) and it was noted (Schell 1979) "the kernel makes it tractable to protect against subversion."

A formal model describing core security policy and properties is mathematically proven to be sound and correct. The system interface operations correspond to one or more rules that represent the security policy. All elements of the system must be demonstrated to be both necessary and sufficient for enforcement of the policy. The implementation is systematically traced to the policy. This includes showing that all of the code is required for policy enforcement and system self-protection. This confirms there is no extraneous code and the system is shown to do what it supposed to do – and *nothing more*.

The system is "proved" to enforce the security policy and to protect itself from penetration using a combination of formal methods and rigorous argument. It not a proof in the absolute sense, but provides confidence through its believability (DeMillo et al. 1977).

A system so designed eliminates opportunities for attackers to plant artifices, as any function that does not relate back to the policy model will be discovered in the correspondence mapping. A successful example of the use of these techniques was provided by the Blacker program (Weissman 1992), a set of coherent devices intended to secure the Defense Data Network (MacKenzie et al 1997). This system was constructed due to a conscious decision to address the concerns described here.

To obtain verifiable protection in a system, it also must be possible to audit the correct functioning of the system after the fact. Although the TCSEC (DOD 1985) was used in the context of the Blacker program, new criteria (ISO 1999) provide details on the minimum requirements to meet these two objectives and cover every part of the system lifecycle.

Rigorous verifiable protection methodologies can be applied to successfully provide appropriate security from a technological standpoint. Unfortunately, knowledge of how to build such systems exists primarily in a select few scattered individuals and a few products such as the GEMSOS security kernel (Schell et al. 1985).

To rescue the lessons learned in past high assurance development projects, the Naval Postgraduate School is developing a Trusted Computing Exemplar (Irvine 2003): a high assurance isolation kernel that offers verifiable protection. All aspects of this effort will be open to observation and participation, and will hopefully result in an open source product available for use as the foundation for further high assurance efforts.

CONCLUSION

The threat of subversion presents a significant risk when considering adversaries engaged in a broad spectrum of information warfare activities. Recent security incidents as well as historical records of subversion demonstrate that the threat is real. Examination of the requirements for carrying out a subversion indicates that it is feasible in the context of the development of contemporary commodity systems. Mitigation of the subversion threat touches the most fundamental aspect of the security problem: proving the absence of a malicious artifice. An analysis of methods that might be applied to mitigate subversion shows that use of a constructive security engineering technique that combines formal methods with rigorous arguments provides a demonstrably feasible approach to mitigation of the subversion threat. With the existing evidence for the reality of the threat, it is possible to conclude that those responsible for critical national and military systems would be negligent by failing to address the subversion threat.

REFERENCES

Anderson, E.A. (2002) A Demonstration of the Subversion Threat: Facing a Critical Responsibility in the Defense of Cyberspace, Master's Thesis, Naval Postgraduate School, Monterey, CA, USA, March.

Benzel, T.V. (1984) Analysis of a Kernel Verification, *Proceedings of the 1984 IEEE Symposium on Security and Privacy*, Oakland, CA, April, pp. 125-131.

DeMillo, R., Lipton, R. J., and Perlis, A. J. (1977) Social Processes and Proofs of Theorems and Programs, *Proceedings of the 4th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages*, Los Angeles, CA, pp. 206-214.

Dijkstra, E. W. (1968) The Structure of the "THE"-Multiprogramming System. *Comm. A. C. M.* **11**(5), pp. 341-346.

DOD 5200.28-STD (1985) Department of Defense Trusted Computer Evaluation Criteria (TCSEC).

Grant, P. & Richie, R. (1983) The Eagle's Own Plume. *Proceeding of the United States Naval Institute*, July.

Hansen, J. C. (2003) *Worm Propagation in Heterogeneous Networks: Weaknesses in the National Strategy for Cysberspace Protection*, Masters Thesis, Naval Postgraduate School, Monterey, CA, March.

Irvine, C. E. (2003) Teaching Constructive Security, *IEEE Security and Privacy*, 1(6), pp 59-61.

ISO/IEC 15408 (1999) Information Technology - Security techniques -- Evaluation criteria for IT Security - Part 3: Security Assurance Requirements, International Organization for Standardisation and International Electrotechnical Commission.

Jbtzhm (2002) *Static Kernel Patching*. In Phrack 60. Available at: http://www.phrack.org/phrack/60/p60-0x08.txt. Downloaded 20 May 2003.

Karger, P. A., and Schell, R. R. (1974) *MULTICS Security Evaluation: Vulnerability Analysis*, ESD-TR-74-193, Vol. II, Electronic Systems Division, Air Force Systems Command, Hanscom AFB, Bedford, MA, June. Also in *Proceedings of the Computer Security Applications Conference*, Las Vegas, NV, USA, December, pp 126-146.

Karger, P.A., and Schell, R. R. (2002) Thirty Years Later: Lessons from the Multics Security Evaluation, *Proceedings of the Computer Security Applications Conference*, Las Vegas, NV, USA, December, pp 119-126. Available at http://www.acsac.org/2002/papers/classic-multics.pdf

Karger, P.A., Zurko, M.E., Bonin, D.W., Mason, A. H. and Kahn, C.E. (1990) A VMM Security Kernel for the VAX Architecture, *Proceedings of the 1990 IEEE Symposium on Research on Security and Privacy*, Oakland, CA, USA, May, pp. 2-19.

Keaton, J. (2000) Microsoft: Big Hack Attack, *CNNMoney*, http://money.cnn.com/2000/10/27/technology/microsoft/. October. Downloaded 20 February 2004.

Lack, L. (2003) *Using the Bootstrap Concept to Build an Adaptable and Compact Subversion Artifice*, Master's Thesis, Naval Postgraduate School, Monterey, CA, USA, June.

Murray, J. (2003) *An Exfiltration Subversion Demonstration*, Master's Thesis, Naval Postgraduate School, Monterey, CA, USA, June.

Lemos, R. (2003) Attempted attack on Linux kernel foiled, *CNET News.com*, http://news.com.com/2100-7355-5103670.html. Downloaded 16 February 2004.

MacKenzie, D. and Pottinger, G. (1997) Mathematics, Technology, and Trust: Formal Verification, Computer Security, and the U.S. Military, *Annals of the History of Computing*, **19**(3), pp. 41-59.

Myers, P. A. (1980) *Subversion: The Neglected Aspect of Computer Security*, Master's Thesis, Naval Postgraduate School, Monterey, CA, USA, June.

Neuman, P.G. and Crawford, M.D. (1998) USS Yorktown Dead in the Water After Divide by Zero, Risks Digest, **19**(88), http://catless.ncl.ac.uk/Risks 22 July. (abstracted from Slabodkin, G. (1998) Software glitches leave Navy Smart Ship dead in the water, Government Computer News, 13 Jul 1998 http://www.gcn.com/gcn/1998/July13/cov2.htm)

Necula, G. C. (1997) Proof Carrying Code, *Proceedings 24th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, Paris, France, pp. 106-119.

Newsbytes (2001) http://www.newsbytes.com/news/01/173039.html. December.

NMCI (2004) NMCI Playbook, http://www.nmci.navy.mil/Primary_Areas/NMCI_Playbook/Secured/NMCI_Playbook?Expand=27504

Parnas, D. (1972) On the Criteria to be Used in Decomposing Systems into Modules. *Communications of the ACM*, **15**(12), pp. 1053-1058.

Rogers, D. (2003) *A Framework for Dynamic Subversion*, Master's Thesis, Naval Postgraduate School, Monterey, CA, USA, June.

Safire, W. (2004) The Farewell Dossier, *The New York Times*, 2 February 2004, Section A, p. 21.

Saltzer, J. and Schroeder, M. (1975) Protection in Information Systems, Proceedings of the IEEE, **63**(9), pp. 1278-1308.

Schell, R.R. (1979) Computer Security: the Achilles' Heel of the Electronic Air Force? *Air University Review*. **30**. pp. 16-33, http://www.airpower.maxwell.af.mil/airchronicles/aureview/1979/jan-feb/schell.html

Schell, R.R., Tao, T. F. and Heckman, M. (1985) Designing the GEMSOS Security Kernel for Security and Performance, *Proceedings 8th DoD/NBS Computer Security Conference*, pp. 108-119.

SSE-CMM (2004) The Systems Security Engineering Capability Maturity Model (SSE_CMM), http://www.sse-cmm.org/model/model.htm. Downloaded 20 February 2004.

Spinellis, D. (2003) Reliable Identification of Bounded-Length Viruses is NP-Complete, *IEEE Transactions on Information Theory,* **49**(1), pp. 280-284.

Thompson, K. (1984) Reflections on Trusting Trust, *Comm. A. C. M.*, 27(8), pp 761-763.

Verisign (2001) *VeriSign Security Alert Fraud Detected in Authenticode Code Signing Certificates,* March, http://www.verisign.com/developer/notice/authenticode/. Downloaded 3 February 2004.

Wang, P. (1997) The Challenge of Information Warfare, in *Chinese Views of Future War*, Revised Edition, Ed. Michael Pillsbury, National Defense University, Institute for National Strategic Studies, Washington, DC.
http://www.au.af.mil/au/awc/awcgate/ndu/chinview/chinacont.html

Weiss, Gus (1996), Duping the Soviets: The Farewell Dossier, in *Studies in Intelligence*. Vol. 39 No. 5. Available: http://www.cia.gov/csi/studies/96unclass/farewell.htm. Downloaded 25 February 2004.

Weissman, C. (1992) BLACKER: Security for the DDN Examples of A1 Security Engineering Trades. *Proceedings of the 1992 IEEE Symposium on Research in Security and Privacy*, Oakland, CA, USA, pp. 286-292.

Yoshihara, T. (2001) *Chinese Information Warfare: A Phantom Menace or Emerging Threat?,* Strategic Studies Institute, Carlisle, PA, November.
ttp://www.carlisle.army.mil/ssi/pubs/2001/chininfo/chininfo.htm