Information Security: Science, Pseudoscience, and Flying Pigs

Dr. Roger R. Schell
Aesec Corporation
schellr@ieee.org

Abstract

The state of the science of information security is astonishingly rich with solutions and tools to incrementally and selectively solve the hard problems. In contrast, the state of the actual application of science, and the general knowledge and understanding of the existing science, is lamentably poor. Still we face a dramatically growing dependence on information technology, e.g., the Internet, that attracts a steadily emerging threat of well-planned, coordinated hostile attacks. A series of hard-won scientific advances gives us the ability to field systems having verifiable protection, and an understanding of how to powerfully leverage verifiable protection to meet pressing system security needs. Yet, we as a community lack the discipline, tenacity and will to do the hard work to effectively deploy such systems. Instead, we pursue pseudoscience and flying pigs. In summary, the state of the science in computer and network security is strong, but it suffers unconscionable neglect.

1. Introduction

The scientific underpinnings of computer and network information security have not changed much in twenty years. This is not surprising because the fundamental properties of information theory and the limits on what is computable are not subject to much evolution.

The science of computer and network information security has for some time given us the ability to purchase an information system from a mortal enemy and then assess its ability to enforce a well defined security policy, gaining sufficient assurance to confidently use the system to protect against massive loss and grave damage, and this has been actually been put into practice. This astonishing capability is known as "verified protection".

On the other hand, the state of the pseudoscience of computer security has changed primarily in its success as a growth industry. Much of the same old snake oil is being peddled, but the manufacturing capacity has ramped up geometrically. The pursuit of pseudoscience remains an incredible source of research dollars and product revenue. To sustain this growth the science of computer security has, on the other hand, been widely abandoned and orphaned, summarily declared a failure and unworkable by a community inclined to ignore scientific success and worked examples. The result of this tension between science and pseudoscience was that our ability to achieve verified protection in fielded computers and networks peaked somewhere near the mid nineties. Figures 1 and 2 are notional illustrations of some of the associated disturbing trends. As many more businesses become more aware of their needs for secure systems, and attract more "experts" there is a smaller percentage of "experts" who actually understand the hard issues, as illustrated in Figure 1. Figure 2 illustrates that at the same time, the ability to procure a bulletproof system is

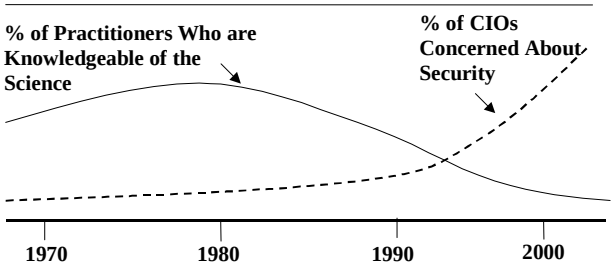

Figure 1: Expertise Diminishes as It is Needed

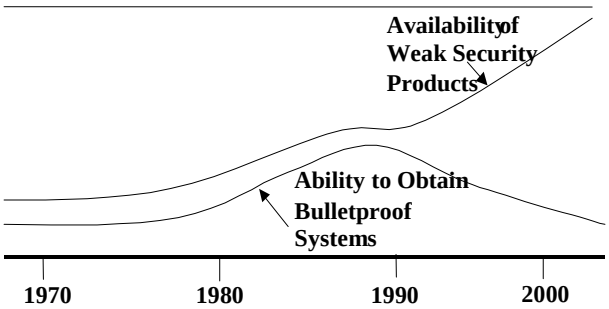

Figure 2: Lack of Evaluated Products

diminished while the market is thick with products having weak security. So how did we get to this unfortunate state? It is hoped that the rest of this essay will help make that clearer.

2. Critical Areas of Focus

This essay strives to provide a critical examination of the state of the science in computer and network security in light of the author's experiences from nearly the beginnings of computer security to the present. However, before we review the scientific advances that preceded the current decline, it is useful to examine what is meant by verified protection, acknowledge significant remaining hard problems, and review what verified protection can and cannot help us achieve. And, we will review the ability to leverage verified protection within systems to divide and conquer seemingly intractable problems.

2.1. Verifiable Protection

Malicious software is the tool of choice for well-planned professional attacks and is the primary threat addressed by systems offering verified protection. To counter malicious software these systems must be designed and built from the very start to have all of the following properties:

- Designed to have no exploitable security flaws.
- Enforce security policies on information flow, thereby bounding the damage of malicious applications software (e.g., Trojan Horses).
- Built to be subject to third party inspection and analysis to confirm the protections are correct, complete and do nothing more than advertised (i.e., no trap doors).

Scientific foundations of information security have provided us with three necessary tools for achieving verifiable protection:

- An ability to identify situations where verified protection is both needed and possible. Central to this is the fundamental distinction between discretionary and mandatory security policies.
- The tools and techniques to implement and field heterogeneous systems where some, but not all, components have verified protection.
- Criteria and methods to independently verify the protections offered by such systems.

Though sufficient to solve many of today's serous challenges, verified protection is not the solution to all computer security problems. Nor is the state of verified protection complete and without areas that could benefit from new research.

2.2. Remaining Hard Problems

Several hard problems remain. These include:

- Verifying the absence of trap doors in hardware, particularly with the amount of automated design currently used. It has been estimated that, "A product designed to cope with subversion could cost 50 to 100 times as much as . . . non-secure products of similar functionality" [1]. Potential ameliorating approaches are analogous to NSA's use of cleared programmers.
- Verifying the absence of trap doors and other malicious software in development tools such as compilers and linkers.
- Covert timing channels remain a problem, however there are demonstrated methods of significantly reducing them.
- Covert channels in end-to-end encryption systems (e.g., VPNs) remain a significant challenge.
- Despite some progress, we lack formal methods for meaningfully corresponding source code to a formal specification, and object code to source.
- Denial of service attacks can be reduced but not effectively eliminated.

2.3. Omniscient Classification of Information

The inability to omnisciently classify the confidentiality or integrity of information is perhaps the single most visible intractable problem in the science of information security, and is likely to remain so. And thus malicious e-mail attachments remain a problem. Science has produced no solution to the problem of rotten apples in a barrel. Users want the convenience of inhabiting the same integrity domain as the least conscience and least informed among them. When some in such a domain behave badly, everyone suffers. Deciding to inhabit different domains at different times largely solves the problem. Otherwise users must live with the vandals, the graffiti artists and the untrained co-workers who invite in the riff raff. Compounding this problem is the trend toward "transportable code". Ultimately however, there is no difference between code and data when it comes to maintaining the integrity of a domain. Many years ago, this very fact led General Motors to establish two parallel domains in their operations and network: one for use in generating and managing engineering data; and one for everything else. Users consciously decided which domain they would inhabit at any given time. This use of closed user groups is a relatively simple solution to an otherwise intractable problem.

The key to maintaining separate domains that permit information to flow in accordance with the desired policy is the use of security labels to enforce a mandatory [2]

security policy – enforcement that is global and persistent. Security labels can be viewed as a social mechanism for understanding the sensitivity of your current context, be it the processing, the reading or the writing of information. Some incorrectly claim that the use of security labels is not a viable solution because it would require the a priori assignment of classifications to data, people and processes on a global basis. In fact, labels can be structured such that their interpretation can be based on a layering of policies such that local policies are only interpreted locally. An example of such a labeling strategy is the distributed dynamic labels for digital certificates defined by Novell [3] and incorporated into their flagship NetWare product, and used by others.

2.4. Success Through Divide and Conquer

History has shown that we need not solve all of the problems at once to have suitably secure systems. The ability to divide and conquer problems has permitted great advances. A core innovation in the science of computer security is to structure systems into those elements of hardware, firmware and software that enforce the security policy and those elements that do not. As a result of such a partitioning, only a subset of the overall system must offer verified protection, and that subset is called the "Trusted Computing Base" (TCB) [4]. In particular, most of what is commonly viewed as operating system functions and all of the applications software need not offer verified protection and are therefore external to the TCB. A basic ability to control access to information can be extended by partitioning other problems into manageable pieces and designing systems to fail secure.

As systems get more complex and unmanageable, you end up with the Internet. There are, however, positive aspects to the evolution of the Internet environment that support divide and conquer techniques. The development and acceptance of line protocols as the primary systems interface has eliminated many of the past "compatibility" problems. Where once an application had to be instruction set compatible with an IBM mainframe, now a lot can be achieved through compatibility with a line protocol. This enables heterogeneous systems with selective use of secure appliances (e.g., Certificate Authorities) whose interface is the network rather than a particular operating system.

The Internet can also greatly benefit from the adoption of the IPSEC standard. Assuming secure protocols for managing keys become widely adopted, VPNs based on IPSEC can do a lot towards bringing strong security to the Internet environment. The use of PKI for managing IPSEC keys holds much promise, particularly because PKI can be built to support a variety of different distributed systems (again permitting a divide and conquer approach). However PKI introduces a new set of vulnerabilities that must be countered with more than emphatic assertion. For example, unless there exists a strong technical basis for assuming certificates cannot be forged (e.g., through the use of trap doors planted in platform operating systems) PKI's foundations will remain mired in pseudoscience.

TCB extensions are a powerful technique demonstrated by Novell [5] to achieve a secure client to interoperate with their server within a well-defined network security architecture. While this was done at a low level of assurance, nothing prevents the same techniques from being applied at high levels of assurance.

Another approach to securing the client in a network is to use a "thin client" that does not execute application software. All application s run on a server and the client provides only interface functions.

A problem faced by both clients and servers is the boot process. Many of today's platforms effectively accept firmware updates from anywhere, making an attractive target for subversion. The Trusted Computing Platform Alliance has define a valuable framework for addressing this problem, but the solution must incorporate verifiable protection if it is to counter malicious software attacks [6].

Cryptographic sealing of objects within directories is another example of getting around intractable problems. This technique is analogous to database "guard" architectures [7] that provided limited solutions to the problem of building trusted database systems. The near term potential for a truly secure directory is roughly nil. However if a trusted system seals the objects it places in an insecure directory, then many of the problems associated with insecure directories goes away.

Security engineering on a systems basis is the key to success without having to solve all of the unsolvable problems. Enough tools exist today to solve the major problems that we face today. But these tools are only useful if we have the basic ability to secure data on a somewhat routine basis. Any solution requiring handcrafted solutions is bound to fail largely because there simply is not enough expertise to apply to every site requiring strong security. We need commercially available products, especially appliances that can provide verifiable demonstrable protection for systems.

2.5. Measuring System Security

How do we know if a system offers verifiable protection? If we buy a commercial product (e.g., an Internet Appliance) that offers verified protection, what metric can be used to measure the strength of its security, particularly in the context of an overall system? A simple business metric is whether the use of the appliance to protect against massive loss can be insured, and the cost of the premium per million-dollars of transactions

protected by the appliance. A truly meaningful product evaluation would permit an insurer to quantify the risks enough to issue such a policy. Such an evaluation must have a well-defined systems context.

Evaluation of a subsystem does the insurer little good if protection depends on software outside the target of evaluation. The Trusted Computer System Evaluation Criteria (TCSEC) [4], is a system evaluation criteria. Its Trusted Network Interpretation (TNI) [8] imposes the context of a "network security architecture" that permits components of systems to be individually evaluated in a way that ensures the eventual composition of the overall system will be secure. On the other hand, the Common Criteria [9] provides a framework for evaluations which do not necessarily answer the question "is the system secure". Common Criteria evaluations need not provide a system context and therefore the insurer would have to perform their own systems evaluation.

2.6. Science and Pseudoscience

> *"The time has come," the Walrus said, "To talk of many things: Of shoes and ships and sealing wax, Of cabbages and kings, And why the sea is boiling hot, And whether pigs have wings."[10]*

The audience of this essay is intended to include the student of computer science having an interest in information security. Such students are likely aware that, as commonly used, the term "computer science" can often be closer to "pseudoscience". In the field of computer and network information security, the challenge of separating science from pseudoscience is quite acute. In some cases, computer security pseudoscience operates from a flawed theory. In other cases, as in flying pigs, there is not so much as a working theory to explain a proposed solution.

The terms "science" and pseudoscience are used as per the "Skeptic's Dictionary" [11] at SkepDic.com:

> A pseudoscience is set of ideas based on theories put forth as scientific when they are not scientific. A theory is scientific if and only if it explains a range of empirical phenomena and can be empirically tested in some meaningful way. Scientific testing usually involves deducing empirical predictions from the theory. To be meaningful, such predictions must, at least in theory, be possible to be false. This quality of scientific theories was called falsifiability by Karl Popper. A pseudoscientific theory claims to be scientific, i.e., be falsifiable, but either the theory is not really falsifiable or it has been falsified but its adherents refuse to accept that the theory has been refuted.

Pseudoscience is evident in what historically has been called application security, which is often characterized by not having any clearly defined security policy. Examples include firewalls, intrusion detection systems, and virtually all security offered by Microsoft and similar vendor's products.

Another application of pseudoscience involves strong mechanisms on weak foundations: Examples includes most cryptography products (i.e., PGP, S/MIME, SSL, VPNs, etc.). It also includes techniques such as layering type-enforcement on top of weak operating systems and additions of access control mechanisms to Linux or Free BSD Unix.

Flying pigs are the goal of too much sponsored research, and the recommendation of many blue ribbon panels. For example, a recent pronouncement from the National Research Council [12] suggested that research focus on something called the "three axioms of insecurity" and pursue techniques for making a network more secure than any of its constituent components. The striking lack of any credible working theory is enough to offend the sensibilities of even the pseudoscientist. Indeed "limitations on new research" was one of the loudest complaints stimulated by the establishment of objective criteria and real world tools for handling system composition. Yes, it would be nice if we could make systems more secure than their constituent components. On the other hand, it is hardly new to find a desire to make a silk purse from sows ears – flying or otherwise.

Verifiable protection has progressed and matured to the point where products can be built, incrementally evaluated and used within heterogeneous networks. Some hard problems remain, however the pressing challenges of today are amenable to divide and conquer techniques that leverage the power of verified protection. Notwithstanding that, these advances have not vanquished pseudoscience, which today dictates too much of our allocation of resources for protecting our computer resources. By examining the history of the advances of the science we can perhaps better understand the depths of our decline in the state-of-art of application of science.

2.7. Epochs of Scientific Advances

In the previous section, we described the general problem of separating science from pseudoscience. The plethora of pseudoscientific pursuits in this field far exceeds our scope. Therefore, the remainder of the essay will largely focus on scientific advances with little attention paid to contemporaneous pseudoscientific adventures except where the contrast is instructional.

The following sections provide a historical perspective on how it is we achieved the capability of verified protection and why it is that verified protection provides important solutions to important problems. A set of historical epochs is presented to provide context to the accumulation of scientific knowledge and its

engineering application. This is not a history essay and no claim is made as to its historical completeness.

Figure 3 identifies three epochs in the advancement of the science of computer security, followed by an epoch of decline. Next these are reviewed chronologically.

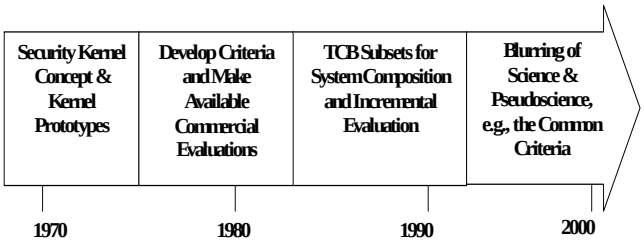

Figure 3: Epochs of Advance Followed by Decline

3. Security Kernel

This epoch culminated in the mid-1970s with the definition, prototyping and fielding of security kernels. Prior to this, secure systems were defined by vendors and aerospace companies as being systems purchased with extra software and functions added to make them secure. Add-on security and applications-level security was promoted by vendors, much as it is today. Since some of the very earliest applications of computers, there has been a recognized critical need for security. Initially this need arose in the military context.

3.1. Multilevel Security

Development of early military systems concluded that some portions of the system require particularly strong security enforcement. Specifically, this enforcement was necessary to protect data whose loss would cause extremely grave damage to the nation. Systems that handled such data, and simultaneously included interfaces and users who were not authorized to access such data, came to be known as "multilevel". Note this term was never intended to mean "hierarchical", it also applied to non-hierarchical, well-defined user groups. The security policy enforced by these systems came to be known as mandatory access controls as introduced in section 2.1, "Verifiable Protection".

One of the first examples of a multilevel system was the SAGE air defense system of the 1950s. Much of the computer security focus was then within the context of "nuclear safety" and had to do with the integrity of guidance data for nuclear-armed anti-aircraft missiles on American soil. Additionally, the system had to protect the secrecy of technical aspects of weapons characteristics, as well as air defense tactics. SAGE supported unclassified clear-text communication interfaces, which meant something had to prevent sensitive information from being disclosed or modified via the unclassified interfaces. The early sixties saw these same issues as part of missile defense systems, which solved some of the problems through use of encryption to protect communications. These early "multilevel" systems can be viewed as deploying applications security with no real allocation of security policy enforcement to distinct components or subsystems.

One of the earliest computer systems specifically designed as a secure system for enforcing a mandatory policy was a Southeast Asia Air Force tactical air control system with a digital interface to NSA intelligence systems for sanitization of data. The system was built with computers specially built in a cleared environment, rather than commercial equipment, because of security considerations. Also, cleared programmers developed all of the software specifically because of the concern of malicious software (e.g., trap doors). In addition, programmers enforced software development practices that were a precursor to the Life-Cycle Assurance requirement of the TCSEC [4] for verified protection. This was one of the earliest systems consciously built and certified as a multilevel system that internally separated data of dramatically different sensitivities.

3.2. Penetrate and Patch

During this time, general purpose ADP often involved aerospace companies wanting to run classified and unclassified processing on the same system. The late sixties saw repeated attempts to achieve the needed security through what became known as "penetrate and patch", which posited that a system could be secured by having experts locate each flaw through penetration testing. Note however that the defender had to find all of the flaws, while the attacker only had to find one. Ultimately the use of a trap door artifice demonstrated the futility of that approach. It is instructive to observe that in many of today's ADP environments, "penetrate and patch" has become "patch and pray", in part because instead of friendly Tiger Teams, many of today's penetrators are hostile hackers.

It was later observed that a failure by a Tiger Team to penetrate a commercial system with add-on security would be extremely damaging to the security of the nation's computer systems [13]. The damage would result from managers falsely concluding that the exercise demonstrated a lack of flaws or trap doors, rather than the proper conclusion that the Tiger Team simply failed to find vulnerabilities that could well have existed.

3.3. Deliberate Design for Evaluation

What was needed was a scientific basis for evaluating the protections offered by a system. The challenge is that you can't evaluate just any software and

hardware. In the general case, verifying software behavior, including security, is often non-computable. In the case of computer security, you must prove a negative, e.g., that the system does not leak sensitive data. To succeed, the system must be specifically designed to be evaluated with the security controls built into the design from the start is a structured manner. Even the earliest SAGE system recognized the need for independent evaluation; in that system, it was driven by the need for nuclear safety. In latter systems, evaluation requirements were driven by the threat of subversion. And, it was recognized that black-box testing for security was useless because it could demonstrate the existence but not the absence of flaws. For example, consider a system having a trap door triggered by a character sequence deliberately designed to not occur in testing. If this triggering sequence were simply a 128-bit key, the universe would die before you finish testing each possibility.

The failure of penetrate and patch to secure ADP systems in the late sixties helped stimulate the Ware Report [14], which represented a codification of the state of understanding, which primarily was a realization of how difficult the problem was. This was one of those points where understanding of concepts came together enough to allow a significant step forward.

The Ware Report [14] clearly identified the problem, but left it unresolved. That led to the Anderson Panel [15], which defined the reference monitor concepts and conceived a program for evaluating and developing kernels. SCOMP [16] was the first commercial result of that immediate effort. Another result was security enhancements to Multics [17]. Both of these had security built into them, but it was recognized that they responded to two different levels of threat.

At about this same time frame, the WWMCCS system was built upon a commercial operating system as an attempt to add security onto a system that was not structurally designed to support security. While WWMCCS had substantial security requirements, it had no fundamental design requirement for the system to be designed from the start to be secure, let alone be evaluatable. It was built by layering applications upon a commercial operating system, and depending on the weak commercial OS security controls. This doomed WWMCCS to a long series of security problems.

AUTODIN, which was the government computerized messaging system for several decades, had definitive design requirements to address security. Even though AUTODIN was built on a commercial machine executive, the executive itself was quite small and primitive. The remainder of the system was built by cleared programmers to a set of security-driven specifications.

The NSA Southeast Asia sanitization system identified above was built substantially around the crypto development model and therefore had development criteria intended to control insertion of trap doors and ensure that the result could to some degree be evaluated.

And while not a true security kernel, Multics incorporated many of the requisite protection mechanisms and separated secret from top-secret information for fifteen years in a critical Pentagon facility without an operational compromise.

3.4. Discretionary and Mandatory Policies

A key scientific underpinning identified during this first epoch is the distinction between discretionary and mandatory security policies. Glimpses of this distinction appeared in the Ware Report [14] of 1969 and in the ADEPT-50 [18] system of around the same time frame. The distinction first appeared in general literature in the mid seventies [19].

By the time of WWMCCS and Multics, there were clear distinctions being made between mandatory access control policies (MAC), discretionary access control policies (DAC) and application policies. And it became clear that only MAC offered verifiable statements about information flow, which requires that the information be assigned global and persistent access classes in the form of a mathematical lattice [20]. This “labeling” of information supports both hierarchical and non-hierarchical relationships between access classes.

3.5. Formal Security Policy Models

As independent parallel efforts, Case Western and MITRE developed formal security policy models. The latter became more generally accepted primarily because its elements more directly mapped to physical computing components (e.g., memory descriptors). So, while the mandatory access controls of the Multics system were built to the Case Western model, an after-the-fact interpretation of the Bell and LaPadula model was generated [[21]]. This was possible because both models were sound math encompassing common security policies. It was found that these formal models could be applied to problems of information integrity as well as to maintaining the secrecy of information [22]. It was later shown that implementations of these models could bring verified protection to security policies thought to be suitable for commercial and other data processing requirements [23].

It became evident that successfully achieving verified protection would require sound math. However, two key security model issues remain a source of misunderstanding to this day, leading to some failed attempts to apply formal models to trusted systems. First, the security model must be a valid representation of the behavior with respect to information protection of the

entire *system*. Second, the model must include a proven security theorem, which establishes that the model's behavior always complies with the security requirements for the policy of interest rather than being a mere formalization of mechanism [24].

3.6. Key Hardware Platform Properties

Several different hardware platforms were the target of kernel prototyping efforts. They each had common properties, however, including memory segmentation and at least three hardware states for use in implementing protection rings (as illustrated in Figure 4): One for the security kernel; one for the operating system; and one for applications. The primary hardware modification that enabled SCOMP was the addition of support for four hardware rings and segmentation.

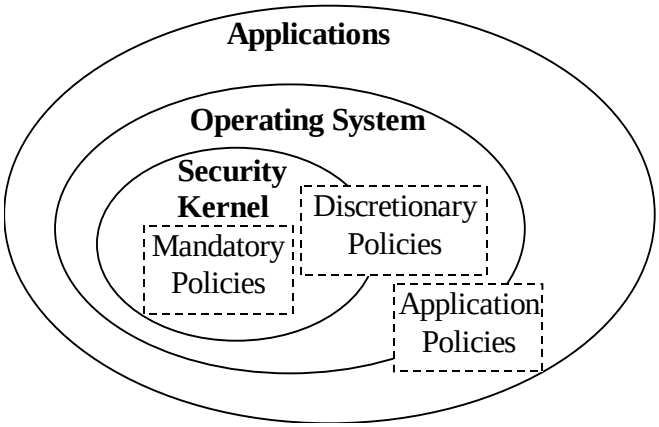

Figure 4: Hardware Protection Rings

Mandatory access control policies are enforced by the security kernel in the most protected ring. Discretionary access control policies can also be enforced by the security kernel, or can be enforced by the OS depending on assurance requirements. Application policies are generally enforced by the applications themselves or by the operating system. In fact, the separation of the operating system from the application programs is itself an application policy enforced by the operating system.

3.7. Advances in the State of the Science

This epoch culminating in the prototyping and fielding of security kernels contributed the following key advances to the state of the science of computer and network security:

- Reference Monitor Concept
- A Simple Security Kernel
- Formal Security Policy Models
- Discretionary vs. Mandatory Access Control Policies
- Hardware Rings and Segmentation

4. Available Commercial Evaluations

The next epoch culminated in an ability to procure a commercial system that is demonstrably secure with respect to some security policy. The key to a "demonstrably secure" system is a scientifically based evaluation. The need to independently evaluate the ability of a system to enforce a given security policy was a recognized problem, even during the previous epoch when security-critical systems were largely built in controlled environments by cleared programmers. The need to evaluate systems becomes more acute when standard commercial products are used, because there is no assurance that they do not contain elements obtained from an outside party who may in fact be hostile.

Note that the focus of this epoch was the ability to obtain evaluated *systems*. An ability to obtain evaluated subsystems is not in and of itself sufficient unless the attacker is willing to agree to attack only the subsystems of your choosing – obviously an unreasonable constraint.

4.1. How to Trust What You Don't Build

Multics was influential in generating interest in commercial evaluations because Multics was a commercial product. Before that, attempts to field secure systems built on commercial security products were characterized by rounds of penetrate and patch, with emphatic assertion being the vendor's argument of choice to explain a systems basis for security.

Amid the increased interest in an ability to evaluate commercial products during the mid seventies, the Air Force and ARPA sponsored MIT to develop auditable systems [25]. This effort concluded that building auditable systems required certain forms of hardware support. Previously, there did not exist many construction techniques for building systems whose security could be assessed after-the-fact. There were ad-hoc techniques and design standards, but little in the way of reproducible theory on how to architect such a system.

The MIT project recognized this as a hard problem and identified a number of solutions components. This effort articulated the notion of layering software modules previously identified by Parnas [26]. This effort also solved challenges related to process management, resulting in the definition of a two-level scheduler. Issues of information flow associated with inter-process communication were a serious challenge, with the contemporary strategies of message passing and P&V semaphores not suitable for the task. This resulted in the invention of eventcounts and sequencers [27] for secure inter-process communication.

The MIT project succeeded in offering solutions to all of the hard problems that stood in the way of building auditable operating systems.

4.2. Formal Methods and Verification

A major step that enables a independent evaluation of products was the development of formal methods [28]. At one point, there were three tools approved for use by the NCSC: FDM, HDM, Gypsy. One of the most significant developments was the Fiertag flow tool [29]. that permitted assessment of implications of design decisions at the interface to a virtual implementation of the system

One driver that led to advances in formal methods was the recognition that Trojan Horses could exploit covert channels. The risk of malicious software violating the system security policy was recognized from the very early secure systems, leading most systems to be developed by cleared programmers. Development of systems (e.g., commercial off the shelf products) by uncontrolled sources (e.g., potential enemies) requires a level of rigor only available with formal methods.

Also during this time frame, limitations on the use of cryptography were becoming more widely known. And, again it is the potential for malicious software that causes these limitations. An example is an exploitable covert channel [30] that exists in end-to-end encryption systems. Trojan Horses can exploit this limitation by modulating large amounts of data within message headers.

Efforts to ignore this problem represented one of the earlier examples of people trying to defend their designs by arbitrarily constraining what an attacker is supposedly permitted to attack. The crypto designers said their system was not considered responsible for this problem. Nor was the operating system subverted to exploit the problem. Yet it was a *system* vulnerability a Trojan horse could exploit.

While covert timing channels remain a challenge to this day, the GEMSOS security kernel showed that covert storage channels could be eliminated and that covert timing channels could be significantly reduced [31].

4.3. Evaluation Classes for Insecure Systems

It is instructive to note that the initial efforts, such as the MIT project, to construct auditable kernels all focused on being "secure", not on being secure with some variable level of assurance. Initially, the efforts to develop system evaluation criteria were synonymous with what latter became called Class A1. The reason is that initially, no assumptions were made about the methods of the attacker. Only latter was it asked if there were any value in defining lesser levels of assurance. Division A of the TCSEC makes no assumptions about limiting the attacker. Division B hypothesizes that the attacker can subvert the applications, but not the operating system. Division C hypothesizes that the attacker uses no subversion at all. And, division D assumes that customers believe that attackers believe the vendor marketing claims.

4.4. Advances in the State of the Science

The epoch culminating in the availability of commercial evaluations (based on the TCSEC) contributed the following key advances to the state of the science of computer and network security:

- Application of Formal Methods
- Architectural Requirements for Evaluation (e.g., layering, least privilege, minimization)
- Covert Channel Reduction and Analysis
- Technically Sound Objective Evaluation Criteria
- Eventcounts for secure synchronization

5. TCB Subset Tools for Composition

Commercial evaluations under the TCSEC made available trusted *systems* whose security properties were independently evaluated. At the time, many inaccurately claimed that use of the TCSEC was limited because it could not be applied to a network of computers. The TCSEC is a complete and reasonable criteria for evaluating systems, including networks as noted in [32] and [33]. However, there were two real-world concerns that motivated published interpretations of the TCSEC:

- The ability to enforce a variety of system security policies at varying levels of assurance; and,
- The ability to incrementally evaluate networks and systems based on well defined modifications (e.g., the addition of a new sub-network).

5.1. Virtual Machines and the Subsetting Problem

Virtual machines such as IBM VM370 were developed to permit the strong partitioning of systems within the same computer, e.g., allow simultaneous development and testing of new operating systems on the same physical machines as the production application environment. These efforts showed that to succeed, the policies must be layered and there must be a partial ordering to the policies. For example, outer layer policies (e.g., discretionary controls on access to data) should not be able to reach in and affect inner layer policies (e.g., separating the production virtual machine from the development virtual machine).

The short early concept papers on trusted VMs and trusted databases recognized that there was a ordering to policies where relatively weak mechanisms could be used

to enforce those policies that have smaller demands placed upon them. This was clearly evident in the VAX VMM Security Kernel [34]. This work provided a formal structure for understanding what previously was not describable. These lessons were applied in the development of the TNI [8] and the understanding of TCB Subsets [35].

5.2. Incremental Evaluation of Distinct Physical Components

A strong driver to the development of the TNI was the existence of continually evolving networks composed of heterogeneous components. It was not practical to require that an entire system be evaluated at once, so strategies were investigated for supporting incremental evaluations. The question was: how to build a system from a set of individual components that were previously evaluated? The answer is found in two key concepts: 1) the concept of a "partitioned TCB", in which individually evaluated components interact in a precisely defined fashion and, 2) a "network security architecture" that addresses the overall network security policy. These concepts enable architectures to evolve without having to go back and reassess the role of each individual component each time a deployment consistent with the architecture is changed. This also led to an ability to recursively "compose" a set of individual components into a new single logical component that has a well-defined role within the network security architecture and a well understood composite security rating.

A litmus test of the practicality of the TNI was whether it could be applied to the Blacker system – a NSA developed Virtual Private Network (VPN) for securing highly sensitive traffic over an insecure Internet [36]. The ability to use products not custom made for a specific network was illustrated by having two major components of Blacker hosted on the commercial GEMSOS security kernel that met the requirements for verified protection [31]. It was concluded that Blacker would be a practical application of the TNI. The notion of a "guard" based on cryptographic checksums [7] was another technique for using insecure components in a secure system.

As another example of the practical use of the TNI, the system architect for the ICL-developed CHOTS system for the UK military reported making significant use of the TNI as an engineering tool in architecting their network of heterogeneous components, and in allocating security policies to the various components. Similarly, Novell confirmed its practicality as a powerful tool in the design of security for a commercial network [5].

5.3. TCB Subsets Within a Single System

The lessons learned from development of the TNI and from the SeaView multilevel DBMS security model [37]were then applied to the initial drafting of the TDI [38] to address the management of TCB subsets within a single physical computer.

Trusted Oracle was structured to exploit the properties of TCB subsetting. It included a mode whereby the mandatory access controls of the database were enforced by the mechanisms of the underlying operating system. This "evaluation by parts" solved the seemingly difficult problem of achieving a Class A1 database system when neither the database vendor nor the operating system vendor was willing to provide the other with the evaluation evidence that would be necessary for a single system evaluation.

Evaluation of Novell's commercial NetWare network under the TNI marks the end of this third epoch. Novell desired an evaluated system, yet was not in the business of building clients. They faced the question of how to specify the role of the client such that other vendors could produce clients that would be secure within the Novell network security architecture. They implemented a TCB extension for their client. Novell completed three distinct but related evaluations: client; server; and network [5].

5.4. Advances in the State of the Science

The epoch culminating in the ability to use TCB subsets to compose and incrementally evaluate systems contributed the following key advances to the state of the science of computer and network security:

- Partitioned TCB
- TCB Subsets
- Rules for Layering Security Policies
- Rules for Composing Systems
- Balanced Assurance
- Cryptographic Checksum Guards
- Multilevel DBMS Data Model
- Secure Client via TCB Extension

6. Common Criteria

The introduction of the Common Criteria delimits the end of our final epoch. While this occurred some years ago, our current situation is dominantly a simple extrapolation from that point.

Although scientific advances had led to a system evaluation criteria, worked examples and engineering tools for composing systems, the building of secure computing systems remained a challenging endeavor requiring significant effort by trained practitioners. Science had provided no way to glom security onto an

existing system. And even though there has been wishful thinking that it would be nice to discover a means of "building trustworthy systems from untrustworthy components" [12], to the current state of science this appears to be intractable. The blurring of distinctions reflected in the Common Criteria provided a vehicle for renewing research into speculative strategies for achieving trusted systems, and opened the door to our current epoch of pseudoscience, emphatic assertion and unconscionable neglect.

A fundamental goal of the TCSEC was that an evaluation must identify the properties of the overall system without having to make system-specific assumptions. There were separate evaluations for "sub-systems" that were called "sub-system evaluations".

The Common Criteria has led to a number of evaluated products, but a dearth of evaluatable *systems*. That is because there is no prescribed distinction between system evaluation and subsystem evaluations. About 20 different network client products were evaluated under the ITSEC (which similarly lacks a systems context), but what do any of the evaluations mean with arbitrary assumptions about the behavior of other parts of the system? In fact been an entire business was pursued based on claimed existence of a "Class A1 chip", apparently inferred from an "EAL7 chip", which said virtually nothing about the security of any system.

We are not in any way saying that this blurring of distinctions is caused by the Common Criteria. In fact, it is not clear that it is having much impact at all on security designs and implementations. We are simply noting that during this epoch many of the distinctive properties of the science are not commonly recognized or applied.

The current failure to apply the existing science of information security is largely the result of three tendencies of pseudoscience:

- A willingness to make baseless assumptions about the behavior of "other" software subsystems, i.e., those falling outside the "target of evaluation"; and,
- A willingness to assume unenforceable prescriptions on the behavior of attackers.
- The classic logic error of assuming all problems are the same and then concluding that certain techniques don't solve any of the problem because they fail to solve some of the problems. For example, because verifiable protection does not fully solve the problems of denial-of-service, some will overlook the fact that verifiable protection permits the procurement of a secure operating system from a mortal enemy

The Common Criteria tends to remove the distinctions between pseudoscience and science, because it has no inherent mechanism for distinguishing system and subsystem evaluations. This leaves the procurers of secure systems at the mercy of an increasingly shallow pool of practitioners of computer security.

This last epoch is not a total loss. Cryptographic subsystems and related standards and protocols have undergone considerable advancement.

6.1. Available Cryptography

DES has grown to triple DES. And the Advanced Cryptography Standard (ACS) holds significant promise. However where once DES was anointed adequate to protect sensitive by unclassified information, there is a troubling lack of such pronouncements for the ACS. Most troubling however, is that most cryptography products are built on platforms that lack a basis for believing their security mechanism cannot be subverted

Today, there is significantly more access to cryptographic algorithms, hardware and turnkey encryption systems than there was twenty years ago. Export restrictions have been greatly relaxed and the government is not knocking on as many doors to respectfully request manufactures to pull products from the market. Almost twenty years ago a large chip vender was encouraged to stop building a quality key generator, leading one manufacturer to replace the use of that chip with the manual rolling of eight sided dice. Since then, the engineering of security products that incorporate cryptography has become considerably easier. On the other hand, the understanding of probing cryptographic solutions for weaknesses has also advanced considerably, with techniques like power consumption analysis threatening many implementations.

Cryptography has become a relatively popular area of study, and new algorithms are subject to systematic review – which is good because there is no real science to cryptography. It has few absolute metrics, relying instead on assertions of relative strength such as: "it is as difficult to break as factoring large numbers to their primes".

The application of cryptography has been greatly enhanced by standards such as IPSEC which can make the engineering of VPNs a whole lot more likely to be sound. Even standards like SSL have usefulness in environments that don't require high assurance.

Cryptography advances have indeed been considerable. However, current products are mostly built on platforms having weak security. And, the recent neglect of the science of computer and network security suggests this won't be corrected soon. As a result of this major weakness in cryptographic deployment, it has in many cases become what has been referred to as the "opiate of the naive".

This is not a unique insight of the author, but has been noted by others as well. For example, with respect to security for modern servers and PCs one long-term expert recently noted [39]:

However, it would appear that we have solved the wrong problem. While encryption codes might represent an intellectual challenge to hackers and criminal decrypters, they invariably choose the more realistic and achievable approach of attacking servers and PCs, the Achilles heel of the network environment.

The irony is that some of our very early PC chips were designed to provide a higher degree of security, but this has never been made available.

In the early 1980s, Intel developed a secure chip called the Intel 286 chip which formed the heart of a secure system known as GEMSOS, one of the most highly trusted computer systems ever built and used by the US Government.

6.2. Advances in the State of the Science

The epoch of decline delimited by the publishing of the Common Criteria did not include much in the way of advances in the state of the science of computer and network security. However, there were considerable advances in the area of available cryptographic tools, including:

- Widely Available Crypto Subsystems
- Digital Certificates and PKI
- IPSEC for VPNs

7. Summary

The state of the science of information security is quite rich with solutions and tools that represent the accumulated knowledge from research over more than 30 years. The state of our assimilation of that knowledge by information security practitioners and understanding of the existing science is very poor. The greatest achievement in the science of computer and network security is the ability to build and deploy truly bulletproof systems having verifiable protection. And this remains the most powerful solution available for many of today's hard problems.

The following list summarizes advances yielding the state of the science of computer and network security.

- Reference Monitor Concept
- A Simple Security Kernel
- Formal Security Policy Models
- Discretionary vs. Mandatory Access Control Policies
- Hardware Rings and Segmentation
- Application of Formal Methods
- Architectural Requirements for Evaluation (e.g., layering, least privilege, minimization)
- Covert Channel Reduction and Analysis
- Technically Sound Objective Evaluation Criteria
- Eventcounts for secure synchronization
- Partitioned TCB
- TCB Subsets
- Rules for Layering Security Policies
- Rules for Composing Systems
- Balanced Assurance
- Cryptographic Checksum Guards
- Multilevel DBMS Data Model
- Secure Client via TCB Extension
- Widely Available Crypto Subsystems
- Digital Certificates and PKI
- IPSEC for VPNs

8. Acknowledgement

The author is deeply grateful to Michael F. Thompson of Aesec Corporation for his most helpful research and review during preparation of this essay.

9. References

[1] Jelen, George F., *Information Security: An Elusive Goal*, *Program on Information Resources Policy*, Harvard University, Cambridge, MA, June 1985.

[2] Brinkley, D.L. and Schell, R.R., "Concepts and Terminology for Computer Security", *Information Security*, M. Abrams, S.Jajodia, and H.Podell, eds., IEEE Computer Society Press, Los Alamitos, Calif., 1995

[3] Jueneman, R.R., *Novell Certificate Extension Attributes—Novell Security Attributes, Tutorial and Detailed Design*, Document Version 0.998, Novell Inc., August 1998.

[4] *Department of Defense Trusted Computer Security Evaluation Criteria*, DOD 5200.28-STD, National Computer Security Center, December 1985.

[5] *Final Evaluation Report, Novell, Incorporated Netware 4.1.1 Server*, National Computer Security Center, June 1998.

[6] Schell, R.R. and Thompson, M.F., "Platform Security: What is Lacking?" Information Security Technical Report, Vol 5, No. 1, 2000, Elsevier Advanced Technology

[7] Denning, D.E., "Cryptographic Checksums for Multilevel Database Security". *Proc. 1984 Symp. on Security and Privacy*, IEEE Computer Society, 1984, pp. 52-61.

[8] *Trusted Network Interpretation of Trusted Computer System Evaluation Criteria*, NCSC-TG-005, National Computer Security Center Version 1, 31 July 1987.

[9] *Common Criteria for Information Technology Security Evaluation*, CCIB-98-026, ISO/IEC, May 1998

[10] Carroll, Lewis, *Through the Looking Glass*, (1872), English writer, mathematician.

[11] Carroll, R.T., *The Skeptics Dictionary*, http://skepdic.com, 1994-2001

[12] *Trust In Cyberspace*, National Academy Press, Washington, D.C. 1998, Fred B. Schneider, Editor

[13] R.R. Schell, "Computer Security: the Achilles' Heel of the Electronic Air Force", *Air University Review*, Vol. 30:2, Jan.-Feb., 1979

[14] Ware, W. H., ed., *Security Controls for Computer Systems: Report of Defense Science Board Task Force on Computer Security*, AD # A076617/0, Rand Corporation, Santa Monica, Calif., February 1970, reissued October 1979.

[15] Anderson, J.P. *Computer Security Technology Planning Study*. ESD-TR-73-51. Bedford MA: USAF Electronics Systems Division. October 1972.

[16] Fraim, L.J., "SCOMP: A Solution to the Multilevel Security Problem," *IEEE Computer*, July 1983.

[17] Whitmore, J.C. et. al., "*Design for Multics Security Enhancements*," ESD-TR-74-176, Honeywell Information Systems, 1974.

[18] Weissman, C., "Security Controls in the ADEPT-50 Time-Sharing System," *Fall Joint Computer Conference*, 1969

[19] J.H. Saltzer and M.R. Schroeder. "The Protection of Information in Computer Systems," *Proc. IEEE* Vol. 63, No. 9, September 1975.

[20] Denning, D.E. "A Lattice Model of Secure Information Flow." *Communications of the ACM*, Vol 19. No 5, May 1976.

[21] Bell, D.E. and LaPadula, L.J., *Computer Security Model: Unified Exposition and Multics Interpretation*, ESD-TR-75-306. MITRE Corporation, Bedford MA, June 1975.

[22] Biba, K.J., *Integrity Considerations for Secure Computer Systems*, ESD-TR-76-372, MITRE Corporation, Bedford, MA, April 1977.

[23] Shockley, W.R., "Implementing the Clark/Wilson Integrity Policy Using Current Technology," *Proc. 11th National Computer Security Conference*, 1988.

[24] Ames, S.R., et al, "Security Kernel Design and Implementation: An Introduction", *Computer*, IEEE, July 1983.

[25] Michael D. Schroeder, David D. Clark, and Jerome H. Saltzer, "The Multics Kernel Design Project", *Proc. Sixth ACM Symposium on Operating System Principles*, November 1977.

[26] Parnas, D.L., "A Technique for Software Module Specification with Examples," *Communications of the ACM, Vol. 13, No. 5*, May, 1972

[27] D.P. Reed, and R.K. Kanodia, "Synchronization with Eventcounts and Sequencers," *Communications of the ACM*, Vol.22, No. 2, February 1979.

[28] Millen, J.K. "Security Kernel Validation in Practice." *Communications of the ACM*. Vol 19. No 5, May 1976.

[29] Feirtag, R.J. et al., "Proving Multilevel Security of a System Design, *Proc. Sixth ACM Symposium on Operating Systems Principles*, November 1977.

[30] Padlipsky, M. A., Snow, D. P., and Karger, P. A., *Limitations of End-to-End Encryption in Secure Computer Networks*, The MITRE Corporation, MTR-3592, Vol. I, May 1978 (ESD TR 78-158, DTIC AD A059221).

[31] *Final Evaluation Report, Gemini Network Processor, Version 1.01*, National Computer Security Center, pp 123-124, June 1995.

[32] W.R. Shockley, et al. "A Network of Trusted Systems". *Proc. AIAA/ASIS/IEEE Third Aerospace Computer Security Conf*., 1987.

[33] Fellows J., et al. "The Architecture of a Distributed Trusted Computing Base." *Proc. 10th National Computer Security Conference*, September 1987.

[34] Karger, P. A., et al., "A Retrospective of the VAX VMM Security Kernel," *IEEE Transactions on Software Engineering*, Vol. 17, No. 11, Nov. 1991.

[35] Shockley, W.R. and Schell, R.R., "TCB Subsets for Incremental Evaluation", *Proc. of the 3rd Aerospace Computer Security Conference*, American Institute of Aeronautics and Astronautics, Washington D.C., 1987.

[36] C. Weissman, "Blacker: Security for the DDN. Examples of A1 Security Engineering Trades". *Proc. 1992 Symp.Research in Security and Privacy*, IEEE Computer Society, 1992, pp. 286-292.

[37] Lunt, T. F., et al., "The SeaView Security Model," *IEEE Transaction on Software Engineering*, Vol. 16, No. 6, June 1990.

[38] *Trusted Database Management System Interpretation of the Trusted Computer System Evaluation Criteria*, NCSC-TG-021, National Computer Security Center, April 1991.

[39] Bill Caelli, "Bring in E-Trading PIN Pads," *The Australian* newspaper, Sydney, Australia, 24 October 2000.