Improving Computing Security during the Development of DOD Computerized Weapons Platforms

Sam Nitzberg
Telos Information Protection Solutions
656 Shrewsbury Ave.
Shrewsbury, NJ 07702
USA

Mike Carson
Telos Information Protection Solutions
656 Shrewsbury Ave.
Shrewsbury, NJ 07702
USA

I Introduction

While there are a multitude of programming and development standards in place for designing and building government software systems, there are few specific guidelines and special processes in place designed to assure the security and integrity of DOD weapons platforms. Such platforms include C2 (Command and Control) and C3 (Computers, Command, and Control) systems. Guidelines such as AR 380-19 do exist but remain eternally out of date when the methods to defeat their mandated security measures are readily available.

The first security issues surrounding computers and computerized weapons systems were first and foremost security issues. Three "eras" indicative of the growing role and coupling of computers and weapons are typified by the following systems:

- ENIAC - physical security
- SAGE - Networked, Centralized Weapons / Stations
- Current Systems - Modern Strategic and Tactical - Increasing decentralization, with Commercial, off-the-shelf (COTS) components.

Systems such as the ENIAC performed their ballistics calculations locally, and thus their systems and data were effectively secured through the use of conventional physical security practices. The next grand era was typified by the introduction of large, sophisticated systems such as the SAGE. This system included a number of facilities with secured sophisticated, centralized computer systems, with connected access terminals. The centralized computers were linked via secure channels. Such systems could effectively be protected through the proper application of source code review, physical security, and communications security based mechanisms. Unfortunately, the current challenges posed by those wishing to build and presumably secure modern weapons systems are not as simply resolved.

Whereas once there were stark distinctions between the platforms to address civilian and military computing requirements, these distinctions are quickly fading away. Modern weapons platforms are typically based on COTS (Commercial Off-the-Shelf) software and hardware systems. These components include network cards, routers, communications switches, computers, computer languages, and operating systems. And whereas once there was great debate over whether someday commercial systems should be utilized, today it is a given that most software subsystems are either COTS products or built on COTS products. What must be kept in mind is that these systems are often built with relatively stable runtime environments in mind, not those of punishing military conflict. Weapons platforms are truly mission-critical, and are to be built with the presumption of their physical, disorderly destruction (or even worse - attempted subversion) while engaged in military operations.

Naturally, the level of determination and funding of an adversary is typically higher with military systems than with civilian systems. It is evident that there are a great many potential avenues by which great

harm may be done to a society by attacking its civilian infrastructure through the use of Information Warfare techniques. However, the average hacker is unlikely to have the steely determination and backing of a significant power interested in penetrating DOD systems engaged in active warfighting efforts, especially when such individuals would be subject to espionage charges, and potentially involving themselves in life-long and life-threatening situations. Naturally, such adversaries are likely to have financial and other backing from hostile world powers, including aggressive funding and cultivation of resources not typically seen in the "Hacker Culture." One example of such sophisticated backing and support provided to undermine systems is recounted by Ira Winkler in the factual tale of Mike, Peter, and Vasya, where sets of commands to undermine systems for military projects were prepared by Soviet Intelligence to be easily invoked by an American in their employ. Protective measures must be in place to defend warfighting and strategic systems from those with the determination and capability to effect their wills - to undermine the security of America's military infrastructure. One of the most efficient vehicles for effecting their will may very well be through corrupting the ability to control sophisticated systems, and thereby subvert control of the battlespace.

While DOD's development efforts are very adept at ensuring that software systems, and weapons systems, consisting of complex integrations of both hardware and software components, meet functional requirements, effective security requirements are often seemingly overlooked. Often, procedural security mechanisms are established in order to effect what is best provided by measures implemented in the systems proper. Government documents offered to assist in providing guidelines and mandates in vital elements of security engineering, such as AR 380-19, may be sound for a moment in time, but unless such documents are "living documents," being updated to address new security threats and concepts as they emerge, can quickly become outdated, ancient checklists, which can readily provide a false sense of security or fulfillment of mission. When the goals of these documents conflict with those of software development teams, there is also the tendency to have waivers produced to exclude the system requirement from meeting the mandated security requirements. Without tracking these waivers, or mechanisms in-place to monitor the ongoing security posture of these systems, new vulnerabilities may be readily exploited by those with specialized knowledge. The incredible rate at which computer systems exploits are shared in the free world, and propagated on the World Wide Web have even produced a new sort of threat from so-called "script kiddies," individuals who on their own would be incapable of exploiting systems, but who obtain scripts (computer programs) from the Internet with which they may exploit systems at will.

II Software Development Methods and Models

Systems designed and built for the DOD have high-level systems requirement documents which serve to define the needs of the system in a manner which may be further formalized. The systems specification documents in turn serve that purpose, to provide specific, testable capabilities which may be used as a guide in constructing and testing the actual system. All too often, the security requirements which do appear in the requirements specification and the accompanying systems specification documents are limited to obvious systems elements associated with the separation of data into sensitivity levels, and procedural issues. Often neglected are the subtler, more insidious issues, such as how to address presently known security vulnerabilities which may be used to exploit the system, and new vulnerabilities which will occur once the system is fielded. The most important vulnerabilities may even include those whose very nature has not yet been discovered for the systems in question, and will be realized in the very near future.

An advantage of aggressively integrating these security issues early in the software development process is increased systems reliability, reduced costs, increased likelihood of meeting deadlines, and an increased likelihood of the system meeting its mission objectives. Since a great many security vulnerabilities intrinsic to operating systems and platforms can be incorporated early in the software development process, costs associated with these changes are much lower than if these need to be performed as a separate effort late in the software development cycle, especially if there is a risk that these oft-hurried later changes may "break" the software, and a fear is present of making any last-minute changes. If all

hardware and software used in the software development effort is selected not only to meet the traditional security requirements, but the assorted security needs of the project, this can avoid very unpleasant surprises late in the development cycle, which may necessitate financially and temporally costly network re-engineering efforts. Coalescing these efforts early in design and analysis will assist in developing a more stable system. Benefits from this added stability may be enjoyed early in development as all incorporated security fixes may be tested from early in the system development stages, and any detrimental consequences to the system noted as regressive systems testing is performed as a normal function of system development. This will reduce last-minute mandates for hardware or software systems changes to fix problems which could have been readily resolved early, at reasonable cost, and without occurring at the tail of a software development project where there may be pressures to expedite systems delivery. There are a number of fundamental studies which indicate the great savings offered through early adoption of accurate and comprehensive requirements, specification, and design. Of particular need are studies revealing the savings associated with producing and incorporating more complete security requirements, specifications, and designs into contemporary projects.

Ideally, a system is immune to all known vulnerabilities when it is ready to be migrated off to Post Deployment Software Support (PDSS). In this case, what is required to maintain a secure system is an active capability and organization with which to identify new vulnerabilities as they become known, as well as an understanding of the capabilities of foreign powers with respect to undermining computer systems security. Besides these capabilities, effective use must be made of the produced intelligence. When systems vulnerabilities are identified, there must be cost and manpower efficient mechanisms available through which the repairs may be effected and tested. Due to the great expense and manpower often demanded in performing systems testing, the expanded use of sophisticated automatic software testing tools would be of great benefit.

III Nature of the Threat

While no organization claims to know the exact number of attacks that have been launched against DOD computer systems, present estimates are in the hundreds of thousands annually. Brock Meeks, a journalist who presented the keynote address at Beyond Hope, a hackers' conference, addressed the issue of the number of hacker penetrations against DOD systems. While he could not condone the majority of hacking activity ("What part of jail don't you understand?"), and advocates hacking as providing checks-and-balances in cyberspace, he chastised the hackers for having attacked only an estimated twenty percent of government systems, and urged them to increase their batting average. It is common knowledge that many hackers consider ".mil" sites to be their most worthy targets.

Computerized weapons programs for the DOD are initiated to solve specific problems in facilitating high speed, efficient communications, managing and manipulating large, complex data sets, and enabling unit commanders to penetrate the "fog of war." Properly handling these problems in an integrated, flexible manner requires the integration of multiple, complex, interrelated systems. Large scale DOD systems rival the largest commercial systems in terms of the volume and intricacy of the information they process.

Modern DOD computer systems implement Commercial Off The Shelf (COTS) software wherever possible, to save costs, reduce development time, and benefit from the extensive design and application experience that goes into commercial products. However, not all of the military's needs can be met by commercial products, and thus a variety of custom military hardware and software products are developed to meet the specific requirements, resulting in a complex "soup" of integrated products seen, literally, in no other class of computing environments extant. COTS products are typically not designed with security in mind; they certainly are not designed with security features meeting the absolute needs demanded in truly mission-critical military applications, leading to a situation where special attention must be paid to secure a system where the underlying infrastructure can not be trusted. Further complicating matters, these subsystems, both military and civilian, interact in complex and sometimes unpredictable ways. Such awkward marriages of systems components necessitate analysis on the macro as well as micro levels.

As systems integrate multiple, large, complex components, individual subsystems will necessarily undergo changes in design and implementation caused by a variety of factors, including errors discovered during testing as well as revised releases of COTS products. This "version drift" may have unexpected security implications. Situations can easily occur where no one member of the development team has a clear understanding of the behavior of the entire system, and changes to the system configuration may readily cause unintended privileges to become available to system users or outsiders.

Version drift is not limited to software. Availability of hardware components, as well as changing requirements and the procurement environment can drive changes to systems hardware at any point in the system development cycle. Even seemingly minor changes can have deleterious effects, or drive software changes with far reaching implications.

Security budgetary allowances for military platforms may be determined by in-house project or contractor rates. Typical analysis for security budgeting is based on considered trade-offs between potential damage and liability in case of failure, the perceived (or better yet, modeled) likelihood of failure of security protocols, and the gains offered by success of the mission or action being so examined. In some cases, spending allowances may be arbitrarily dictated based on programmatic concerns.

Discovering a hacker intrusion into a system is complicated by several factors. First, most system administration personnel consider an attack on their facilities unlikely, making them slow to notice, accept, and respond to evidence of an intrusion - if personnel even take adequate steps to identify any potential penetration. A successful attack often leads to compromise of sufficiently privileged accounts that audit logs are compromised, disguising the breach. Beyond these problems is the simple fact that system intrusion is often indistinguishable from system failure. Especially in the case of Denial Of Service (DOS) attacks, which often issue errors and cause symptoms similar to well known and far more common system failures, attacks can occur often before the specter of hostile action is even raised. Even DOS attacks that do not mimic normal errors are likely to be treated as "weird" errors rather than hostile actions by most inexperienced personnel.

DOS attacks suppress a required system function, interfering with proper systems operations. A recent rash of such attacks, bearing names like SYNC flood, boink, smurf, pepsi, land and teardrop attack the protocol or implementation of the TCP/IP networking stack through which most systems communicate. Their effects range from temporarily halting communication to selected services accessed over the network to completely halting the target system. DOS attacks may also be triggered locally. For example, the Pentium FOOF bug may be so invoked, causing the local system to halt. DOS attacks can be extremely difficult to detect, as they often leave no audit trail, and may emulate random system failures.

The damages inflicted by DOS attacks pale in comparison to the losses that can result from compromising system accounts. Minimally, if a legitimate user subverts another user's account, that user may launch attacks on accounts with higher privilege with relative impunity, and access the victim's data. An outsider that gains access to a non-privileged user account can cause local denial of service attacks, attempt to gain access to privileged accounts and compromise user or application-accessible data on the system. Of course, the ultimate goal of attacking strategic weapons platforms is to gain privileged systems access, and effectively command the entire platform.

In a military environment, the consequences of a non-authorized party gaining access to a system can be extreme. The highlights include compromise of classified data, loss of life, and failure of the mission. To address the threats in conventional systems, trusted systems, using provably secure kernels and data channels, such as SCOMP, have been produced, and are subject to most severe design and analysis constraints. These systems are typically proprietary, very expensive, and run a very limited range of software. Systems such as System V MLS do enforce multilevel military security models consisting of subjects and objects, and offer reduced costs when compared to more rigidly defined trusted systems, while affording protections not available in most "Off-the-Shelf" operating systems. For example, MLS systems would not allow Trojan Horses to compromise access or data, since programs and their descendants are

restricted in privilege. One shortcoming of conventional systems is worth noting here. They may not have a formal memory reuse policy, which may lead to disclosure of sensitive operational data.

The UNIX operating system was first implemented in 1970 by Ken Thompson on a DEC PDP-7. It was designed to be a modular, scaleable and portable operating system. UNIX was designed to provide a high level of abstraction from the hardware on which it was implemented, so that applications written for it could readily be ported to other computer systems. It was, however, not designed with significant security in mind.

UNIX is known for providing an extremely flexible, powerful and consistent interface for users. It is this very flexibility and power which makes securing a UNIX system difficult. For example, Sendmail, the mail transport that delivers email between systems, has been a consistent cause of vulnerabilities almost since its inception. Means of exploitation have quickly been found to allow unintended access for both local users and attacks across the network in each new version. The reason for this, largely, is that Sendmail does a good deal more than simply move mail between systems. Nor is this the only UNIX tool to be exploited. Many other components, including rlogin, ftp, Web servers, even text editors have fallen victim to ingenious hackers who take advantage of their flexibility and power. The UNIX operating system has been directly exploited by hackers taking advantage of problems ranging from file permissions that allow more access than is desirable to buffer overflows that grant privileged access through unintended use of otherwise innocuous system commands. Naturally, it is not desirable to incorporate known security-defective software products in systems supporting combat.

Microsoft's Windows NT suffers from its share of problems. Source code for the operating system is not available for review, making objective analysis impossible. Also, its password system has been shown to have very serious weaknesses; programs to decipher encrypted NT passwords are available from multiple sources. Presentations dissecting these problems in painstaking detail have been presented in a major metropolitan hacking congress, with members in attendance from many nations.

Many operating systems and network infrastructure devices suffer from various abuses to the implementation of networking protocols, causing various forms of denial of service. These attacks take advantage of the fact that many programmers do not perform proper bounds checking to handle situations that should never occur, such as a connection being opened from a particular port on a given machine to that same port on the system. Since the software that implements the networking stack does not handle the packet correctly, it will often fall into an unstable state and cease to function correctly.

Other networking attacks are caused by problems with the protocols themselves. For example, the SYN flood attack is perpetrated by an attacker that forges a series of packets from an address that has a route, but will not respond (i.e., a machine that is down, or an address that is not used) to the target with the SYN flag set. This causes many half-open connections that will take some time to close, causing a denial of service on the target port. Also, since each connection uses a portion of memory, it is possible that this attack will cause the victim to crash. Unfortunately, software to effect this and many other attacks is freely available from the Internet and so-called computer underground publications.

The following table provides a summary of some major security considerations which are typical of and tend to distinguish operational characteristics of Windows, UNIX, and trusted systems. Of course, little precludes a trusted system from also being a UNIX system, but such a system would necessarily be built from the ground up towards this end, and would deviate in significant ways from traditional UNIX systems.

ISSUE OR VULNERABILITY	WINDOWS	UNIX	TRUSTED SYSTEMS
Proprietary Source Code	Yes	Yes Source code licenses are often available for a fee.	Yes Source is available for analysis. Government may sponsor and own source code for an entire system if desired.
Weak Password Systems	Yes	No	No
Mail Vulnerabilities **Mail Bombing** **Breaking Root**	 Yes No	 Yes Yes	 No No
Application Escapes	Yes	Yes	No
Protocol Exploits	Yes	Yes	YES and NO If intrinsically flawed protocols are utilized by the system, mitigating the associated vulnerabilities may be impractical.

Major Security Characteristics

IV Trust

Fundamental to any critical mission or flow information are questions associated with trust. Intelligence officers, General officers, the soldiers, and the engineers must be able to trust one another and the systems at their disposal. While it may be difficult to trust individuals and subordinates (even peers), it can often be more difficult to trust the machinery at one's command.

Subtly hinting at some of the grave problems which may be experienced is the problem of composition of secure systems. That is, relatively secure systems can be composed, with the resultant system being non-secure. A concrete example of a fundamental trust issue relates to a truly classic Unix system exploit. A Trojan Horse had been planted in the Unix system's compiler code. Since Unix systems were constructed in the C programming language, this exploit was propagated to Unix systems as they were built. The result of this subterfuge is that Ken Thompson was capable of logging into any Unix system at will. This example also serves as an additional warning of welcoming COTS products with open arms.

Communications hardware and software are now serving increasingly important and dedicated roles, requiring special attention and demanding trust. Communications equipment and software serve to process and propagate information, and if compromised can serve as a difficult-to-trace avenue for further systems compromises. A recent incident highlights the situation. An Israeli product, Firewall - I , was discovered to have serious significant security shortcomings. By comparison, the NetFortress by Fortress Technologies has been reviewed by the NSA (National Security Agency) / SPOCK (Security Proof of Concept Keystone) to validate its functional claims. Such third party assurances assist in establishing

systems trust. Similarly, the Approved Products List provides a catalogue of systems which have been studied in detail to provide systems security assurances.

There are certain naturally imposed limits on the degree of testing which any system may undergo, and any testing effort brings its associated costs. Among the most severe considerations are some very pragmatic and obvious issues. It would be inadvisable and counterproductive to initiate a total war merely to test military systems and how they may be penetrated. Fortunately (or perhaps, unfortunately), there are documented cases of successful penetrations which may be used as a guide in how DOD may better secure its systems. These experiences also indicate some of the tactics employed by hostile forces, even if we may not command a total knowledge of their exact methods, capabilities, and operations.

V Strengthening the Security Posture

The view that security requirements are functional requirements must be nurtured and allowed to flourish in all stages of weapons work. Sufficient training and guidance must be made available to software developers and systems managers, so that they may better design and build their systems with the intention of minimizing the number of unfortunate surprises which may arise either later in development, or after fielding, when the consequences are most serious.

Centralized organizations established either to support groups of DOD efforts, or a single, centralized organization could act as an all-in-one support facility, a common library of knowledge of current systems vulnerabilities and exploitations, a coordinated center for coordinating and tracking corrections to such vulnerabilities, and thus, provide economies of scale. Such an organization could also maintain "Templates," which would provide standardized points of reference with which the security of weapons systems could be evaluated and compared against known vulnerabilities as software development proceeds. As these templates are maintained and kept up - to - date, the developers would be responsible to maintain currency with newly discovered security hazards. Standardizing the system review process and centralizing knowledge of vulnerability would also assist in evaluating the overall security posture of all DOD platforms at any point in time, and since such a process would will allow relative comparisons between systems security profiles, as well as the production of relevant statistical figures relating to security strengths, the DOD may more efficiently expend its resources to address its computing security needs.

One obstacle to be carefully addressed concerns issues relating to updating systems in the field. Once systems are deployed, they must be kept current with known, and preferably unknown attacks. Regimentation of appropriate processes should be applied to aggressively and consistently use the most current security tools to both attack and defend mission-critical systems. Certain methodologies, such as enlightened engineering practices and the use of traditional trusted systems methodologies can assist in reducing the number of attacks whose natures may be unforeseen. Unfortunately, for practical reasons, these methods can not practically be relied on in a great many cases. Methods which could rapidly be brought to bear to address these needs, however, include the incorporation of rapid, sophisticated, and highly automated systems test capabilities for the systems concerned. This can greatly reduce the lag time between the discovery of security vulnerabilities and the time when they are actually closed in DOD systems. Then, secure and reliable methods for closing these security holes in the field must be developed. The critical window of vulnerability must be minimized.

VI Conclusion

Traditional, sound software engineering practices have been known and accepted to reduce development and Post Deployment Software Support (PDSS) costs, and should offer risk mitigation through a heightened systems security profile across DOD baselined platforms. Placing consideration of security issues early in the development cycle yields a more efficient use of resources, providing better security for less money. It is crucial to avoid the attitude expressed by one former DOD official that, "Infosec does not fly, sail into harm's way or grind its way across the desert," and is therefore not important. Quite to the

contrary, failure to properly recognize the importance of information security and take appropriate, strong measures can lead to utter devastation, including compromise of classified information, loss of life and failure to accomplish military missions.

VII REFERENCES

Mc Cullough, Daryl, "A Hookup Theorem for Multilevel Security," IEEE Transactions on Software Engineering, Vol. 16, No. 6, June 1990.

Gasser, Morrie, "Building a Secure Computer System," Van Nostrand Reinhold, New York, 1988.

General Accounting Office, "Report to Congressional Requesters: Information Security - Computer Attacks at Department of Defense Pose Increasing Risks," GAO/AIMD-96-84, May 1996. http://www.epic.org/computer_crime/gao_dod_security.html

Gibbs, Wayt, "Profile:Dan Farmer From Satan to Zen," Scientific American, April 1997

Meeks, Brock, "Introduction and Keynote Address," http://www.hope.net

Secure Networks, Inc., "Checkpoint-I Security Advisory," December 9, 1997. http://www.ers.ibm.com/tech-info/advisories/oar/1997/ERS-OAR-E01-1997:140.1.txt

Thompson, Ken, "Reflections on Trusting Trust," Turing Award Lecture, Communications of the ACM, August 1984, Vol. 27, Number 8.

Winkler, Ira, "Corporate Espionage," Prima Publishing, 1997.

VIII Contacts and Biographies

Sam Nitzberg was born in and lives in Long Branch, New Jersey, USA. He graduated from Monmouth University with a Bachelors degree in Computer Science, and a Masters degree in Software Engineering. His Masters thesis concerned the performance benchmarking of Unix audit trail systems. He is presently studying for his Ph.D. in Computer Science at Stevens Institute of Technology in Hoboken, New Jersey, USA. Mr. Nitzberg has been working for Telos for the last five years, specializing in software engineering and computing security.

Michael Carson resides in Brick, NJ. Born in Neptune in 1972, he received his Bachelor of Science degree in Computer Science from Monmouth University in 1994. He is currently employed by the Telos Information Protection Solutions, specializing in Information Security.

Sam Nitzberg and Michael Carson may be contacted at: Telos Information Protection Solutions, 656 Shrewsbury Avenue, Shrewsbury, NJ 07702, USA. Sam's electronic mail address is sam.nitzberg@telos.com, and Michael's is mike.carson@telos.com.

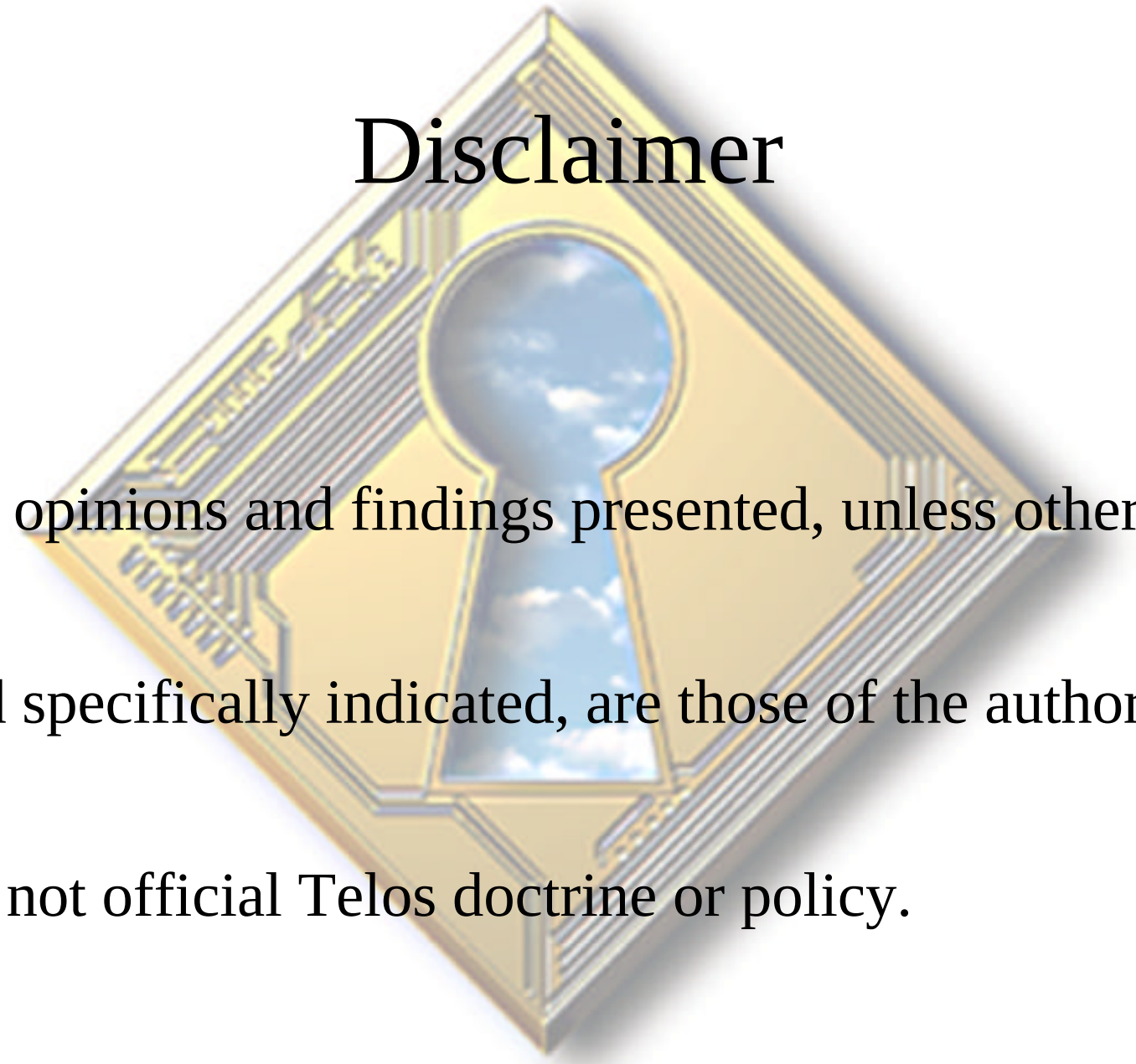
Improving Computing Security

During the Development of
DOD Computerized Weapons Platforms

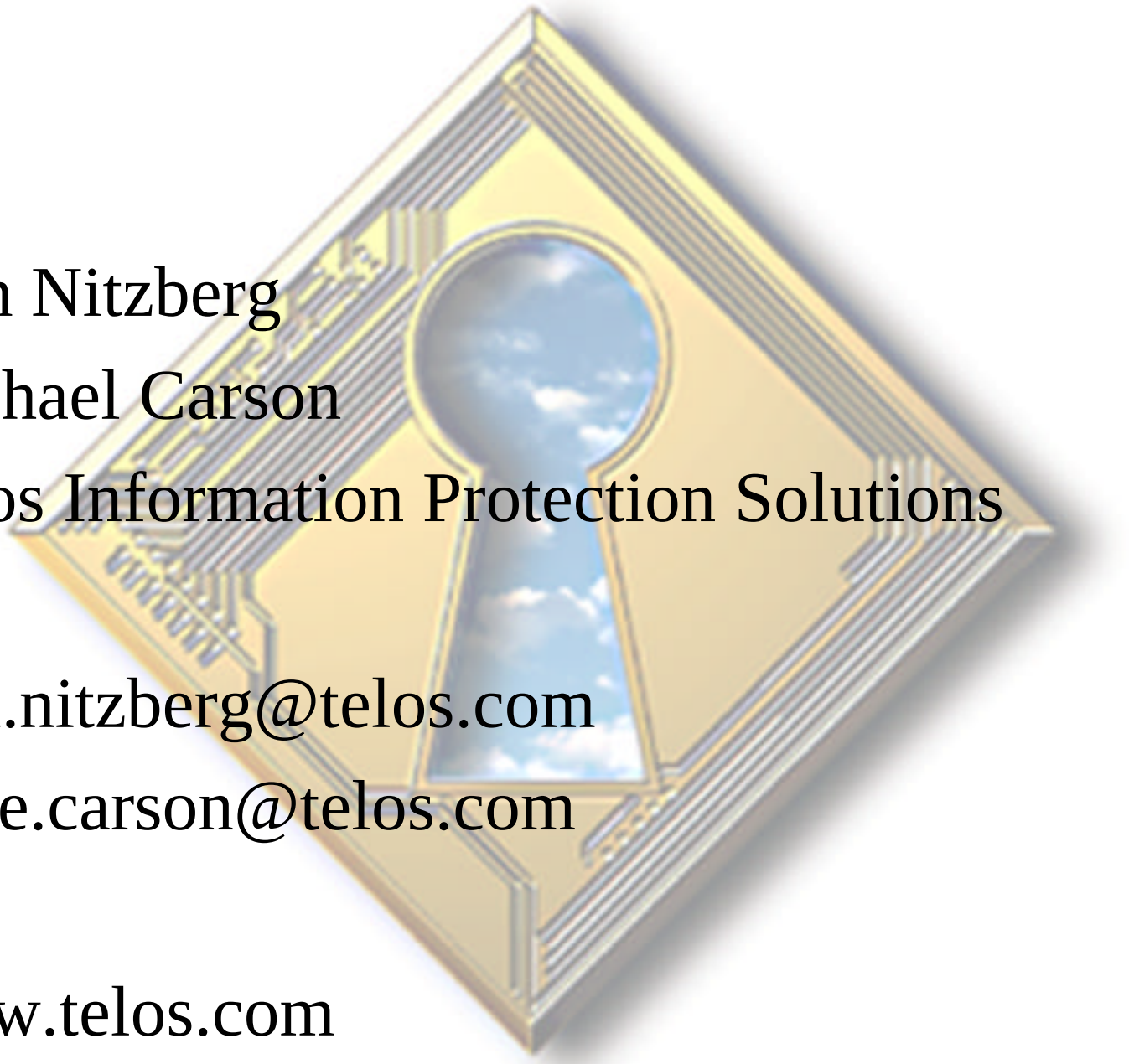
National Information Systems Security Conference - 1998
Crystal City, VA

Sam Nitzberg
Michael Carson
Telos Information Protection Solutions

Disclaimer



The opinions and findings presented, unless otherwise and specifically indicated, are those of the authors, and not official Telos doctrine or policy.



Sam Nitzberg

Michael Carson

Telos Information Protection Solutions

sam.nitzberg@telos.com

mike.carson@telos.com

www.telos.com

ORGANIZATION

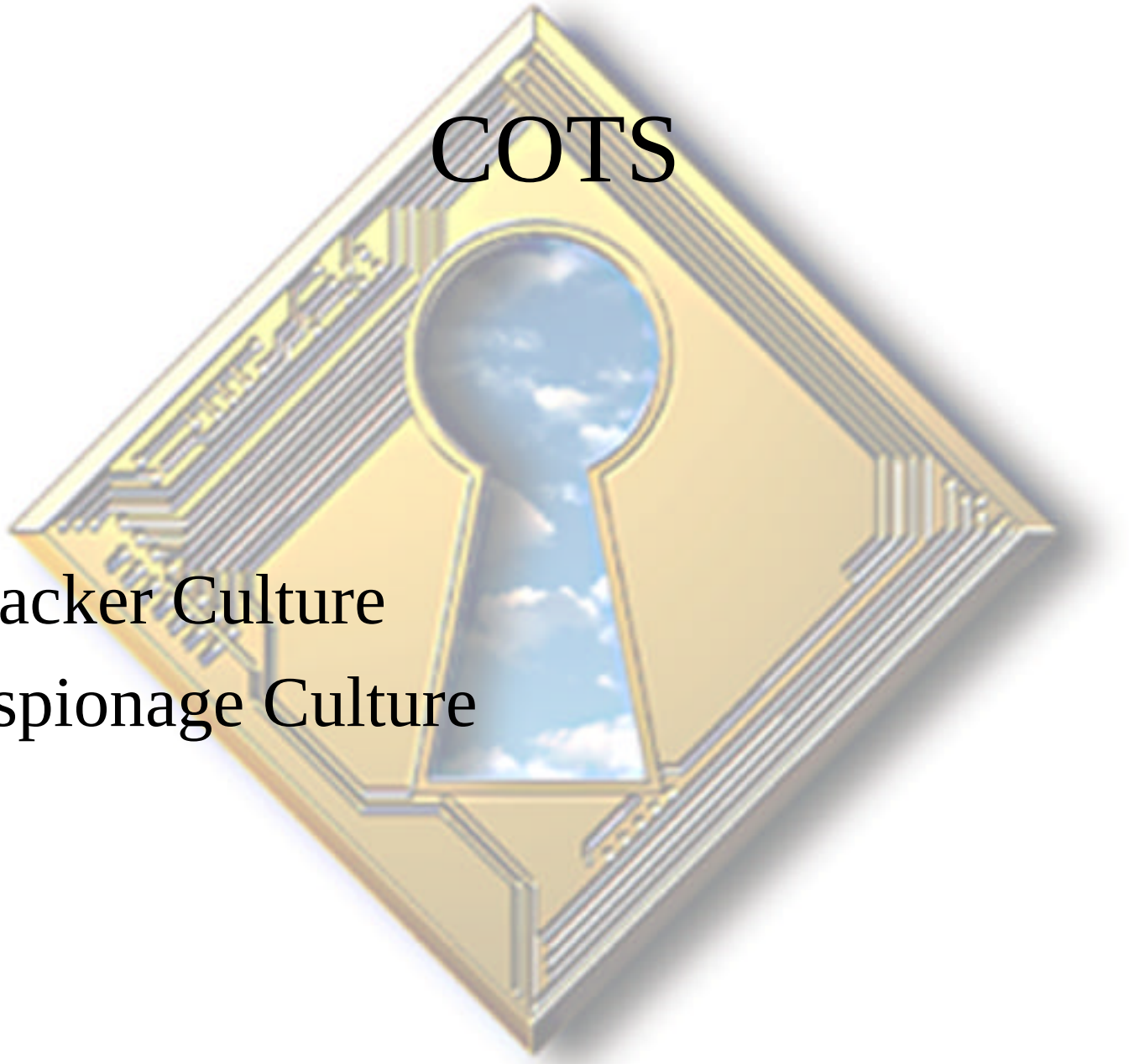
I. Introduction
II. Software Development Methods and Models
III. Nature of the Threat
IV. Trust
V. Strengthening the Security Posture
VI. Conclusion

I INTRODUCTION

Three Eras

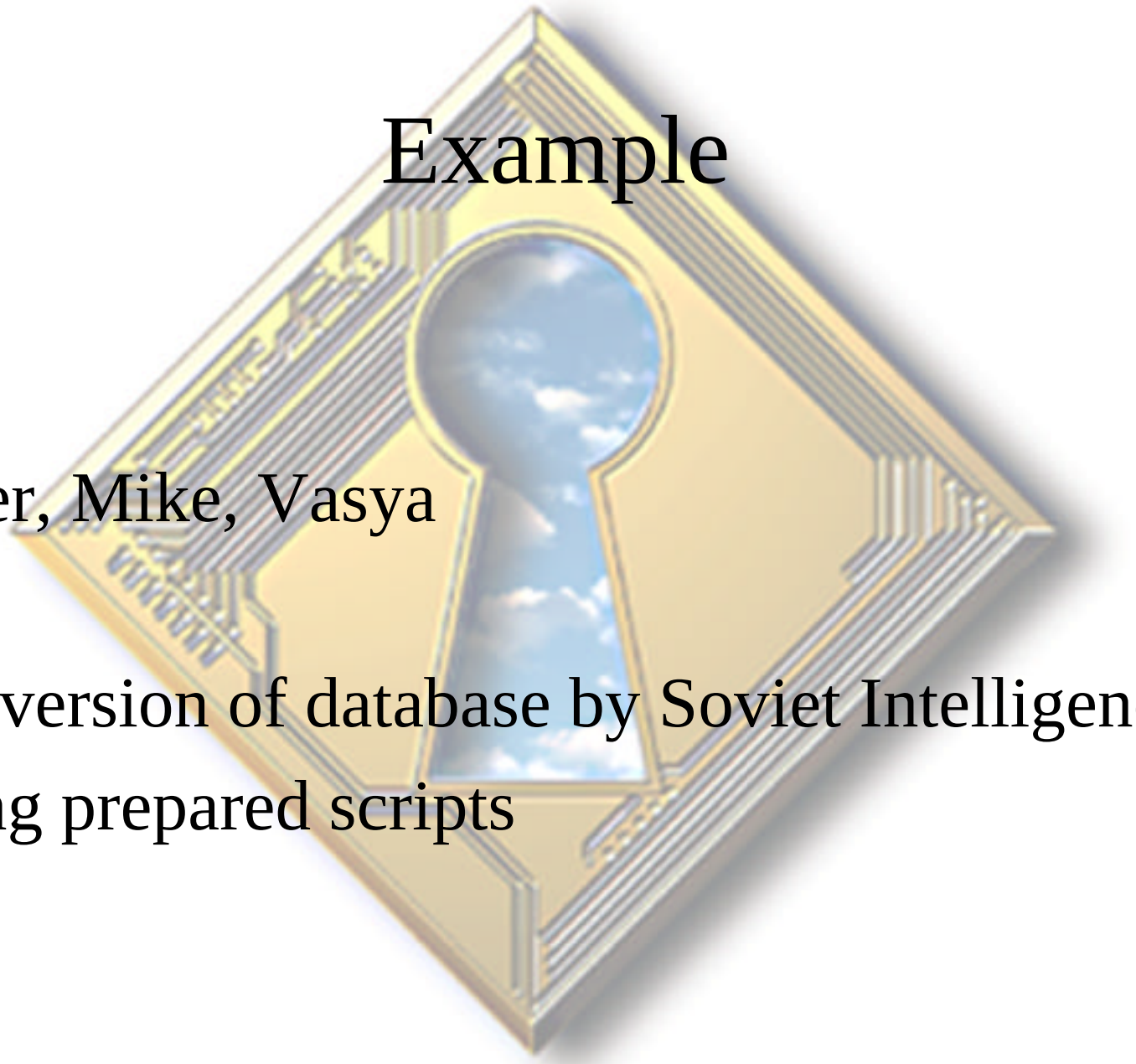
- ENIAC - Physical Security
- SAGE - Networked, Centralized weapons
- Current Systems - Increasing decentralization with COTS components

COTS



- Hacker Culture
- Espionage Culture

Example



Peter, Mike, Vasya

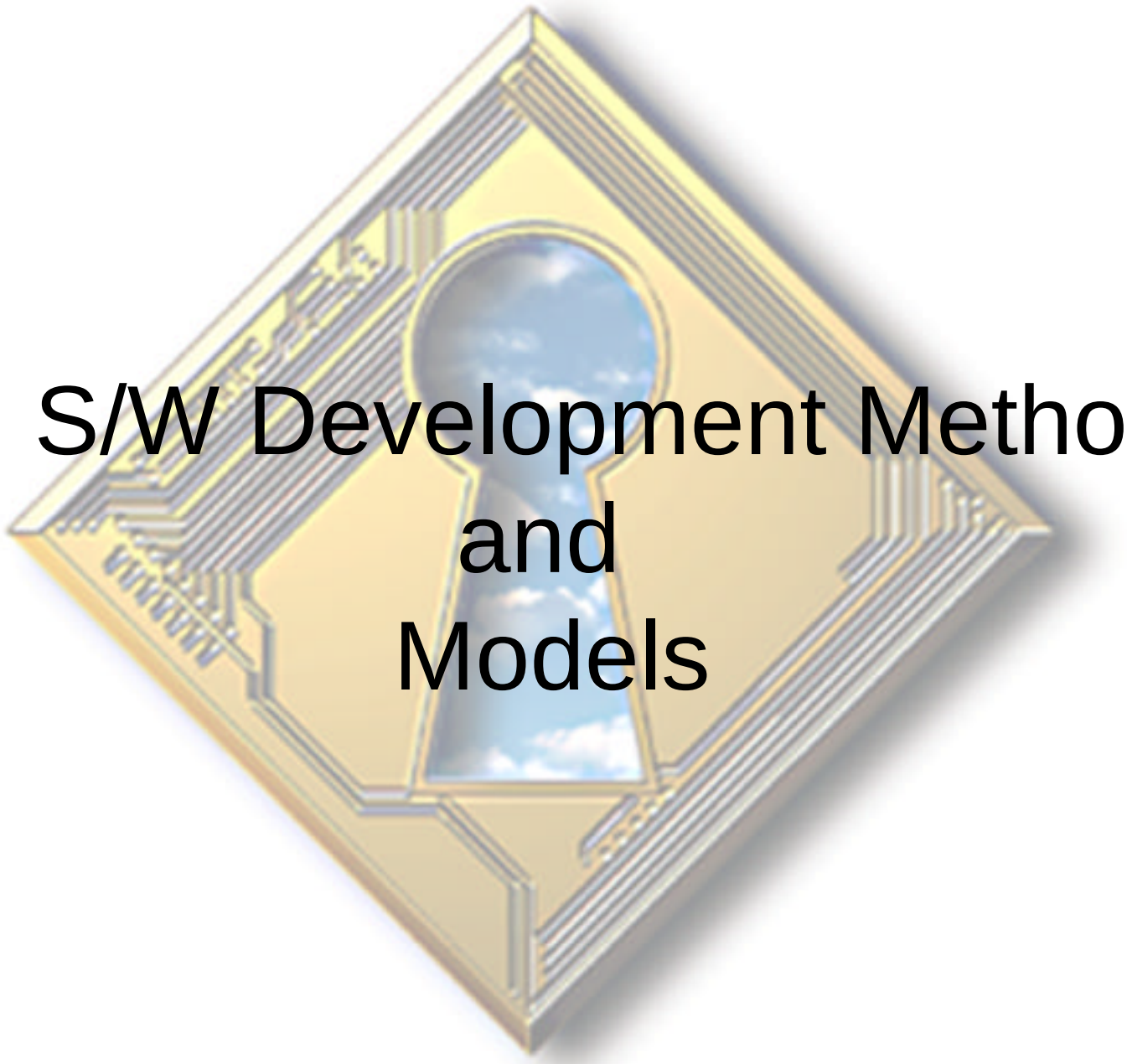
Subversion of database by Soviet Intelligence using prepared scripts

Control

Must not permit an adversary to subvert control of the battlespace by subverting DOD systems

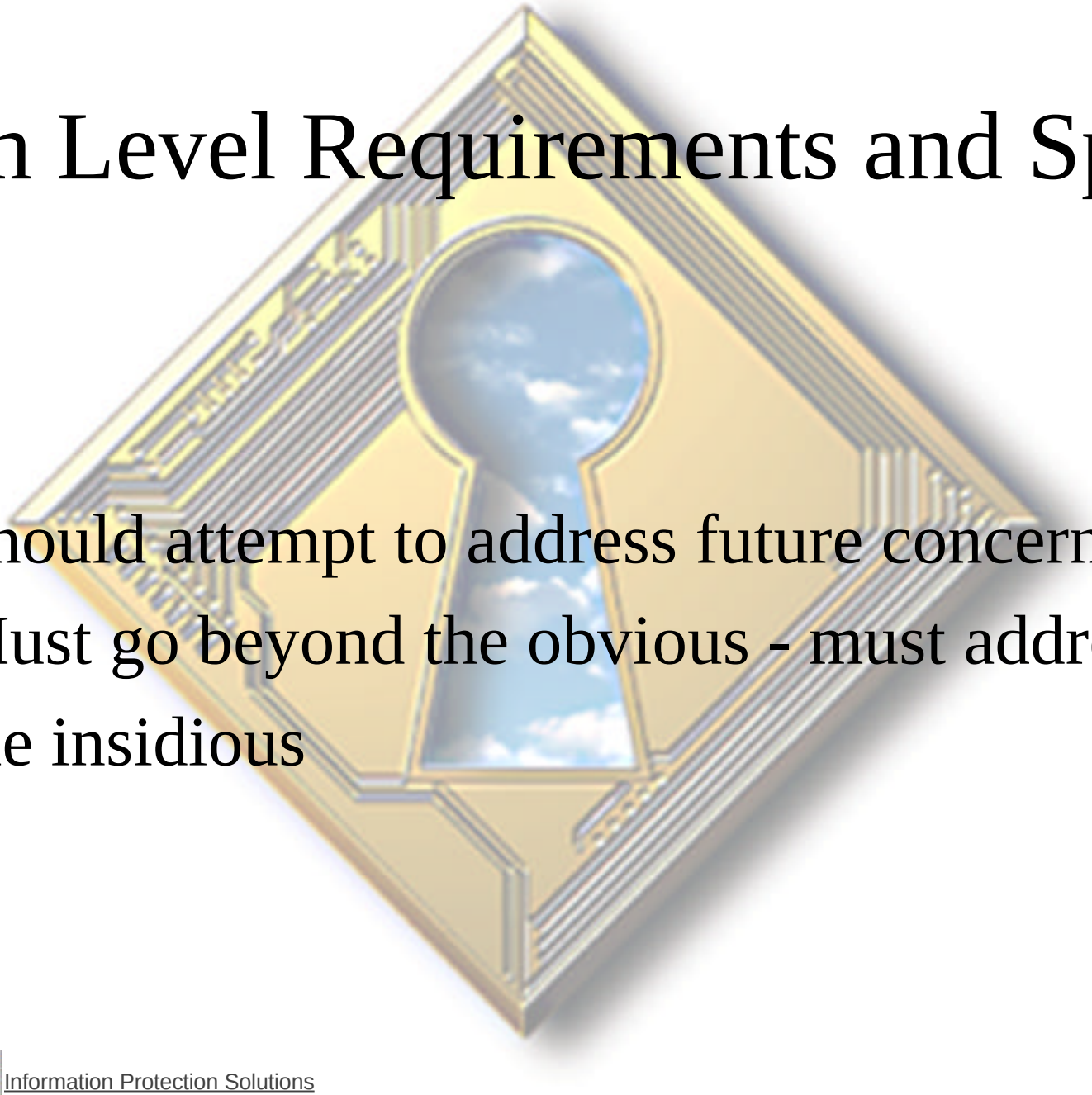
Living Documents Required

- Must go beyond procedural mechanisms
- Dynamic security posture is necessary
- Must address security threats, vulnerabilities, and concepts as they emerge as they impact all systems.



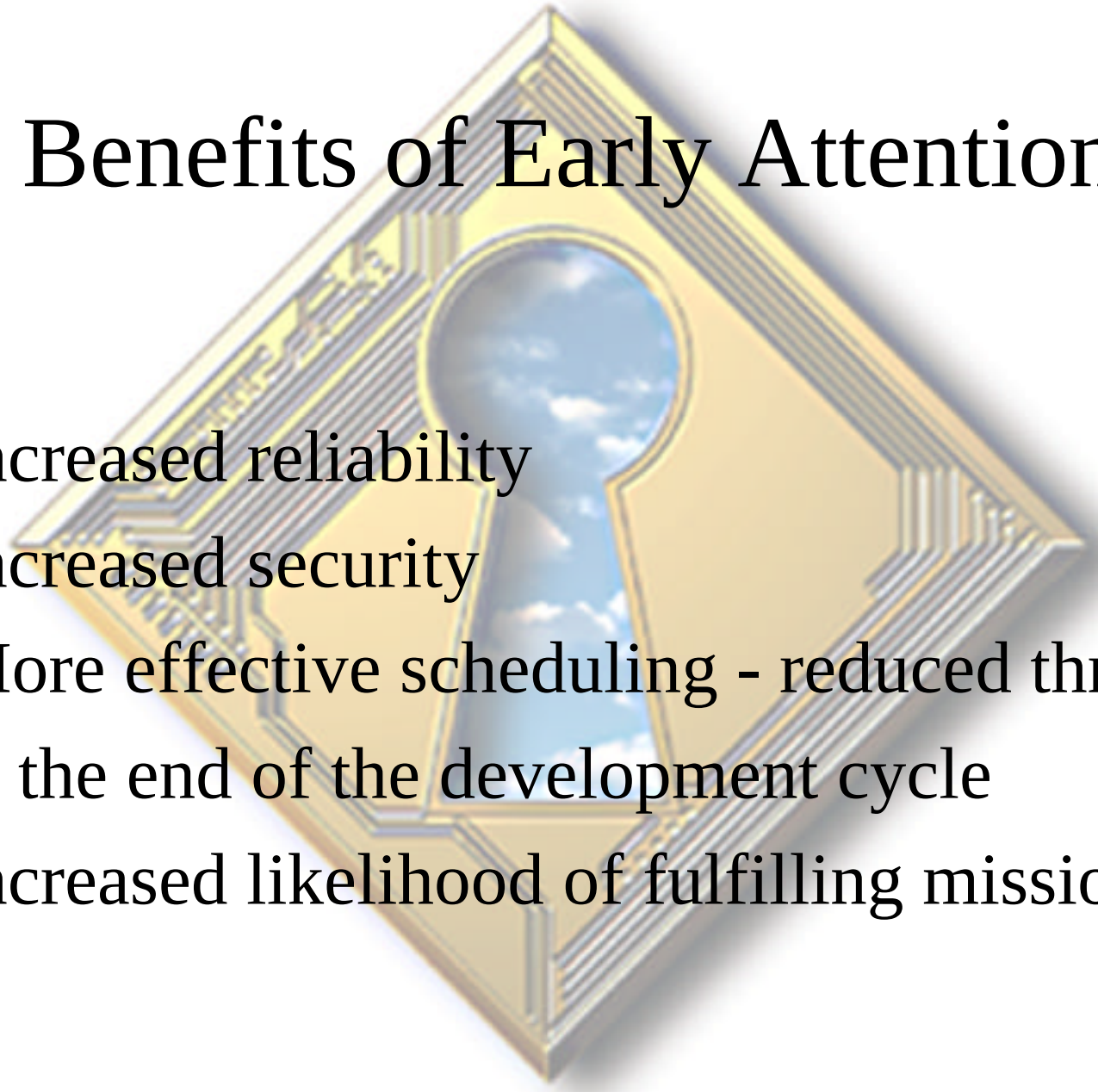
II S/W Development Methods and Models

High Level Requirements and Specs



- Should attempt to address future concerns
- Must go beyond the obvious - must address the insidious

Benefits of Early Attention



- Increased reliability
- Increased security
- More effective scheduling - reduced thrashing at the end of the development cycle
- Increased likelihood of fulfilling missions

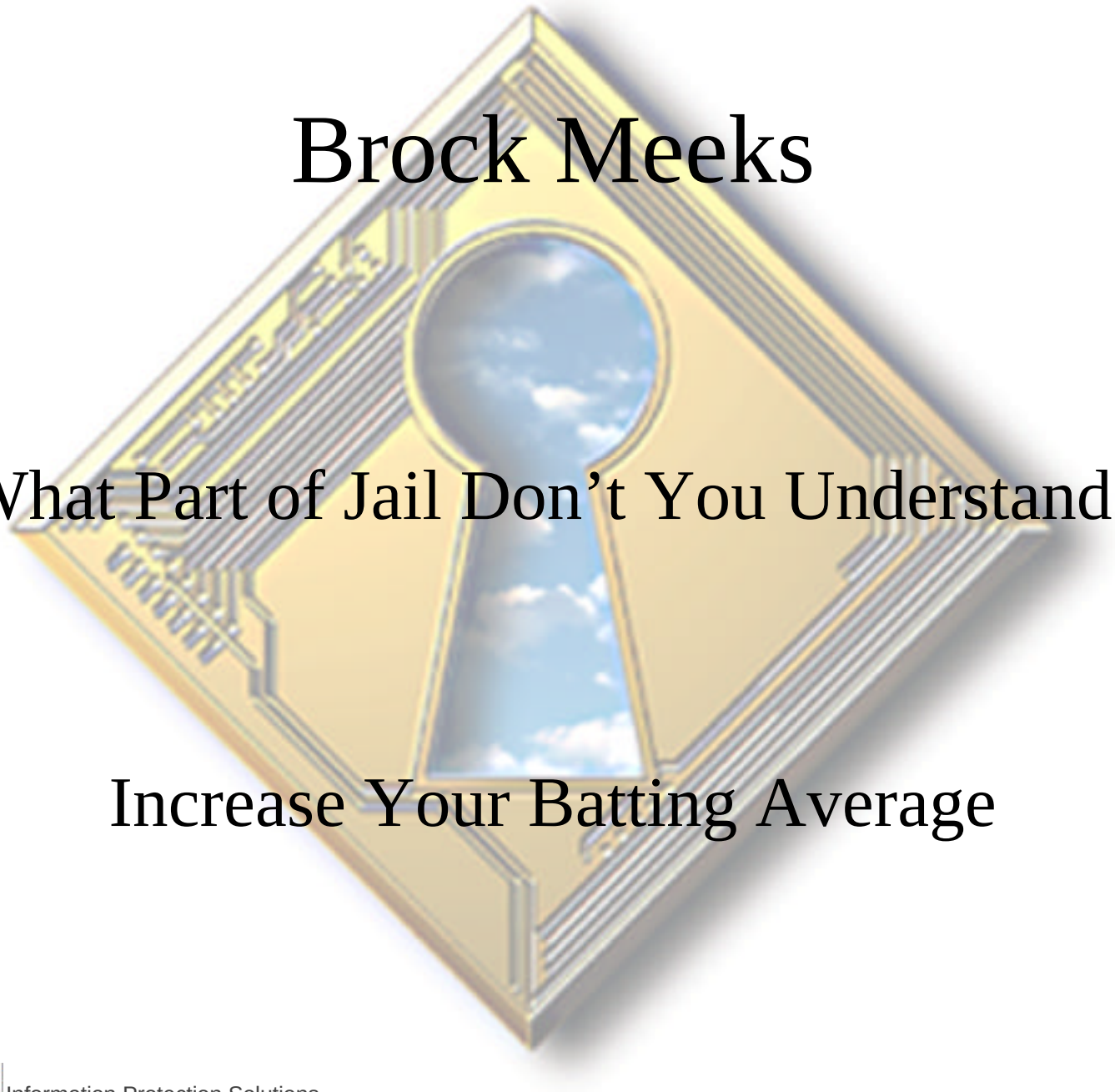
Re-Engineering Avoidance

- Do not rely solely on traditional security requirements
- Aim H/W and S/W selection to meet security objectives and requirements

The Ideal

- All known vulnerabilities are addressed
- Process is in-place to remedy new vulnerabilities
- Efficient solution mechanisms
- Controlled testing expense

III Nature of the Threat



Brock Meeks

What Part of Jail Don’t You Understand?

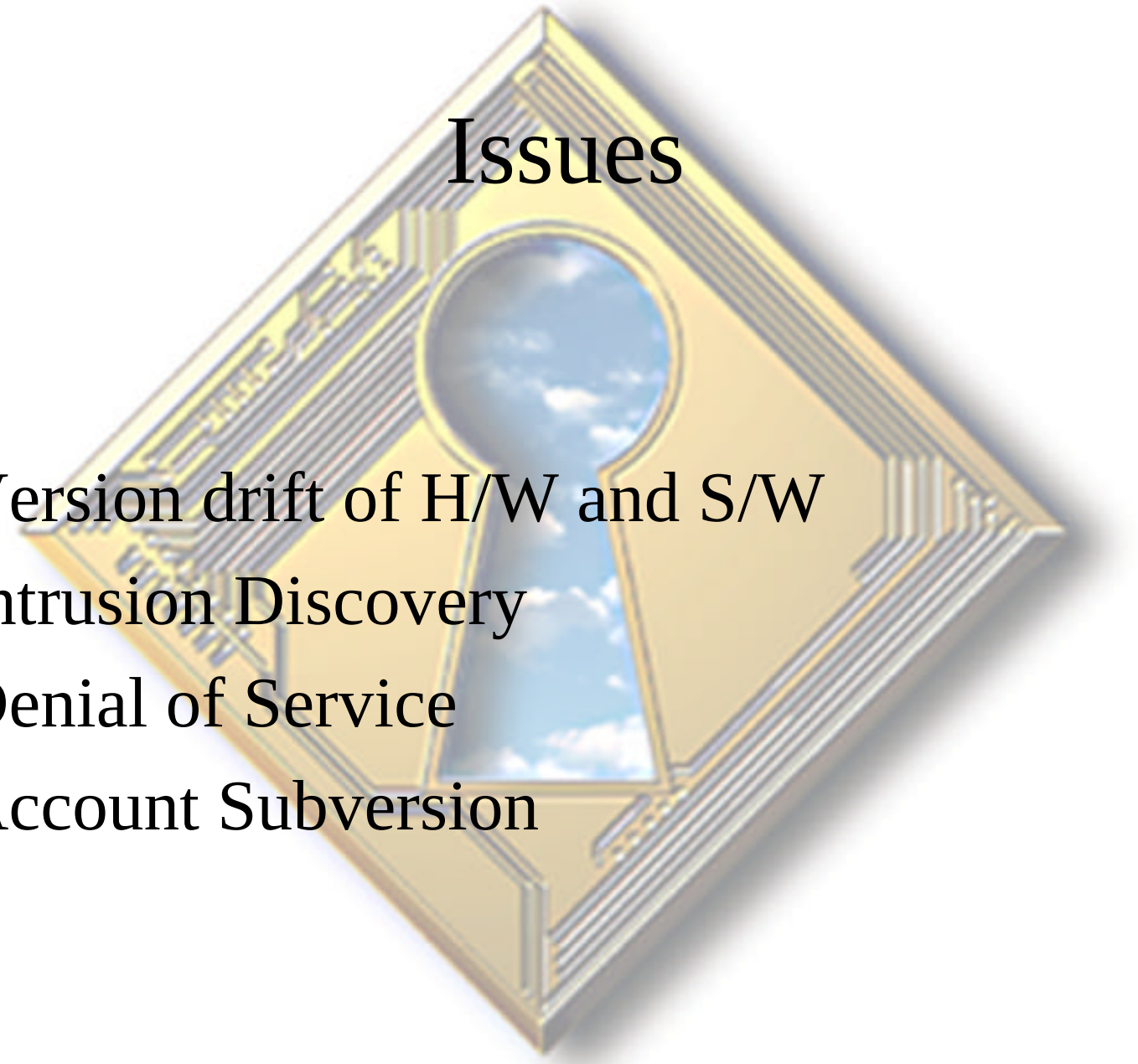
Increase Your Batting Average

Nature of Systems

An Awkward Marriage

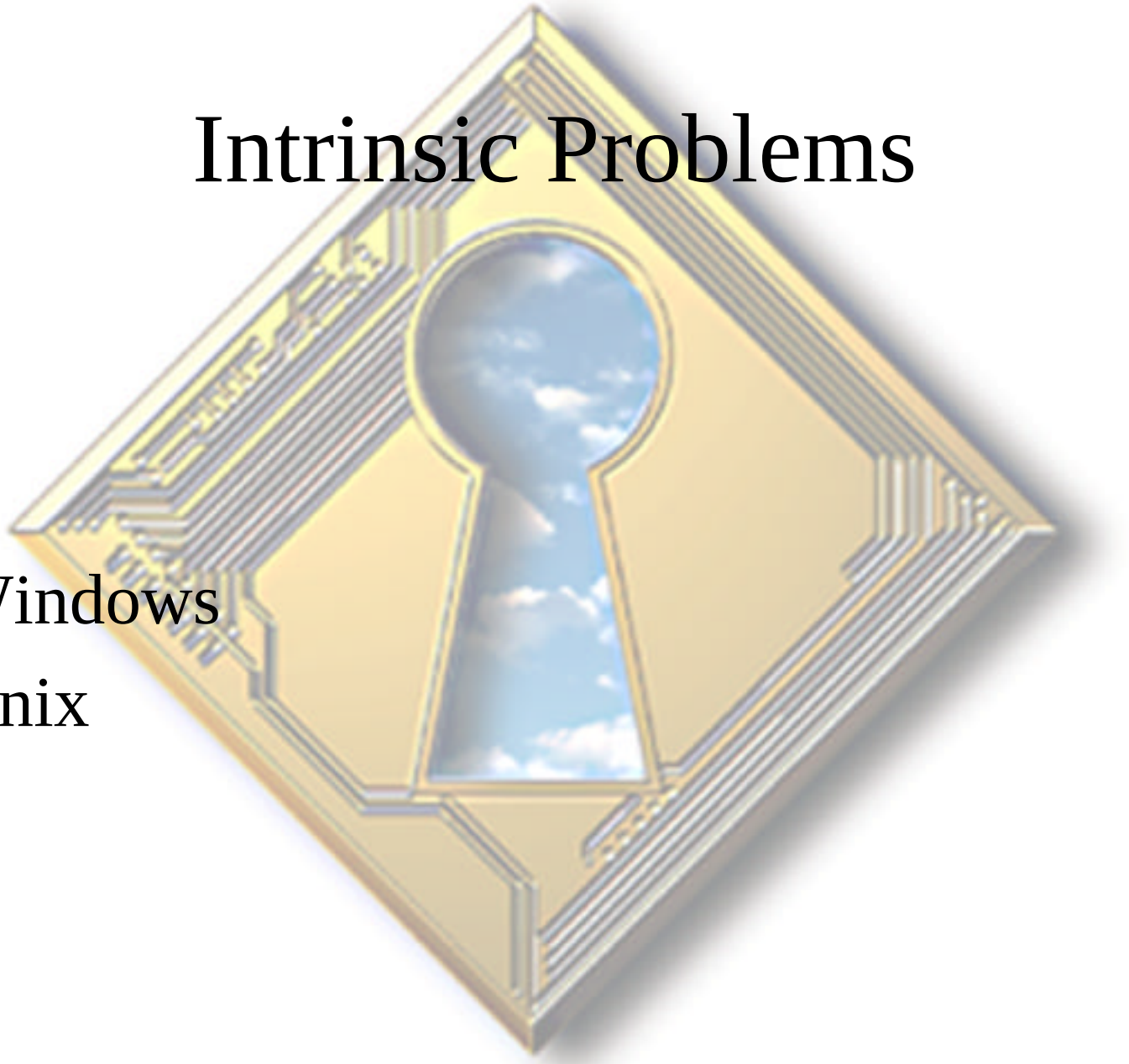
- Penetrate the Fog of War
- Complex “Soup” of Products
- Rely on COTS for truly Mission-Critical applications

Issues



- Version drift of H/W and S/W
- Intrusion Discovery
- Denial of Service
- Account Subversion

Intrinsic Problems



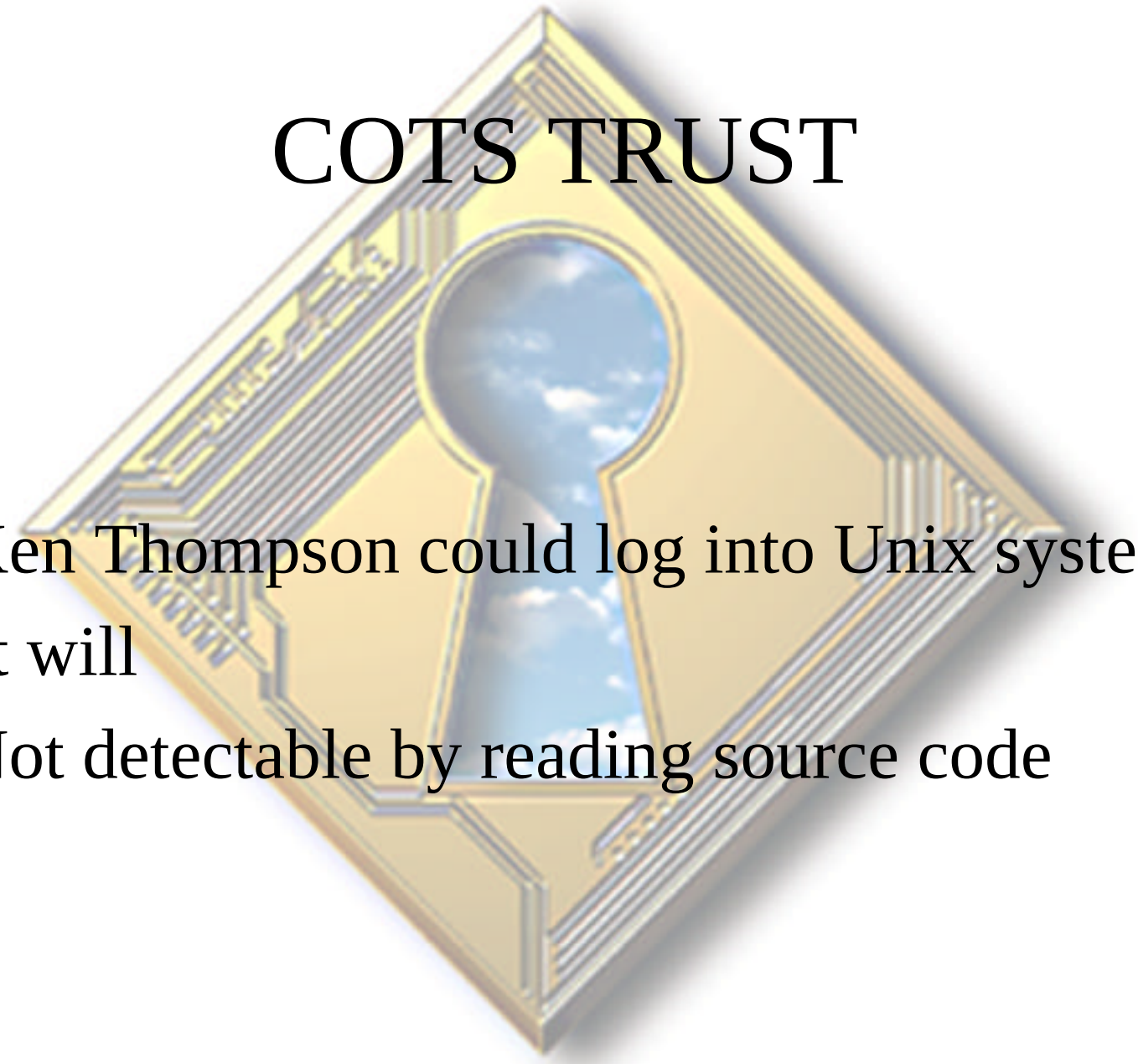
- Windows
- Unix

Some Security Considerations

ISSUE OR VULNERABILITY	WINDOWS	UNIX	TRUSTED SYSTEMS
Proprietary Source Code	Yes	Yes Source code licenses are often available for a fee.	Yes Source is available for analysis. Government may sponsor and own source code for an entire system if desired.
Weak Password Systems	Yes	No	No
Mail Vulnerabilities			
Mail Bombing	Yes	Yes	No
Breaking Root	No	Yes	No
Application Escapes	Yes	Yes	No
Protocol Exploits	Yes	Yes	YES and NO If intrinsically flawed protocols are utilized by the system, mitigating the associated vulnerabilities may be impractical.


IV
TRUST

COTS TRUST

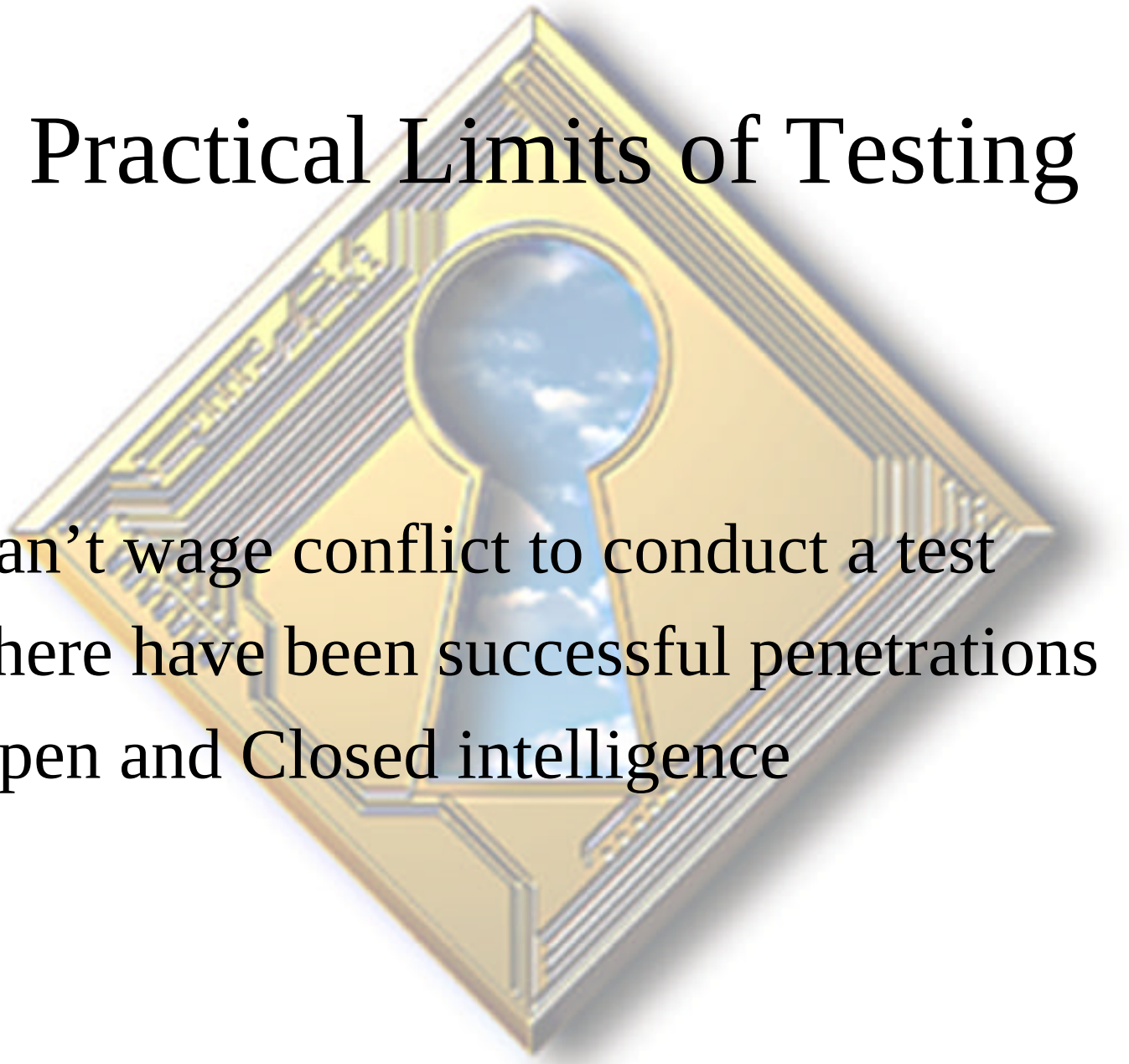


- Ken Thompson could log into Unix systems at will
- Not detectable by reading source code

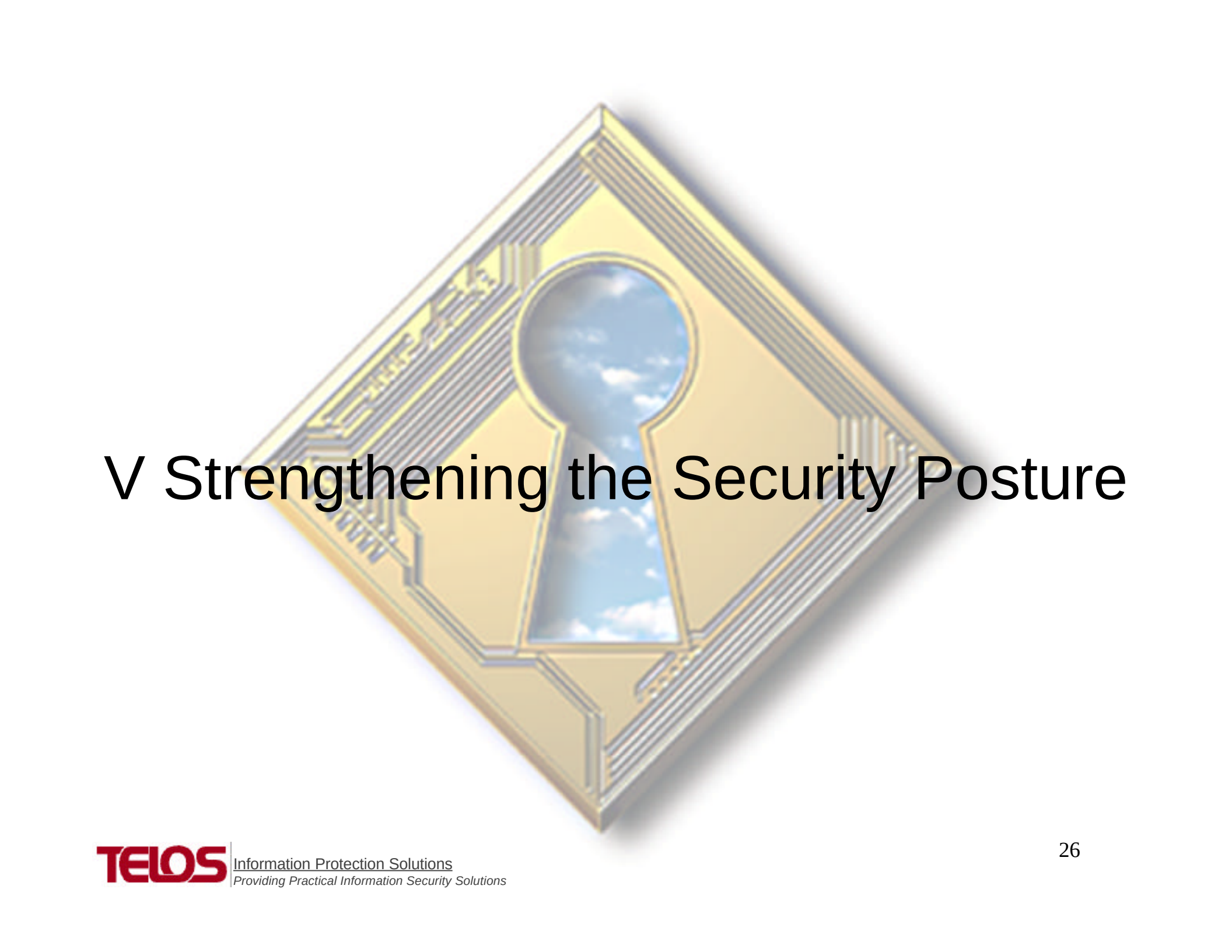
SPOCK

Security Proof of Concept Keystone

Practical Limits of Testing



- Can’t wage conflict to conduct a test
- There have been successful penetrations
- Open and Closed intelligence



V Strengthening the Security Posture

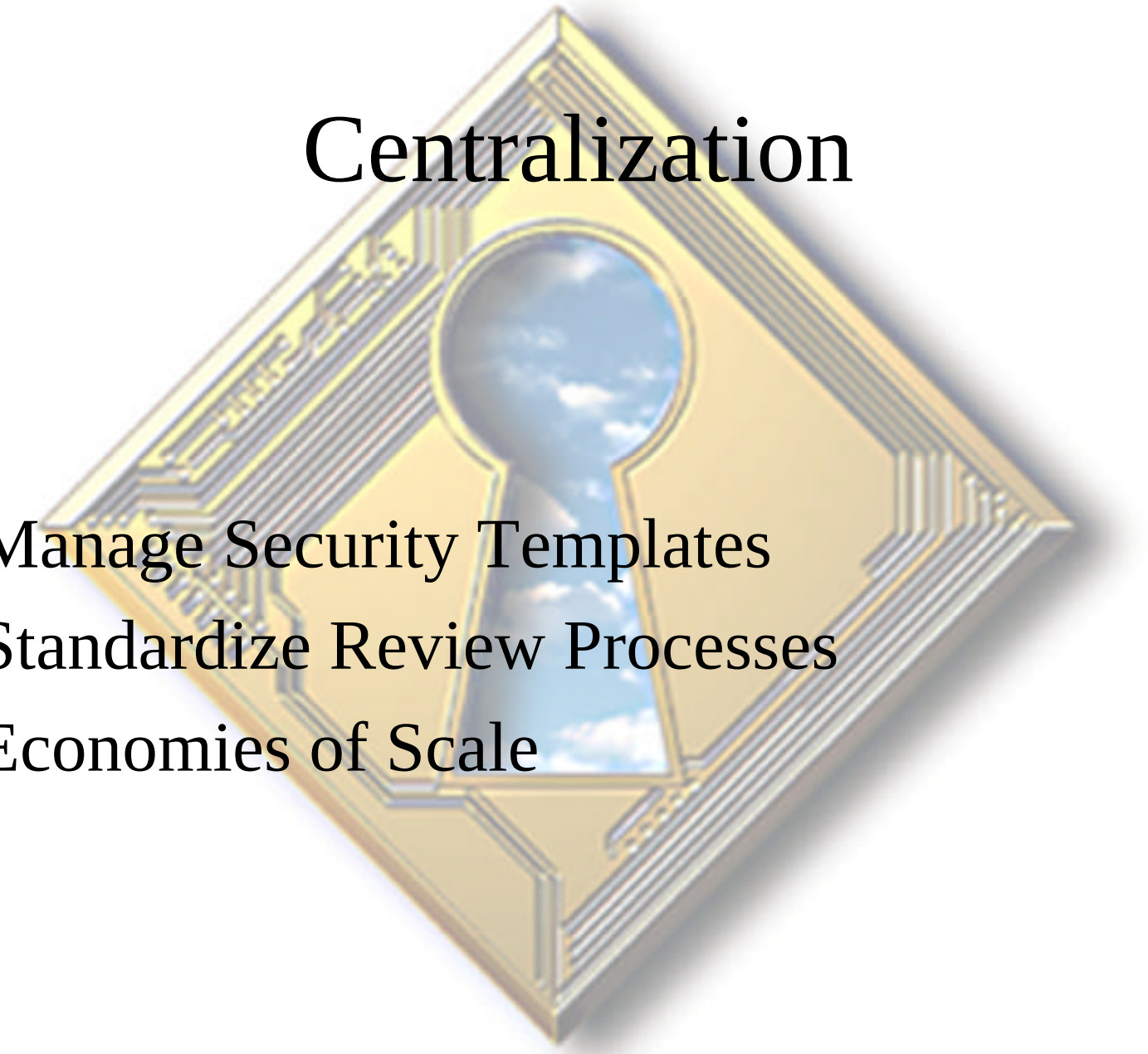
POSITION

Security Requirements

are

Functional Requirements

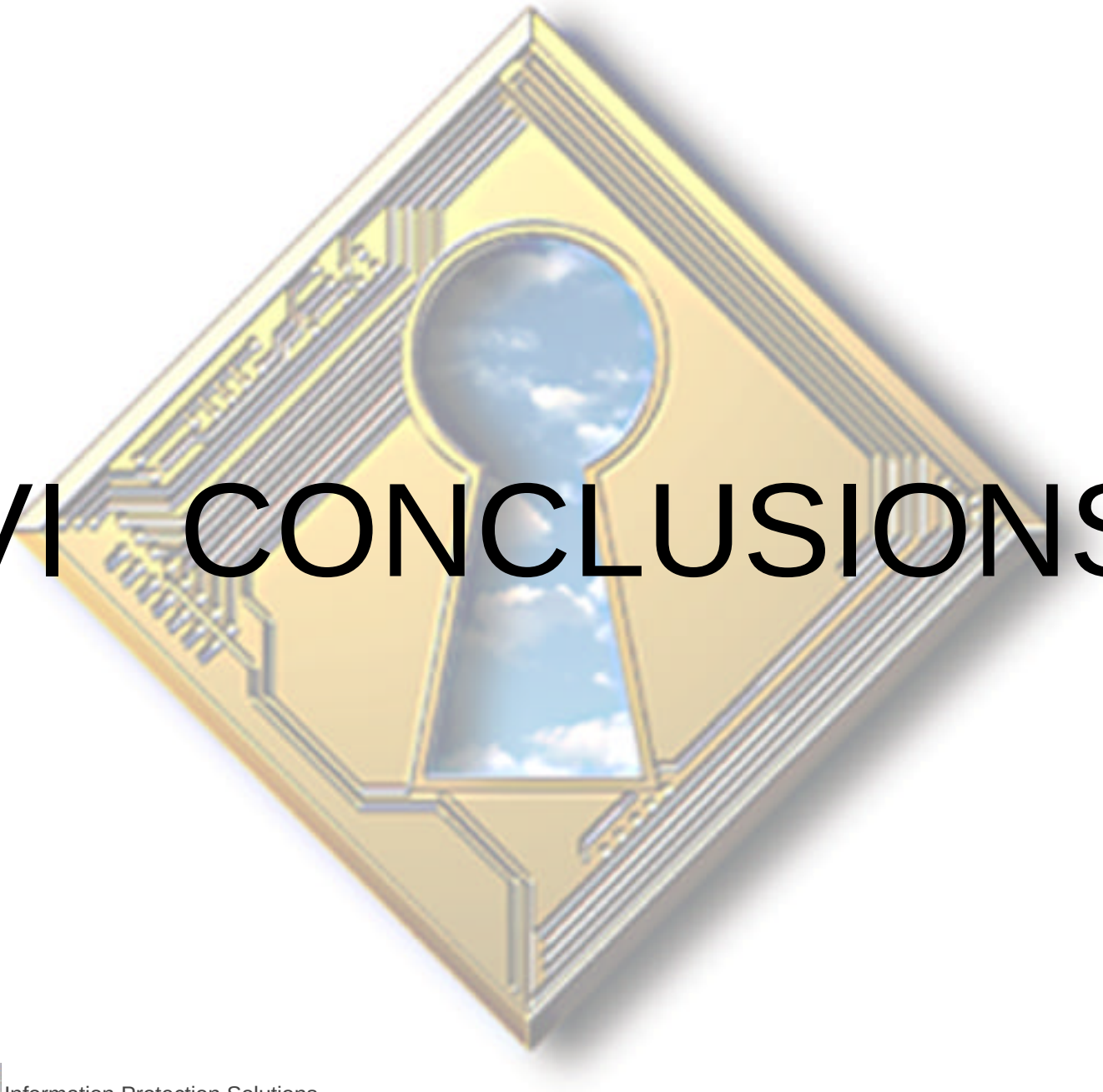
Centralization



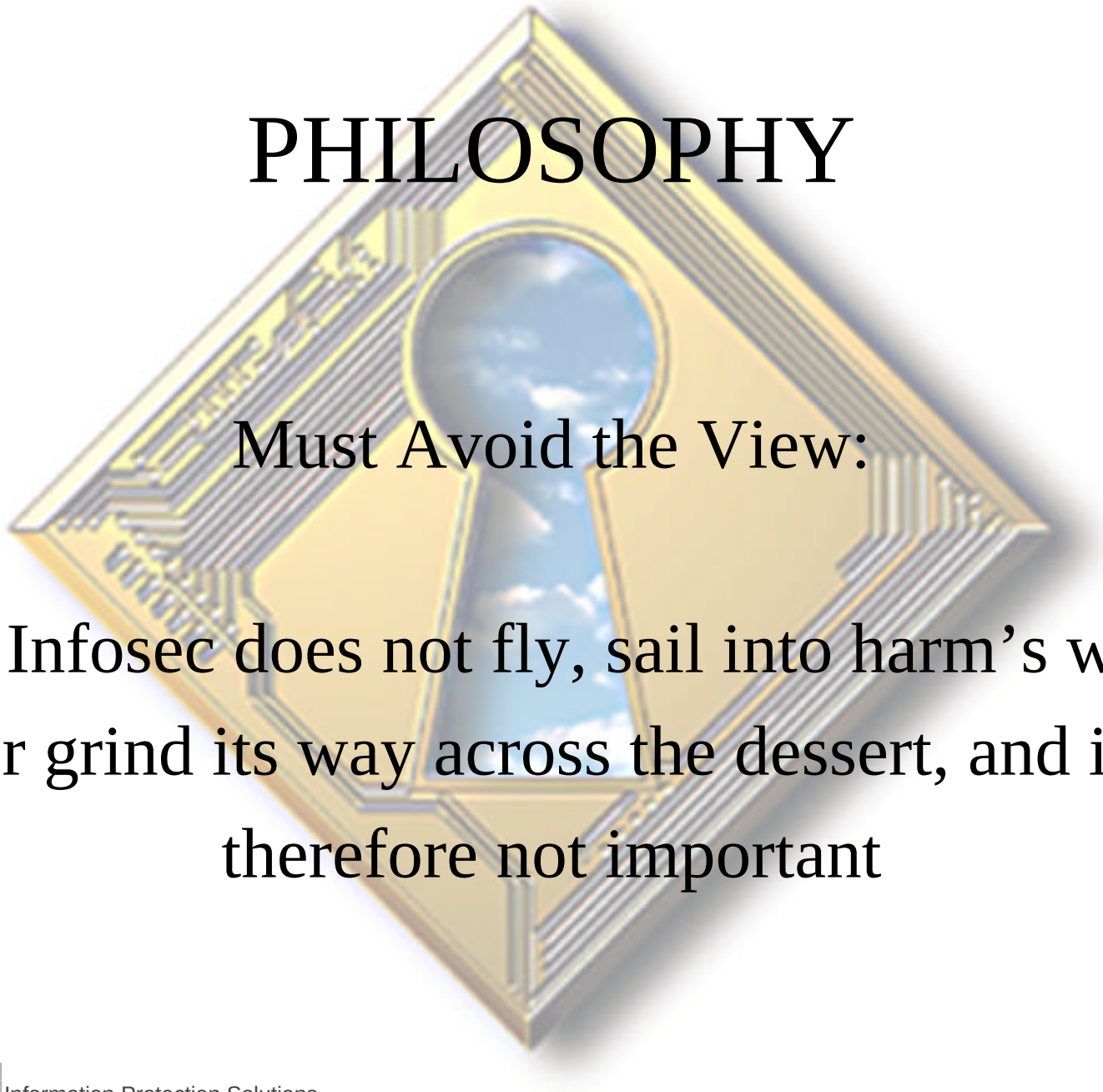
- Manage Security Templates
- Standardize Review Processes
- Economies of Scale

Field Updates

Methods must be used and developed to minimize window of vulnerability



VI CONCLUSIONS

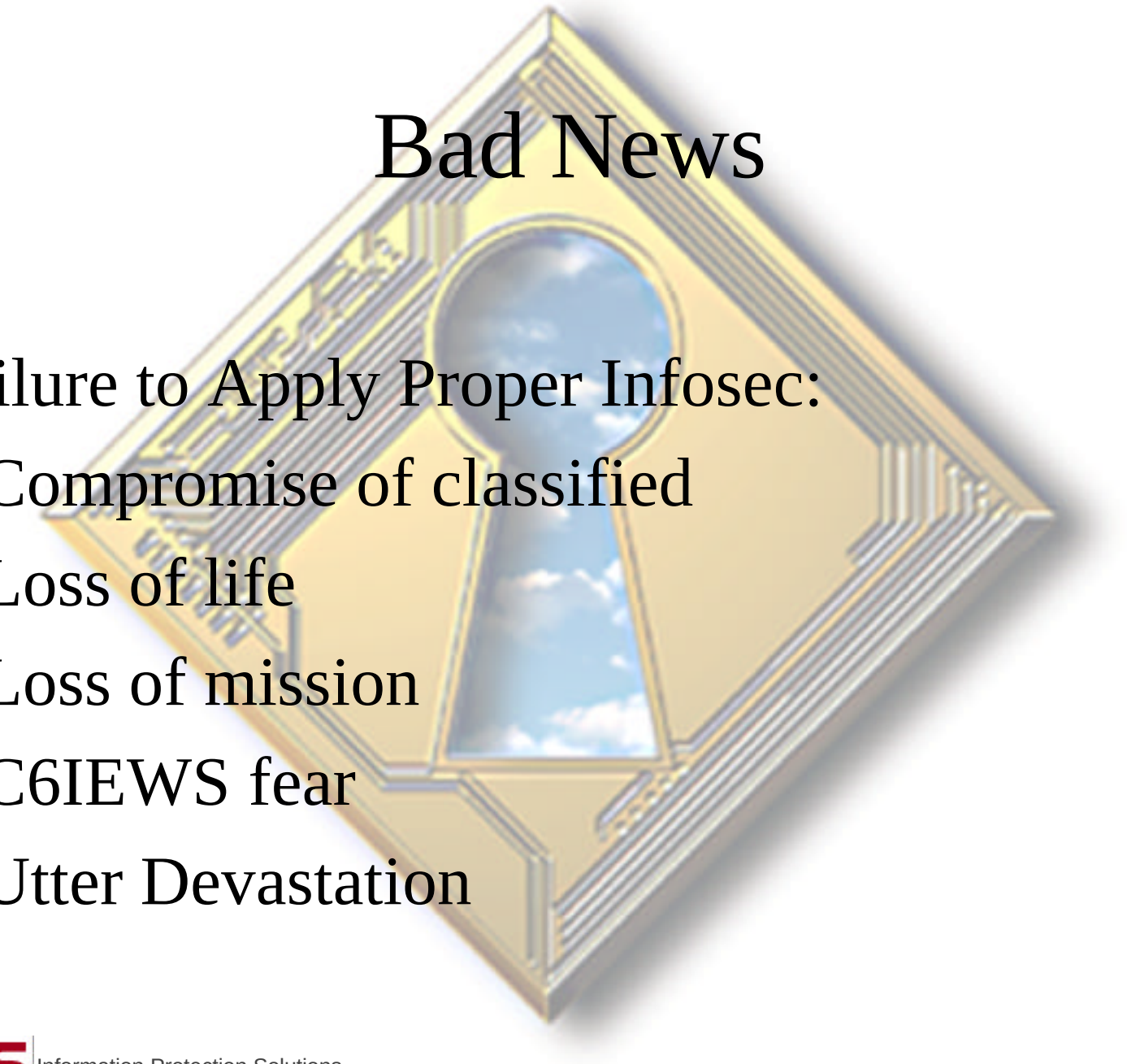


PHILOSOPHY

Must Avoid the View:

… Infosec does not fly, sail into harm's way or grind its way across the dessert, and is therefore not important

Bad News



Failure to Apply Proper Infosec:

- Compromise of classified
- Loss of life
- Loss of mission
- C6IEWS fear
- Utter Devastation

Good News

Proper addressing of Infosec:

- Mitigates risk
- More efficient use of resources

  Better security for less money
- Reduce cross-project impacts