Cyber Security

Edited by
John G. Voeller

WILEY

CYBER SECURITY

CYBER SECURITY

Edited by

JOHN G. VOELLER
Black & Veatch

WILEY

Published by John Wiley & Sons, Inc., Hoboken, New Jersey
Published simultaneously in Canada

ePDF: 9781118651735
ePub: 9781118651681

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

CONTENTS

Preface vii

1. Cyber Security Metrics and Measure 1
2. Multilevel Security 9
3. Trusted Platforms: The Root of Security 29
4. Cyber Security Technology Usability and Management 41
5. Security of Distributed, Ubiquitous, and Embedded Computing Platforms 57
6. Advanced Attacker Detection and Understanding with Emerging Honeynet Technologies 69
7. Security of Web Application and Services and Service-Oriented Architectures 79
8. Industrial Process Control System Security 87
9. Cyber Security for the Banking and Finance Sector 97

Index 113

PREFACE

Adapted from the Wiley Handbook of Science and Technology for Homeland Security.

The topic of homeland security did not begin with the World Trade Center or the IRA or the dissidents of past empires, but began when the concept of a nation versus a tribe took root and allegiance to a people was a choice, not a mandate. The concept of terrorism is part of homeland security but there are other risks to homeland security; such as those that come from Mother Nature or negligence of infrastructure maintenance. Indeed, these factors have much higher probabilities of causing substantial damage and loss of life than any group of terrorists could ever conceive. Hence, the focus here is on situations that put humans at risk and can disrupt and damage infrastructure, businesses, and the environment, and on scientific and technological tools that can assist in detecting, preventing, mitigating, recovering, and repairing the effects of such situations.

The number of science and technology (S&T) related topics that are involved in the physical, cyber and social areas of homeland security includes thousands of specialties in hundreds of disciplines so no single collection could hope to cover even a majority of these. Instead, our intention is to discuss selected topics in ways that will allow readers to acquire basic knowledge and awareness and encourage them to continue developing their understanding of the subjects.

Naturally, in the context of homeland security and counterterrorism, some work has to be classified so as not to "communicate our punches" to our adversaries and this is especially true in a military setting. However, homeland security is concerned with solutions to domestic situations and these must be communicated to officials, law enforcement, and the public. Moreover, having experts speak in an open channel is important for informing researchers, academics, and students so that they can work together and increase our collective knowledge.

There are many ways to address homeland security concerns and needs, and many different disciplines and specialties. An ongoing open conversation among experts which will allow them to connect with others and promote collaboration, shared learning and new relationships is needed. Certainly, creating a forum in which theories, approaches, solutions and implications could be discussed and compared would be beneficial. In addition, reliable sources from which experts and lay persons alike could learn about various facets of homeland security are needed. It is equally important that policy and decision makers get the full picture of how much has been done and how much still needs to be done in related areas.

Even in places that have dealt with terrorism for over a century, there are no strong, cost-effective solutions to some of the most pressing problems. For example, from a distance, we have very limited ability to spot a bomb in a car moving toward a building to allow decision making on whether to destroy or divert the car before it can damage the target. Even simpler, the ability to spot a personnel-borne improvised explosive device (IED) in a crowd coming into a busy venue is still beyond our collective capability. Therefore, the bounding of what we know and don’t know needs to be documented.

Finding additional uses for technologies developed originally to solve a homeland security problem is one of the most important aspects of the economics involved. An inescapable issue in many areas of homeland security S&T, is that even a successful solution when applied to only a small market will likely fail because of insufficient returns. For example, building a few hundred detectors for specific pathogens is likely to fail because of limited demand, or it may never even receive funding in the first place. The solution to this issue is finding multiple uses for such devices. In such a case, a chemical detector for contraband or dangerous materials could be used also to detect specific air pollutants in a building; thus, help allergy sufferers. In this way capabilities developed for homeland security may benefit other, more frequently needed uses, thereby making the invention more viable.

The editors of this work have done a superb job of assembling authors and topics and ensuring good balance between fundamentals and details in the chapters. The authors were asked to contribute material that was instructional, discusses a specific threat and a solution, or provides a case study on different ways a problem could be addressed and what was found to be effective. We wanted new material where possible. The authors have produced valuable content and worked hard to enhance quality and clarity of the chapters. And finally, the Wiley staff has taken on the management of contributors with patience and energy beyond measure.

Senior Editor
John G. Voeller

1

CYBER SECURITY METRICS AND MEASURES

Paul E. Black, Karen Scarfone, and Murugiah Souppaya
National Institute of Standards and Technology, Gaithersburg, Maryland

1.1 INTRODUCTION

Cyber security metrics and measures can help organizations (i) verify that their security controls are in compliance with a policy, process, or procedure; (ii) identify their security strengths and weaknesses; and (iii) identify security trends, both within and outside the organization's control. Studying trends allows an organization to monitor its security performance over time and to identify changes that necessitate adjustments in the organization's security posture. At a higher level, these benefits can be combined to help an organization achieve its mission by (i) evaluating its compliance with legislation and regulations, (ii) improving the performance of its implemented security controls, and (iii) answering high-level business questions regarding security, which facilitate strategic decision making by the organization's highest levels of management. This chapter defines some terms, and then discusses the current state of security metrics, focusing on the measurement of operational security using existing data collected at the information system level. This chapter explains the importance of selecting measures that support particular metrics and then examines several problems with current practices related to the accuracy, selection, and use of measures and metrics. The chapter also presents an overview of a security metrics research effort, to illustrate the current state of metrics research, and suggests additional research topics.

1.2 CONTRASTING METRICS AND MEASURES

The term *metric* is often used to refer to the measurement of performance, but it is clearer to define metrics and measures separately. A *measure* is a concrete, objective attribute, such as the percentage of systems within an organization that are fully patched, the length of time between the release of a patch and its installation on a system, or the level of access to a system that a vulnerability in the system could provide. A *metric* is an abstract, somewhat subjective attribute, such as how well an organization's systems are secured against external threats or how effective the organization's incident response team is. An analyst can approximate the value of a metric by collecting and analyzing groups of measures, as is explained later.

Historically, many metrics efforts have focused on collecting individual measures, and given little or no thought as to how those measures could be combined into metrics. Ideally, organizations should first select their metrics, and then determine what measures they can perform that support those metrics. An organization should also have multiple levels of metrics, each geared toward a particular type of audience. For example, technical security staff might be interested in lower-level metrics related to the effectiveness of particular types of security controls, such as malicious code detection capabilities. Security management might be interested in higher-level metrics regarding the organization's security posture, such as the overall effectiveness of the organization's incident prevention and handling capabilities. Lower-level metrics facilitate making more tactical decisions, whereas higher-level metrics are well suited for making more strategic decisions. The lower-level metrics are often used as input to the higher-level metrics.

Organizations can use measures and metrics to set goals, also known as *benchmarks*, and determine success or failure against the benchmarks. For example, suppose that an organization determines that 68% of its systems are in compliance with a particular policy. The organization could set a benchmark of 80%, implement changes in its practices to increase compliance, and then measure compliance again in six months to see if the benchmark has been achieved. Benchmarks are organization specific and are typically based on baselines from an operational environment.

1.3 SELECTING MEASURES TO SUPPORT METRICS

Once an organization has identified its metrics, it then needs to determine what measures can feasibly be collected to support those metrics. Organizations should favor measures that can be collected via automated means because they are more likely to be accurate than manual collection (e.g. self-evaluation surveys) and can also be collected as often as needed. Organizations should also seek opportunities to use existing data sources and automated collection mechanisms because of the cost of implementing and maintaining new systems and software simply for data collection purposes.

As measures are collected, organizations need a way to analyze them and generate reports for the metrics they support. Organizations can analyze the measures and metrics in many ways, such as grouping them by geographic location, logical division within the organization, system type, system criticality, and so on. Some organizations use products that roll up measures into metrics and present the metrics in a security dashboard format, with the measures underlying each metric available through drill-down. This allows a dashboard user to see the values of the presented metrics and changes in those metrics over time, as well as to examine the metrics and measures comprising those metrics.

1.4 PROBLEMS WITH THE ACCURACY OF MEASURES

The accuracy of a metric is by definition dependent on the accuracy of the measures that comprise the metric. Organizations currently face several problems related to the accuracy of measures. One problem is that measures are often defined imprecisely. Consider the percentage of systems that are fully patched: does this only include operating system patches or does it also include service and application patches? Does it only mean that the patches have been installed or that subsequent actions necessary to activate the patch (such as rebooting the system or changing configuration settings) have also been performed? Another issue with measure definition is the terminology itself, such as measuring the number of port scans performed. What is the minimum number of ports that must be scanned in a port scan? If an attacker scans ports on 100 hosts, is it one port scan or 100 port scans? If the attacker performed the same scan but only scanned one host each day, is it one port scan or 100 port scans?

Measuring port scans is also a good example of a related common problem—inconsistent measurement methods. Port scans are often identified by intrusion detection systems (IDSs), but each IDS uses its own proprietary algorithms for identifying port scans, so activity identified as a port scan by one IDS may not be identified as such by another. This causes inconsistency in measurement if the organization uses multiple IDSs or if a single product is in use but its sensors have different port scan settings (e.g. the minimum number of ports in a scan or the maximum length of time to track a scan). Another example is system patch status—one operating system might report only on operating system patches, while another operating system might also include some application patches. In such a case, an organization could use multiple measures instead of one, with each measure corresponding to a different measurement method, and then combine the measures into a metric that approximates the collective values of the measures.

Many instances of problems with imprecise measure definitions have been mentioned in the security community, but to date no concerted effort has been made to exhaustively gather information on these problems, document it, and make it available to the security community. It would be much more helpful to identify the factors that organizations should consider when defining their measures than to attempt to provide a single definition for each measure. The best definition for an organization is driven by what the organization is trying to accomplish. For example, in the patching example mentioned above, an organization might be trying to gain insights on general patch distribution and installation practices to verify that all applications deemed critical by the organization have been patched or to verify that the organization's patch management software is functioning properly (e.g. patches are installed and operate properly based on a predefined schedule).

Another common problem with the accuracy of measures is the use of qualitative measures. As mentioned earlier, data collection methods such as self-evaluation surveys often produce inaccurate or skewed results, depending on the types of questions asked. For example, if users or administrators are asked if their systems comply with the organization's policies, they are very likely to say that they do. It would be more accurate to instead use quantitative measures that assess the systems' compliance. Qualitative measures that do not have well-defined scales or units of measure can be particularly problematic in terms of accuracy. For instance, asking a user to rate the reliability of their computer on a scale of 1–5 where 1 is simply defined as "poor" and 5 as "excellent" is subjective and imprecise. A qualitative measure may be useful if each rating is defined clearly without overlap between the ratings, so that different people, when given the same information, would be likely to assign the same rating. An objective scale might be 5—no crashes or hangs in six months, 4—one crash or hang, 3—two or three instances, 2—four to six instances, and 1—more than six instances. The rating may still be somewhat subjective because it is dependent on the user's recall or because "crash" and “hang” are not defined. Nonetheless, this qualitative measure is more precise than “poor” to “excellent”.

Some measures are also considered qualitative because they provide absolute counts without a context, norm, or goal. For example, a measure that indicates that 100 attacks were attempted has no context. What is the period of time? Is 100 a lot or a little? A measure that indicates that 100 attacks were attempted out of 1,000,000 incoming Web server connections adds context.

Context is very important to measures and metrics. Most measures individually have little meaning. Even the example above—the number of attempted attacks per million incoming Web server connections—does not have much meaning by itself. Is the rate of attempted attacks rising, falling, or staying steady? Have any changes been made to the organization’s security controls that would change how effectively they can detect attacks or has there truly been a change in the number of attacks? Do changes in the rate of attempted attacks correspond to observations about attack trends reported by other organizations? A single measure may need to be analyzed in context with several other measures, as well as separate events such as security control changes and external trends, to determine its true significance. It would be helpful to organizations to have additional information compiled on the relationships between measures and between measures and separate events, particularly if it includes empirical information based on analyses of real-world operational environments.

Also, because cyber technology is so dynamic, the meaning of measures and metrics changes over time. For example, a measure may have shown an increase in attacks succeeding last year, and the organization determined through other measures and knowledge of external events that this was primarily due to an increase in phishing attacks. This year antiphishing technologies are deployed, but the success rate for attacks continues to increase. Is this due to improved attack techniques, improperly configured antiphishing technologies, inadequately trained users, or other factors? Next year, there may be additional factors that influence the significance of the measure, as well as different relative importances for the existing factors.

1.5 PROBLEMS WITH THE SELECTION OF MEASURES

Most organizations have many existing sources of security measures, automatically generated by enterprise security controls such as antivirus and antispyware software, intrusion detection systems, firewalls, patch management systems, and vulnerability scanners. There may be accuracy issues with some of these measures, but this can still leave an organization with many existing measures from which to choose. Organizations could also create additional measures such as utilities to extract information from security logs, but there may be considerable cost in creating and deploying software and, in some cases, entire systems to collect such measures.

Some organizations collect many measures under the assumption that it is better to have more information than less information, or because it is easier to collect a lot of measures than it is to create a set of metrics and then determine which measures support those metrics. Collecting measures without evaluating their usefulness and having a plan for how to use them has several disadvantages. Firstly, it can waste considerable time and resources to collect, analyze, and report measures: only the measures that support the organization-selected metrics are generally needed. Secondly, if the measures are not selected and organized so that the dependencies between the measures are clear and accurately represented in the corresponding metrics, analysis of the measures and related metrics is likely to generate misleading results. Thirdly, it often causes people involved in the measure collection process to feel that the effort is a waste of time, because it is unclear what value there is in collecting so many measures. Another reason is that if people are allowed to choose which measures they will collect and share with others, they are more likely to collect measures that demonstrate positive results (e.g. 100% of desktop computers have antivirus software installed) than measures that demonstrate negative results (e.g. 15% of antivirus software installations are up to date).

Currently, there are many suggestions in the security community for what measures organizations should collect. However, little work has been done to determine the value of these measures in real-world operational environments, including which measures are most supportive of particular metrics. For example, suppose that an organization wants a metric for how effectively its security controls detect and stop attacks. Dozens, if not hundreds, of measures that could support this metric have been suggested by the security community, but little research has been done as to which of those measures are most closely correlated with the metric. If the characteristics of real-world operations were studied and analyzed, it may become apparent which measures are most indicative of the overall security posture and which measures are of little or no value. It might also be possible to approximate a metric by using just a few carefully selected measures.

1.6 PROBLEMS WITH THE USE OF MEASURES

In addition to issues with measure accuracy and selection, many organizations also face challenges involving the use of measures. Some of these challenges, such as ensuring that the selected measures support the determination of the chosen metrics, have been discussed earlier. Another common challenge is determining how to combine the values of the measures into a metric. The measures may use different units of measurement, have different scales, and have varying precision; these issues can be addressed through careful creation of equations to combine the values. Also, some of the measures may be more important than others in the scope of the metric; however, it is often difficult to quantify what weight each measure should be given. Empirical research in this area could provide organizations with a factual basis for weighting measures instead of either guessing or weighting each measure equally.

Organizations need to recognize that over time, they will need to alter their measures and metrics. Although high-level metrics may stay the same, low-level metrics need to change over time as the security posture of the organization changes. For example, an organization with relatively immature security practices may need to initially focus on measures and metrics involving its most basic security controls, such as what percentage of computers are protected by antivirus software. As the organization's security controls mature, these metrics may become less useful, and the organization may want to answer new questions, such as how effective its security controls are at stopping malware. This may require the development of new metrics and corresponding measures, and the collection of the old metrics and measures can be stopped if the organization no longer finds them to be of value.

1.7 COMMON VULNERABILITY SCORING SYSTEM (CVSS)

To better illustrate the current state of research on security metrics, we will examine an ongoing research effort for metrics that indicate the significance of vulnerabilities in systems. The Common Vulnerability Scoring System (CVSS) is a standard for assessing the severity of flaws in operating system and application software [1]. CVSS is composed of three sets of measures: base measures that are constant over time, temporal measures that change over time but are the same for all environments, and environmental measures that may be different for each environment. There is a different equation for each set of measures, and the result of each equation is a score—a base, temporal, or environmental score—that is in essence a metric. The measures are for particular characteristics of each vulnerability, such as whether it can be exploited remotely (over a network) and to what degree the confidentiality, integrity, and availability of a target could be impacted. The score metric is intended to give a general indication of the relative severity of the vulnerability.

The initial version of the CVSS measures, equations, and metrics were released in 2005. On the basis of feedback from their real-world use, particularly examination of empirical measure and metric data by security experts, deficiencies in the CVSS standard were identified. The measures, equations, and metrics, as well as the corresponding documentation, have all been revised to make the measures more consistent and to improve the accuracy of the metric scores. Version 2 of the CVSS standard was released in mid-2007.

CVSS is most commonly used by organizations to prioritize their vulnerability mitigation activities, such as applying patches to systems. However, researchers are investigating other uses for CVSS. For example, work has been done at the National Institute of Standards and Technology (NIST) on using CVSS to determine metrics for security-related software configuration settings [2]. Researchers at Veracode are looking at using CVSS to rate software weaknesses [3]. There is also interest in bringing CVSS scores down from the enterprise level to the individual system level so that CVSS could be used to help assess the overall vulnerability of individual systems. To accomplish this, considerable research and empirical validation are needed for applying CVSS to software configuration settings and weaknesses, as well as a new way of measuring the strength of security controls on individual systems.

1.8 RESEARCH DIRECTIONS

Literature contains hundreds of measures. Research is needed to validate connections between measures and security, determine correlations, and model effects. Although there is some readily accessible data, for example, National Vulnerability Database [4] and A Chronology of Data Breaches [5], such research and analysis require more and higher-quality data. State laws requiring companies to notify consumers of data breaches have the benefit of supplying data. Additional information can come from developing or articulating motivations for organizations to share information, as is the practice for business case studies or the airline industry.

Beyond measures, a secure nation needs research to understand which metrics lead to higher security, the measures supporting those metrics, and analytical methods to aggregate measures.

REFERENCES

1. Common Vulnerability Scoring System. (2008). *Forum for Incident Response and Security Teams (FIRST)*, http://www.first.org/cvss/.
2. Scarfone, K. and Mell, P. (2008). The Common Configuration Scoring System (CCSS) (Draft), NIST, NIST Interagency Report 7502, http://csrc.nist.gov/publications/PubsNISTIRs.html.
3. (2008). https://securitymetrics.org/content/attach/Metricon2.0/Wysopal-metricon2.0-software-weakness-scoring.ppt.
4. National Institute of Standards and Technology (NIST). (2008). *National Vulnerability Database*, http://nvd.nist.gov/.
5. Privacy Rights Clearinghouse. (2008). *A Chronology of Data Breaches*, http://www.privacyrights.org/ar/ChronDataBreaches.htm.

FURTHER READING

Corporate Information Security Working Group. (2005). *Report of the Best Practices and Metrics Teams*, Subcommittee on Technology, Information Policy, Intergovernmental Relations and the Census, Government Reform Committee, United States House of Representatives, 17 November 2004, revised 10 January http://www.educause.edu/ir/library/pdf/CSD3661.pdf.

Dorofee, A., Killcrece, G., Ruefle, R., and Zajicek, M. (2007). *Incident Management Capability Metrics*, Version 0.1, Software Engineering Institute/CERT, http://www.cert.org/archive/pdf/07tr008.pdf.

Herrmann, D. S. (2007). *Complete Guide to Security and Privacy Metrics*, Auerbach Publications, Boca Raton, FL.

Jaquith, A., (2007). *Security Metrics: Replacing Fear, Uncertainty, and Doubt*, Addison-Wesley, Upper Saddle River, NJ.

McCurley, J., Zubrow, D., and Dekkers, C. (2007). *Measures and Measurement for Secure Software Development*, Software Engineering Institute, https://buildsecurityin.us-cert.gov/daisy/bsi/articles/best-practices/measurement/227.html.

Safety and Security Measurement Technical Working Group. (2005). *Security Measurement*, Practical Software and Systems Measurement (PSM), http://www.psmsc.com/Downloads/Other/Security%20White%20Paper%202.0.pdf.

(2008). securitymetrics.org, http://www.securitymetrics.org/.

Chew, E., Swanson, M., Stine, K., Bartol, N., Brown, A., and Robinson, W. (2008) *Performance Measurement Guide for Information Security*, NIST, NIST SP 800-55 Revision 1, http://csrc.nist.gov/publications/PubsSPs.html.

Conferences and workshops

(2008). *Third Workshop on Security Metrics (MetriCon 3.0)*, http://www.securitymetrics.org/content/Wiki.jsp.

(2008). Security measurements and metrics. *Fourth International Workshop on Quality of Protection (QoP 2008)*, http://dit.unitn.it/∼qop/.

(2005). *IEEE International Symposium on Software Metrics*, http://www.informatik.unitrier.de/∼ley/db/conf/metrics/ (latest year was 2005).

2

MULTILEVEL SECURITY

Cynthia E. Irvine
Naval Postgraduate School, Monterey, California

2.1 INTRODUCTION

Multilevel security (MLS) refers to policies and techniques where the sensitivity of the information is immutably bound to an equivalence class. (One can think of *equivalence classes* as subsets of a set where there is no overlap or intersection among the subsets. For example, pens could be subdivided into red pens, blue pens, black pens, green pens, and so on. Information might be subdivided into *CRITICAL* and *NONCRITICAL* information or *PUBLIC* or *PROPRIETARY* information.) The active entities that access the information are also statically associated with equivalence classes. On the basis of the relationships between the equivalence classes, rules determine whether and with what rights an active entity can access the information. The mandatory policies associated with MLS can apply to integrity as well as confidentiality. Specific models and mechanisms have been developed to support MLS in computer systems. Requirements for multilevel secure systems span the private sector, the government, and the military.

2.2 BACKGROUND

Most organizations maintain information that is either protected or openly available. In government, information often is categorized as either classified or unclassified. Within the context of classified information, various levels of information sensitivity may be established based upon the damage caused should that information become accessible to adversaries. The more grievous the damage resulting from unauthorized access, the more sensitive the information. For example, the recipe for Uncle Joe's secret sauce may be considered critical to the continued well being of a producer of barbeque sauce: it must neither be revealed to competitors, nor should be corrupted by changing the proportions of the ingredients. Physical documents containing sensitive information are protected through a variety of physical and procedural controls. Computer systems introduce new challenges.

Throughout the 1960s, as multiprocessing computer systems evolved, it became evident that the separation provided by the resource management mechanisms of typical

Cyber Security, Edited by John G. Voeller

operating systems was insufficient to prevent highly sensitive information from becoming accessible to unauthorized individuals. These controls were so inadequate that instead of utilizing the power of multiprocessing, classified information processing was conducted separately. At times, this meant that those with classified tasks had to wait until after hours, when the system could be dedicated to processing the sensitive information. Following the completion of the classified tasks, the system was purged of all sensitive information and restored to unclassified activity. This is what is called *periods processing*. If the amount of classified processing merited the additional expense, a *dedicated system* might be allocated to sensitive tasks.

Both these approaches were insufficient to meet the requirements of organizations that depended upon rapid access to information for military command and control. Periods processing could result in unacceptable delays and dedicated systems incurred both the expense of additional equipment and a high cost of ownership in terms of system maintenance and support personnel. If simultaneous processing at several classification levels, such as *CONFIDENTIAL*, *SECRET*, and *TOP SECRET*, was required, then the resources for either periods processing or dedicated systems could be inadequate. In addition, these approaches could be wasteful if the computing resources allocated to particular classification levels were underutilized.

In organizations where access to a broad spectrum of information is required for making informed decisions, the temporal and spatial separation of information with various sensitivities afforded by periods processing and dedicated systems was more than inconvenient: it could mean the difference between victory and defeat, life or death. Those at the management level wanted computer systems that would mimic the kind of access to information possible when using physical documents: timely simultaneous access to both classified and unclassified information, by properly authorized individuals.

MLS addresses these requirements. To understand MLS, it is necessary to understand the nature of the policies to be enforced, the challenges associated with enforcement of those policies in automated systems, how multilevel systems and networks are implemented, current approaches to MLS systems, and emerging technologies for MLS systems.

2.3 MULTILEVEL SECURITY POLICIES

Security policies are embodied in the laws, procedures, and rules used to manage and protect information. In general, policies reflect an organization's requirements for information confidentiality, integrity, and availability. MLS is applicable to both confidentiality and integrity policies. An organization may use labels to associate a particular sensitivity level with particular piece of information, and people are vetted for access to sensitive information through checks that result in some form of authorization. For example, extensive and costly background checks are required to vet individuals as sufficiently trustworthy to merit access to *TOP-SECRET* information. Individuals lacking appropriate authorization will be unable to access any sensitive information, whereas those with many or high authorizations have access not only to nonsensitive information but also to highly sensitive information. MAC policies are policies that are both global in scope and persistent in time; users cannot override the policy during normal use. Sometimes mandatory policies are called *nondiscretionary* policies; the two terms are equivalent. In contrast, *discretionary access control* policies permit modification of the

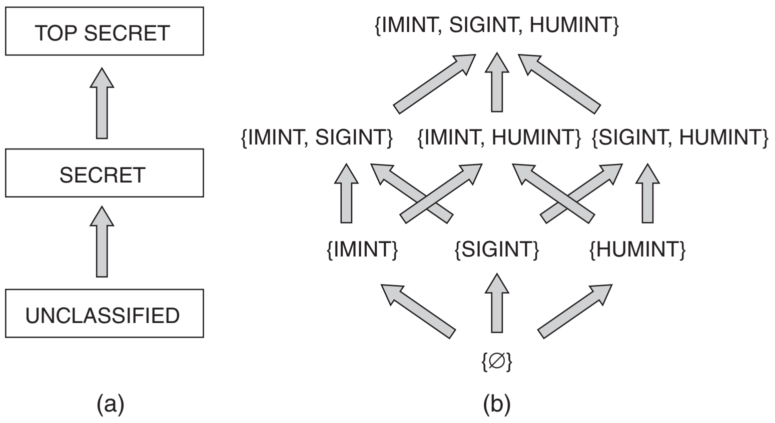


FIGURE 2.1 A hierarchical ordering of classes is shown in (a). In (b), a set of noncomparable classes is depicted. Arrows show the allowed direction of information flow.

rules pertaining to access to information: a run-time interface is provided through which properly authorized users may modify policy. Consequently, it is up to the discretion of the individual to determine who will have access to information. A test for determining whether a policy is mandatory or discretionary is to examine the punishment associated with its violation [1]. Disclosure of state secrets can result in prison or firing squads, whereas violation of discretionary policy may only result in a reprimand.

Sensitivity levels are identifiers for equivalence classes of information and are based upon the secrecy and integrity attributes of the information. The choice of equivalence classes is up to the organization. For a private enterprise, the sensitivity levels might be *PROPRIETARY* and *PUBLIC*, whereas a military organization might choose *SECRET*, *CONFIDENTIAL*, and *UNCLASSIFIED*. Consider a few examples.

In a large company, only personnel in the *PRODUCT-RESEARCH* group may have access to *PRODUCT-RESEARCH* information, whereas only personnel in *CORPORATE-STRATEGY* group may access the *CORPORATE-STRATEGY* for next year. Information on the company web pages is *PUBLIC* and is readable by anyone, although the company is likely to restrict write access to its webmasters and system administrators. Management determines the membership of the respective groups. Lipner provided a discussion of the applicability of MAC policies in the commercial sector [2] and concluded that a very large number of labels would be required when many enterprises were involved. Military organizations may organize classified information into *TOP SECRET*, *SECRET*, and *CONFIDENTIAL* levels, and all nonsensitive information is *UNCLASSIFIED*. Individuals are given background checks and are assigned clearances such as *TOP SECRET* and *SECRET*.

There may be a hierarchical relationship between the major equivalence classes. For example, a user cleared for *TOP SECRET* is able to access *TOP SECRET*, *SECRET*, and *UNCLASSIFIED* information. In the corporate example, everyone in *PRODUCT-RESEARCH* may be allowed to access both *SHIPPING* and *PUBLIC*. Often an organization may impose further granularity on its access controls by imposing a mandatory *need-to-know* policy. Additional metadata is associated with both subjects and objects to reflect mandatory need-to-know policies. For example, an individual cleared for *TOP SECRET* may be vetted for access to information that is in special compartments such as *Imagery Intelligence* (*IMINT*), *Signals Intelligence* (*SIGINT*), and *Human Intelligence* (*HUMINT*). Such labels are commonly used by the intelligence community where the work of analysts is compartmented so that individuals have access only to the information required to do their job. These mandatory policies reflect a requirement to enforce the notion of least privilege [3]. Figure 2.1 shows both a hierarchical ordering of access classes and classes created from combinations of noncomparable attributes. For a mandatory confidentiality policy, information is allowed to flow from lower classification levels or combinations to higher ones. In the latter, any classes may receive information from classes with attributes that are a subset of its own.

A common misconception is that MLS applies only to the enforcement of mandatory confidentiality policies. MLS may also be used to enforce mandatory integrity policies. The semantics of integrity are as follows. If information is of high integrity, then it must not be corrupted by low integrity information. However, adding high integrity information to low integrity information does no harm, although there is no real improvement unless the low integrity information is somehow cleaned up. For example, the Royal Observatory at Greenwich, England, is a reliable and high integrity source for astronomical time, while an amateur sundial might be a relatively low integrity time source. Thus high integrity information should be readable by everyone, whereas low integrity information should be confined and readable within quarantine-like processes. Note that processing of high integrity information by low integrity software, whether being executed by a high integrity process or not, lowers the integrity of that information [4].

The sensitivity labels form a set of equivalence classes that can be organized according to the information flow intended in the system. Denning showed that these equivalence classes can be represented mathematically with respect to the flow of information via read and write operations [5]. This work demonstrated that all mandatory policies could be simply combined and that the resulting sensitivity classes could be easily compared with one another.

For both the corporate and the military examples, the policy regarding access to information is inflexible with respect to location. Even though the corporation may have offices across the world, only *PRODUCT-RESEARCH* group personnel may handle *PRODUCT-RESEARCH*; and, analogously, military access to *TOP-SECRET* information requires a *TOP-SECRET* clearance regardless of the location. These policies are also temporally inflexible. Information marked as *PRODUCT-RESEARCH* does not become *PUBLIC* at certain times of the week while it is marked *PRODUCT-RESEARCH* the rest of the time, just as the classification of information designated *TOP SECRET* does not become *UNCLASSIFIED* just for the weekends. Thus, we say that mandatory policies are *global* and *persistent*.

The global nature of MAC policies does not dictate that all components of a multilevel system, whether individual machines or elements of a highly distributed network, must enforce all policies. The system may be organized so certain components only process information at a single classification level and need not be required to enforce mandatory policy. Such components are *single level*.

In practice, certain information once highly classified may after some time be downgraded and released to the public. Certain situations may require the rapid *regrading* of selected intelligence information so that it is available to operational personnel. This might occur during military operations or in disasters involving first responders who are typically not cleared for access to sensitive information. Similarly low integrity information may be analyzed and judged qualified for use in a high integrity context. For example, integrity regrading might take place in the creation of software for a manned space mission. Initially, the software might be labeled *DEVELOPMENT*, but following proper analysis, it could be upgraded to *MANNED_FLIGHT*. These examples illustrate that the notion of persistence is not absolutely rigid; however, the processes used to regrade information, whether procedural or automated, must be carefully controlled.

Two techniques are commonly used to enforce mandatory policies. The first is physical and the second, logical. Enforcement of mandatory policies in a context without computers involves only individuals and documents. Individuals who have been properly vetted to handle sensitive documents are trusted to ensure that the sensitive information contained in those documents is not transferred to documents with inappropriate sensitivity markings. Organizations go to great lengths to ensure that corrupted insiders or spies do not cause the release of sensitive information. Physical protection can ensure that sensitive information is accessible only within confined spaces. Locks and guards ensure that only authorized individuals enter the sensitive enclave, and logging of entry and exit to the space as well as check-in and check-out of information while in the space deter malicious insiders. Special construction methods may protect the enclave from eavesdropping technologies.

Because an *organizational security policy* may be stated in very general terms, its translation to a form that can be implemented in computer systems results in an *automated security policy* [6]. This articulation of the policy permits access to information to be described in terms of the active and passive entities of the system. Ultimately this translates to execution of an instruction by the CPU that accesses resources managed by the system.

The active system entities, *subjects*, access information contained in passive entities, *objects*. Subjects are not the cleared individuals themselves, but generally are processes that execute software [7]. Subjects are surrogates for users who may be either administrators or normal users. The mechanism that implements and enforces the MAC policy exports subjects and objects at its interface. The enforcement mechanism binds a sensitivity level, either implicitly or explicitly, to each subject and object.

2.3.1 Confinement

In the world external to computers, individuals use their judgment to ensure that only authorized individuals have access to information. Subjects within a computer are programs in execution. Programs do not exercise judgment and thus may violate the intended policy if they are programmed to do so. An example of this problem is a *Trojan Horse*, that is, clandestine malicious software placed within an application. While the user innocently uses the application, the Trojan Horse abuses the user's privileges to cause unauthorized information flow so that it may be accessed by unauthorized individuals.

To better appreciate the Trojan Horse problem, consider a system that enforces a discretionary policy using access control lists (ACLs). Each process has a binding to the individual upon whose behalf it is running, and a separate ACL is associated with each file and directory. Each ACL contains a list of individuals and their access rights to the object.

Suppose that Alice has *PRODUCT-RESEARCH* information in the file called *research-stuff* and that the ACL on *research-stuff* allows read and write access only to Alice and Bob, who are both members of the *PRODUCT-RESEARCH* team. Elmo is in the shipping department and has a file called *shipping-stuff* with an ACL that allows

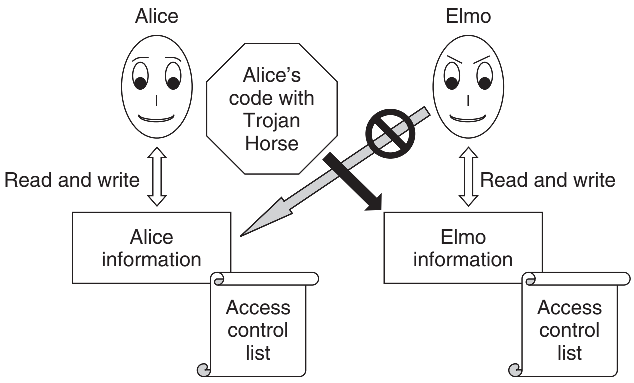


FIGURE 2.2 A Trojan Horse executing in Alice's code is able to write to Elmo's information, thus circumventing Alice's intent to block Elmo's read access to her information.

anyone in the company to have read and write access to it. In theory, Alice could use her computer to read information out of *research-stuff* and write it into *shipping-stuff*, but Alice is a good employee and would not do this. What Alice does not know is that her program, *research-SW*, contains a Trojan Horse. While she uses *research-SW* in the normal course of her job, it is clandestinely writing *PRODUCT-RESEARCH* information to Elmo's file, as illustrated in Figure 2.2. Elmo, a corporate spy, will sell this information to competitors who are stealing the intellectual property of Alice's company in order to dominate the marketplace. Of course, since a system enforcing discretionary access controls has a run-time application programming interface that may change the policy, it is possible that the Trojan Horse in research-SW might change the ACL on research-stuff, thereby allowing direct read by Elmo. However, the first approach is somewhat preferred as it is less likely to be detected by an auditing mechanism.

Integrity Trojan Horses are also possible. In this case, an integrity Trojan Horse would write to a high integrity object. An example of an integrity attack might be modification of critical avionics information on a commercial aircraft.

In an automated system, the mandatory policies can be enforced in a way that will prevent Trojan Horses from causing unauthorized information flows. Thus, unlike the world of people with judgment and paper, that of subjects and objects is constrained by rules that thwart attempts to violate policy.

Users set their session level before starting work on an MLS system. This sets the label on any subjects that will act on behalf of the user. Normally these subjects are single level. The systems' objects possess immutable labels indicating a single sensitivity level. Subjects may have access to objects with labels different from their session levels, but in the case of confidentiality they are neither authorized to read from objects that are more sensitive, nor permitted to write to objects less sensitive than the current session level. The former rule reflects the laws and practices imposed in the world of people and paper, whereas the latter is imposed specifically to prevent either accidental or malicious flow of information from high to low sensitivity. This is called *confinement*, and in formal security policy models is called the *confinement-property* (also called the **-property*) [8]. The rules for enforcement of integrity policies have an opposite symmetry. Subjects are permitted to read information of higher integrity and to write to low integrity objects, but they may not read low integrity information or write to higher integrity objects.

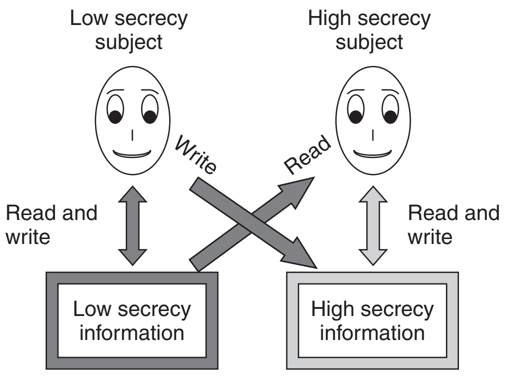


FIGURE 2.3 High confidentiality subjects have read access to low confidentiality information; at the same time mandatory policy prohibits the flow of high information to low.

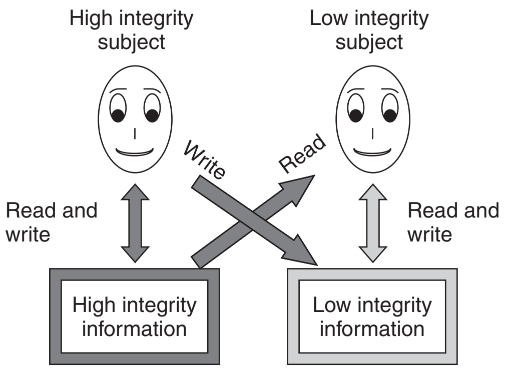


FIGURE 2.4 Mandatory integrity policies prevent corruption of high integrity information by low integrity subjects, while low integrity domains benefit from high integrity information.

Mathematical models allow precise articulation of the properties to be enforced when mandatory policies are required. The Bell and LaPadula model [9] provides a formal representation of a mandatory confidentiality policy and the Biba model describes a mandatory integrity policy [9]. Figures 2.3 and 2.4 illustrate the rules of the Bell and LaPadula and Biba models, respectively.

The SeaView model was the first to illustrate that a single set of equivalence classes could be used to enforce combined secrecy, integrity, and least privilege policies in a system with mandatory controls [10]. The least privilege policy, implemented using protection rings [11], addressed process-internal integrity. Extensions to this work have been presented in the context of multilevel secure smart cards [12].

2.3.2 Supporting Policies

In an operational system, there must be a binding between the real user and the subjects acting on behalf of the user [3]. First, a user must be identified to the system, and then the user must be authenticated to the system by presenting something that only come from that user. This might involve a password, token, biometric factor, or some combination of these.

Once logged in, the user must set a session level. To accomplish this, the password file might be augmented to contain the maximum sensitivity level, such as a clearance, at which the user can work. Although this maximum sensitivity level might be possible, users may wish to work at lower levels and must select a *session level*. For example, Jane, who is cleared to *TOP SECRET*, could select any one of the following session levels for her current session: *TOP SECRET*, *SECRET*, or *UNCLASSIFIED*. Suppose the user logs on at *SECRET*, then, when a subject is instantiated on behalf of the user, one of its attributes will be the user's current session level, for example, *SECRET*.

The implementation of an identification and authentication policy requires a *trusted path*. The idea behind the trusted path is that the user is trustworthy, as is the identification and authentication mechanism. So, both the system and the users require an unforgeable connection that assures the user protected communication with the trusted system that communication is with the user and not a man-in-the-middle or some other malicious entity. Users invoke the trusted path using a *secure attention key*: a single key or some special combination of keys designated solely for the purpose of establishing a trusted path.

Since the attributes bound to subjects acting on behalf of users are the basis for access control decisions, it is clear that a well-defined identification and authentication policy is essential for multilevel systems.

Audit provides a record of security-relevant events. If bound to a rule-checking mechanism, audit may provide alerts of impending security violations. Policies must be established to determine what should be audited. For example, one might choose to audit all accesses to a particular object, but to no others; all activity on the system could be audited; the activities' subjects at a particular sensitivity level might be recorded; the use of selected system calls could be audited; and so forth. It is important to audit the activities of the security administrator and other trusted individuals so that a record of security-critical activities can be maintained. Also, good audit reduction tools are needed, otherwise voluminous audit records are not likely to be particularly useful.

In systems enforcing mandatory policies, the management of labels and the labeling of information being transferred into and out of the system should be reflected in supporting policies. For example, there may be a requirement that all printed documents contain markings in their headers and footers indicating whether the document is CORPORATE-STRATEGY or PUBLIC.

System support is also required for the enforcement of administrative policies, such as user account management and security configuration, including the configuration of mandatory sensitivity levels.

2.3.3 Trusted Subjects

To perform many of the tasks associated with supporting policies as well as to provide functionality such as regrading, systems enforcing MLS must support subjects that have the ability to both read and write to a label range that spans multiple sensitivity levels [13]. Trusted subjects do not violate the MLS policy, but are explicitly part of the system and adhere to a relaxed MLS policy. To ensure that the information flows resulting from the actions of trusted subjects are not in violation of the intent of the overall system policy, the code executed by these subjects must be analyzed for possible incorrect or malicious execution. As a result of this analysis, we can say that they are trustworthy [14]. A challenge associated with the design and analysis of trusted subjects has been the modeling of their correct behavior [15].

Trusted subjects execute at the login interface to allow each user to set a session level and create untrusted, single-level subjects at that session level. They are also used by security administrators to set the levels associated with single-level devices. For multilevel devices where the sensitivity level of incoming or outgoing information can require the use of explicit sensitivity labels, trusted subjects ensure that correct label-to-information bindings occur. Another service performed by trusted subjects is in regrading information. Many organizations enforce mandatory policies; however, operationally, they also require mechanisms where exceptions to these policies may be implemented. For example, consider a system that encrypts information before transmitting it on the network. The process of encryption can transform bits that represent proprietary sensitive information into bits that represent no information and, thus, can be seen by anyone. In essence, encryption is *downgrading* the information, viz., changing its sensitivity level from high to low. Modern systems may require additional downgrading functions that move certain information from high networks or repositories to low ones. A downgrader or *guard* will consist of a trusted subject that is able to read information from a high sensitivity level, for example, TOP SECRET, and write that information to an object at a lower sensitivity level, for example, SECRET.

Sometimes sensitive information is scanned for certain critical words that are then expunged from the data. This is called *sanitization*. A danger in downgrading systems is *steganography* [16]. Steganography, the art of hidden writing, involves a secret encoded by malicious code in seemingly innocuous data so that it is not visible to the casual observer. Clearly, filtering and regrading must be conducted using systems for which there is a very high confidence that only the correct actions will be taken.

Note that in the regrading example given above, the labels on the subjects and objects are immutable. A trusted subject does not change the label of the object, but rather reads the information from the source object and writes it to a destination object that has a different sensitivity level. This is an example of *tranquility*. Because of its lower complexity, label tranquility for both subjects and objects has been considered to be an essential aspect of highly trustworthy implementations of MLS policies.

The construction of *trusted systems* represents one of the great challenges in security modeling and engineering. Underlying operating system controls are used to enforce mandatory policies on its processes and, at the same time, permit *trusted* applications to be subject to relaxed mandatory constraints.

2.4 ENFORCEMENT OF MULTILEVEL SECURITY POLICIES

When constructing an MLS system, several approaches are possible. Threats will determine system requirements, which include those that specify functionality and those that determine its trustworthiness.

2.4.1 Design Approaches

To enforce policy using physical mechanisms, one must construct a separate single-level network for each sensitivity level. All users must be authorized for the sensitivity level of the network and all information created and managed in that network must be considered to be at the network's sensitivity level. The advantages of single-level systems or networks include the ability to identify and manage the access to information in a manner that is easy to understand. An isolated network may be maintained in a special facility and only authorized users may be granted access to the premises. Although costly, extremely critical information may warrant the protection afforded by isolated networks and systems.

In the case of physically isolated systems, users must either move from room to room to access different networks or they may have multiple systems on the desktop. The latter can lead to clutter and confusion. The user could use a keyboard, video, mouse (KVM) switch to minimize consumption of desktop space; however, multiple processors are still required and the possible advantage of simultaneously seeing information at different sensitivity levels is lost.

If nonsensitive information can be moved to networks of higher sensitivity, but without a highly trustworthy binding of sensitivity labels associated to the information, the users cannot distinguish nonsensitive from sensitive information. Also, to share nonsensitive information with individuals having lesser authorizations, users must go back to the nonsensitive system. If a user wishes to transmit nonsensitive information directly from the more sensitive enclave, complex procedures including automated guards are required, and the possibility of steganography [16] and other techniques for clandestine information hiding must be addressed.

Logical isolation depends upon an underlying mechanism that enforces the security policy. Because mandatory policies can always be characterized by comparisons between equivalence classes, it is possible to construct a relatively simple mechanism to determine whether a particular subject may have access to a given object. Systems that enforce logical isolation can provide users with a coherent view of all information at or below their sensitivity level. They also allow users to log into the system at any sensitivity level at or below their maximum authorization or clearance. Thus, from a single system, an authorized user is able to both access company proprietary information when logged in at *PROPRIETARY* and the Internet when logged in at the *PUBLIC* sensitivity level.

2.4.2 Threats to MLS Systems

The critical nature of information to be protected in MLS systems requires that threats to correct policy enforcement be carefully examined before system development. This allows requirements to be elicited that will ensure threats are eliminated as part of a carefully articulated system life cycle process.

Threat analysis reveals that there are two broad classes of threats: *developmental threats* and *operational threats* [17]. Developmental threats include the introduction of flaws into the system through mistakes in design and implementation and through deliberate system subversion. The former introduces exploitable flaws, whereas the latter introduces trapdoors.

Operational threats occur when the system is in use. Adversaries can include malicious insiders as well as external activities. The mechanisms that have been designed into the system are intended to counter operational threats; however, system security also depends upon adequate user and administrator training, as well as good configuration management and system maintenance. Constructive techniques counter developmental

TABLE 2.1 Examples of Errors Resulting in Security Flaws

General Error Category	Example
System design errors	Absence of least privilege
	Inappropriate mechanism for shared objects
	Poor choice of data types
Design errors	Error recovery results in exploitable side effects
	System modifications that deviate original intent of security mechanisms
Implementation errors	Buffers sizes are not checked, resulting in “buffer overflow”
	Failure to initialize variables
	Absent parameters are erroneously assumed
User interface errors	Gratuitous active execution
	Passwords too short
	Default access control lists are too permissive
Configuration errors	Insecure defaults render the system vulnerable
	Critical resources remain unprotected because of bad configuration choices

threats, and systems must be designed and implemented so that operational threats are addressed.

A large number of failures in policy enforcement result from the presence of unspecified functionality in systems, for example, vulnerabilities in the form of system flaws and unintended artifacts that permit an adversary to bypass the policy enforcement mechanism of a system. These failures in design and implementation can be exploited by adversaries intent on gaining system privileges for the purpose of avoiding the constraints of the protection mechanism. Flaws range from inadequate bounds checking of interface parameters to pathological interactions between synchronizing processes. Such flaws were identified by Anderson in 1972 [18] and are still common. The Common Vulnerabilities and Exposures [19] website lists thousands of unique entries. A few of the major categories of flaws derived from Linde [20] and Anderson [18] are provided in Table 2.1.

The most insidious form of unspecified functionality is subversion [21], where a member of the system’s development team intentionally adds clandestine functionality that permits the adversary to bypass system security mechanisms. The term *subversion* is generally applied to the operating system or kernel, whereas other forms of malicious software, for example, Trojan Horses, function in the context of applications. Karger and Schell [22] suggested a subversion in which a compiler could insert an artifice into an operating system. This concern regarding untrustworthy tools was popularized by Thompson [23].

A subversion mechanism executing within the operating system has full system privileges and is unconstrained by policy enforcement mechanisms. If it contains triggers for activation and deactivation, the adversary will have control over its execution.

In many cases, malicious code may be introduced into systems in the form of downloadable executables or scripts, updates, and patches. Code of unknown provenance should be given an integrity label such as *POND-SCUM* and be confined using mandatory integrity controls thus preventing the infliction of pervasive damage.

2.4.3 Assurance

As is the case with security, assurance is a term that is often misused. For example, some state that “software assurance” will improve the “security” of systems. Both these terms are meaningless without context. In the case of software assurance, some might say that a system possesses this quality if it functions as specified and if various tests indicate that the software behaves as expected over a set of inputs, but this definition assumes no malicious intent in the construction of the system. If, on the other hand, one assumes a malicious adversary, then assurance means correct policy enforcement in the face of sophisticated attacks.

The critical nature of the information processed within multilevel secure systems requires assurance through a carefully chosen combination of environmental and technical measures.

Given a particular security policy, an organization may seek more or less assurance that the policy is correctly enforced. For example, greater assurance might be required to show that only authorized individuals have access to trade secrets, whereas less assurance might be required for the protection of the agenda for the next staff meeting. Barriers to achieving high assurance multilevel secure systems include product development pressures that can lead to shortcuts and specious claims. For example, a vendor may claim to have a “secret” technique that makes a system secure, but close inspection by knowledgeable reviewers usually reveals serious flaws [17, 22]. (Such “secret” techniques often involve a combination of cryptography and handwaving.) It is generally accepted that an objective third party must provide an independent assessment of system assurance. The current framework for third party evaluation of system assurance is that of the Common Criteria [24], which provides for high-level requirements for various classes of security products. Through analysis and testing, product team evaluators establish that the product meets both functional and assurance security requirements. A second round of testing and analysis by independent evaluators validates the team’s results.

2.4.4 Secure MLS System Development

Over time, a set of design principles that can guide the development of highly secure systems has emerged [14]. These principles take traditional constructive methods into account, as well as recent advances in hardware and networking.

A secure system should exhibit all of the characteristics of a classic *reference monitor* [17]: resistance to tamper, continuous policy enforcement, and an understandable implementation.

Dependencies are of great importance in designing a secure system. If the system is organized as a set of hierarchical layers, then it must be organized so that each layer of the system depends only upon layers that are of equal or higher assurance and that enforce equivalent or stronger policies. The number and extent of the layers depend upon whether the mandatory policy is void or richly populated, but it is important that mandatory policy enforcement mechanisms do not depend upon discretionary, viz., modifiable, policy components.

Once the system architecture has been delineated and policy has been allocated to its various layers, it is possible to focus on the construction of each layer. Here, the techniques used to develop a high assurance, low-level layer are sketched.

To start, a formal security policy model is developed. The model provides a proof that if the system starts in a secure state, then all operations will maintain that secure state [8, 9]. The formal model serves two important purposes: first, it demonstrates that the intended policy is logically self-consistent and not flawed, and second, it provides a mathematical description of the system to which the implementation can be mapped. The objective of this mapping is to demonstrate that everything in the implementation is both necessary and sufficient for the enforcement of the policy and that no unspecified functionality is present.

Because the objective in constructing the lowest layers of the system is to develop a coherent mechanism for the enforcement of the system's MLS policy, a combination of hardware and software is used to create the exported abstract machine. Rigorous security engineering techniques employing the concepts of layering, modularity, and data hiding ensure that the system has a coherent loop-free design and provides understandable abstractions. Within the system itself, the principle of least privilege can be applied as part of the engineering process. Ultimately, both the formal and informal efforts provide a mapping of the implementation to the formal security policy model as well as evidence that the system is correct and complete.

Although testing cannot prove that a system is secure, it can lessen the likelihood of obvious flaws. Traditional testing demonstrates that each function and module performs as specified. At the system level, this is supplemented by penetration testing. Here, the tester behaves as an adversary and attempts to abuse the system interfaces in an unexpected manner. A useful approach to penetration testing is the *flaw hypothesis methodology* [20].

System administrators and users must be provided with documentation and training. If a system with useful security mechanisms is configured and used improperly, a false sense of security may result that may have consequences far worse than if management believed security was inadequate.

Thorough documentation of the development process is needed so that the system can be assessed with respect to its security requirements by third party evaluators. To date, no viable alternative to either the reference monitor concept or the need for third party evaluation has been proposed. Future research may result in more streamlined approaches to secure system construction and assessment.

2.4.5 Covert Channels

A multilevel secure system is designed to export resources, including subjects and objects, at its interface and to actively mediate information flows between its exported subjects. Rules enforced by the MLS system will ensure that unauthorized flows do not occur as a result of interactions among exported resources; however, design measures must ensure unintended information flows do not occur via covert channels [25] among elements internal to the policy enforcement mechanism.

Covert channels result from the manipulation of system interfaces in ways that cause unintended information flow and result from incomplete resource virtualization by the underlying protection mechanism. This means that some abstract data type presented at the system interface involves operating system constructs that can be manipulated to allow signaling to take place in violation of the system security policy.

An effective technique for covert channel analysis is the shared-resource matrix method [26]. Using this technique, the effects of each system call on operating system-level data structures are analyzed and their visibility, perhaps through exceptions or timing delays, to other processes is identified. When the effects could result in unintended transmission of information, either a system flaw or a covert channel is present.

Covert channels may take one of two forms: *storage channels* and *timing channels* (e.g. [27]). In the case of the former, a resource such as secondary memory is incompletely virtualized so that effects and exceptions viewed by the receiver can be manipulated by the sender. This causality permits the sender to signal a sequence of effects and exceptions, which may be interpreted as a series of 1s and 0s by the receiver, thus creating a binary channel. The bandwidth of the channel will depend upon the speed with which the sender can manipulate the observables for the sender. Complete elimination of covert storage channels is possible for classic monolithic single-processor systems. Multicore processors present new challenges to covert channel mitigation. Covert timing channels result from the ability of the sender to modulate the temporal activities of the system as seen by the sender. Because time is a shared resource that cannot be completely virtualized by the underlying security mechanism, requirements may only call for a reduction of the channel bandwidth below a certain threshold. This can prove challenging, as increases in processor speed can dramatically increase the bandwidth of a covert timing channel, and timing channel reduction may result in imprecise knowledge of system time, significant loss in overall performance, or both [28]. Careful configuration and scheduling can mitigate timing channels; their complete elimination has yet to be demonstrated in useful systems. If covert channel analysis is an objective, then the design and implementation framework and development methodology for the target system must ensure that adequate information for this analysis is available.

Another form of information leakage is via *side channels*. In this case, information leakage is not internal, but is possible through monitoring of external observables (e.g. [29]). Side channels are possible in any system intended to protect some secret, such as encryption key, from external observation.

2.4.6 Object Reuse Considerations

Objects, the information containers in systems, are constructed using system resources, usually primary and secondary memory, but devices must also be considered. When objects are deleted, the memory from which they were constructed is returned to a pool. To prevent inadvertent access to information previously stored in deleted objects, an object reuse mechanism is needed that will remove information from resources before their reuse. The system implementation determines whether the information is purged immediately after object deletion or before its allocation to a new object.

2.4.7 Target Environment

The requirements levied upon a multilevel system usually depend upon the conditions within which it is intended to operate. As a result, there are several *operational modes*.

System high operation requires that all users be cleared to the highest sensitivity level and all information is stored at the system high sensitivity level. Any labels are only advisory and a review is required to downgrade information for use in a less sensitive system. Physical controls commensurate with the highest sensitivity of information within the system are used to isolate the system and grant users physical access to it.

In a *compartmented mode* system, all information is treated to be at the highest sensitivity level. Users must be vetted to not disclose information on the system. Formal compartments subdivide the information and users are granted formal access to those compartments on a need-to-know basis. Information to be moved out of the system to lower sensitivity levels must be submitted to a review and downgrading process. Physical controls commensurate with the highest sensitivity of information within the system are used to both isolate the system and to control physical access to it.

A *multilevel mode* system permits information at more than one sensitivity level to be processed within the system for which users do not have authorizations to access all information on the system.

Multilevel mode does not imply that all sensitivity levels must be processed by the system, nor does it imply support for a full range of user authorizations. The range of information sensitivity levels and the range of user clearances will be dictated by the assurance of correct policy enforcement that the system provides. For example, in 1985, even with the highest assurance systems of the time, the risk was considered too high to permit the minimum clearance of users to be UNCLASSIFIED and the maximum sensitivity of data on a system to be TOP SECRET with multiple compartments [30].

2.4.8 Cascade Problem

Organizations may wish to link several MLS systems into a network. When doing so, it is necessary to consider the assurance characteristics of the interconnected components.

Consider two systems that have been certified to have sufficient assurance to separate two sensitivity levels. System A can process TOP-SECRET and SECRET information, and System B can process SECRET and CONFIDENTIAL information. As stand-alone systems, we are satisfied that although some information flow might occur, they will not be particularly damaging due to the small range in sensitivity levels being managed by the system. Now suppose that the systems are networked together so that the SECRET domain of System A is connected to the SECRET domain of System B, as shown in Figure 2.5. In this configuration, information might inadvertently flow from TOP SECRET to CONFIDENTIAL. This could pose an intolerable risk. The interconnection introduces an information flow cascade [31]. Because of the complexity of finding and correcting cascades, networks should be carefully designed to avoid the cascade problem from the outset.

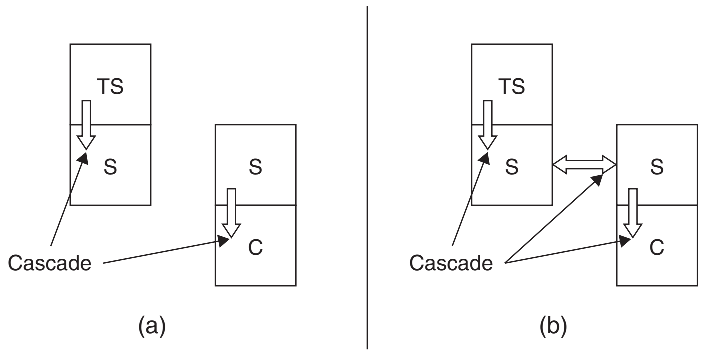


FIGURE 2.5 Systems in (a) have sufficient assurance to address information flow between two equivalence classes. By interconnecting the same systems, as shown in (b), the assurance is insufficient with respect to flows between three equivalence classes, and an unacceptable cascade results.

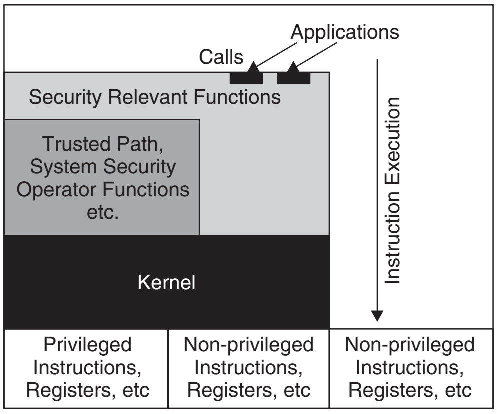

Applications
Calls
Security Relevant Functions
Trusted Path, System Security Operator Functions etc.
Kernel
Instruction Execution
Privileged Instructions, Registers, etc
Non-privileged Instructions, Registers, etc
Non-privileged Instructions, Registers, etc

FIGURE 2.6 A trusted computing base encompasses all security-relevant functionality and may contain a kernel.

2.5 PLATFORMS AND ARCHITECTURES FOR MULTILEVEL SECURITY

All multilevel secure systems must provide separation of the information equivalence classes derived from policy, enforcement of rules for interaction between the equivalence classes, a mechanism to map the equivalence classes to human readable labels, and assurance that the system is complete and correct. Because enforcement of supporting policies, for example, identification and authentication and audit, would complicate the minimal kernel, an MLS system is usually organized as a *trusted computing base* (TCB) of which the kernel is a part. Figure 2.6 is a simplistic illustration of a kernel within a TCB.

Current approaches to multilevel secure platforms include classic security kernels [32] and separation kernels [33]. In the former, an internal policy module mediates access based upon the rules established for the equivalence classes it maintains. Information exported to networks will have either implicit labels, in the case of single-level networks, or explicit labels associated with each packet, in the case of multilevel networks. In these kernels, each process will be assigned either a single level or will be trusted over some range of labels. When combined with the use of ring-based privilege domains [11], traditional security kernels provide considerable granularity for information flow control. Traditional security kernels dynamically allocate resources as different sensitivity levels are required.

A separation kernel creates partitions to which it exports resources based upon assignments loaded into the kernel in the form of a configuration table. Once the configuration is loaded, the sensitivity levels to be managed are fixed until a new configuration is applied, which in conservative cases requires a system restart [34]. Each partition represents an equivalence class, and an interpartition flow policy determines how subjects within a partition can cause flows to other partitions. These kernels are sometimes called *partitioning kernels* or *multiple independent levels of security* (MILS) architecture kernels, for example [35]. For these kernels, enforcement of the interpartition flow policy is allocated to a partition that has read and write access to all other partitions and contains a trusted subject to mediate and effect the interpartition flows.

Least privilege separation kernels [36] provide higher granularity with regard to flows than is present in MILS separation kernels. By combining a more granular configuration table with the use of hardware-supported rings and the ability to create many subjects per equivalence class, a least privilege separation kernel does not need a special partition for interpartition communication, but can create trusted partitions for other functions as needed.

The distinct advantages and disadvantages of each type of kernel should be considered when selecting a kernel to meet specific requirements [37].

Requirements exist for architectures that support heterogeneously trusted users, for example, [38]. Blacker was an early example of a multilevel secure network [39]. In essence, it was an MLS virtual private network (VPN) and was designed to provide both the platform assurance and cryptographic mechanisms required for highly distributed host-to-host communication. Various other solutions have been adopted as well. In system high and compartmented environments, workstations of low or medium assurance provide multilevel functionality, but do little to address the subversion threat. The cost is a specialized workstation for each user. Certain high assurance MLS systems can be used in distributed environments and a major vendor once developed a high assurance MLS virtual machine monitor (VMM), although the product was never released. Recent progress in VMMs has led to a resurgence of interest in their use for MLS.

Architectures that combine the use of popular commodity office productivity applications with components for high assurance of policy enforcement are possible [40]. This approach results in a network architecture that depends upon the use of high assurance multilevel components for servers as well as high assurance elements to support a distributed trusted path and for label enforcement at single-level network connections. An alternative approach uses many microarchitectural elements [41].

2.5.1 Use of Applications in MLS Systems

A challenge facing the developers of MLS systems has been the design of applications able to take advantage of the system's underlying multilevel technology. To avoid requiring every application to be a trusted subject, applications must be organized to execute as multilevel-aware untrusted single-level subjects that can easily read and write to objects as allowed by policy.

Early work in this area involved the design of a multilevel file system intended to provide applications with a complete set of file system features while constrained by an underlying security kernel [42]. Subsequent work [43] has demonstrated that a wide variety of applications can be made multilevel aware.

User acceptability is often a problem for operational MLS systems. The rules to constrain Trojan Horses and other malicious software do not exist in the world of people and paper. A person cleared for TOP SECRET can hand write an UNCLASSIFIED memo without clearing the desk of all classified information. Typical users do not understand why the rules for automated MLS security policies exist and merely conclude that those developing MLS do not trust them. Others seek alternative solutions with relaxed rules or "vendor magic", which can endanger highly sensitive information.

2.6 CONCLUSION

Multilevel secure systems are required when heterogeneously trusted users must access information resources. They can be used to protect information from unauthorized disclosure and unauthorized modification of information. Although the challenges associated with the design and development of highly trustworthy multilevel secure systems have been understood since the late 1960s, their implementation and use have remained elusive. Consequently, many organizations adopt architectures comprising multiple single-level networks for which timely access to information at different sensitivity levels is often difficult or impossible. Challenges to the use of MLS systems include their proportionately higher cost relative to commodity products, user acceptability of the policy enforcement rules, and modern platform architectures that introduce opportunity for the existence and exploitation of high-bandwidth covert channels.

REFERENCES

1. Brinkley, D. L., and Schell, R. R. (1995). Concepts and terminology for computer security. In *Information Security: An Integrated Collection of Essays*, M. Abrams, S. Jajodia, H. Podell, Eds. IEEE Computer Society Press, Los Alamitos, CA, pp. 40–47.
2. Lipner, S. (1982). Non-discretionary controls for commercial applications. In *Proceedings of the 1982 IEEE Symposium on Security and Privacy*, IEEE Computer Society Press, Los Alamitos, CA, pp. 2–20.
3. Saltzer, J. H., and Schroeder, M. D. (1975). The protection of information in computer systems. *Proc. IEEE* **63**(9), 1278–1308.
4. Irvine, C. E., and Levin, T. (2002). A cautionary note regarding the data integrity capacity of certain secure systems. In *Integrity, Internal Control and Security in Information Systems*, M. Gertz, E. Guldentops, L. Strous, Eds. Kluwer Academic Publishers, Norwell, MA, pp. 3–25.
5. Denning, D. E. (1976). A lattice model of secure information flow. *Commun. ACM* **19**(5), 236–243.
6. Sterne, D. F. (1991). On the buzzword "security policy". In *Proceedings of the IEEE Symposium on Research in Security and Privacy*, IEEE Computer Society Press, Los Alamitos, CA, pp. 219–230.
7. Lampson, B. W. (1971). Protection. In *Fifth Princeton Conference on Information Sciences and Systems*, pp. 437–443; Reprinted in ACM SIGOPS *Oper. Syst. Rev.* **8**(1), 18–24.
8. Bell, D. E., and La Padula, L. (1973). *Secure Computer Systems: Mathematical Foundations and Model*, (Tech. Rep. No. M74–244). MITRE Corp, Bedford, MA.
9. Biba, K. J. (1977). *Integrity Considerations for Secure Computer Systems*, (Tech. Rep. No. ESD-TR-76-372). MITRE Corp, Bedford, MA.
10. Lunt, T. F., Neumann, P. G., Denning, D. E., Schell, R. R., Heckman, M., and Shockley, W. R. (1989). *Secure Distributed Data Views Security Policy and Interpretation for DMBS for a Class A1 DBMS (RADC-TR-89-313*, Vol **1**), Rome Air Development Center, Griffiss Air Force Base, NY.
11. Schroeder, M. D., and Saltzer, J. H. (1972). A hardware architecture for implementing protection rings. *Commun. ACM* **15**(3), 157–170.
12. Schellhorn, G., Reif, W., Schairer, A., Karger, P., Austel, V., and Toll, D. (2000). Verification of a formal security model for multiapplicative smart cards. *Proceedings of the 6th European Symposium on Research in Computer Security (ESORICS 2000)*, Lecture Notes in Computer Science Vol. 1895, Springer-Verlag, pp. 17–36.
13. Landauer, J., Redmond, T., and Benzel, T. (1989). Formal policies for trusted processes. *Proceedings of the Computer Security Foundations Workshop II*, IEEE Computer Society Press, Los Alamitos, CA, pp. 31–40.

14. Levin, T. E., Irvine, C. E., Benzel, T. V., Bhaskara, G., Clark, P. C., and Nguyen, T. D. (2007). *Design Principles and Guidelines for Security*, NPS-CS-08-001, Naval Postgraduate School, Monterey, CA.
15. Benzel, T., and Tavilla, D. (1985). Trusted software verification: a case study. In *Proceedings of the 1985 IEEE Symposium on Security and Privacy*, IEEE Computer Society Press, Los Alamitos, CA, pp. 14–31.
16. Kurak, C., and McHugh, J. (1992). A cautionary note on image downgrading. In *Proceedings of the Eighth Annual Computer Security Applications Conference*, IEEE Computer Society Press, Los Alamitos, CA, pp. 153–159.
17. Irvine, C. E., Levin, T., Wilson, J. D., Shifflett, D., and Pereira, B. (2002). An approach to security requirements engineering for a high assurance system. *Requirements Eng.* **7**(4), 192–208.
18. Anderson, J. P. (1972). *Computer Security Technology Planning Study*, (Tech. Rep. ESD-TR-73-51, Vols. 1 and 2, NTIS Document No. AD758206). Air Force Electronic Systems Division, Hanscom Air Force Base, Hanscom, MA.
19. MITRE Corp (2007). *Common Vulnerabilities and Exposures*, Last Accessed 22 August 2007 http://www.cve.mitre.org/.
20. Linde, R. R. (1975). Operating system penetration. In *Proceedings of the National Computer Conference*, AFIPS Press, Montvale, NJ, pp. 36–368.
21. Anderson, E. A., Irvine, C. E., and Schell, R. R. (2004). Subversion as a threat in information warfare. *J. Inf. Warf.* **3**(2), 52–65.
22. Karger, P. A., and Schell, R. R. (1974). *Multics security evaluation: Vulnerability analysis.* Hanscom Air Force Base, Information Systems Technology Application Office Deputy for Command and Management Systems Electronic Systems Division (AFSC), Bedford, MA.
23. Thompson, K. (1984). Reflections on trusting trust. *Commun. ACM* **27**(8), 761–763.
24. ISO/IEC (2004). *Common Criteria for Information Technology Security Evaluation*, (Rep. No. CCIMB-2004-01-001, Ver. 2.2, Rev. 256). International Organization for Standardisation, Geneva, Switzerland.
25. Lampson, B. W. (1973). A note on the confinement problem. *Commun. ACM* **16**(10), 613–615.
26. Kemmerer, R. (1982). A practical approach to identifying storage and timing channels. In *Proceedings of the IEEE Symposium on Security and Privacy*, IEEE Computer Society Press, Los Alamitos, CA, pp. 66–73.
27. Levin, T., and Clark, P. C. (2004). A note regarding covert channels. In *Proceedings of the Sixth Workshop on Computer Security Education*, Naval Postgraduate School, Monterey, CA, pp. 11–15.
28. Hu, W. (1992). Lattice scheduling and covert channels. In *Proceedings of the IEEE Computer Society Symposium on Research in Security and Privacy*, IEEE Computer Society Press, Los Alamitos, CA, pp. 52–61.
29. Kocher, P. C. (1996). Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In *Advances in Cryptology—CRYPTO'96, Lecture Notes in Computer Science*, Springer-Verlag, Vol. 1109, pp. 104–113.
30. Department of Defense (1985b). *Technical Rationale Behind csc-std-03-85: Computer Security Requirements –Guidance for Applying the Department of Defense Trusted Computer System Evaluation Criteria in Specific Environments*, June, US Government Printing Office, Washington, DC. URL: http://csrc.nist.gov/publications/secpubs/rainbow/std004.txt.
31. Millen, J. (1988). The cascading problem for interconnected networks. In *Proceedings of the Fourth Aerospace Computer Security Applications Conference*, IEEE Computer Society Press, Los Alamitos, CA, pp. 269–273.

32. Ames, S. H., Gasser, M., and Schell, R. R. (1983). Security kernel design and implementation: an introduction. *IEEE Comput.* **16**(7), 14–22.
33. Rushby, J. (1981). Design and verification of secure systems. *ACM Oper. Syst. Rev.* **15**(5), 12–21.
34. National Security Agency (2007). *U.S. Government protection profile for separation kernels in environments requiring high robustness*, Version 1.03. 29 June 2007.
35. Vanfleet, W. M., Beckwith, R. W., Calloni, B., Luke, J. A., Taylor, C., and Uchenick, G. (2005). MILS: architecture for high assurance embedded computing. *CrossTalk* **18**(8), 12–16.
36. Levin, T. E., Irvine, C. E., and Nguyen, T. D. (2006). Least privilege in separation kernels. *Proceedings of the International Conference on Security and Cryptography, Setubal, Portugal,* pp. 355–362.
37. Levin, T. E., Irvine, C. E., Weissman, C., and Nguyen, T. D. (2007). Analysis of Three Multi-level Security Architectures, *In Proceedings of the Computer Security Architecture Workshop*, Fairfax, VA, pp. 37–46.
38. National Security Agency (2004). *(U) Global Information Grid Information Assurance Capability/Technology Roadmap*, Version 1.0 (Final Draft), October. National Security Agency, Fort Meade, MD.
39. Weissman, C. (1992). Blacker: Security for the ddn examples of a1 security engineering trades. In *Proceedings of the 1992 IEEE Symposium on Research in Security and Privacy*, IEEE Computer Society Press, Los Alamitos, CA, pp. 286–291.
40. Irvine, C. E., Levin, T. E., Nguyen, T. D., Shifflett, D., Khosalim, J., Clark, P. C., Wong, A., Afinidad, F., Bibighaus, D., and Sears, J. (2004). Overview of a high assurance architecture for distributed multilevel security. In *Proceedings of the 2004 IEEE Systems, Man and Cybernetics Information Assurance Workshop*, IEEE Computer Society Press, Los Alamitos, CA, pp. 38–45.
41. Weissman, C. (2003). MLS-PCA: a high assurance security architecture for future avionics. In *Proceedings of the Annual Computer Security Application Conference*, IEEE Computer Society Press, Los Alamitos, CA, pp. 2–12.
42. Irvine, C. E. (1995). A multilevel file system for high assurance. In *Proceedings of the 1995 IEEE Symposium on Security and Privacy*, IEEE Computer Society Press, Los Alamitos, CA, pp. 78–87.
43. Nguyen, T. D., Levin, T. E., and Irvine, C. E. (2005). MYSEA Testbed. In *Proceedings of the 6th IEEE Systems, Man and Cybernetics Information Assurance Workshop*, IEEE Computer Society Press, Los Alamitos, CA, pp. 438–439.

3

TRUSTED PLATFORMS: THE ROOT OF SECURITY

ROGER L. KAY
Endpoint Technologies Associates, Inc., Wayland, Massachusetts

3.1 INTRODUCTION

The ancient adage admonishes that a chain is only as strong as its weakest link, but this maxim might as well have been coined for computer security. A chain of trust is only as trustworthy as its most vulnerable node or layer. The computing industry has long understood this principle and has even generated adequate technological solutions. Two challenges remain unmet, however: standardization and adoption. Toward the end of the Millennium, IBM and its partners in the development of trusted architectures realized that no solution would work unless it was adopted universally. The industry embraced the principle of standardized security, even before 9/11, but all the more ardently since. Hardware manufacturers and software platform developers agreed that hardware-based security was necessary as the root of trust, and the industry embodied the basic standard in hardware circuitry, essentially a silicon chip. This circuitry ships with an ever growing proportion of personal computers, smart phones, and particularly embedded devices, but has not been put into use widely, particularly in its broader manifestation among devices across a network. Although the circuitry is in the computer, it remains inactive for the most part. Usage, such as it is, is mainly restricted to user-to-platform authentication and password management. Thus, the pace of implementation of secure base computing in the decade succeeding 9/11 has slowed to a crawl, as users await software that makes better use of the secure hardware, and remaining standardization issues get sorted out. There are some exceptions, islands of adoption, and areas of usefulness, but much of the work lies ahead.

3.2 THE STATE OF TRUSTED COMPUTING

It did not take the computing industry long to realize what an optimal security architecture looked like. Work done on encryption had shown that key pairs based on large prime numbers, generated through Public Key Infrastructure (PKI) technology [1], were the

Cyber Security, Edited by John G. Voeller

mathematically simplest and best way to create robust keys that could be used to encode and decode data securely as well as to identify a user publicly and authenticate a user privately. PKI creates key pairs, sometimes several sets. Each pair has a private and public key. Each is mathematically related to the other, but neither can be derived from the other. True to its name, the public key, everyone knows. If you send me something encoded in my public key, no one can intercept it or make sense of it, but I can open it with my private key. If I send you something encoded with my private key and you can decrypt it with my public key, knowing it is me because no one else has my private key.

Important to this architecture's viability is the fact that sender and receiver need never share a secret. And the same relationship between sender and receiver holds between any two nodes in the infrastructure (e.g. hardware and firmware, operating system and application within a system, or any two adjacent communications points within the network). As long as each element or layer can be publicly identified and privately authenticated, the chain of trust can be extended through it.

Keys are large and difficult to crack. Theoretically, a key domain can be expanded to the point of precluding a crack, but this degree of security is not desirable for practical reasons. Arguably, "commercial-grade" security, whose primary job is to rapidly encrypt and decrypt transactional data in high volumes, is sufficient and the type of security appropriate for military-grade protection is unnecessary. It is typical of the commercial world that none of the transactions is of earth-shattering importance. However, all these transactions taken together represent a profile of the business in question and therefore need to be kept from prying eyes.

In fact, the PKI algorithm is asymmetrical; that is, it is a "one-way" formula. The metaphor is breaking a champagne glass. It is easy to turn a glass into shards by throwing it into a fireplace, but extremely difficult to reassemble a glass from shards in a fireplace. This characteristic helps make PKI hard to break, but it is also a reason that PKI is slow [2]. For this reason, a different algorithm, Advanced Encryption Standard (AES), a method that has the virtue of being symmetrical but the weakness of requiring a shared secret, is used for bulk encryption. Called *Triple AES*, the data is wrapped three times in AES encryption, a method deemed sufficient to stave off most intruders. The shared secret, which is a relative handful of code, is encoded with the heavier weight, more robust PKI algorithm, and thus the data is safe, whether in flight (i.e. traveling over data communication lines) or at rest (i.e. sitting on a hard drive).

In the late 1990s, when the original architecture for trusted computing was being forged, IBM, the senior partner on the development team, realized that the elegance of PKI could only be realized if everyone agreed to adhere to the same standard. After all, if the public is to know that *I am me*, it must be all the public and not just some of it. And if I want to send something to you that only you can see, I can do so only if I have access to your public key and you and only you have your private key within a system that recognizes both as being part of the same pair. None of this can work if separate domains exist. Potentially, one needs to interact with someone else.

Seeing developments swinging in this direction, IBM realized that it could not propagate the system on its own. Rivals would perceive the move as an attempt to control the industry at a choke point. So, the company changed course and decided to contribute its intellectual property to a neutral industry body and work with other vendors to bring everyone over to the new standard. That group evolved into the Trusted Computing Group (TCG), which is now comprised of most of the key players in the information technology industry.

The membership of the TCG, 140 companies drawn from all geographies, has spent the past decade refining and promoting standardized security for a number of platforms and circumstances. The TCG board includes AMD, Fujitsu Limited, Hewlett-Packard, IBM, Infineon, Intel, Lenovo, Microsoft, Seagate, Sun Microsystems, and Wave Systems [3].

3.2.1 Why Hardware Security?

A primary reason that the TCG decided to back hardware-based security is its clear superiority over software-based solutions. Of course, every solution has software, but the critical issue is how keys are created and handled. Because a software-only solution requires that a key, the algorithm that uses the key, and the data to be encrypted all reside in main memory at the same time in order to function, the key is vulnerable.

This fact was proven in January 2000, when researchers at nCipher in Cambridge, England, came up with an algorithm that can search main memory, looking for a high degree of entropy [4]. A good private key is going to be exceedingly entropic; that is the random numbers in the key will be well dispersed in numeric space. Other elements in memory—such as the clear text to be encrypted and the encryption program itself—will not be. Since all three—the program, the data, and the key—have to be in main memory at the same time for software encryption to take place, the nCipher algorithm could allow an intruder to scan the contents of main memory and find the user's private key. The nCipher program has been able to find a 1,024-bit private key, the best in commercial use at the time [5].

Another weakness of software solutions is that they cannot prevent "hammering," the practice of trying one possible key after another in rapid fire, because they are unable to keep a counter. A hacker can always freeze the state of the machine and continue to bombard it with attempts. Hardware security cannot be bypassed in this way and thus can institute a progressive lock-out program if too many password attempts are made. This type of program increases the delay between log-in offers after a set number of failed attempts. But, as a reason to avoid software-only security solutions, the hammering flaw pales beside the problem of leaving highly entropic private keys around in main memory.

3.2.2 Essentials of Trusted Computing

Thus, the industry agreed to pursue an architecture based on a hardware implementation of established security principles. The actual hardware involved circuitry insulated from the main computing mechanism. This circuitry, named the Trusted Platform Module (TPM), contained read-only memory, preloaded with the basic security program and a small amount of memory for storing keys generated by the program [6]. The circuitry, in its discrete implementation (a separate chip as opposed to a section of a larger chip), was connected to the processor via a secure bus, and in the off chance that this bus was attacked, the mechanism was designed to flag that it was in an insecure state. A bus attack became less of an issue with integrated implementations, as when the TPM circuitry is added as a module to core logic, the I/O chip, or even the processor itself.

A generic view of TPM architecture shows how it sits on a bus connected to core logic (Fig. 3.1). The trusted platform provides three basic elements: protected capabilities, integrity measurement, and integrity reporting. Protected capabilities are a set of commands that can access shielded locations (e.g. in memory, in the registry), where data operations can take place safely. The TPM uses shielded locations to protect and

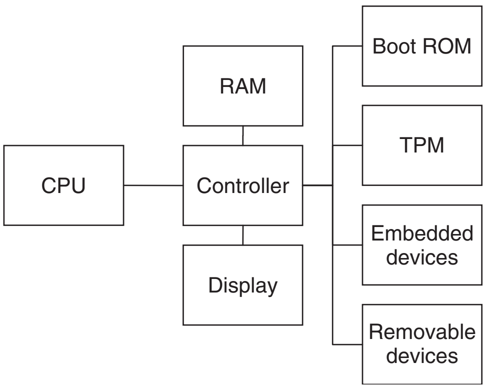


FIGURE 3.1 Reference PC platform containing a TPM.

report integrity measurements. In addition, the TPM generates and stores cryptographic keys used to authenticate reported measurements.

Among other functions, the TPM can attest to the reliability of a platform, providing proof of a set of the platform's integrity measurements through digital signing. Also, the TPM can be used to authenticate a platform, a user, or both. Since the TPM is capable of generating an unlimited number of keys, it can be used to authenticate an unlimited number of identities, but in practice the number of identities is relatively few.

Integrity measurement assesses known platform metrics against the system state, stores this information, and reports it. This function is separated from judging the output, which is reserved for a policy entity. All the integrity measurement has to do is report accurately on the state of the computing platform. To do this, three functions—measurement, storage, and reporting—must be the basis for the roots of trust.

Outside the protected area, other building blocks based on the roots of trust do not have to be shielded. These building blocks include elements that may reside in main memory, core logic, the boot ROM, the processor, and I/O devices (Fig. 3.2). A combination of the roots of trust and the trusted building blocks constitutes a trusted boundary, within which measurement, storage, and reporting activity can take place. The basic operation consists

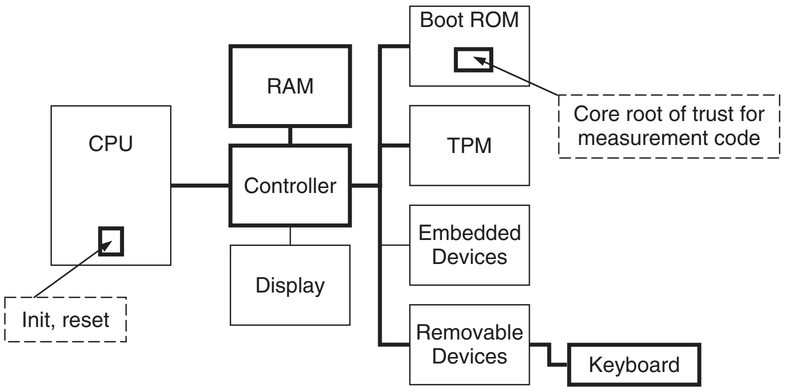


FIGURE 3.2 Bold indicates part of the trusted building blocks of a trusted platform.

of measuring an element in its known state, storing that information, and reporting this value when the element is encountered again in an unknown state. If the newly measured value agrees with the stored value, then the element is considered to be in a trusted state.

This basic function can be expanded through a property called *transitivity*. Essentially, the trusted boundary can be expanded if the already-trusted elements are used to attest to the trustworthiness of a secondary set of elements, assuming that the initial state of these secondary elements can be determined to be trustworthy. These secondary elements themselves can then be used to attest to the trustworthiness of a tertiary set, and so on, and the trusted boundary can be expanded theoretically infinitely in a chain of trust, so long as the roots of trust retain their integrity.

An example of transitivity demonstrates how the trusted boundary can be extended through various levels of the computing stack, right up to the application level (Fig. 3.3).

The TPM is capable of storing mathematically derived representations (hashes) of integrity measurements, which can be compared from time to time or on an event basis with system elements being measured.

A protocol has been established for reporting integrity to a challenger. This protocol involves an exchange of messages between the challenger and the platform, during which an agent on the platform receives the challenge, collects signed information from the TPM, and returns them to the challenger. The protocol is independent of transport and delivery mechanisms and is typical of remote procedure calls, messaging, and communications commonly in use [6].

From this infrastructure, a whole set of credentials can be developed, which allow platforms to gain access to services by disclosing only the minimum amount of identity information. Thus, if a service requires only that a platform be compliant, then it is not necessary to expose to the service the exact identity of the platform, only that it can be reliably said that it meets the compliance requirements. Hence, for example, a mapping service may only need to know that a platform is virus free in order to distribute geographic information to it, not that the platform belongs to a particular entity or individual.

As a communications' endpoint, the TPM is minimally comprised of asymmetric keys, key storage, and processing that protects protocol data items. With these capabilities, the

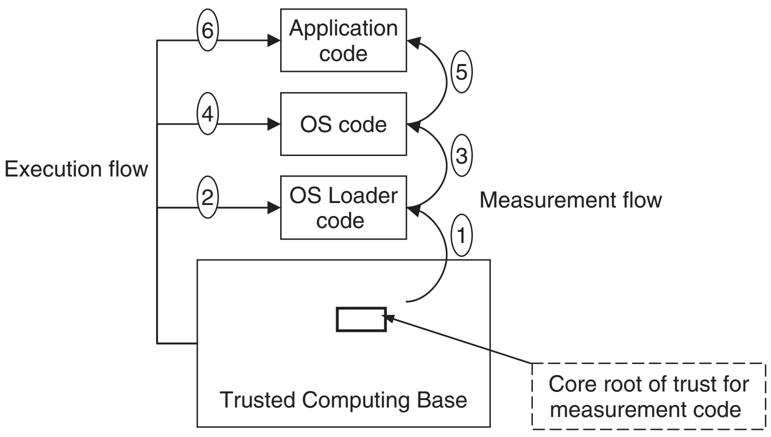


FIGURE 3.3 Transitive trust applied to system boot from a static root of trust.

TPM can perform traditional binding (encrypting a message with a public key), signing (using a separate key to compute a hash of the signed message and encrypt it), sealing (binding a message further with platform metrics), and signed-sealing (a higher level of verified signing that takes into account the platform's configuration).

3.2.3 Extension to Storage

Although TPMs, despite widespread attachment, have not been put to much use, the architecture has been extended to storage devices successfully. Probably one of the most useful applications for TPM architecture to date has been its implementation in full- and partial drive encryption. Since the chain of trust can be extended to progressively larger areas, widening the trusted boundary, the hard drive is a natural candidate for inclusion inside the boundary.

Hard drives contain, in static form, all the most valuable information on a computer. Particularly given that an increasing proportion of personal computers are now mobile, the hard drive is constantly exposed to potential threats. The portable computer can be outright stolen, the drive can be read through the standard user interface, or its contents can be extracted by various means. It is highly important to encrypt the drive contents in such a manner that the information is readable only by authorized individuals and processes, but it is also critical that this encryption is easy enough to perform and fast enough so that the user is not tempted to eschew it. Drive encryption can be enforced by policy, but it still needs to be sufficiently usable so that ordinary operations can be conducted in a straightforward way. In schema available commercially today, encrypted data is written to the hard drive when the user execute a simple "save," and it is decrypted when the user performs a simple "open." The overhead of the encryption process is low enough so that, with modern high-speed processors, the user barely notices.

3.2.4 Biometric Devices as Physical Interface

One of the key issues with deployment, as opposed to implementation, is that the TPM circuitry is difficult to use. The software available thus far has not been user friendly, and many information technology (IT) managers have, after experimentation, declined to deploy the TPM widely in their organizations, despite its benefits, because of user resistance. This circumstance has left many organizations needlessly vulnerable.

One fairly straightforward way to bring the value of the TPM to the surface of the computer interface is through biometric devices, specifically fingerprint readers. A reader connected to the TPM allows a user to get immediate physical access to the encryption capabilities.

Fingerprint reader suppliers, notably UPEK and AuthenTec, offer software suits that allow a user to tie his or her unique identity to a particular platform without having to remember passwords or carry a secure token. In some circumstances, dual-factor authentication—such as fingerprint and password—may be desirable, but in many cases a fingerprint alone is sufficient.

Although the readers can operate without reference to the TPM, all available models allow for TPM interaction, which is useful since other utilities (e.g. full-drive encryption) may be tied to the TPM. The reader, which resides on the surface of an open laptop near the keyboard, on the keyboard peripheral of a desktop, or in an external device attached via USB connection, is the visible face of encryption, the simple, comprehensible interface through which secure procedures take place.

3.2.5 Usage Model

After failing to reach initial lofty goals of getting the whole world to interact across trusted platforms via standard hardware security, the industry turned to more modest objectives. The goal became to make the single-node experience workable. The areas of focus became user authentication to the platform, most often actuated by fingerprint sensor, file and folder encryption, and password management. File and folder encryption has evolved into full-drive encryption (FDE), which is one of the fastest growing areas of TPM adoption. Because of the hardware implementation, when file and folder encryption is working, the user perceives little latency. Nonetheless, TPM calculations do put a tax on system performance, which will disappear over time as hardware technology improves.

Password management has become the most popular consumer usage. The number of passwords users are called upon to manage has grown along with the Internet, and most people have far more than they can remember, particularly when strong password rules are enforced. It can be a great relief for a user to simply enter the user name and password once, and assign them to a fingerprint-actuated, encrypted password bank. And some utilities can generate strong user-password combinations automatically. Thereafter, the account owner can invoke the site with nothing more than a swipe of the finger. As trusted networking increases, the value of this capability will rise, aided by multifactor authentication backup.

3.3 INTERNATIONAL SCOPE

Over time, the TPM standard has been incorporated into basic technology. Microsoft uses the TPM to drive BitLocker, the FDE solution that the company incorporated into Vista, its mainstream operating system. The company continues to extend BitLocker technology to encompass server and mobile form factors. In the second half of 2008, Intel began including TPM support in high-end dual and quad desktops and advanced chipsets. Seagate offers a TPM-based FDE solution for its drives. Among PC hardware OEMs, TPM penetration has been heavier in commercial client lines and less so in consumer, but ecommerce opens a horizon of potential TPM usage among consumers over time. Virtually all embedded applications on kiosks, ATM machines, and factory automation systems now ship with TPM. In addition, the International Organization for Standardization (ISO) has recently approved the TPM specification. Finally, the governments of most countries in Western and Eastern Europe, including the EU as an entity, in Asia, and in North America have backed TPM architecture.

Support for TPM has been widespread geographically. Manufacturers of modules exist on most major continents, and many hardware OEMs ship systems with TPMs. China, Germany, Israel, Japan, Korea, Switzerland, and the United States all host factories that produce chips incorporating TPMs in various ways. TPM circuitry has been produced widely as a dedicated module and has also been realized as an integrated part of other subsystems, such as the processor, I/O chip, or core logic.

3.3.1 Integration

The inexorable economics of trusted computing rests on the simple idea that the wider the standard the better. It is not so much what the standard is exactly as the fact that it is a standard at all. That is what makes the long-term adoption of this schema for secure storage and transmission of data more likely. As the volume rises, so the likelihood of incorporation in basic silicon, the way Intel has done. Thus, the modules are "free" in the sense that they come with, are there anyway, no extra charge, whether or not they are activated. A sufficiently large transistor budget, thanks to improved silicon processes, has made it simple to add the circuitry to core logic. Over time, this integration will likely lead some of the specialty subsystem vendors to leave the TPM market, but good arguments remain for others to continue to supply discrete modules. Funding in building the trusted stack will thus largely come from vendors in the field, with broad agreement about the architecture by participants and regulatory authorities.

3.3.2 Trusted Software

Software is readily available from many hardware OEMs and some software-only companies. Net-net, the interface, could stand an upgrade. Some of the lack of utilization by patrons of systems shipped with TPMs is due to poor usability. Nonetheless, hardware vendors such as Dell, Hewlett-Packard, and IBM, and software vendor Wave Systems all provide suites for individual and networked platform environments.

3.3.3 Networking Trusted Platforms

Another important and related TCG standard is Trusted Network Connect (TNC), which defines a protocol for any network element to conduct a dialog with any other over a trusted Internet Protocol (IP) network. This technology is in the infancy of its adoption, but long term it provides a powerful basis for ecommerce.

Essentially, a user is authenticated to a platform and a network, and the combination of the user and platform information becomes the basis for a grant of rights and privileges within a trusted network (Fig. 3.4). Authentication is performed via biometric, often fingerprint, and system information is read. The hash of this calculation becomes the stored value, which has to match on future entries. If it does not, the requestor can be denied entry. The technology is fine-tuned enough so that the concept of quarantine can be introduced. A system can be deemed friendly, but damaged; that is, it might be recognizable, but, for example, its virus signature file is out of date. This system can be granted provisional entry to a quarantined site, where it can be remediated, the virus definition file topped off, or whatever else needs to be done to prepare it for a retry. If it passes after remediation, it is granted its full rights and privileges, even if it is a mobile system connecting over wireless (Fig. 3.5).

For purchasing purposes, the Department of Defense (DoD), Office of the CIO, is on a several year program, moving toward requiring three infrastructure elements on all clients: a smart card, a fingerprint reader, and TNC connect activated through a TPM. This development is significant because DoD requirements are typically pushed out to contractors and vendors, expanding penetration.

In addition, the National Security Agency (NSA) has approved the TPM for its multiyear "high-assurance program" (HAP), a framework for next generation computing platforms. The effort has involved representatives from industry, academia, and other institutions, who are assembling a series of recommendations. HAP members are working with vendors to generate products and create initial applications for three levels of security clearance. With virtualization, these HAP platforms will allow three virtual machines,

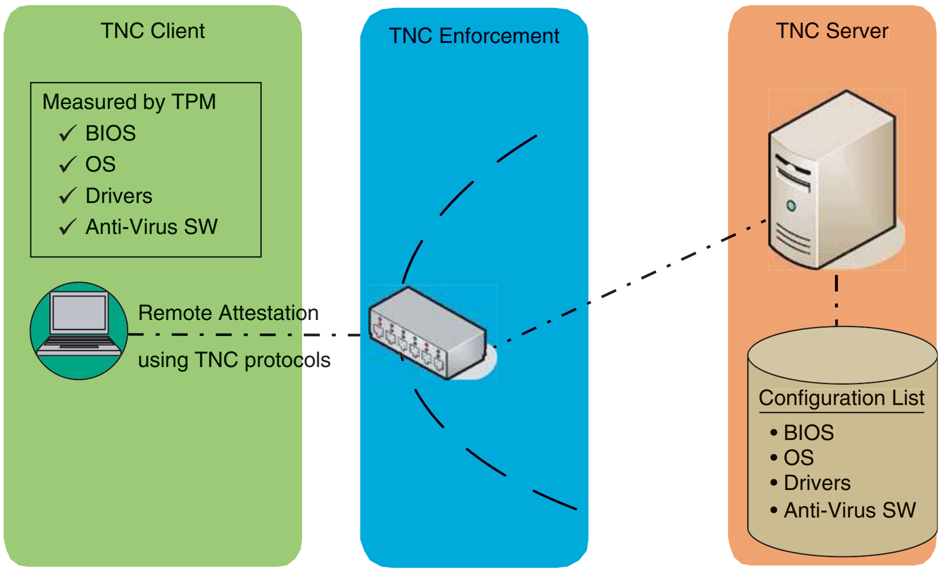


FIGURE 3.4 Trusted hardware via the TPM and remote attestation via TNC verify critical client software.

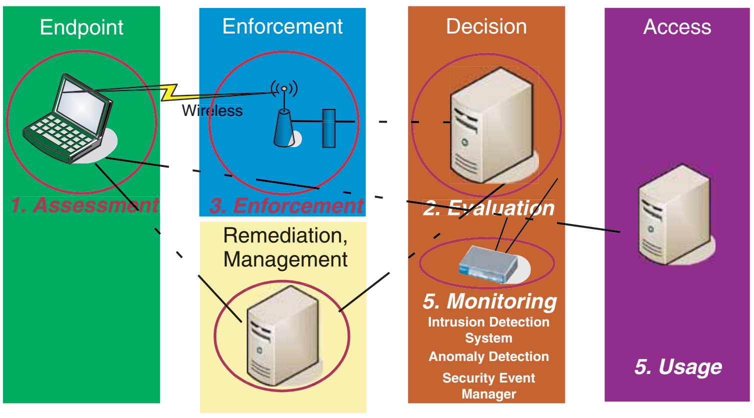


FIGURE 3.5 Nodes with various roles challenge the endpoint. Its credentials are evaluated, policies are enforced, remediation occurs as necessary, and ongoing monitoring ensures that the endpoint's credentials remain valid.

of varying levels of security, to reside together in a single physical computer. Each virtual machine will use its own TNC credentials to gain access to different networks, depending on its rights and privileges [7].

3.4 EXPANDING THE RING OF TRUST

The trusted computing base, grown as the rings of trusted computing expand, can be extended to vast networks, once the infrastructure is established, providing an umbrella under which eCommerce can be conducted safely. Any extension of trusted architecture benefits the whole. Any node can trust any other node. Friends and enemies are distinguishable. They can be sorted or not, depending on the need.

Thus, promoting the standard is advisable for the U.S. government, specifically with reference to homeland security and counterterrorism departments. Standards can be enforced within specific agencies as the need suggests itself. With an increasingly mobile workforce, trusted architecture can speed and secure remote access by laptops, which often seek a network connection from an unknown address.

While it is true that enemies can also conduct secure conversations and pass around secure communications among themselves, this behavior itself is indicative, and those who are interested can monitor those particular behavior patterns quite closely. Metadata, you could call them, information about the information. That notwithstanding, the value of the open use of common secure architecture outweighs the cost of allowing potentially unfriendly parties to use the same technology.

3.5 REMAINING CHALLENGES

Of course, every system is fallible, but it is useful to think of some helpful secrets sufficiently costly to crack so as to discourage all but the most determined adversary. It is not that back doors are built into these implementations of secure base computing. It is that by their nature, all secrets are subject to discovery. While the potential domain of keys in random space may be large enough to dissuade casual search, that space may be subject to preprocessing designed to intelligently limit it to a smaller portion for brute force hammering. Such techniques, when well funded, can yield results.

Thus, cracking a code could be a \$64 problem or a \$1 million problem. When the cracks start hitting \$10 million, commercial entities back off. Only governments and very focused, well funded players remain in the game. And the stakes are generally small on any given device. Even given that valuable trade secrets reside on hard disks and credit card numbers are bought and sold on the open market, the average hard drive will yield information likely priced in the few hundreds of dollars, if that.

So, with the template for the infrastructure mostly in place, it is reasonable to ask why trusted base computing has not been more widely adopted by the public; that is, utilized, put into service, by IT departments and end users. Some industry watchers put the blame on lack of basic demand, saying the standard is being pushed by the industry and has no core constituency in IT departments and among users [8]. The real answer is probably somewhat less crisp. In fact, software needs to be made easier and public awareness increased. The whole trusted experience needs to be made smoother, and people find it difficult to imagine a capability they knew nothing about previously. The government could take a leading role here, creating a unifying voice. The cost is converging on negligible, and it will soon be included anyway.

Some of the attitude toward authentication and security is in flux in post-9/11 American culture. Before the attack, Americans ranked privacy at the very top of rights they valued, whereas afterward their right to be proven who they are gained in stature. People want to be known as themselves. If they have to sacrifice a degree of privacy to achieve a clear identity, the cost seems relatively small compared to the value gained. For example, if swiping a finger on a reader gave a traveler rapid access to the flight areas of an airport, speeding the process through security, many users would agree to register a fingerprint.

The government should look at funding the promotion of trusted standards in wide usage. The flourishing of secure eCommerce alone promotes the interests of the United States, which is then the host to a leading-edge technical and social development. Over time, this ideal eCommerce infrastructure can be promoted to international audiences.

REFERENCES

1. Yin, Y. L. (Ed.) (2004). *IEEE 1363-2000: Standard Specifications For Public Key Cryptography*, http://grouper.ieee.org/groups/1363/P1363/index.html.
2. Becker, P. (2004). *CoreStreet Cuts the PKI Gordian Knot, Digital ID World*, p. 22. http://magazine.digitalidworld.com/Jun04/Page22.pdf.
3. Trusted Computing Group membership, Website: https://www.trustedcomputinggroup.org/about/members/, 2008.
4. Shannon, Claude E. 1950. Prediction and entropy of printed English. *Bell Syst Tech J* **30**, 50–64.
5. Kay, Roger L. The Coming of Age of Client Security Technology, IDC, January 2003, P. 13.
6. Rotondo, S., Hammersley, J., Kotani, S., Balacheff, B., Perez, R., Rosteck, T., Vishik, C., Challener, D., Jong-Chen, J. D., Thibadeau, B., Berger, B., 2007. *TCG Specification Architecture Overview*.
7. NSA High Assurance Program. (2008). http://www.nsa.gov/ia/industry/HAP/HAP.cfm?MenuID=10.2.1.6.
8. IDC, Trusted Platform Module (TPM): Adoption Dynamics, by John Daly, Roseann Day, Charles Kolodgy, Bob O'Donnell, Shane Rau, and Bill Roch, August 30, 2006.

FURTHER READING

Tamvada, J. H. 2007. *Posture of an End Point*, Helsinki University of Technology. Framingham, MA. http://www.tml.tkk.fi/Publications/C/25/papers/Tamvada_final.pdf.

Mayne, M. NAC: Growing pains, SC Magazine UK 2008. http://www.scmagazineuk.com/NAC-Growing-pains/article/112204/.

Santha, M., Vazirani, U. V. *Generating quasi-random sequences from slightly-random sources*. Proceedings of the 25th IEEE Symposium on Foundations of Computer Science: pages 434–440, University of California. October, 24, 1984. ISBN 0-8186-0591-X.

von Neumann, John (1963). Various techniques for use in connection with random digits. In *The Collected Works of John von Neumann*, Pergamon Press, Sunnyvale, CA. pp. 768–770. ISBN 0-08-009566-6. http://www.merrymeet.com/jon/usingrandom.html.

4

CYBER SECURITY TECHNOLOGY USABILITY AND MANAGEMENT

Diana K. Smetters
PARC, Palo Alto, California

4.1 INTRODUCTION

Why does usability matter? Particularly when placed in the context of security, it sounds nice but is not *necessary*—after all, who would place their systems or data in jeopardy just for a little convenience? In practice, the answer to this question is anyone for whom maintaining system security is not the primary job, or in other words, almost everyone. People use computers to accomplish particular tasks, and anything ancillary to those tasks, and particularly anything that gets in the way of their accomplishment will be worked around, disabled, or avoided [1]. The net result of an unusable security measure is likely to be a system less secure than the one that started out as more insecure in the beginning. Luckily, recent work at the intersection of computer security and human—computer interaction (HCI) has begun to demonstrate that the "human element" is a critical component of security, and that it is possible, with care, to build systems that are both usable and secure.

4.2 USABILITY AND SECURITY: CURRENT RESEARCH

In their seminal 1974 paper, Salzer and Schroeder listed eight principles for the design of secure systems. The last was "psychological acceptability"—usability, which they saw as critical if mechanisms designed according to the other seven principles were to be applied correctly [2]. After 25 years of relative quiet, there has been an explosion of interest in how to make secure systems "psychologically acceptable", and how to make "acceptable", or *usable* systems secure.

We can divide the body of research in this new field, often referred to as *HCISEC* (referring to the interface between the fields of *human—computer interaction* and security), into three loose classes. The first takes existing secure systems, identifies their usability flaws, and then (sometimes) attempts to improve them through interface redesign. The second tries to design new systems from the ground up to be both usable and secure, often arriving at significantly different solutions than traditional, solely security-focused designs. The last attempts to develop design principles, or guidelines, to aid practitioners in better designing future systems. An overview of research in each class is given below.

Throughout, it is useful to consider the identity of the "user" in *usable*—while most usability work focuses on end users, usability for systems administrators and developers is also important to the building of effective systems. It is also useful to keep in mind the difference between "security technologies"—technologies designed to accomplish an explicitly security-focused task, such as login interfaces or policy management systems, which permit a certain level of "security focus" in their design and interfaces, and systems that have security requirements, but whose primary goal is something other than security, for example, file sharing systems and e-mail systems [3].

4.2.1 Improving Existing Security Technologies

4.2.1.1 Failures of Existing Systems. Perhaps the largest body of research in security usability looks at systems that are not studies of real users and the ways in which they find existing systems impenetrable or use them incorrectly. The first of these, due to Whitten and Tygar [4], asked users in a laboratory setting to encrypt their e-mail using relatively popular commercial e-mail encryption software (PGP 5.0), and demonstrated that the majority were simply unable to do so.

Even more disturbing is that users not only find managing security difficult, but they are also often unable to determine when they have accomplished security-critical tasks and when they have not. Good and Krekelberg looked at use of a popular peer-to-peer file sharing program, and discovered that the majority of users were unable to determine what files they were sharing with others, sometimes sharing the entire contents of their hard drives when they thought they had exposed nothing [5]. Notable industry studies have shown that although 92% of users think they have up-to-date antivirus software, only 51% actually do; even though 73% of users think they have a firewall, only 64% have turned it on; and that only a small fraction of users who think they have antispam and antiphishing software actually do have [6].

In case we forget that usability problems plague not only end users but also trained system administrators, a large-scale automated survey of secure web servers shows that as of January 1, 2008, 68% of web server SSL/TLS certificates were invalid [7].

4.2.1.2 Passwords. Passwords are the most common form of security technology experienced by users, and perhaps the most overused—many users manage 15 or more [8–10]. Studies confirm that users do not know to pick good passwords, and are asked to remember far too many of them and change them too often, resulting in poor password choice, or passwords that are written down [9, 11].[1] Situations that require individuals to act on each others' behalf result in passwords that are shared [9, 12]. Users frequently reuse passwords across sites [10], reducing cognitive load but allowing an attacker to capture a password in one place and successfully use it in others.

[1]Merely writing down a password is not always a security risk. Using a complex on-line banking password that you must write on a note attached to your monitor is much better protection against hackers on the internet than using a shorter password you can remember. However, it is not very good protection against your cleaning people.

A number of attempts have been made to help users choose better passwords and remember them more effectively [11, 13]. Better password choice algorithms suggest that users derive passwords from memorable phrases rather than words, resulting in passwords that appear as difficult to crack as random passwords but are more memorable [14]. An increasing number of graphical password systems ask users to draw, remember, or select regions of images rather than entering text-based passwords [15, 16]. Unfortunately, the very features that make things memorable to humans—their patterns and predictability—also make them vulnerable to attack. Humans can indeed remember phrase-derived passwords better than random passwords, but they also tend to pick the *same* phrases as easily memorable, making it possible to build a phrase-based dictionary that is highly effective at breaking such passwords [17]. Similarly, the properties of an image or image component that makes it likely a person will remember it or choose it as part of a graphical password are also predictable, resulting in the ability to construct "graphical dictionaries" effective in attacking such passwords [18, 19].

An alternate approach to this problem can be seen in the design of tools that help users remember their existing passwords, such as password toolbars. The best of these help the user create stronger passwords, while reducing the user burden. They do this by taking one (user-remembered) "master" password and generating from it individual, random passwords for each site the user visits. This limits the risk to the user if any single password is compromised [20, 21]. Such systems also attempt to protect the user from phishing attacks (see below) by detecting when the user attempts to enter such a protected password into a site that does not match the one it was created for, either warning the user [20, 21], or actually generating an incorrect password as a result [22]. These are difficult to implement in a fashion that can be used effectively and correctly, however [23], leaving the ultimate effectiveness and practicality of such approaches open to future research.

4.3 SYSTEMS MANAGEMENT

Systems management contains the largest fraction of "security-focused" tasks, and so is perhaps the area where usability of security comes first to mind. The "users" involved in systems management, however, range tremendously. There are typical end users struggling to update and secure their home PCs, who are asked to take on increasingly sophisticated management tasks [24]. There are systems administrators who, as "trained professionals", are often considered not to require special attention to usability, but who frequently have no special expertise in security. And there are specialized security administrators, often dealing with vast amounts of time sensitive information, usually with text-based tools [25].

Several recent ethnographic studies have documented the tools and practices of systems [26, 27] and security administrators [25]. These studies have documented that administrators are often skeptical of graphical management tools, particularly that they will scale effectively, and prefer to build tools themselves or adopt tools built by others within their professional communities. At the same time, these studies have examples where more effective tools would help administrators in visualizing information to more effectively monitor and understand situations, or to configure systems in ways that minimize error or increase speed or effectiveness. It is also the case that "user-friendly" tools designed to help end users with management tasks are sometimes adopted by professional administrators when they see that they can indeed simplify their work (e.g. [28], discussed below, where a secure wireless local area network (WLAN) configuration tool designed for end users was adopted for enterprise use at the request of administrators, and which has been used by those administrators in deployment for over 4 years at this writing).

Well-designed information visualization tools can aid in almost any complex task, including systems administration. An early example of a visualization tool designed specifically to illustrate the frequency of network attacks is the compellingly named *Spinning Cube of Potential Doom* [29].[2] Though not designed for use in real-time administration of networks, one could easily imagine that a successor to this tool could be a powerful aid in understanding current network state. Recent work has begun to leverage visualization tools to assist even untrained end users in understanding and managing their computers. *Sesame* [30] is a visualization system that allows end users to graphically "drill down" and examine the state of their computer, its processes, and network connections, and to manage basic aspects of their systems. Using Sesame, users were more effective at simple security tasks than users using traditional tools (e.g. firewalls).

A number of studies have looked at whether better interfaces or tools can improve administrator effectiveness on standard security tasks.[3] Perhaps the largest group of these have looked at access control or the configuration of permissions to determine who can use files or other resources. The results are very encouraging, suggesting that effective attention to usability in system design can really improve performance on security tasks.

One of the earliest efforts to identify a link between usability and security is that of Zurko et al. [31, 32], who demonstrated that appropriate attention to usable interface design could improve the effectiveness of a role-based access control system. More recent work [33, 34] has demonstrated that improved graphical interfaces to traditional access control lists can improve users' ability to rapidly and correctly specify access policies. Going further, the SPARCLE system [35] allows minimally trained administrators to specify privacy policies in natural language; perhaps signaling the ultimate direction such management systems must go in to reach the ever-larger classes of individuals that must use them. Finally, Cao and Iverson [36] attempted to build the first systems—"intentional access management"—capable of taking a simple specification of what policy the user intends to apply, and automatically navigating the often complex and conflicting maze of effector mechanisms that can "make it so", avoiding common user error in the process.

4.4 WEB SECURITY AND PHISHING

Perhaps the most significant battleground wherein the end user is on the front lines, directly determining the success or failure of an attack, is the war against phishing. *Phishing* is the attempt to get users to hand over personal information or credentials to an attacker via subterfuge.[4] A user receives an e-mail message containing a request for information (e.g. "Update your account information to make sure your access is not disabled!"), and usually a clickable link to an attacker's website where that information

[2]So compelling is the name that I challenge any reader of this manuscript *not* to go and look at the corresponding website.

[3]By effectiveness here, we mean the administrator's ability to accomplish their stated goal rapidly and without error.

[4]Phishing is perhaps the most technical form of traditional social engineering attacks.

can be entered. The e-mail, link, and website are carefully crafted by the attacker to resemble communications from a trusted provider with whom the user already has a relationship (e.g. their bank or an Internet provider such as PayPal or Amazon). Successful attacks enable the attacker to drain the user's bank or PayPal account, or to engage in further forms of identity theft [37].

The fundamental problem underlying the success of phishing attacks is the lack of effective *mutual authentication* in web-based interactions. Logins and passwords are designed to protect websites from unauthorized use. Unfortunately, the mechanisms designed to protect users from being tricked into communicating with sites other than the ones they intend are much less effective. In the former case, a web server can automatically verify user credentials, in the latter a human user must look at a set of security indicators—for example, the use of TLS/SSL, the URL they are connecting to, and the content of the page—and determine whether they are connected to the correct server. Though some effort has been made to design features into modern web browsers to aid users in making this determination, most users do not look at them, instead primarily make trust decisions based on insecure aspects of the web content [38, 39] or use ineffective decision strategies [40].

Approaches to prevent or defend against phishing attacks largely take three forms. The first class of approaches attempts to educate the user: teaching them basic security information and heuristics to help them authenticate the websites they communicate with [41]. Some of these efforts are effective, increasing users' ability to distinguish good from malicious sites [41], and can even be enjoyable—embedding the learning experience in a game [42]. However, other attempts to educate the user about security indicators have resulted in subjects misidentifying malicious sites as good with improved confidence after training [39].

The second class of antiphishing efforts attempts to detect "bad" sites and warn users about them. All major web browsers incorporate increasingly sophisticated tools to do this, using a combination of website analysis and blacklisting. Add-on "antiphishing toolbars" abound. Unfortunately, users tend to ignore such warnings [43]. Even the best of current detection technology is highly inaccurate [44], making it currently unacceptable to simply prevent users from visiting identified supposedly malicious websites. Research continues on improving malicious website detection, from better approaches to content analysis [45], to combined analysis of phishing e-mails and the websites they point to [46], to attempts at large-scale surveys of every malicious site on the Internet [47].

The third class of antiphishing efforts focuses on improving users' abilities to authenticate the websites they interact with—to detect correct sites, rather than to be warned about impersonators. Dynamic security skins [48] allow the user to associate each website with a personal image, stored by that web server, and carefully integrate into the user interface (UI) presented to them in a manner that resists spoofing. Unless the server is able to present the correct image, the user is supposed to reject it as being a malicious impersonator. Limited features of this approach have been integrated into commercial banking sites in a system called *SiteKey* [49]. Though a promising approach, studies have determined that users do not notice the absence of their security image or other security indicators [50]. Password protection toolbars, mentioned above, can help prevent users from giving away their passwords. Some toolbars ensure that the user has a unique password for each site, and resist attempts by users to enter a protected password into the wrong site [20, 21]. Others operate by keying each password to the url of the site being visited, so that the password presented to a phishing site will not be the password required for the corresponding legitimate site [22]. However, they are often difficult to use correctly, and users may achieve a lower level of effective protection than they think they have [23].

4.4.1 Designing New Technologies with Usability in Mind

One of the most common complaints of security experts is that systems designers often attempt to "add security on at the end", after a system has already been built. This is highly ineffective. Usability experts similarly run into efforts to "add-on usability", for example, to bring in an interface designer late in the development of a system in the hopes that better-designed dialog boxes will make up for a fundamentally flawed system interaction design. To design systems that are both usable and secure, perhaps the only truly effective approach is designing for both usability and security from the start [51]. Also the key is to take both seriously—for example, discounting neither security to get greater functionality in an end user focused application nor usability to get greater security. As we have seen, this often ends with users avoiding or abusing the provided security mechanisms, resulting in a system that is less secure than one that had not tried so hard for security to start with. Most interestingly, attempts to "design *usable* security in" from the start often result in systems that use existing technologies in creative and effective ways, as well as designing new technologies potentially useful for future systems. We sample a few such systems here.

The cryptographic literature abounds with techniques for securing network connections; however, difficulties in managing and distributing keys limit their use. *Public key infrastructures* (PKIs), an approach to binding public keys to user identities via signatures by trusted authorities on digital certificates, have been proposed as the universal solution to this problem, but they require that users obtain keys and certificates, and that there be a mechanism for distributing trust in the certificates of the very authorities used to bootstrap the system. Although used extensively to verify web transactions through the use of the SSL (TLS) protocol, digital certificates are in general only used to authenticate servers. Distributing client certificates to authenticate users is considered too difficult, so they are generally authenticated using only passwords.

The largest difficulties in deploying PKIs come firstly from attempting to manage PKIs "in the large" as they were originally designed—to identify people as part of a large-scale, even global, naming infrastructure; and secondly from forcing users to explicitly play their part in the PKI enrollment process—namely generating public/private key pairs and having them certified. By rethinking these assumptions, and considering PKI, or infrastructure "in the small", one can create flexible, small-scale public key infrastructures designed to meet particular goals [52]. If these PKIs, or "instant" PKIs, are tailored to a particular application context, the process of enrollment can then be made transparent to end users, by embedding it in that application's context. This approach tends to use such certificates as simple group membership credentials—group members have a certificate, others do not—rather than the more traditional identity credentials that at least X.509 digital certificates were designed to be.

Balfanz et al. [28] used this idea of an "instant" PKI to address the problem of allowing end users to easily set up highly secure WLANs in a system called *Network in a Box* (NiaB). The NiaB access point configured itself into all the components necessary to secure a WLAN using digital certificates and strong authentication. Users wishing to join a given NiaB-controlled WLAN simply "point out" the access point controlling the network using the infrared port of their laptop or PDA, as if they were using a remote control. Using this infrared connection as a form of *gesture-based authentication* [53], the prospective client and NiaB access point (AP) exchange public key information over this out of band channel, allowing them to authenticate each other and set up a secure connection over the WLAN. Over this wireless connection, the client is automatically given a digital certificate and configured to use this wireless network in the future. The resulting system is both intuitive and secure; providing a simple trust model wherein only people with physical access to the NiaB AP (able to communicate with it via infrared) are able to join the WLAN. The user-friendly approach of using an automated out of band channel to bootstrap authentication between devices [53, 54] has been extensively used in recent years (e.g. [47, 55]). Similar approaches for automatic provisioning of certificates have also been used, for example, to provide client credentials for authenticating to secure websites [48, 49, 56, 57].

Another intuitive approach to delivering and establishing trust in public keys is termed *key continuity management* (KCM) [50, 58]. This simple model, originally adopted by the program *ssh*, simply promiscuously accepts the first key proffered for any identity for which it does not currently know one (offering the user the chance to verify that key first, though they rarely do), and sounding a warning if that key then changes. This approach significantly lowers the barriers to entry for using public key cryptography, in return for slightly reduced initial security. Garfinkel and Miller applied this model to the problem of key management for e-mail encryption, embedding it into a popular e-mail client program [51, 59]. Their results suggest that KCM offers promise for easing practical deployment of e-mail encryption.

Finally, one of the most promising approaches to easing the interface between security and the user is through the use of portable personal devices, such as cell phones, which can carry credentials and perform cryptographic protocols on behalf of end users. Such devices have been used as intermediaries to protect users from giving credentials away in response to phishing attacks [52, 60] or as intuitive tools for authenticating users [28, 61] or creating [53, 62] or effecting [54, 63] access control policy.

4.4.2 Design Guidelines for Building Better Systems

Finally, there has been extensive work developing design guidelines that aim to help systems designers come up with systems that are both usable and secure. Perhaps the most influential, clearly stated, and comprehensive of these is due to Yee [64, 65]. Yee's guidelines blend long-agreed security design goals, such as *the principle of least authority* (or *privilege*)—which says that systems should operate with only the ability to access those resources necessary to do their job, with the best interests of end users. The results are principles such as *the path of least resistance* [64], which say that the easiest way to perform a task ought to also be the one that requires the least granting of authority. These guidelines argue for a distinctly user-focused view of security, arguing that systems should respond to the user's expressed intent to grant or remove authority, where intent should be expressed in terms of the task at hand—terms that are relevant, and understandable to the user. Other guidelines emphasize both that the input from the user is privileged, and must be protected so that user intent can be correctly captured; and that clear and understandable feedback of state to the user is critical for allowing the user to achieve both their task and security goals. These guidelines can be further refined and specialized to address particular user aims; for example, Chiasson et al. present a set of usable security design guidelines tailored toward the needs of systems administrators [66].

4.5 OPEN CHALLENGES AND TAKE-AWAYS

The field of usable security is still very young, and it is marked more by the number of open questions than of accepted answers. The good news is that users generally want to "do the right thing" with regard to security—if only they can figure out what that is, and it does not keep them from getting their primary tasks done [9]. The goal of usable system design is to help them do just that.

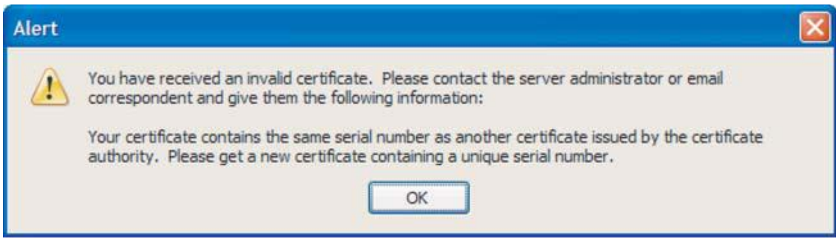
One of the most open areas of research mentioned above is that of developing usable tools for secure, and *correct*, systems management. Because the users in this case are often administrators, not end users,[5] the requirement for improved usability is often underestimated (even by the users themselves [26, 66]). Because the tasks and tools involved are *security focused*—designed to solve a task wherein security management is often an explicit goal—it is perhaps easier to achieve effective security while keeping the user's primary task foremost [3].

The failure of existing tools and approaches for usable systems management can most clearly be seen in the prevalence of configuration errors in deployed systems—for example, the fact that 68% of deployed web server security certificates are currently invalid [7]. Such misconfigurations are surprisingly common, and reflected to the user in potentially confusing ways (Figs. 4.1 and 4.2).

While it is well known that configuration errors may result in insecure systems, there is a more subtle effect of such mistakes—every misconfiguration, on any system, decreases the ability of all other existing systems to recruit end users in defending their own security. Consider the *error attack*. If a malicious server is attempting to impersonate a critical system to a user (e.g. to capture passwords or other credentials), the attacker can explain the absence of any security indicator the user has been trained to rely upon simply by suggesting that "there is a problem with the system" [50].

As long as attacks and system configuration errors generate warnings that are indistinguishable to the end user, the end user must make a determination—"does this warning signal an actual attack or merely a misconfiguration?" Evidence suggests that overwhelmingly they assume the latter [67]—that the warning simply reflects a *false alarm*, and that they should proceed with what is for them, their much more urgent primary task. Given the relative frequency of genuine attacks and simple configuration errors, *this is the rational decision for them to make*. And it will continue to be, until configuration errors are vanishingly less common than genuine attacks. There are two ways to achieve this state: the first is to wait until the attack frequency rises to the point that systems are completely unusable (and hope that configuration errors do not rise in parallel). The better option is to reduce the incidence of configuration errors to the point where warnings of attack are, with overwhelming likelihood, just that. To reduce configuration errors to that degree, we must either reduce the amount of configuration that must be performed or ensure that more of it is done correctly—this requires better systems management and

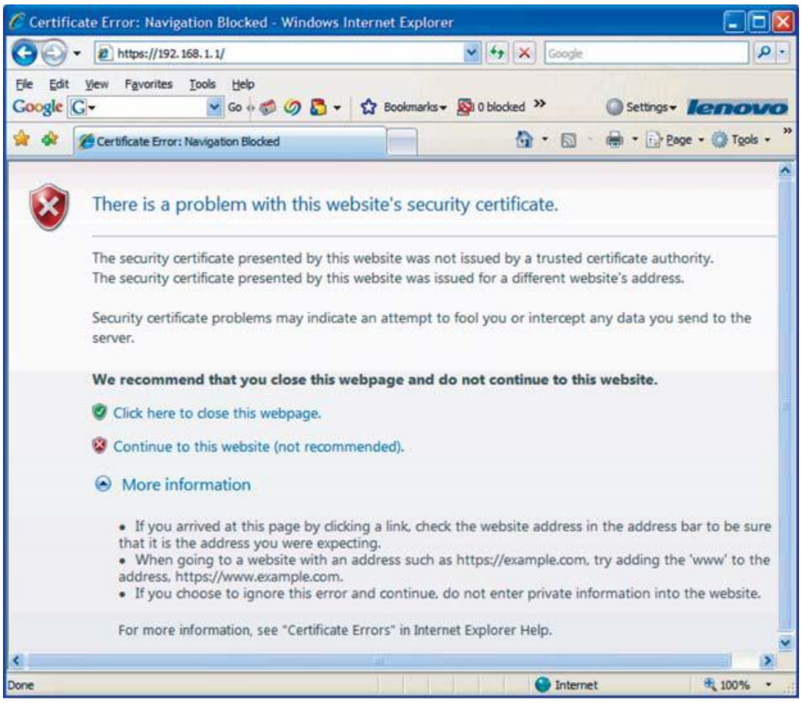
[5]Though as computers get more critically involved in every aspect of day-to-day life, "average" users are more and more often systems administrators as well [24].


Alert

You have received an invalid certificate. Please contact the server administrator or email correspondent and give them the following information:

Your certificate contains the same serial number as another certificate issued by the certificate authority. Please get a new certificate containing a unique serial number.

FIGURE 4.1 This figure shows an error message generated by the Mozilla Firefox web browser when attempting to visit the default secure (SSL-protected) management page offered by a common consumer-grade wireless access point. The access point has been provisioned by the vendor with an invalid certificate. The antiphishing protections built into Mozilla Firefox make it impossible to actually visit this site and administer the device.


Certificate Error: Navigation Blocked - Windows Internet Explorer

https://192.168.1.1/

File Edit View Favorites Tools Help
Google
Go
Bookmarks
0 blocked
Settings
lenovo
Certificate Error: Navigation Blocked
Page
Tools

There is a problem with this website's security certificate.

The security certificate presented by this website was not issued by a trusted certificate authority.
The security certificate presented by this website was issued for a different website's address.

Security certificate problems may indicate an attempt to fool you or intercept any data you send to the server.

We recommend that you close this webpage and do not continue to this website.

Click here to close this webpage.

Continue to this website (not recommended).

More information

- If you arrived at this page by clicking a link, check the website address in the address bar to be sure that it is the address you were expecting.
- When going to a website with an address such as https://example.com, try adding the 'www' to the address, https://www.example.com.
- If you choose to ignore this error and continue, do not enter private information into the website.

For more information, see "Certificate Errors" in Internet Explorer Help.

FIGURE 4.2 This figure shows the error message generated by Internet Explorer 7 when attempting to visit the same SSL-protected configuration page as displayed in Figure 4.1. Using IE, one can override the security warnings and determine that the device contains a self-signed certificate with the serial number 0; this might be a certificate generated automatically by the device (and all devices incorrectly select 0 as a serial number, rather than say choosing a random value) or all devices might even be configured with the same certificate and private key. Note that the errors presented by Firefox and IE cite completely different problems with the certificate.

configuration tools. Though those tools may not be "security tools" per se, they clearly play a security-critical function.

4.6 CONCLUSIONS

This chapter has presented a review of the current state of research in usable security. This is an extremely active, fast-moving field, as evidenced that most of the cited work was performed within the last 5 years. In any such rapidly moving area, new results are always appearing; the further reading list (below) contains pointers to places to find them. Equally important to the work reviewed here is the large body of work omitted for reasons of space—for example, this review does not consider work on usable *privacy*, itself an active area of research.

Perhaps the most important lessons to be learned from this body of work are that usability is indeed key to effective security and that it is possible to design systems that are simultaneously usable and secure—as long as you think it is important enough to do so.

REFERENCES

1. RSA. (2007). *The Untold Insider Threat: Office Workers Confess to Everyday Behavior that Places Sensitive Information at Risk*, [Online] 12 10, 2007. [Cited: 1 11, 2008.] http://www.rsa.com/press_release.aspx?id=8992.
2. Schroeder, J. H. and Salzer, M. D. (1975). The protection of information in computer systems. *Proc. IEEE* **63**, 1278–1308.
3. Smetters, D. K. and Grinter, R. E. (2002). Moving from the design of usable security technologies to the design of useful secure applications. *Proceedings of the New Security Paradigms Workshop 2002*. Virginia Beach, VA.
4. Whitten, A. and Tygar, J. D. (1999). Why Johnny can't encrypt: a usability evaluation of PGP 5.0. *Proceedings of the 8th Usenix Security Symposium*. Usenix, Washington, DC, pp. 169–183.
5. Good, N. S. and Krekelberg, A. (2003). Usability and privacy: a study of Kazaa P2P file-sharing. *CHI '03: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM Press, Lauderdale, FL, pp. 137–144.
6. McAfee Corporation and National Cyber Security Alliance. (2007). *McAfee/NCSA cyber security survey*. *McAfee Corporation*. [Online] September 25, 2007. [Cited: January 6, 2008.] http://download.mcafee.com/products/manuals/en-us/McAfeeNCSA_Analysis09-25-07.pdf?cid=36665.
7. SecuritySpace (2008). *Secure Server Survey*. SecuritySpace, [Online] January 1, 2008. [Cited: January 6, 2008.] http://www.securityspace.com/s_survey/sdata/200712/certca.html.
8. RSA. (2006). *RSA Security Research Shows Volume of Business Passwords Overwhelming End Users and Hindering IT Security Efforts*, [Online] September 12, 2006. [Cited: 1 10, 2008.] http://www.rsa.com/press_release.aspx?id=7296.
9. Adams, A. and Sasse, M. A. (1999). Users are not the enemy: why users compromise computer security mechanisms and how to take remedial measures. *Commun. ACM* **42**, 40–46.
10. Gaw, S. and Felten, E. (2006). Password management strategies for online accounts. *SOUPS '06: Proceedings of the Second Symposium on Usable Privacy and Security*. ACM Press, Pittsburgh, PA, pp. 44–55.

11. Sasse, M. A., Brostoff, S., and Weirich, D. (2001). Transforming the “weakest link”: a human-computer interaction approach to usable and effective security. *BT Tech. J.* **19**(3), 122–131.
12. Singh, S., Cabraal, A., Demosthenous, C., Astbrink, G., and Furlong, M. (2007). Password sharing: implications for security design based on social practice. *CHI ’07: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM Press, San Jose, CA, pp. 895–904. doi = http://doi.acm.org/10.1145/1240624.1240759.
13. Brostoff, S. and Sasse, M. A. (2003). Ten strikes and you’re out: Increasing the number of login attempts can improve password usability. *Workshop on Human-Computer Interaction and Security Systems*, part of CHI2003, Ft. Lauderdale, FL.
14. Yan, J., Blackwell, A., Anderson, R., and Grant, A. (2005). The memorability and security of passwords. In *Security and Usability: Designing Secure Systems that People Can Use*, L. F. Cranor and S. Garfinkel, Eds. O’Reilly & Associates, Sebastopol, CA.
15. Thorpe, J. and van Oorschot, P. C. (2004). Graphical dictionaries and the memorable space of graphical passwords. *Proceedings of the 13th Annual USENIX Security Symposium*. Usenix, San Diego, CA, pp. 135–150.
16. Chiasson, S., Biddle, R., and Oorschot, P. C. (2007). A second look at the usability of click-based graphical passwords. *ACM Symposium on Usable Privacy and Security (SOUPS 2007)*. ACM Press, Pittsburgh, PA, pp. 1–12.
17. Kuo, C., Romanosky, S., and Cranor, L. F. (2006). Human selection of mnemonic phrase-based passwords. *SOUPS ’06: Proceedings of the Second Symposium on Usable Privacy and Security*. ACM Press, Pittsburgh, PA, pp. 67–78.
18. Thorpe, J. and van Oorschot, P. C. (2007). Human-seeded attacks and exploiting hot-spots in graphical passwords. *Proceedings of the 16th Annual Usenix Security Symposium*. Usenix, Boston, MA, pp. 103–118.
19. Ahmet, E. D., Memon, N., and Birget, J.-C. (2007). Modeling user choice in the PassPoints graphical password scheme. *ACM Symposium on Usable Security and Privacy (SOUPS07)*. ACM Press, Pittsburgh, PA, pp. 20–28.
20. Yee, K.-P. and Sitaker, K. (2006). Passpet: convenient password management and phishing protection. *SOUPS ’06: Proceedings of the Second Symposium on Usable Privacy and Security*. ACM Press, Pittsburgh, PA, pp. 32–43.
21. Wu, M., Miller, R. C., and Little, G. (2006). Web wallet: preventing phishing attacks by revealing user intentions. *SOUPS ’06: Proceedings of the Second Symposium on Usable Privacy and Security*. ACM Press, Pittsburgh, PA, pp. 102–113.
22. Ross, B., Jackson, C., Miyake, N., Boneh, D., and Mitchell, J. C. (2005). Stronger password authentication using browser extensions. *Proceedings of the 14th Conference on USENIX Security Symposium—Volume 14*, (Baltimore, MD, July 31—August 05, 2005). USENIX Association, Berkeley, CA, pp. 17–32.
23. Chiasson, S., Oorschot, P. C., and Biddle, R. (2006). A usability study and critique of two password managers. *Proceedings of the 15th Annual Usenix Security Symposium*. Vancouver, BC, pp. 1–16.
24. Shehan, E. and Edwards, W. K. (2007). Home networking and HCI: what hath God wrought? *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI 2007)*. San Jose, CA, April 28-May 3, 2007.
25. Kandogan, E. and Eben, M. H. (2005). Security administration tools and practices. In *Security and Usability: Designing Secure Systems that People Can Use*, L. F. Cranor and S. Garfinkel, Eds. O’Reilly Media, Sebastapol, pp. 357–378. http://www.plunk.org/eben/PublishedPapers/Security-ch18.pdf.
26. Barrett, R., Kandogan, E., Maglio, P. P., Haber, E. M., Takayama, L. A., and Prabaker, M. (2004). Field studies of computer system administrators: analysis of system management tools

and practices. *Proceedings of the 2004 ACM Conference on Computer Supported Cooperative Work*, CSCW '04. (Chicago, Illinois, USA, November 06–10, 2004). ACM, New York, pp. 388–395. DOI = http://doi.acm.org/10.1145/1031607.1031672.

27. Botta, D., Werlinger, R., Gagné, A., Beznosov, K., Iverson, L., Fels, S., and Fisher, B. (2007). Towards understanding IT security professionals and their tools. *Proceedings of the 3rd Symposium on Usable Privacy and Security*, SOUPS '07, vol. 229. (Pittsburgh, Pennsylvania, July 18–20, 2007). ACM, New York, pp. 100–111. DOI = http://doi.acm.org/10.1145/1280680.1280693.
28. Balfanz, D., Durfee, G., Grinter, R. E., Smetters, D. K., and Stewart, P. (2004). Network-in-a-box: how to set up a secure wireless network in under a minute. *13th Usenix Security Symposium*. San Diego, CA, August, 2004.
29. Lau, S. (2003). *The Spinning Cube of Potential Doom*. *National Energy Research Scientific Computing Center*, [Online] December 10, 2003. [Cited: January 20, 2008.] http://www.nersc.gov/nusers/security/TheSpinningCube.php.
30. Stoll, J., Tashman, C., Edwards, W. K., and Spafford, K. (2008). Sesame: informing user security decisions with system visualization. *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI 2008)*. Florence, April 5–10, 2008.
31. Zurko, M. E. and Simon, R. T. (1996). User-centered security. *Proceedings of the 1996 Workshop on New Security Paradigms*, NSPW '96. (Lake Arrowhead, California, United States, September 17–20, 1996). ACM, New York, pp. 27–33. DOI = http://doi.acm.org/10.1145/304851.304859.
32. Zurko, M. E., Simon, R., and Sanfilippo, T. (1999). A user-centered, modular authorization service built on an RBAC foundation. *Proceedings of the 1999 IEEE Symposium on Security and Privacy*. Oakland, CA, pp. 57–71.
33. Maxion, R. A. and Reeder, R. W. (2005). Improving user-interface dependability through mitigation of human error. *Int. J. Human Comp. Studies* **63**(1-2), 25–50. [DOI: 10.1016/j.ijhcs.2005.04.009]
34. Reeder, R. W., Bauer, L., Cranor, L. F., Reiter, M. K., Bacon, K., How, K., and Strong, H. (2008). Expandable grids for visualizing and authoring computer security policies. *ACM SIGCHI Conference on Human Factors in Computing Systems* (CHI '08). Florence, Italy.
35. Karat, J., Karat, C.-M. Brodie, C., and Feng, J. (2005). Privacy in information technology: designing to enable privacy policy management in organizations. *Int. J. Human Comp. Studies* **63**(1-2), 153–174.
36. Cao, X. and Iverson, L. (2006). Intentional access management: making access control usable for end-users. *Proceedings of the Second Symposium on Usable Privacy and Security*, SOUPS '06, vol. 149. (Pittsburgh, Pennsylvania, July 12–14, 2006). ACM, New York, pp. 20–31. DOI = http://doi.acm.org/10.1145/1143120.1143124.
37. Jakobsson, M. and Myers, S. A. Ed. (2006). *Phishing and Countermeasures: Understanding the Increasing Problem of Electronic Identity Theft*. Wiley, Hoboken, NJ.
38. Dhamija, R., Tygar, J. D., and Hearst, M. (2006). Why phishing works. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '06. (Montréal, Québec, Canada, April 22–27, 2006). R. Grinter, T. Rodden, P. Aoki, E. Cutrell, R. Jeffries, and G. Olson, Eds. ACM, New York, pp. 581–590. DOI = http://doi.acm.org/10.1145/1124772.1124861.
39. Jackson, C., Simon, D., Tan, D., and Barth, A. (2007). An evaluation of extended validation and picture-in-picture phishing attacks. *Proceedings of the 1st International Conference in Usable Security (USEC07), part of the Proceedings of the Conference on Financial Cryptography*. Lowlands, Scarborough, Trinidad/Tobago.

40. Downs, J. S., Holbrook, M. B., and Cranor, L. F. (2006). Decision strategies and susceptibility to phishing. *Proceedings of the Second Symposium on Usable Privacy and Security*. Pittsburgh, PA, July 12–14, 2006. ~[doi>10.1145/1143120.1143131]

41. Kumaraguru, P., Rhee, Y., Acquisti, A., Cranor, L. F., Hong, J., and Nunge, E. (2007). Protecting people from phishing: the design and evaluation of an embedded training email system. *CHI 2007: Conference on Human Factors in Computing Systems*, San Jose, CA, April 28–May 3, 2007, pp. 905–914.

42. Sheng, S., Magnien, B., Kumaraguru, P., Acquisti, A., Cranor, L. F., Hong, J., and Nunge, E. (2007). Anti-phishing phil: the design and evaluation of a game that teaches people not to fall for phish. *Proceedings of the 2007 Symposium On Usable Privacy and Security*. Pittsburgh, PA, July 18–20 200.

43. Wu, M., Miller, R. C., and Garfinkel, S. L. (2006). Do security toolbars actually prevent phishing attacks? *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '06. (Montréal, Québec, Canada, April 22–27, 2006). R. Grinter, T. Rodden, P. Aoki, E. Cutrell, R. Jeffries, and G. Olson, Eds. ACM, New York, pp. 601–610. DOI = http://doi.acm.org/10.1145/1124772.1124863.

44. Zhang, Y., Egelman, S., Cranor, L., and Hong, J. (2007). Phinding phish: evaluating anti-phishing tools. 2007. *Proceedings of the 14th Annual Network & Distributed System Security Symposium (NDSS 2007)*, San Diego, CA, February 28th–2nd March.

45. Zhang, Y., Hong, J., and Cranor, L. (2007). CANTINA: a content-based approach to detecting phishing web sites. 2007. *Proceedings of the 16th International World Wide Web Conference (WWW2007)*. Banff, AB, May 8–12, 2007, pp. 639–648.

46. Cook, D. L., Gurbani, V., and Daniluk, M. (2008). Phishwish: a stateless phishing filter using minimal rules. *Proceedings of Financial Crypto*, El Cozumeleno Beach Resort, Cozumel, January, 2008.

47. Provos, N., McNamee, D., Mavrommatis, P., Wang, K., and Modadugu, N. (2007). The ghost in the browser analysis of web-based malware. *Proceedings of the First Conference on First Workshop on Hot Topics in Understanding Botnets*. (Cambridge, MA). USENIX Association, Berkeley, CA, pp. 4–4.

48. Dhamija, R. and Tygar, J. D. (2005). The battle against phishing: dynamic security skins. *Proceedings of the 2005 Symposium on Usable Privacy and Security*, SOUPS '05, vol. 93. (Pittsburgh, Pennsylvania, July 06–08, 2005). ACM, New York, pp. 77–88. DOI = http://doi.acm.org/10.1145/1073001.1073009.

49. Bank of America (2006). *How Bank of America SiteKey Works for Online Banking Security*. *Bank of America*, [Online] 2006. [Cited: January 19, 2008.] http://www.bankofamerica.com/privacy/sitekey/.

50. Schechter, S. E., Dhamija, R., Ozment, A., and Fischer, I. (2007). *The Emperor's New Security Indicators*. IEEE Computer Society, Washington, DC, SP '07: Proceedings of the 2007 IEEE Symposium on Security and Privacy. pp. 51–65.

51. Balfanz, D., Durfee, G., Grinter, R. E., and Smetters, D. K. (2004). In search of usable security—five lessons from the field. *IEEE J. Secur. Priv.* **2**(5), 19–24.

52. Balfanz, D., Durfee, G. and Smetters, D. K. (2005). Making the Impossible easy: usable PKI. In *Security and usability: Designing Secure Systems that People Can Use*, L. F. Cranor and S. Garfinkel, Eds. O'Reilly Media, Sebastopol, CA, pp. 319–334.

53. Balfanz, D., Smetters, D. K., Stewart, P., and Wong, H. C. (2002). Talking to strangers: authentication in ad-hoc wireless networks. *Network and Distributed System Security Symposium*. Internet Society, San Diego, CA, February 6–8, 2002.

54. Stajano, F. and Anderson, R. J. (2000). The resurrecting duckling: security issues for Ad-hoc wireless networks. In *Proceedings of the 7th international Workshop on Security Protocols*, Lecture Notes In Computer Science, vol. 1796 (April 19–21, 1999). B.

Christianson, B. Crispo, J. A. Malcolm, and M. Roe, Eds. Springer-Verlag, London, pp. 172–194.

55. McCune, J. M., Perrig, A., and Reiter, M. K. (2005). Seeing-Is-believing: using camera phones for human-verifiable authentication. *Proceedings of the 2005 IEEE Symposium on Security and Privacy*, (May 08–11, 2005). IEEE Computer Society, Washington, DC, pp. 110–124. DOI = http://dx.doi.org/10.1109/SP.2005.19.
56. Balfanz, D. (2003). Usable access control for the world wide web. *Proceedings of the 19th Annual Computer Security Applications Conference*, ACSAC, (December 08–12, 2003). IEEE Computer Society, Washington, DC, p. 406.
57. Gutmann, P. (2003). Plug-and-play PKI: a PKI your mother can use. *Proceedings of the 12th Conference on USENIX Security Symposium—Volume 12*, (Washington, DC, August 04–08, 2003). USENIX Association, Berkeley, CA, pp. 4–4.
58. Gutmann, P. *Underappreciated security mechanisms*. *Peter Gutmann*, [Online] [Cited: 1 20, 2008.] http://www.cs.auckland.ac.nz/∼pgut001/pubs/underappreciated.pdf.
59. Garfinkel, S. L. and Miller, R. C. (2005). Johnny 2: a user test of key continuity management with S/MIME and outlook express. *Proceedings of the 2005 Symposium on Usable Privacy and Security*, SOUPS '05, vol. 93. (Pittsburgh, Pennsylvania, July 06–08, 2005). ACM, New York, pp. 13–24. DOI=http://doi.acm.org/10.1145/1073001.1073003.
60. Parno, B., Kuo, C., and Perrig, A. (2006). Phoolproof phishing prevention. *Financial Cryptography and Data Security 10th International Conference*. British West Indies, February 27–March 2, 2006.
61. Corner, M. D. and Noble, B. D. (2002). Zero-interaction authentication. *Proceedings of the 8th Annual international Conference on Mobile Computing and Networking*, (Atlanta, Georgia, USA, September 23–28, 2002). ACM, New York, pp. 1–11. DOI=http://doi.acm.org/10.1145/570645.570647.
62. Bauer, L., Cranor, L. F., Reeder, R. W., Reiter, M. K., and Vaniea, K. (2008). A user study of policy creation in a flexible access-control system. *ACM SIGCHI Conference on Human Factors in Computing Systems* (CHI '08).
63. Smetters, D. K., Balfanz, D., Durfee, G. E., Smith, T., and Lee, K. (2006). Instant matchmaking: simple, secure virtual extensions to ubiquitous computing environments. Ubicomp 2006, *Proceedings of the 8th International Conference of Ubiquitous Computing*. Springer Verlag, Irvine, CA, September 17–21, 2006; LCS 4206: pp. 477–494.
64. Yee, K.-P. (2002). User interaction design for secure systems. In *Proceedings of the 4th International Conference on Information and Communications Security*, Lecture Notes in Computer Science 2513, R. Deng, S. Qing, F. Bao, and J. Zhou, Eds. Springer-Verlag, Heidelberg, http://zesty.ca/sid/.
65. Yee, K.-P. (2005). Guidelines and strategies for secure interaction design (Chapter 13). In *Security and Usability: Designing Secure Systems that People Can Use*, L. F. Cranor and S. Garfinkel, Eds. O'Reilly, Sebastopol, CA.
66. Chiasson, S., Biddle, R., and Somayaji, A. (2007). Even experts deserve usable security: design guidelines for security management systems. *Workshop on Usable IT Security Management (USM'07) held with the ACM Symposium on Usable Privacy and Security (SOUPS 2007)*, July 2007.
67. Mannan, M., van Oorschot, P. C. Security and usability: the gap in real-world online banking. *New Security Paradigms Workshop (NSPW)*. New Hampshire. Sept. 18–21, 2007.

FURTHER READING

Cranor, L. F. and Garfinkel, S. *Security and Usability: Designing Secure Systems that People Can Use*. O'Reilly & Associates, 2005.

Gutmann, P. (2008). *Usable Security Fundamentals*, http://www.cs.auckland.ac.nz/∼pgut001/pubs/usability.pdf.

The HCISEC Bibliography. http://gaudior.net/alma/biblio.html.

Yee, K.-P. *The Usable Security Blog*. http://usablesecurity.com/.

5

SECURITY OF DISTRIBUTED, UBIQUITOUS, AND EMBEDDED COMPUTING PLATFORMS

Anthony D. Wood and John A. Stankovic
University of Virginia, Charlottesville, Virginia

5.1 INTRODUCTION

Computer systems and networks are becoming more capable—and more vulnerable—as they are embedded more deeply into our environment. In this chapter, we describe security challenges faced by ubiquitous distributed systems: ad hoc networks of handheld computers, sensor networks for directly interacting with the world, and radio frequency identification (RFID) tags that instantiate real-world objects with elements in our virtual computer systems. We review promising research approaches, and identify important future directions in these application areas.

5.2 SCIENTIFIC OVERVIEW

The confluence of wireless networking, increasing transistor densities (Moore's Law), and miniaturization of manufacturing processes has accelerated the deployment of computer networks. Computing devices are now lightweight, portable, unobtrusive, powerful, and more well connected than ever. Adding environmental and biological sensors tightens the connection with the real world, so that computing is not just embedded in non computing devices (like the proverbial Internet toaster), but is embedded in our living spaces.

We focus on three developing technology areas, represented in Figures 5.1 and 5.2: ad hoc networks, wireless sensor networks (WSNs), and RFID tags. Their applications range widely and are expanding, including military battlefield awareness, airport surveillance, emergency medical care, disaster response, critical infrastructure monitoring, container tracking, facilities access control, firearm and vehicle immobilizers, currency and travel document fraud detection, and border enforcement.

Security requirements are unique for each application, but overall they are becoming increasingly significant due to several factors. The systems being monitored, controlled, or protected are often critical for economic or safety reasons. Technological societies are

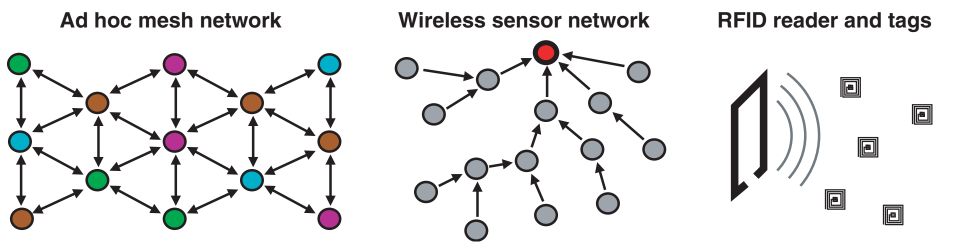


FIGURE 5.1 Example network connectivity for an ad hoc network, sensor network, and RFID reader and tags.

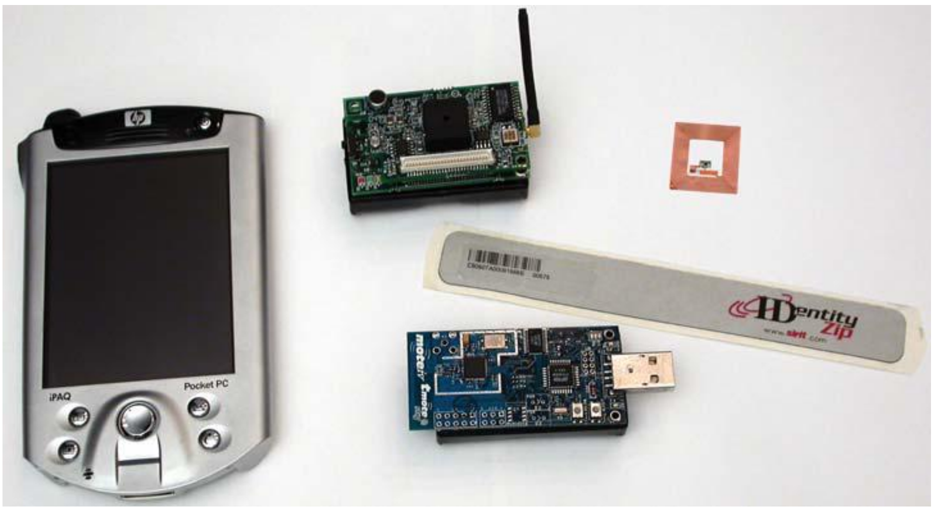


FIGURE 5.2 Examples of embedded devices: PDA, sensors, and RFID tags.

becoming more dependent on their proper operation and real-time response. The networks are pervasive in many environments, where they are easily accessible and, therefore, exposed to greater threats. For example, wireless accessibility, while a great convenience, also makes it easier for attackers to find and interact with devices. Finally, the deepening of familiarity with and acceptance of computing devices extends to the unscrupulous, as well. The constant attacks that occur daily on the Internet, from ego-boosting web defacements to vengeful distributed denial of service botnets, may eventually be the norm on any accessible network.

5.2.1 Security Properties

Security properties can be distilled to a core set, many of which may be important for a given application. Many others are defined in the literature [1], and here we describe those discussed in this chapter, giving brief examples of their use.

- *Confidentiality.* Secrecy of communication between parties. Exposure of communications in wireless networks makes eavesdropping a constant threat.

- *Integrity.* Assurance that data has not been modified by an unauthorized party. This applies to messages in transit, records stored in databases, and even data possessed by attackers (such as on stolen smart cards).
- *Authenticity.* Assurance that a message originated from a known other party. Among others, command and control systems require high confidence that actions with large or dangerous effects have been issued by appropriate means. Message authentication codes (MACs) are often appended to protocol messages to provide this property.
- *Identification.* Determination of a contextually unique label for a party. It enables authentication of a party and authorization of actions it may take. Also, a persistent ID allows goods to be tracked through supply chains, from manufacturers to shelves.
- *Authorization.* Determination of privileges from a party's identity. System designs may change the authorized set of actions a party may take based on environmental contexts, for example, granting additional access during medical emergencies.
- *Access control.* Limitations on exposure or modification of protected resources to authorized parties. An RFID token that serves as a "key" for an automobile is a form of access control.
- *Availability.* A service or system performs its function in a timely manner for legitimate users. Denial of service attacks may crash a system completely, or may only slow it down enough to cause significantly disrupted service.
- *Auditability.* Logging of security-relevant actions or events for later analysis. Many attacks cannot be reliably detected in real time, but can be analyzed after the fact to help with future defenses.
- *Tamper resistance.* Ability of a device's packaging and design to withstand physical modification or interrogation. Smart cards, though in public possession, often contain secret keys, which must remain secret to prevent changing credit balances.
- *Nonrepudiation.* Inability of a user or device to deny participation in a protocol or performance of an operation after the fact. This is often related to auditability.

5.2.2 Constraints on the Design Space

Constraints on design are imposed by considerations such as available power, cost to manufacture and maintain, form factor and size, tamper resistance, development effort, the ability to dynamically reprogram, and intended architectures for deployment. Devices in ubiquitous embedded networks form a spectrum of capabilities, from PDAs to passive RFID tags, and are connected together in varying ways.

Ad hoc networks [2] connect (frequently) mobile devices together in a relatively flat mesh and usually depend on peer routing for connectivity (Fig. 5.1). Hardware typically consists of cell phones, mobile handheld computers (PDAs), and laptop computers, with relatively powerful processors such as the Intel PXA255 running at 400 MHz. They may use networks with high bandwidth, for example Institute of Electrical and Electronics Engineers (IEEE) 802.11b/g, to deliver multimedia. Storage on internal and removable flash drives with capacity up to 2 GB is common.

WSNs [3] may also use node-to-node ad hoc connectivity, perhaps organized hierarchically with one or more nodes to act as sinks for generated data. Devices are primarily constrained by size, cost, and power. For example, the Crossbow Mica2 family of motes uses the 8-bit Atmel ATMEGA128 processor operating at 8 MHz, with 4 KB RAM and 128 KB flash. Simple Frequency Shift Keying (FSK) modulation at 900 MHz, IEEE 802.15.4, or Bluetooth radio communication is common.

RFID tags [4] are even more limited. Most are completely passive, using the energy of a reader's transmission to briefly power the tag's processing circuit. The tag communicates by modulating the reader's transmission. Tags may be smaller than 1–2 mm (without the antenna), and operate in the high-frequency (HF) band (13.56 MHz) for intermediate range.

Security comes at the cost of memory, computation, and messaging [5, 6]. Ad hoc network devices may be able to afford expensive asymmetric cryptography, storing 1024-bit keys for their neighbors, and participating in multi-round key establishment protocols. Sensor network devices cannot afford this computation and storage expense, unless very efficient elliptic-curve cryptographic (ECC) methods are used only infrequently. Instead, most researchers focus on lighter-weight symmetric cryptography and hashing in this context. Many RFID tags provide no security at all. Those that do may use only hashing or very efficient symmetric methods.

Hence, there are considerable differences in the security approaches that are practical and possible in distributed, embedded, and ubiquitous networks. Next, we describe the state of important research areas in security for these types of systems.

5.2.3 Solution Approaches

Distributed devices typically use layering to modularize hardware and software. Figure 5.3 shows generic software stacks for ad hoc and wireless sensor devices, and how services may be classified by layer. Because of their limited capabilities, RFID tags may be considered to have only a couple of layers. For this discussion, we abstract away many details unique to each network type.

Strong security mechanisms at higher layers may be completely subverted by design or coding flaws at lower layers. Nowhere isthis more evident than at the physical layer. Therefore, we describe attacks and defenses proposed in the state of the art by focusing on solutions to securing services at the critical physical, network, and middleware/application layers of the stack, starting at the bottom. We discuss ad hoc, sensor, and RFID networks together in each layer.

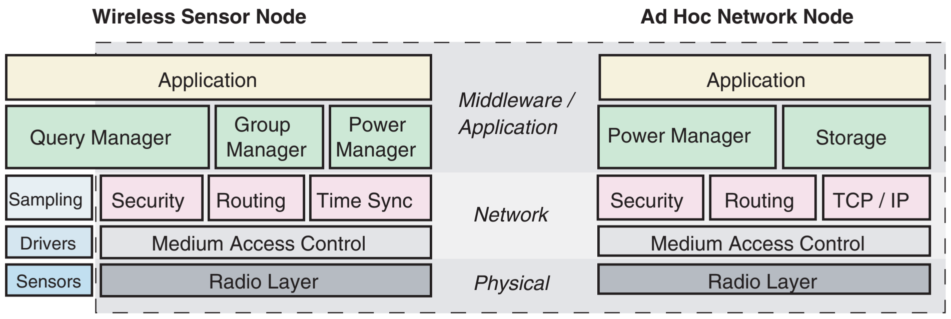


FIGURE 5.3 Typical software stacks for wireless sensor network and ad hoc network devices. A dashed box surrounds the communication layers, which contain various services often found in the networks.

5.2.4 Physical Layer

The ubiquity of network devices means that they are easily inspected and probed by attackers. There is by definition no physical access control to sensors that are deployed throughout a public building, in parks, forests, or other open spaces. RFID tags attached to books, clothes, or supplies are necessarily as accessible as the asset they help to track.

The simplest attack is to destroy or disable the devices entirely, creating a denial of service. This is as low-tech as briefly putting a banknote or passport in a microwave oven. However, a destruction attack can often be mitigated using fault-tolerant protocols. For example, a mesh network can continue to operate despite some fraction of the devices being destroyed—remaining connected nodes take over routing. Or, if a passport's RFID tag is destroyed, backup procedures such as optical scanning of barcodes may be used instead.

Probing of the physical device to deconstruct its internals is more powerful and damaging. By reading the contents of memory cells, the secret keys are recovered and then programmed into another device, which can fully masquerade as the original—yet is under attacker control. Messages originated by the clone are fully authentic, and the device can actively participate in formerly inaccessible transactions, as between a smart card and a payment terminal [7].

In addition to invasive techniques that usually require partial depackaging of a device, various physical properties of the circuits can be inspected without leaving a trace. Data-dependent computation affects the power consumption and timing of circuits, which can be analyzed statistically over many trials to determine bit patterns of a key [8]. Faults may be injected using heat or radiation, while the observed behavior is compared with correct behavior. Electromagnetic emissions may be inspected similarly to power consumption.

Proposed solutions include tamper-resistant packaging [7], better attack detection, fault recovery mechanisms, and reducing trust in external components [9]. For example, if a device can detect that it is being tampered with, it may erase its memory to prevent disclosure of sensitive data. Circuits may be shielded by distributing logical components across the die, bus traffic may be encrypted, and data values and timing can be randomized or "blinded" to thwart side-channel attacks.

Devices' use of wireless communication leaves them vulnerable to denial of service by radio jamming, which can be perpetrated at large distances and unobtrusively. Xu et al. propose channel hopping and "retreats" [10] to physically move away from the jammer. This is most appropriate for ad hoc networks, as it may be too energy consuming for sensor devices. Law et al. propose data blurting and changing transmission schedules as countermeasures [11]. Another approach, when the jamming cannot be avoided, is for nodes to determine the extent of the jammed area in a wide-scale network by collaboratively mapping and avoiding the region [12].

5.2.5 Networking Layers

We group the networking layers of the stack together to examine the security needs and vulnerabilities created when connecting multiple devices in networks. Services provided include channel arbitration, link establishment, one-hop data transmission, routing, and end-to-end data transport.

A primary concern is keeping data private, given the innate vulnerability to eavesdropping of wireless networks. Ad hoc networks often use protocols developed for the

Internet, such as IPSec [13] and SSL/TLS [14]. Both these protocols allow the use of suites of cryptographic mechanisms to provide authenticity, integrity, and confidentiality of messages. IPSec is commonly used to establish a secure virtual private network (VPN) connection between peers. Secure Sockets Layer/Transport Layer Security (SSL/TLS) operates end-to-end, above an existing Transaction Control Protocol (TCP) connection, and further allows the client and server to negotiate a common set of capabilities. Though asymmetric methods may be used to establish keys and authenticate certificates, symmetric cryptographic protocols are used for data transfer.

In WSNs and RFID devices, symmetric mechanisms are encapsulated in lightweight protocols to provide data security. SPINS [15] provides two-party confidentiality and authenticity with the SNEP protocol. TinySec [16] similarly provides these properties using Skipjack or RC5 ciphers, in a fully implemented and compact form with low overheads.

Because of energy constraints, sensor devices cannot use asymmetric cryptographic operations often. TinyPK [17] is an implementation of the relatively less demanding signature verification and key agreement for sensor devices. Though processing times for a single message may be over a minute (depending on the key length), they argue that it is acceptable for rare events, like code updating. Recent elliptic-curve implementations [18] improve efficiency, making slightly more frequent use of public-key infrastructures possible. For RFID tags with modest resources, researchers have proposed simple authentication and encryption to prevent "skimming," or physical proximity-based interrogation of tags [4].

In addition to neighbor-to-neighbor communication, many networks require secure broadcast and multicast communication. Often a control station must broadcast parameter changes to an entire network, and authentication of these messages is imperative. Both Timed Efficient Stream Loss-tolerant Authentication (TESLA) [19] (for ad hoc networks) and uTESLA [15] (for sensor networks) provide broadcast authentication. A base station commits to a chain of one-way hashes, and then uses each in reverse sequence as a key to authenticate a message. After the message has been distributed, the next key in the chain is released. Network nodes validate that $K_i = H(K_{i-1})$ and deliver the message. If all keys in the chain are exhausted, the base station must again securely distribute the commitment (last value) for a new chain to every network node.

LKHW [20] merges logical key hierarchy (LKH) with directed diffusion to provide secure multicast for groups in sensor networks. Directed diffusion is a routing protocol in which sinks diffuse interests for events, and sources send messages along "interest gradients" that find all sinks. LKHW allows group membership to change and provides backward and forward secrecy.

Any protocol that uses cryptographic protection for confidentiality, integrity, or authentication relies on the presence of shared keys. Many approaches for creating and distributing these keys have been proposed. For public-key algorithms, a traditional centralized key distribution architecture may be used, such as Kerberos [21]. Centralized key distribution centers can become performance bottlenecks and attractive targets for attacks, however. By using threshold cryptography, the certification function is distributed among multiple authorities, such that at least k out of n are required to grant certificates [22]. This is more resistant to compromise than a centralized approach, but has higher overhead.

Ad hoc network devices often must collaborate together in groups, using secure multicast communication. In the group key management protocol (GKMP) [23], a centralized controller for each group generates and distributes pairwise keys to the other members. The secure spread service [24] provides multiparty key creation using Group Diffie–Hellman, in which each member contributes to the key.

Any scheme that requires cryptography also requires keys. Much research on key distribution in WSNs has focused on distributing secrets prior to deployment [25–28]. Nodes are preloaded with multiple keys from a large key space. After deployment, nodes discover neighbors with whom they share keys, and use these paths to indirectly establish keys with other neighbors. Adding the requirement for nodes to share q common keys improves the protocols' resistance to compromise. Other protocols, such as Localized Encryption and Authentication Protocol (LEAP) [29], use a globally shared key to create pairwise-shared keys with neighbors during a short initialization period. The network is assumed to be free from compromises during this time, and the global key is erased thereafter.

RFID tags interact only with readers and certain special purpose tags, so the security concerns mostly center on identification and authentication. To prevent cloning attacks, a tag may implement lightweight symmetric cryptography or hashing and be programmed with a unique key. A challenge-response protocol prevents replay attacks, and provides simple identification or authentication. With the most constraints on size and cost, RFID tags are often vulnerable to physical attacks such as those described in the previous section.

Tags that respond to any reader or that respond using the same ID or key pose privacy risks. Weis et al. propose using key-search techniques to conceal a tag's identity from any except legitimate readers [30]. The reader receives $H(k_i, N)$ from a tag, for key k_i and nonce N, and searches through all keys known to the reader for a match, identifying tag i. This is expensive for large numbers of tags, however. Others propose tree-based and synchronization-based schemes to limit the searching necessary, for example, by computing outputs based on an increasing counter.

Ad hoc and sensor networks use devices connected together wirelessly for multihop routing. The use of redundant, dynamic routing paths provides protection against link failures, but it increases the risk of relying on a compromised or adversarial node.

Approaches to securing ad hoc routing have focused on retrofitting existing protocols, or creating new ones to provide desirable properties. Secure Ad hoc On-Demand Distance Vector (SAODV) [31] provides authentication, nonrepudiation, and integrity by means of a protocol extension to Ad hoc On-Demand Distance Vector (AODV) that relies on digital signatures and hashing.

Secure Efficient Ad hoc Distance-vector (SEAD) [32] also addresses security in distance-vector routing protocols, which are suitable for networks with limited mobility. It uses hash chains to secure routing updates, an approach that is more computationally efficient than SAODV and which provides some defense against denial of service attacks.

Protocols such as SEAD and SAODV rely on periodic routing updates, which have high overhead or poor performance when node mobility is high. In these networks, on-demand protocols like Ariadne [33] may be more suitable. Ariadne provides secure on-demand routing based on the Dynamic Source Routing (DSR) protocol, and requires one of the following: network-wide pairwise-shared keys, neighborhood pairwise-shared keys and broadcast authentication (such as TESLA), or digital signatures.

WSNs require very efficient routing mechanisms, since radio transmission consumes so much of their energy budget. Their unique characteristics also pose special difficulties for secure routing [34]. Addressing the many attacks given the constraints of low-end sensor devices is problematic.

Aggregation of information to a centralized base station is a common communication pattern in WSNs. The authors of SPINS [15] suggest using underlying secure unicast and broadcast links (SNEP and uTESLA, respectively) to form routing trees from nodes back to base stations. LKHW targets communication within groups of collaborating devices (as described above), and secures directed diffusion for routing.

Secure Implicit Geographic Forwarding (SIGF) [35] is a family of routing protocols for WSNs, which allows very lightweight operation when no attacks are present, and stronger defenses—at the cost of overhead—when more attacks are detected. It is a form of on-demand routing based on geographic forwarding, where the message destination is specified as a location toward which each hop makes progress. The set of candidates considered at each hop may be increased, and their selection is randomized to prevent persistent selection of neighboring compromised nodes.

INtrusion-tolerant routing protocol for wireless SEnsor NetworkS (INSENS) [36] is an intrusion tolerant protocol for WSNs that need little or no sensor-to-sensor communication, but which have well-defended base stations. Network topology is collected from sensor devices by the base station. Routes are centrally computed and securely distributed to sensors using one-way hash chain sequence numbers, similar to uTESLA.

5.2.6 Middleware and Applications

Above the networking layers, which are concerned with relatively low-level details of interconnection, middleware and application-layer software provide rich and varied services. Networks connected to the physical world must provide mechanisms for extracting important data to authorized parties for analysis and manipulation. Several protocols have been proposed for querying, aggregating, and validating sensor data collected by WSNs.

Secure information aggregation (SIA) [37] uses special nodes in the network to aggregate sensor data. As data are collected and aggregated, results are reported to the base station along with a commitment to the data. Commitments are formed using a binary Merkle hash tree, which reduces the size of the verification information. The base station may then request particular sensor values from the aggregator in an interactive proof, until results are verified with a desired probability.

For large-scale networks where events of interest are witnessed by multiple sensors, Ye at al. propose statistical en-route filtering (SEF) of injected false data [38]. Nodes compute message authentication codes that are aggregated and sent with the reported data. Intermediate nodes check the MACs probabilistically, dropping incorrect messages. A Bloom filter is used to decrease the cost of aggregating multiple MACs.

Reprogramming widely distributed systems is expensive if it requires manual retrieval and manipulation of unattended devices. Over the network reprogramming alleviates this practical difficulty, but presents significant security concerns. All other hardware and software defenses may be subverted by a flaw that allows an attacker to replace nodes' programs with custom code.

Deng et al. [39] propose related schemes for securely distributing code in WSNs. The first uses a chain of hashes, where each message contains segment i of code and a hash of segment $i + 1$. Upon receipt of a message, the previous code segment can be immediately and efficiently verified. To bootstrap the chain, an ECC signature of the first hash value is computed using the base station's private key. This method is suitable when there is little message loss and packets are received mostly in order. The second scheme uses a hash tree to distribute all the hashes in advance, so that out-of-order packets can be quickly checked. Resistance to denial of service is improved since packets need not be stored if they are corrupt.

SCUBA [40] is a protocol for detecting and recovering compromised nodes in sensor networks. Base stations verify code images on nodes using an indisputable code execution (ICE) facility, which ensures that unmodified self-checksumming code runs on the target in the expected time. The ICE code computes checksums over the ROM, ICE function, and main executable. Incorrect checksums or executions that take too long indicate that malicious code is interfering with proper operation of the device. The result of the full Secure Code Update By Attestation (SCUBA) protocol is a repaired or blacklisted node.

Many applications may be built upon the foundations provided by the protocols we have reviewed: physical and radio-layer protections, secure node-to-node communication, multihop routing, data aggregation, and code updating. System designers must determine the attack model most appropriate for their application domain and deployment environment, carefully choosing protocols that defend against possible attacks and that do not create additional points of vulnerability.

5.3 GLOBAL RESEARCH AND FUNDING

National Science Foundation (NSF)'s Embedded and Hybrid Systems (EHS) Program [41] supports research in many aspects of embedded systems technology. A pervasive theme of the EHS program is the high-confidence integration of real-time and other service guarantees with the coordination requirements of next-generation complex, secure, networked, embedded systems.

NSF's Cyber Trust (CT) Program [42] envisions computer networks that are more reliable, accountable, and resistant to attacks, and a workforce that is well trained and educated to operate them. Research proposals that will target security for applications, security for computer systems, security for networks, and new security foundations are solicited. The entire system life cycle may be considered, and multidisciplinary projects with behavioral and social science participation are encouraged.

The European ARTIST2 Consortium [43] supports the Network of Excellence on Embedded Systems Design, which intends to strengthen European research in this area. The testing and verification cluster targets verification of security properties in designs.

5.4 CRITICAL NEEDS ANALYSIS

Embedded systems have already become ubiquitous, but their composition into large-scale systems for monitoring and controlling the physical world is nascent. Advancements in this field will enable many advanced applications, such as secure communication for emergency personnel, disaster-site coordination, border patrol, container tracking and inspection, biological and radiological sensing, traffic control, and civil infrastructure monitoring. Realization of these critical applications will be subject to research progress on many technical fronts, including security and privacy.

5.5 RESEARCH DIRECTIONS

Physical-layer security is often a weak spot in embedded devices even when higher layers are provably sound. Tamper-resistant packaging and designs for smart card, RFID, and sensor devices will be necessary for ubiquitous deployments and deserve more research.

Wireless devices expose the system to monitoring by and remote interaction with attackers. More research in resistance to denial of service attacks by jamming, flooding, and invoking expensive computations is needed to enable continued operation of critical components even while attacks are ongoing.

Connecting virtual and physical worlds raises many privacy concerns. Controversies over RFID-enabled passports and banknotes, urban camera networks, tracking of consumer-products post sale, and disclosure of aggregated data by companies and government agencies all portend a complex future of interdependent technical, legal, and political effects on personal privacy. More fundamental research is needed in ways to preserve privacy despite the collection of unprecedented amounts of data in the public and private sectors.

Data collected by WSNs will be useful for many purposes, but may inadvertently disclose sensitive information—even if the data payloads in network are encrypted. Traffic analysis or simple radio-activity detection may reveal to an attacker whether a home is occupied, the nationality of a traveler in a crowd, or the location of important control devices. Comprehensive research that crosses traditional functional layers and includes noncryptographic approaches to information hiding is needed.

REFERENCES

1. NIST. (2006). *Glossary of Key Information Security Terms*, R. Kissel, Ed. NIST IR, Gaithersburg, MD, 7298. April 25.
2. Royer, E., and Toh, C. A. (1999). Review of current routing protocols for ad-hoc mobile wireless networks. *IEEE Personal Commun*. **6**(2), 46–55.
3. Akyildiz, I. F., Su, W., Sankarasubramaniam, Y., and Cayirci, E. (2002). Wireless sensor networks: a survey. *Comput. Networks* **38**(4), 393–422.
4. Juels, A. (2005). *RFID Security and Privacy: A Research Survey*. Technical Report. RSA Laboratories, pp. 1–19.
5. Perrig, A., Stankovic, J. A., and Wagner, D. (2004). Security in wireless sensor networks. *Commun. of the ACM* **47**(6), 53–57.
6. Ravi, S., Raghunathan, A., Kocher, P., and Hattangady, S. (2004). Security in embedded systems: Design challenges. *Trans. on Embedded Computing Sys*. **3**(3), 461–491.
7. Anderson, R., and Kuhn, M. (1996). *Tamper resistance—a cautionary note*. *Usenix Workshop on Electronic Commerce*. Oakland, California, pp. 1–11.
8. Ravi, S., Raghunathan, A., Chakradhar, S. (2004). Tamper resistance mechanisms for secure, embedded systems. *Proceedings of 17th International Conference on VLSI Design*. Washington, DC, p. 605.
9. Suh, G., Clarke, D., Gassend, B., van Dijk, M., and Devadas, S. (2003). AEGIS: architecture for tamper-evident and tamper-resistant processing. *Proceedings of ICS.* San Francisco, CA, pp. 168–177.
10. Xu, W., Trappe, W., Zhang, Y., Wood, T. (2005). The feasibility of launching and detecting jamming attacks in wireless networks. *Proceedings of MobiHoc*. New York, pp. 46–57.

11. Law, Y. W., Hartel, P. H., den Hartog, J. I., and Havinga, P. J. M. (2005). Link-layer jamming attacks on S-MAC. *Proceedings of EWSN*. Alexandria, Virginia, pp. 217–225.
12. Wood, A. D., Stankovic, J. A., and Son, S. H. (2003). JAM: a jammed-area mapping service for sensor networks. *Proceedings of IEEE RTSS*. Cancun, Mexico, pp. 286.
13. Kent, S., and Seo, K. (2005). *Security Architecture for the Internet Protocol*. IETF RFC-4301.
14. Dierks, T., Allen, C. (1999). *The TLS Protocol*, Version 1.0. IETF RFC-2246.
15. Perrig, A., Szewczyk, R., Wen, V., Culler, D., and Tygar, J. D. (2001). SPINS: security protocols for sensor networks. *Proceedings of MobiCom*. Rome, Italy, pp. 189–199.
16. Karlof, C., Sastry, N., and Wagner, D. (2004). TinySec: a link layer security architecture for wireless sensor networks. *Proceedings of SensSys*. Los Angeles, CA, pp. 162–175.
17. Watro, R., Kong, D., fen Cuti, S., Gardiner, C., Lynn, C., and Kruus, P. (2004). TinyPK: securing sensor networks with public key technology. *Proceedings of SASN*. New York, pp. 59–64.
18. Malan, D. J., Welsh, M., and Smith, M. D. (2004). A public-key infrastructure for key distribution in TinyOS based on elliptic curve cryptography. *Proceedings of SECON*. Santa Clara, CA.
19. Perrig, A., Canetti, R., Tygar, D., and Song, D. (2002). The TESLA broadcast authentication protocol. In *RSA Cryptobytes*, RSA Laboratories, Bedford, MA, Vol. 5.
20. Pietro, R. D., Mancini, L. V., Law, Y. W., Etalle, S., and Havinga, P. J. M. (2003). LKHW: a directed diffusion-based secure multicast scheme for wireless sensor networks. *32nd International Conference on Parallel Processing Workshops (ICPP 2003 Workshops).* Kaohsiung, Taiwan.
21. Steiner, J. G., Neuman, C., and Schiller, J. I. (1988). Kerberos: an authentication service for open network systems. *Proceedings of USENIX*. San Francisco, CA, pp. 191–200.
22. Zhou, L., and Haas, Z. (1999). Securing ad hoc networks. *IEEE Network* **13**(6), 24–30.
23. Harney, H., and Muckenhirn, C. (1997). *Group Key Management Protocol (GKMP) Architecture*. IETF RFC 2094.
24. Amir, Y., Kim, Y., Nita-Rotaru, C., Schultz, J. L., Stanton, J., and Tsudik, G. (2004). Secure group communication using robust contributory key agreement. *IEEE Trans. on Parallel and Distributed Syst*. **15**(5), 468–480.
25. Du, W., Deng, J., Han, Y. S., and Varshney, P. K. (2003). A pairwise key pre-distribution scheme for wireless sensor networks. *Proceedings of ACM CCS*. New York, pp. 42–51.
26. Chan, H., Perrig, A., and Song, D. (2003). Random key predistribution schemes for sensor networks. *IEEE Symposium on Security and Privacy.* Oakland, CA, pp. 197–213.
27. Liu, D., and Ning, P. (2003). Efficient distribution of key chain commitments for broadcast authentication in distributed sensor networks. *Proceedings of NDSS*. San Diego, CA, pp. 263–276.
28. Eschenauer, L., and Gligor, V. D. (2002). A key-management scheme for distributed sensor networks. *Proceedings of ACM CCS*. Washington, DC, pp. 41–47.
29. Zhu, S., Setia, S., and Jajodia, S. (2003). LEAP: efficient security mechanisms for large-scale distributed sensor networks. *Proceedings of ACM CCS*. Washington, DC.
30. Weis, S., Sarma, S., Rivest, R., and Engels, D. (2003). Security and privacy aspects of low-cost radio frequency identification systems. In *International Conference on Security in Pervasive Computing (SPC 2003).* v. 2802 of Lecture Notes in Computer Science, D. Hutter, G. Mueller, W. Stephan, and M. Ullmann, Eds. Springer-Verlag, Berlin, Germany, pp. 454–469.
31. Zapata, M. G. (2001). *Secure Ad hoc On-Demand Distance Vector (SAODV) Routing*. IETF Internet Draft, draft-guerrero-manet-saodv-00.txt.

32. Hu, Y.-C., Johnson, D. B., and Perrig, A. (2002). SEAD: secure efficient distance vector routing for mobile wireless ad hoc networks. *Proceedings of the Fourth IEEE Workshop on Mobile Computing Systems and Applications*. Callicoon, NY.
33. Hu, Y.-C., Perrig, A., and Johnson, D. B. (2002). Ariadne: a secure on-demand routing protocol for ad hoc networks. *Proceedings of ACM MobiCom*. Atlanta, Georgia.
34. Karlof, C., and Wagner, D. (2003). Secure routing in wireless sensor networks: attacks and countermeasures. *In First IEEE International Workshop on Sensor Network Protocols and Applications*. Anchorage, AK.
35. Wood, A. D., Fang, L., Stankovic, J. A., and He, T. (2006). SIGF: a family of configurable, secure routing protocols for wireless sensor networks. *Proceedings of SASN*. Alexandria, VA.
36. Deng, J., Han, R., and Mishra, S. (2005). INSENS: Intrusion-tolerant routing for wireless sensor networks. *Elsevier J. on Comput. Commun., Special Issue on Dependable Wireless Sens. Networks* **29**(2), 216–230.
37. Przydatek, B., Song, D., and Perrig, A. (2003). SIA: secure information aggregation in sensor networks. *Proceedings of ACM SenSys*. Los Angeles, CA.
38. Ye, F., Luo, H., Lu, S., and Zhang, L. (2004). Statistical en-route detection and filtering of injected false data in sensor networks. *Proceedings of IEEE INFOCOM.* Hong Kong.
39. Deng, J., Han, R., and Mishra, S. (2006). Secure code distribution in dynamically programmable wireless sensor networks. *Proceedings of ACM/IEEE IPSN.* Nashville, TN, pp. 292–300.
40. Seshadri, A., Luk, M., Perrig, A., van Doorn, L., and Khosla, P. (2006). SCUBA: Secure Code Update By Attestation in Sensor Networks. *Proceedings ACM WiSe*. Los Angeles, CA, pp. 85–94.
41. NSF's Embedded and Hybrid Systems (EHS) Program, (2006). URL: http://www.nsf.gov/funding/pgm_summ.jsp?pims_id=5139.
42. NSFs Cyber Trust (CT) Program, (2006). URL: http://www.nsf.gov/funding/pgm_summ.jsp?pims_id=13451.
43. European ARTIST2 Consortium, (2006). URL: http://www.artist-embedded.org/FP6/.

FURTHER READING

Anderson, R. (2001). *Security Engineering*, John Wiley & Sons, New York.

Karl, H., and Willig, A. (2005). *Protocols and Architectures for Wireless Sensor Networks*, John Wiley & Sons, England.

Zhao, F., and Guibas, L. (2004). *Wireless Sensor Networks: An Information Processing Approach*, Morgan Kaufmann, San Francisco, CA.

Murthy, C. S. R., and Manoj, B. S. (2004). *Ad Hoc Wireless Networks: Architectures and Protocols*, Prentice Hall Ptr, Upper Saddle River, New Jersey.

Basagni, S., Conti, M., Giordano, S., and Stojmenovie, I., Eds. (2004). *Mobile Ad Hoc Networking*, Wiley-IEEE Press, New York, NY.

6

ADVANCED ATTACKER DETECTION AND UNDERSTANDING WITH EMERGING HONEYNET TECHNOLOGIES

RONALD C. DODGE JR.
United States Military Academy, West Point, New York

THORSTEN HOLZ
Aachen University, Aachen, Germany

ANTON CHUVAKIN
LogLogic, San Jose, California

6.1 HONEYPOT ESSENTIALS

Honeypots and honeynets are well known to security processionals. Newly developed honeynet techniques and technologies are a hot topic in information security. However, the amount of technical information available on their setup, configuration, and maintenance is still sparse as are qualified people with the capability to run them, interpret the activity, and make security recommendations. Higher-level guidelines, such as a need and business case determination, are similarly absent. In addition, honeypot risks, such as legal authority and ethical use, need to be fully evaluated prior to honeynet deployments. In this chapter, we present a brief introduction to honeynet technologies, a short word on ethical and legal considerations, and then describe four of the most productive honeynet technologies.

What is a honeypot? Lance Spitzner, a founder of Honeynet Project [1] defines a honeypot as “a security resource whose value lies in being probed, attacked or compromised”. Thus, a goal of such a masochistic system is to be compromised and abused. Hopefully, each time a honeypot goes up in smoke, the researcher learns a new technique. For example, you can use a honeypot to find new rootkits, exploits, or backdoors before they become mainstream.

The term *honeynet*, originated by the Honeynet Project, means a network of honeypots. The configuration of the honeypots is specific to the network and services

Cyber Security, Edited by John G. Voeller
© 2014 John Wiley & Sons, Inc. Published 2014 by John Wiley & Sons, Inc.

architecture of the environment you are defending. Further the honeynet is configured so that if a honeypot is compromised, the attacker cannot use that system to attack systems in your production network or external systems. Details of this configuration are described in Section 3.1. In some configurations, the system software of the honeypots is slightly modified (in the same manner as a rootkit works) to help monitor activity and encrypted communication of attackers. The Honeynet Project defines such honeypots as "high-interaction" honeypots, meaning that attackers interact with a fully functional deception system exactly as they would with a real victim machine. On the other hand, various honeypot and deception daemons are "low interaction", since they only provide an illusion to an attacker. These systems only hold attacker attention for a short time. Examples of low-interaction honeypots include, Honeyd [2], Specter [3], and KFSensor [4]. Such honeypots have value as an early attack indicator collecting statistical data, and collecting malware, but do not yield in-depth information about the attackers.

Honeypots can further be separated into client and server honeypots. Client honeypots or "honeyclients" masquerade not as a legitimate server, but as a legitimate client system, such as a web surfer. A honeyclient might be compromised when trying to connect to a malicious or compromised server. Honeyclients are perfect for collecting web-deployed malware and web exploits.

What are some of the common sense prerequisites for running a honeynet? First, security basics should be covered. If your firewall crashes or your IDS misses attacks, you are clearly not yet ready for a honeypot deployment. Running a honeypot also requires advanced knowledge in computer security, network, platform, and application levels. Obviously, the compromised honeypot systems (whether client or server) should not be allowed to attack other systems.

The Honeynet Project defines guidelines on data control (capability required to control the network traffic flow in and out of the honeynet in order to contain the blackhat actions within the defined policy) and data capture (defines the information that should be captured on the honeypot systems for future analysis, data retention policies, and standardized data formats) for the deployed honeynet. They distill the above ideas and guidelines into a well-written document "Honeynet Definitions, Requirements, and Standards" [5].

The deployment of honeynets must be carefully planned and guidance sought to ensure that legal requirements are followed. Do you have the authority to monitor network traffic? What requirements exist for the data contained within the network packets? What must be done if your honeypots are actually attacked? If the attacker then engages in illegal activity using your honeypots, are you liable?

6.2 HONEYPOT RISK; LEGAL AND ETHICAL ISSUES

Running a honeypot incurs some risks. There are many laws, in virtually all nations that cover a person's expectation of privacy and the culpability incurred when your systems are involved in a cyber crime. Despite the risks, running the honeypot is an exciting and educational experience, which also contributes to a state of the art in information security.

What are some of the legal issues typically associated with running a honeypot?

Liability risk is the most common issue. Can you get sued if an attacker uses your honeynet to attack other, possible sensitive, systems? Given that there is very little case law, it is still too early to decide how significant this is. However, the more freedom is given to the attacker for the sake of creating a realistic “production-like” environment, the more risk of such liability is present.

The privacy and handling of the data captured is also a concern. The data packet may contain communications from unknowing bystanders or sensitive data (such as credit card or other personal information). Proper safeguards must be used to ensure that this type of data is protected from disclosure. A bizarre issue that is sometimes brought up is whether the honeypots infringe upon the rights (such as the privacy right) of the hackers? While this sounds truly preposterous, the cases where burglars sued the victims who wounded them while defending their property are no less ridiculous—and they actually happened!

Richard Salgado, former senior counsel for the Department of Justice’s computer crime unit, investigated some of the legal issues related to honeynet operation. He did confirm that “There are some legal issues here, and they are not necessarily trivial, and they’re not necessarily easy.” For example, in one of his media interviews [6], Mr Salgato mentioned a case where “an accused kidnapper who was using a cloned cell phone sued for the interception of the cell phone conversations and won.”

The discussion of the legalities of deploying honeynets can be a very long one. The specific legal restrictions of the implementer’s nation or province must be well understood. As the primary focus of this chapter is to describe emerging honeynet technologies, the reader should review the material available specifically covering legal issues on the Honeynet Project website [7] as well as other publications [8, 9].

6.3 HONEYNET TECHNOLOGIES

Honeynet technologies have matured over the last 9 years to a collection of sophisticated architectures that enable a relatively safe deployment of honeypots in the traditional high-interaction format to the new Global Distributed Honeynet (GDH) or as a honeyclient. The high-interaction honeypot is the most recognized honeynet technology. The latest version, generation III, brings the latest in ease of deployment, data control and capture, and analysis.

6.3.1 Generation III Honeynet

The generation III honeynet consists of an architecture where the honeypots reside behind a data control and capture device called a *honeywall*. The honeywall is a data link layer (layer 2) proxy. The software for the Honeynet Project honeywall is distributed on a bootable CD and is currently built using a CentOS kernel. This layer 2 proxy intercepts all inbound and outbound packets and uses a variety of mechanisms including rate limiting, selective dropping, and “bit-flipping,” to mitigate the risk of a honeynet being used to exploit or scan yours or other’s networks.

As shown in Figure 6.1, the honeywall is placed between the honeynet and the rest of the network. The organizations production systems (for example, e-mail or web services), shown as Production 1 and 2, are placed in front of the honeywall. The honeywall uses a combination of Snort [10] and Snort-inline to log and, where necessary, scrub incoming and outgoing packets. A detailed communication flow for a honeywall is shown in Figure 6.2.

In most cases, the packets coming into the honeynet through the honeywall are allowed to pass unchallenged to the honeypots. Only packet capture is done on the honeypot

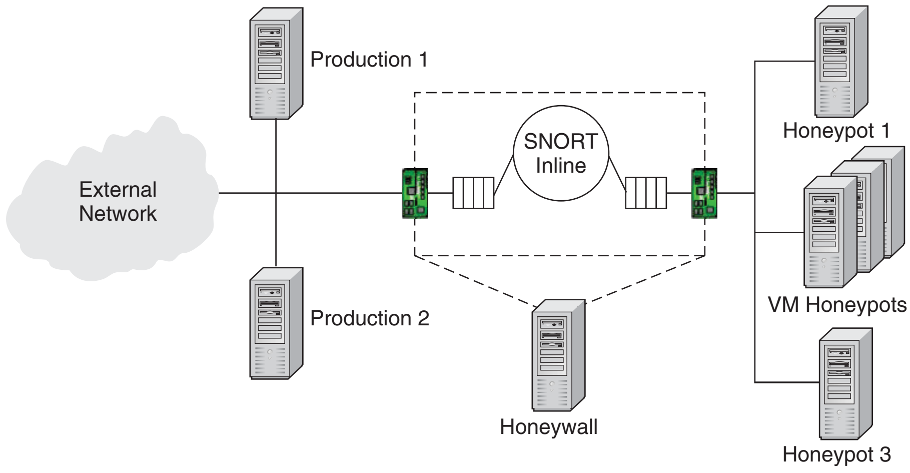


FIGURE 6.1 Honeynet architecture.

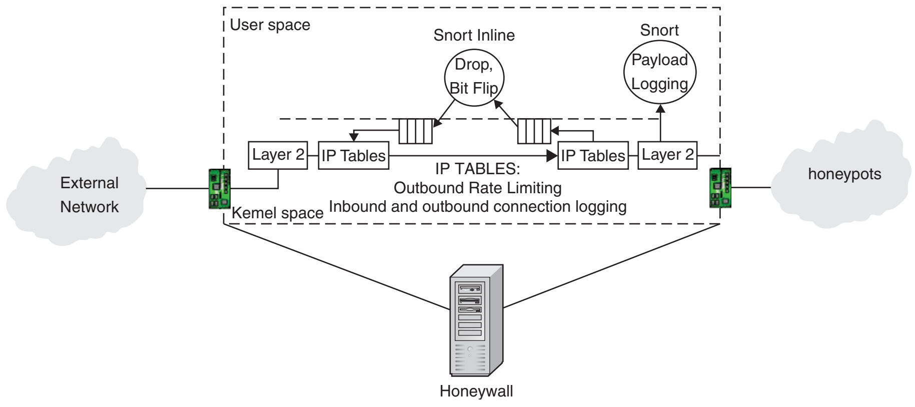


FIGURE 6.2 Honeywall connection detail.

facing NIC. Outbound packets are subject to rate limiting (implemented in IP tables) and analysis by the Snort-inline process. The rate limiting of outbound activity is based on protocol and number of packets and can be specified in the scripts that start IP tables.

The task of data control and protecting the production and internet hosts from attacks originating from your honeypots is handled by the Snort-inline process. This process receives packets from the IP table queuing feature and matches the packets against a modified Snort rule set. If the packet matches a rule, either the packet could be allowed to continue (not likely) or be simply dropped, or Snort-inline could slightly modify the outbound packet's payload thereby rendering the exploit impotent when it reaches its intended target. Using the last approach, it is less likely that the attacker will become discouraged and move on.

Data Capture is accomplished as described above by the Snort process on the internal interface. Additionally a host-based technology called *Sebek* [11], logs all attacker activity on the honeypot and sends it to the honeywall for capture. The original version of Sebek was based on the Adore rootkit. The current version of Sebek replaces the read() function in the system call table in the 2.4 Linux kernel with a read() that is configured to copy all data that it sees into a packet and sends it to a server that collects Sebek packets as well as the normal processing handled by the read() function. Using this method not only captures all activity by the attacker on the honeypot, but also any data that was once encrypted between the honeypot and the attacker's SSH server. Sebek also allows data to be sent covertly out of the honeypot without being observed by the attacker. This is accomplished by bypassing the TCP/IP stack (and the corresponding raw socket interface) and sending the Sebek data packets directly to the network device driver using the transport protocol UDP.

While Sebek provides a tremendous opportunity to record everything an attacker does, it is not invisible. Just as we are using attacker techniques by modifying the kernel functions for the purpose of monitoring, an attacker could determine the build of the OS on the honeynet and compare the various kernel functions known to be used by Sebek (most commonly through an MD5 checksum) and determine that he is on a honeypot. While this might be true and the attacker would probably leave, we have still gained a tremendous amount of information about the attacker—including the fact he is *not* a script kiddie. There are other detection mechanisms for honeypots and Sebek that can be found; the first one published is known as *NoSEBrEak* [12].

6.3.2 Global Distributed Honeynet (GDH)

Until last year, the deployment of honeynets was very geographically constrained. The GDH project is an attempt to develop, deploy, operate, and analyze data based on a worldwide network of honeynets. GDH was developed and led by David Watson of the UK Honeynet Project. The idea is to set up at different network locations an identical honeypot sensor system and store all collected information in a central database. Each GDH node consists of different components:

- virtual honeywall for data capture and data control;
- virtual Nepenthes honeypot for automated malware collection;
- one or more high-interaction honeypots running in a virtual machine.

All honeypot components are executed within virtual machines such that the whole honeynet can be deployed on one physical machine. Besides needing only a limited amount of hardware for running a GDH node, this approach has the additional advantage of the administration and system maintenance of the honeypot being easier. Setting up a new node only requires booting from a GDH CD-ROM, which automatically sets up the base platform on the system.

The whole honeynet is structured in a star-based network model with many GDH nodes and one central GDH data server. Each day all collected information, such as raw network data, IDS log files, and binary samples of malware, is transferred from each sensor node to the central data server. This enables a central correlation and analysis mechanism and all reports about malicious activities are generated based on this data.

The full GDH sensor system was operated for a period of 3 months between March and May 2007 with 11 GDH nodes. This GDH phase one was an attempt to test the whole infrastructure in a real-world environment. More than 730 million network packets were captured during this period, resulting in more than 122 GB of raw network data. In total, about 300,000 unique source IP addresses were observed at all honeypot sensors and this shows that such a global network of honeypots can collect quite a lot of information about network-based attacks. Furthermore, about 670,000 brute force attacks against SSH servers could be observed and 1680 malware samples were collected with the virtual Nepenthes honeypots.

Besides these automated attacks, several attacks by blackhats were also observed and lot of information about the typical tools, tactics, and motives of the attackers were collected. The major incidents observed include, for example:

– Polish cyber crime botnet used to attack other system with distributed denial-of-service (DDoS) attacks
– Brazilian group of blackhats who attacked web applications
– Romanian group of blackhats who are specialized in SSH brute force compromises.

The GDH phase one demonstrated the ability to successfully deploy and operate a globally distributed, standardized honeynet with identical sensor nodes at each location. The whole operation has also demonstrated that large scale distributed data collection and analysis are complex and time-consuming efforts, but the collected data compensates the effort since a lot of data about blackhat attacks were collected. In the future, the basic design of a GDH will be refined and adopted to new threats observed in the wild. The goal is to continuously operate a global network of both low- and high-interaction distributed honeynets based on current honeynet technology. Such a system can then be used to study and analyze current events and threats against systems connected to the Internet. The whole infrastructure can also be used as a test bed to study current honeynet technology in a real-world environment.

6.3.3 Honeyclients

One of the new research areas for Honeynet technologies is the honeyclient. Honeyclients, which are sometimes also called *client honeypots*, are the opposite of server honeypots. The main idea is to simulate the behavior of humans and then closely observe whether or not the honeypot is attacked. For example, a honeyclient can actively surf websites to search for malicious web servers that exploit the visitor's web browser and thus gather information of how attackers exploit clients. Another example are honeyclients that automatically process e-mails by clicking on attachments or following links from the e-mail and then closely observing if the honeypot is attacked. The current focus in the area of honeyclients is mostly based on the analysis of web client exploitation, since this attack vector is often used by blackhats in order to compromise a client.

Honeyclients also have another advantage: honeyclients initialize every analysis and thus control the maximum number of possible attacks. It is possible that a server honeypot may not be attacked for weeks or even months, or it is also possible that the honeypot is attacked occasionally by many attackers at the same time. In general, it is not possible to predict how frequently attacks will occur on a honeypot, and therefore the analyses get more complicated.

Honeyclients can also be classified as high-interaction or low-interaction honeyclients. High-interaction honeyclients are usually real, automated web browsers on real operating systems which interact with websites like real humans would do. They log as much data as possible during the attack and allow a fixed time period for an attack. Since they provide detailed information about the attack, high-interaction honeyclients are in general rather slow and not able to scan broad parts of the web. Low-interaction honeyclients, on the other hand, are often emulated web browsers, usually webcrawlers, which have no or only limited abilities for attackers to interact with. Low-interaction honeyclients often make use of static signature or heuristics-based malware and attack detection tools and thus lack the detection of zero-day exploits and unimplemented attack types.

For all available honeyclients, a common general architecture or process chain, which consists of three serial steps depending on each other, can be observed. At first, a queue is filled with objects to analyze. Second, a honeyclient draws targets from the queued objects. Third, the collected information about the object is analyzed regarding their maliciousness. Several examples of different kinds of honeyclients and first results obtained with these tools are provided in the following paragraphs.

The HoneyMonkey project is a web browser (Internet Explorer) based high-interaction honeyclient developed at Microsoft Research in 2005 [13]. The HoneyMonkey architecture consists of a chain of virtual machines with different flavors of the Windows operating system in various patch levels. Starting with a fully unpatched system, the Monkey Controller initiates a so-called “monkey” program that browses previously scheduled websites. The monkey opens the website and waits for a predefined time. After the time-out, the virtual machine is checked for signs of a successful intrusion. If an attack is detected, the website is revisited with the next machine having a higher patch-level in the pipeline. During their research in May/June 2005, the researchers found that unpatched Windows XP SP1 systems could be exploited by 752 different URLs and a fully patched Windows XP SP2 had no exploits. They also claim to have detected a zero-day exploit in July 2005.

Capture [14] is a high-interaction honeyclient developed at the Victoria University of Wellington, New Zealand. Capture has two functional areas in its design, namely, a Capture client and a Capture server. The clients are hosting the actual high-interaction honeypotclient on a virtual machine, whereas the server coordinates and controls the clients. Capture concentrates on three aspects of high-interaction honeyclients:

- Capture is designed to be fast. State changes on the clients are triggering malicious actions in real time to the server.
- Capture is designed to be scalable. The central Capture server can control numerous clients across a network
- Capture supports different clients. The current version supports the three web browsers Firefox, Opera and Internet Explorer.

The server takes a URL as input and distributes it to one of the honeyclients in a round-robin fashion while controlling the clients in means of starting and stopping them. The clients report back any state changes: each client monitors its own state for changes on the file system, registry, and processes while browsing a website. An exclusion list for known, benign system changes are used to identify a nonmalicious state change; any other operation triggers a malicious classification of the web server and sends this information to the Capture server. Since the state of the client has been changed, the client resets its state to a clean state and retrieves new instructions from the server. If no state change was detected, the client requests new instructions from the server and continues its browsing without resetting. An interesting feature in development is to support nonbrowser clients, such as multimedia players and Microsoft Office applications.

6.3.4 Low-Interaction Malware Collectors

Several low-interaction honeypots were developed in the last few years to automatically capture binary copies of autonomous spreading malware. The basic idea of each of these honeypots is to emulate an actual vulnerability. If a honeypot thus emulates a vulnerability and behaves like a vulnerable system, the malware will be tricked into thinking that the machine can actually be compromised and attacks the honeypot. By analyzing the received attack data and sending back packets that emulate an actual exploitation phase, the honeypot can collect enough information to acquire a binary copy of the malware.

One of the well-known honeypots from this area is Nepenthes [15]. Nepenthes is a low-interaction honeypot which aims at capturing malicious software artifacts that spread in an automated manner, like for example, worms or bots. The tool is based upon a very flexible and modularized design. The core—the actual daemon—handles the network interface and coordinates the actions of the other modules. The actual work is carried out by several modules, which register themselves in the Nepenthes core, and currently there are several different types of modules:

– Vulnerability modules emulate the vulnerable parts of network services. In total, this honeypot emulates more than 20 different vulnerabilities, corresponding to commonly exploited network services.
– Shellcode parsing modules analyze the payload received by one of the vulnerability modules. These modules analyze the received shellcode, an assembly language program, and extract information about the propagating malware from it.
– Fetch modules use the information extracted by the shellcode parsing modules to download the malware from a remote location.
– Submission modules take care of the downloaded malware, for example, by saving the binary to a hard disc, storing it in a database, or sending it to antivirus vendors.
– Logging modules log information about the emulation process and help in getting an overview of patterns in the collected data.

With the help of Nepenthes, it is possible to collect a large number of malware binaries, which spread autonomously. For example, during an 8-week measurement study between December 2006 and January 2007 a Nepenthes sensor running on about 16,000 IP addresses in parallel was able to collect more than 2500 unique malware samples [16]. The uniqueness is determined by the MD5 hash of each binary: two binaries that have the same MD5 hash are considered to be the same malware. Other operators of low-interaction malware collectors report a similar amount of malware collected with these honeypots [17].

REFERENCES

1. The Honeynet Project, http://www.honeynet.org last accessed on 20 January 2008.
2. Honeyd, http://www.honeyd.org last accessed on 20 January 2008.
3. Specter, http://www.specter.com last accessed on 20 January 2008.
4. KFsensor, http://www.keyfocus.net/kfsensor/ last accessed on 20 January 2008.
5. Honeynet guideline, http://www.honeynet.org/alliance/requirements.html last accessed on 20 January 2008.

6. http://www.securityfocus.com/news/4004, last accessed on 20 January 2008.
7. http://honeynet.org/book/Chp8.pdf, last accessed 20 January 2008.
8. Spitzner, L. *Honeypots: Are They Illegal?* http://www.securityfocus.com/infocus/1703.
9. Spitzner, L. *The Value of Honeypots, Part Two: Honeypot Solutions and Legal Issues,* http://www.securityfocus.com/infocus/1498.
10. SNORT, last accessed on 20 Jan 2008 at www.snort.org.
11. Balas, E. *Know Your Enemy: Sebek*, http://www.honeynet.org.
12. Dornseif, M., Holz, T., and Klein, C. (2004). NoSEBrEaK—attacking honeynets. *IEEE Information Assurance Workshop*. West Point, NY, June 11, 2004.
13. Wang, Y., Beck, D., Jiang, X., Roussev, R., Verbowski, C., Chen, S., and King, S. (2006). Automated web patrol with strider honeymonkeys: Finding web sites that exploit browser vulnerabilities. *Proceedings of 13th Network and Distributed System Security Symposium (NDSS'06).*
14. Seifert, C. *Capture High-Interaction Client Honeypot*, Internet:https://www.client-honeynet.org /capture.html.
15. Baecher, P., Koetter, M., Holz, T., Dornseif, M., and Freiling, F. (2006). The Nepenthes platform: an efficient approach to collect malware. In Zamboni, D., and Kruegel, C. Eds. *RAID 2006. LNCS*, Springer, Heidelberg, Vol. 4219, pp. 165–184.
16. Goebel, J., Holz, T., and Willems, C. (2007). Measurement and analysis of autonomous spreading malware in a university environment. *Proceedings of DIMVA*. pp. 109–128.
17. Watson, D. (2007). *Deploying and Operating a Global Distributed Honeynet*, PacSEC 2007, 29-30 November 2007, http://www.honeynet.org/speaking/PacSec07_David_Watson_Global_Distributed_Honeynet.pdf.

7

SECURITY OF WEB APPLICATION AND SERVICES AND SERVICE-ORIENTED ARCHITECTURES

Marc Goodner
Microsoft, Redmond, Washington

7.1 INTRODUCTION

Service Oriented Architecture, or SOA, has had many characteristics ascribed to it. It has been hailed as a way for organizations to integrate heterogeneous systems, provide for increased reuse of software assets, provide multichannel access to common components, and as the way to build distributed systems, all while reducing complexity and increasing interoperability. SOA can mean all of these things depending upon how its principles are applied in a given project. Seen as a natural outgrowth of previous software design paradigms, the key distinguishing characteristic it has is facilitating interoperability between distributed software components. As an architectural style for designing software SOA does not mandate one implementation choice over another. While there are many realizations of these principles, Web services is the most pervasive implementation choice for SOA. This is because Web services are platform-agnostic protocols; thus they greatly facilitate interoperability between different implementations. This is critically important in enabling software systems that need to connect new and existing applications in and across environments as heterogeneous as a typical corporation, government agency, or even a home. The security requirements for these applications will vary greatly depending on the type of information they use, the policies of the application users, and the threats to the messages being exchanged. The techniques available to mitigate threats to the security of an SOA-based software application will be the same regardless of whether the threats are from non-state actors, corporate espionage agents or common criminals. From the perspective of homeland security, many of the sources of each of these threats are increasingly blurred due to an emerging common underground economy for those perpetrating the attacks. In that respect it is an imperative that security be a foremost concern in all distributed software implementations. This chapter will describe the security challenges in SOA, relate them to Web service protocols that address these challenges, and introduce some areas for further consideration.

7.2 SOA SECURITY CHALLENGES

In order to properly discuss the security challenges of SOA let us look a little closer at what SOA is. Newcomer and Lomow provide a succinct description: “A service-oriented architecture is a style of design that guides all aspects of creating and using business services throughout their life cycle (from conception to retirement). An SOA is also a way to define and provision an IT infrastructure to allow different applications to exchange data and participate in business processes, regardless of the operating systems or programming languages underlying those applications.” [1] The following properties of SOA can be derived from the above description:

- *Explicit boundaries.* When exposing functionality as a service it is important that the choice of what to expose is carefully considered. Only what is needed to exchange information should be exposed. What is critical here is that while services are explicitly exposed, nothing should ever be exposed as a service by default;
- *Service autonomy.* The application that supports a service should be independent of the service definition itself. This allows for revisions of the application while keeping the interface used by client applications stable. Services may also need to change their location over time;
- *Contracts.* There needs to be machine-processable formats to produce the necessary message structures for the service. Classes should never be shared to enable invocation of services;
- *Compatibility based on policies.* There also needs to be machine-processable policies in place at a service so that client applications can determine if they are compatible;
- *Interoperability.* Services need to be exposed in a way that facilitates use by the broadest base of applications.

What you can see in SOA are services exposed in a highly interoperable and machine-processable fashion. As SOA has gained the most traction within larger enterprises, it is important to step back and recognize that the environments in which these services are commonly deployed are often secured via tight perimeter controls. To properly secure these services a more granular approach is needed. Remember that a key promise of SOA is re-use. Services may not only find re-use within more areas of the organization that created them, but potentially externally as well as they begin to offer value to an organization’s partners. The remainder of this section will focus on specific security concerns and concepts that need to be considered in the design of an SOA-based system. The next section will map these security concerns and concepts to specific Web services protocols.

It is important to state that this section is not intended as a substitute for proper risk analysis of your software design. The concerns and concepts below should be used in a formal risk assessment, not as a checklist. The following advice is based on applied experience in this area: “Before you take any technical steps to implement a specific security structure, you should capture the security requirements for all relevant pieces of software during a risk analysis. Based on this analysis, you build the software architecture to incorporate the defined security requirements” [2].

Choosing explicitly what to expose as a service is an important decision. If something does not need to be exposed as a service, it should not be. Services that are exposed should present the minimum amount of surface area for a malicious client to probe. At the point that a choice to expose a service is made, the decisions about how to protect the service endpoint and the messages it exchanges should also be made. The value of the information contained within the messages exchanged will guide some of the choices during a full risk assessment of what mechanisms to employ. From the client's perspective, the analysis of the value of the information being exchanged is just as important. Clients should not interact with services that do not meet their own security requirements.

SOA-based systems are vulnerable to threats targeted at the messages they exchange over networks. These threats revolve around classic man-in-the-middle scenarios where messages are intercepted for inspection or alteration. In particular, substitution of messages in whole or in part must be guarded against. For instance, an attacker should not be able to change the message content while leaving security credentials intact; such a condition must be detectable. Similarly messages should not be allowed to be replayed, as that can be used to introduce application errors or cause other problems.

There are many other security challenges that need to be considered in designing an SOA-based system. We will attempt to answer many of these in the remaining space but this list should not be considered exhaustive.

- *Access.* How is access to a service granted?
- *Confidentiality.* How can the contents of a message be kept secret?
- *Integrity.* How can a message be protected from alteration en route to a recipient?
- *Non-repudiation.* How can it be proven that a message was received?
- *Trust.* How can parties in a given interaction rely upon the security credentials of each other?
- *Sessions.* How can a set of messages be exchanged securely?
- *Description.* How can the security requirements of a service be expressed?
- *Federation.* How can resources in one security domain be provided to clients whose identities are managed in another?
- *Identity.* How are the participants in an exchange known?
- *Privacy.* How is inadvertent information disclosure prevented?

There are other questions that cannot be answered within the scope of this paper as the answers are specific to the technologies used in an implementation, or to local requirements.

- *Performance.* Is there a resource penalty for securing services?
- *Audit.* How can messages be traced in a distributed environment?
- *Regulatory compliance.* Are there mandated security requirements from government agencies?

An SOA-based infrastructure may have many different types of client applications present, from large back-end systems and middleware to rich desktop applications and lightweight web clients. This chapter has not discussed the security issues involved in developing the client applications themselves.

7.3 SECURE WEB SERVICES FOR SOA

The use of Web services is one of the most common implementation strategies for SOA. Web services are widely supported and provide a high degree of interoperability. What helps to make Web services interoperable is that they are based on XML [3] and a set of standards and specifications that have been developed by a diverse group of software vendors and users, and validated through implementation.

When a service is exposed as a Web service, it is typically exposed with a contract consisting of Web Services Description Language (WSDL) [4] and XML Schema [5]. The WSDL describes the message exchange patterns supported by the service. The XML Schema describes the message formats themselves, especially the application payload. Additional semantics around usage of the service, characteristics such as security and reliability, are expressed via WS-Policy [6]. This allows a client to determine if it is compatible with a given service or not by comparing the service's capabilities to its own. A Web service must have a service location, or address, where it can be accessed. The address is expressed in terms of WS-Addressing [7]. The messages themselves are based on SOAP (Simple Object Access Protocol) [8]. SOAP messages are broken into two basic wrapper elements, the header and the body, both of which are contained in a common "envelope" element. The SOAP header contains infrastructure information; the SOAP body contains the application message payload.

Before going on, it is worth noting that most other strategies for implementing SOA are specific to a particular vendor's software platform. As such, they each must have their own unique strategies for meeting the security challenges with the SOA outlined above. This situation is similar in a standards-based environment of Web services as well which has been described by Thomas Erl: "However, the SOAP messaging communications framework upon which contemporary SOA is built, emphasizes particular aspects of security that need to be accommodated by a security framework designed specifically for Web services." [9] Here, we look at the specifics without needing to consider a specific vendor's platform as we have publicly available specifications and standards to refer to.

Broadly speaking, there are two primary ways to secure Web services: transport, or message-based security.

7.3.1 Transport-Based Security

The most common way to secure Web service messages at the transport layer is SSL (secure sockets layer) or TLS (transport layer security) [10]. It can provide for both confidentiality and integrity, and it has an in-built mechanism for exchanging credentials. It is also widely supported and well-understood. Its use can even be described using WS-SecurityPolicy [11].

SSL/TLS protects communications from one point to the next point. This is satisfactory protection in many cases but it is also a limitation that must be considered when choosing to use it. At any time that communications need to be routed through an intermediary, the SSL/TLS session ends and the message will be in the clear. The lack of the client's visibility to this occurring makes SSL/TLS unacceptable in many cases.

While many think of Web services as being used exclusively over HTTP, they are not bound to that choice alone; they may be used over TCP (Transmission Control Protocol), SMTP (Simple Mail Transfer Protocol), UDP (User Datagram Protocol) etc. SSL/TLS can limit these choices of your underlying transport layer as some, like UDP, cannot support it at all, while others are not broadly supported. SSL/TLS is also limiting in the types of security credentials that can be used in establishing the secure channel. Both of these factors will be overly constraining in many instances.

SSL/TLS is suitable for short-lived sessions, but it does not provide the same capability for long-running protected sessions that something like WS-SecureConversation [12] does. While SSL/TLS does provide the capability for resumption, where the same SSL/TLS session is re-established after it is interrupted, it is not commonly implemented or used. In practice SSL/TLS is best used for short exchanges where communications are not expected to be interrupted or where it is acceptable for them to be resumed over a new protected channel. Note that in many instances it is not acceptable to resume communications from an invalidated secure channel, over a new one.

SSL/TLS should not be ruled out as a way to protect Web services but it is important to keep its constraints in mind. It is often best used in conjunction with the message-based security mechanisms described below.

7.3.2 Message-Based Security

Web Services Security (WSS) [13] enables the exchange of trusted messages. It defines an information structure for use in the SOAP header that provides the capability for expressing security information about a message. It leverages the capabilities of the SOAP header to facilitate securing parts of messages to specific roles in a message exchange, thus enabling end-to-end message security even in the presence of intermediaries.

A key part of WSS is its support for attaching and referencing security tokens. WSS was designed to be as flexible as possible in its support for different token types. It defines a Username Token and extensible binary and XML token types. These extensible types have been further refined by WSS token profiles that include X.509, SAML, Kerberos and others.

WSS builds on the work of XML-Signature [14] to address threats related to message alteration and to facilitate nonrepudiation. WSS uses signatures as a means to verify a message's integrity, that it was not altered in transit, and as a way to validate the claims of security tokens associated with the message. WSS also allows for the inclusion of multiple signatures and signature formats. This is important in distributed applications where different parts of a message may need to be signed by different parties involved in the processing of the information.

Message confidentiality is provided in WSS by leveraging XML Encryption [15]. There are facilities provided for protecting individual SOAP header or body elements or sub-elements.

WSS also introduces a security time-stamp element for use in protecting against message replay threats.

In order to ensure broad interoperability of WSS, the WS-I Basic Security Profile (BSP) [16] was created. The BSP constrains many of the options present within WSS, its security token profiles, as well as the carrying-forward requirements that are important for interoperability.

While WSS describes how to use different security token types, it does not describe how to get them. WS-Trust [17] describes a Web service-based interaction for the request and issuance of security tokens from a Security Token Service (STS). This takes the form of a Request Security Token (RST) request and a corresponding Request Security Token Response (RSTR). The tokens requested and issued are also capable of being scoped to a specific purpose. It is recommended that this scope be as constrained as possible to prevent possible abuse of issued tokens.

There is also the capability in WS-Trust for more complex negotiations for the establishment of the requested security tokens. This is accomplished by adding Signature Challenge and Response legs in between the initial RST and eventual RSTR. In addition to these capabilities, WS-Trust also provides capabilities for the validation, renewal, and cancellation of security tokens that have been issued.

In order to address the need to authenticate a series of messages, WS-SecureConversation [12] was defined. WS-SecureConversation builds upon WS-Trust to define a protocol that allows for the establishment of a security session including the establishment of efficient keys and key material. It defines the Security Context Token (SCT) to be used over the lifetime of messages that are part of an exchange. The SCT is referenced like any other token using the capabilities of WSS. It is important to note that an SCT need not contain any identity-related information. An SCT is often used just for a single exchange and may be discarded when it is completed.

The security requirements of a Web service are expressed using policy assertions that are defined by WS-SecurityPolicy (SP) [11]. There are assertions that allow the expression of a service's requirements of features provided by WSS, WS-Trust, and WS-SecureConversation. These assertions provide the necessary information for a client to determine its compatibility with a service and configure itself to produce messages that meet the service's security requirements. The assertions provide a great deal of flexibility for expressing the necessary token types, cryptographic algorithms, and transport requirements of the service.

7.4 WEB SERVICE SECURITY DIRECTIONS

7.4.1 Federation

Looking to more advanced security requirements, one that stands out is the need to enable different security realms to work together, where a realm is seen as a single unit of administration or trust. This is known as federation, where resources managed in one realm are made available to security principals whose identity is managed in another. The WS-Federation [18] specification addresses this topic by building upon WS-Trust.

WS-Federation provides capabilities for the sharing of identity, authentication, authorization, and privacy data when appropriate. It allows for the remote access of services without requiring the service provider to maintain local user identities. It provides optional pseudonym services that allow for the protection of identity information by providing alternative representations of it. A common claims dialect is also defined that allows for the expression of basic claims in WS-Trust messages. WS-Policy assertions are defined to allow a service to advertise its support for the features defined by WS-Federation.

WS-Federation defines bindings of WS-Trust for use with web-browser clients. This enables the use of web browsers in interactions in which they cannot directly make Web service requests on their own.

7.4.2 Identity

Another evolving area in WSS is identity. The internet has no identity layer. This is becoming increasingly problematic as mechanisms for managing who is connecting to what on the internet are increasingly under attack, especially the most common form of user name and password. Kim Cameron's Laws of Identity [19] were arrived at through a dialog of many experts in digital identity and serve as the principles for establishing an Identity Metasystem for the internet to address this problem. The core idea is to put the user back at the center of managing their identity on-line. The realization of these concepts has been through the use of Web service protocols discussed above, especially WS-Trust.

The Identity Metasystem has three essential roles. Note, however, that participants may fulfill more than one of the following roles:

- *Identity providers.* the issuers of digital identities. Examples of potential identity providers are credit card issuers, banks or government agencies;
- *Relying parties.* parties that require a digital identity for use. On-line merchants are a fine example of a relying party that would require a digital identity issued by a credit card company;
- *Subjects.* the individual or other entity that claims are made about, for example, an end user or company.

The Identity Metasystem is an open and nonproprietary concept. The concepts that it consists of, described above, are realized using web service protocols that are not restricted to a specific platform. Specifically an identity provider is an STS as described in WS-Trust. Relying parties describe their own requirements using WS-SecurityPolicy and make them available through WS-MetadataExchange [20].

Users are always aware of interactions that are made on their behalf between parties, in the Identity Metasystem. This awareness is made possible through a consistent experience provided by an identity selector. The identity selector presents a user with the credentials that satisfy the requirements of the relying party. The relying party is clearly revealed as are the identity providers to the user. The user has visibility to whom is making the request (the relying party) for their digital identity. The user also has visibility to who issued the digital identity (the identity provider) they are going to release to the requestor. This identity selector is invoked by an application typically, but not necessarily, a web browser, and performs the necessary negotiations between identity providers and relying parties. The user has complete visibility to the claims being requested and released as part of this interaction.

The ideas of the Identity Metasystem have been reflected in implementations of both commercial and open-source software such as Microsoft's CardSpace, Mozilla Firefox, and Higgins which has backing from IBM and Novell.

7.5 SUMMARY

SOA will remain a popular approach to building distributed software applications for many years. Web services are, and will remain, a common SOA implementation strategy on many platforms. The above chapter has described many of the common threats present to SOA applications and how those threats are addressed via Web service security protocols. The threats to SOA applications are the same irrespective of the attacker; for example, a criminal, espionage agent, or common vandal. While transport security will remain an important option for securing Web service messages, message-based security provides more flexibility, capability and generally better protection when properly deployed. This is an evolving area, particularly in the areas of federated trust and identity. The reader is encouraged to follow the references from this chapter to stay up-to-date.

REFERENCES

1. Newcomer, E., and Lomow, G. *Understanding SOA with Web Services*. Addison Wesley Professional.
2. Krafzig, D., Banke, K., and Slama, D. *Enterprise SOA: Service-oriented Architecture Best Practices*. Prentice Hall.
3. *W3C Recommendation. Extensible Markup Language (XML) 1.0 (Second Edition).* Available at http://www.w3.org/TR/2000/REC-xml-20001006.
4. *W3C Note. Web Services Description Language (WSDL 1.1).* Available at http://www.w3.org/TR/2001/NOTE-wsdl-20010315.
5. *W3C Recommendation. XML Schema Part 1: Structures Second Edition*. Available at http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/.
6. *W3C Member Submission. Web Services Policy 1.2—Framework*. Available at http://www.w3.org/Submission/2006/SUBM-WS-Policy-20060425/.
7. *W3C Recommendation. Web Services Addressing (WS-Addressing).* Available at http://www.w3.org/TR/2006/REC-ws-addr-core-20060509.
8. *W3C Note. SOAP: Simple Object Access Protocol 1.1*. Available at http://www.w3.org/TR/2000/NOTE-SOAP-20000508/.
9. Erl, T. *Service-oriented Architecture: Concepts, Technology, and Design*. Prentice Hall.
10. *IETF Standard. The TLS Protocol*. Available at http://www.ietf.org/rfc/rfc2246.txt.
11. *OASIS Committee Draft. WS-SecurityPolicy 1.2*. Available at http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200512.
12. *OASIS Committee Specification. WS-SecureConversation 1.3*. Available at http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512.
13. *OASIS Standard. OASIS Web Services Security: SOAP Message Security 1.0 (WS-Security 2004).* Available at http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf.
14. *W3C Recommendation. XML-Signature Syntax and Processing.* Available at http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/.
15. *W3C Recommendation. XML Encryption Syntax and Processing*. Available at http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/.
16. *WS-I Working Group Draft. "Basic Security Profile Version 1.0*. Available at http://www.ws-i.org/Profiles/BasicSecurityProfile-1.0.html.
17. *OASIS Committee Specification. WS-Trust 1.3*. Available at http://docs.oasis-open.org/ws-sx/ws-trust/200512.
18. Kaler, C., Nadalin, T., et al. *WS-Federation*. Available at http://schemas.xmlsoap.org/ws/2006/12/federation/.
19. Cameron, K. *The Laws of Identity*. Available at http://www.identityblog.com/?page_id=354.
20. Curbera, F., Parastatidis, S., Schlimmer, J., et al. *WS-MetadataExchange*. Available at http://schemas.xmlsoap.org/ws/2004/09/mex/.

8

INDUSTRIAL PROCESS CONTROL SYSTEM SECURITY

Ivan Susanto, Rich Jackson Jr., and Donald L. Paul
Chevron Corporation, San Ramon, California

8.1 INTRODUCTION

Process control systems or industrial automation and control systems (IACS) used in the O&G Industry are vulnerable to new threats with potentially serious consequences. Vulnerabilities come from many sources, including, but not limited to increasing access to IACS, increased digital intensity in the form of digital oil fields, smart sensors generating ever increasing amounts of data, real-time optimization, reservoir modeling, and global value chains that are highly leveraged on information and connectivity. In order to address these vulnerabilities, a public–private partnership called *Project LOGIIC* was formed to create and execute projects that address critical O&G cyber security Research and Development (R&D) needs, and produce solutions upon their completion, which can be deployed in the industry. ISA Security Compliance Institute (ISCI) also combines the talents of industry leaders from a number of major control system users and manufacturers to create a collaborative industry certification-based program.

8.2 BACKGROUND

Process control systems or IACS are used by O&G companies at their offshore platforms, pipelines, refineries, plants, and other industrial assets. IACS are collections of personnel, hardware, and software that can affect or influence the safe, secure, and reliable operation of an industrial process. The systems include, but are not limited to [1]:

1. Industrial control systems including distributed control systems (DCSs), programmable logic controllers (PLCs), remote terminal units (RTUs), intelligent electronic devices, supervisory control and data acquisition (SCADA), networked electronic sensing and control, and monitoring and diagnostic systems. (In this context, process control systems include basic process control systems and

Cyber Security, Edited by John G. Voeller

safety-instrumented system [SIS] functions, whether they are physically separate or integrated.)

2. Associated information systems such as advanced or multivariable control, on-line optimizers, dedicated equipment monitors, graphical interfaces, process historians, manufacturing execution systems, and plant information management systems.
3. Associated internal, human, network, or machine interfaces used to provide control, safety, and manufacturing operations functionality to continuous, batch, discrete, and other processes.

There is an increased reliance on IACS for safe, secure, and reliable operations of facilities. Historically, it was thought that IACS were secure because they relied on proprietary networks and hardware and were considered immune to network attacks that plague corporate information systems. This is no longer true.

While no solution can offer a complete solution, defense-in-depth methods can help detect and delay or even prevent breaches. Without the right information at the right time, there cannot be an appropriate response to threats.

8.2.1 The Problem

IACS used in the O&G industry are potentially vulnerable to new threats. Standardization and integration with corporate business systems have increased the potential exposure to these systems. IACS data were traditionally used in a contained environment only by those in that environment. Now, government agencies, business partners, suppliers, and others want access to the IACS data, causing more time to be spent on filling requests and less attention to monitoring for potential breaches.

Most importantly, this integration requires network connections that provide access and raise risks and threats.

8.2.2 New Threats

Most people will click on interesting links, especially when they are sent by someone known to them. Employees and vendors often use thumb drives, CDs, or DVDs to support IACS, and these portable media are readily inserted into an IACS environment without scanning for viruses first. It takes real effort to stop and think about risk; whether it is real or a cleverly disguised threat.

Removable drives and e-mail links are just two ways that these threats can be introduced. Threats to energy industry systems have expanded beyond the typical physical attacks of the past. When these physical attacks are combined with cyber attacks on the control systems, the results could be much more damaging. The changing nature of control systems means that attackers ranging from hackers through organized cyber criminals and sophisticated insiders can have physical effects through cyber means.

The new networked control systems and commercial off the shelf (COTS) technology are vulnerable to attacks that are not specifically aimed at them. For example, the Port of Houston had to shut down operation of its control system in September, 2001. This system controlled ship movement, docking, mooring, loading, and unloading. They were affected by a “denial of service” attack, which was not aimed at them but which affected them just the same. The attack was the result of a “botnet” or robot network of computers, typical to those used by organized crime.

There are other known security incidents happening in the industries as well, such as the Maroochy Shire Sewage Spill, an IP Address change shut down chemical plant, and a slammer-infected laptop shutting down a DCS.

These are the factors that contribute to risk in the IACS environment [2]:

- Adoption of open standardized technologies susceptible to known vulnerabilities;
- Connectivity of Control Systems with other networks, including the Corporate network;
- Insecure remote connections;
- Widespread availability of technical information about control systems.

On the basis of a recent industry trend, both security risks from insiders and outsiders still continue to be of most concern, with hackers gaining a greater understanding of IACS.

8.2.3 The Solution

LOGIIC-1 Team [3] within a critical infrastructure environment, addressing security risk is a shared problem that can only be addressed and solved collaboratively. In the LOGIIC partnership, the following were the goals:

- Demonstrating a forward-looking opportunity to reduce vulnerabilities of O&G process control environments.
- Creating a working model to leverage the collective resources of the O&G industry, government agencies, and national laboratories for future cyber security projects.
- Leveraging existing SCADA cyber security knowledge and tools from the O&G industry, government, and vendors to
  - align with existing and future activities being performed in the SCADA industry, National Laboratory Testbeds, and O&G industry;
  - assist the National Laboratory Testbeds with the research and development of new solutions focused on the O&G industry, which will address existing security weaknesses (evolutionary) and breakthrough security solutions (revolutionary).

ISA Certification is one resource that promises to provide asset owners [4] a well-designed and managed product security certification process, leading to improved process reliability and safety. Certification responds to a common need for a shared security vision to be executed by suppliers, asset owners, and consultants. It also will promote better field-tested standards that are clearly followed by industry.

8.3 SCIENTIFIC STUDY

In the LOGIIC-1 Project (Event Correlation), a defense-in-depth solution can collect all raw events (data) from IACS to business/corporate network, correlate it and analyze abnormal events to provide information to decision makers enabling them to validate threats and take appropriate action.

Monitoring is the key to building better defenses, especially for new unknown threats and vulnerabilities, but implementing even a simple perimeter intrusion detection system (IDS) can produce such volumes of data that it can become overwhelming. Too much data from an IDS would then become a hindrance rather than a help.

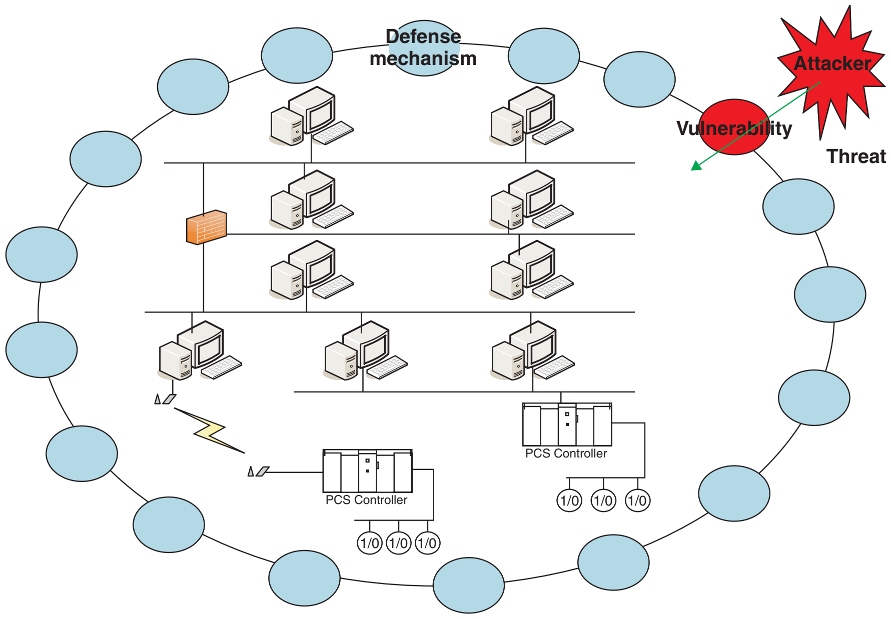


FIGURE 8.1 Threats and vulnerabilities.

And as illustrated in Figure 8.1, for systems without layered security architecture, it only takes a single vulnerability for an attacker to bring a system down. Even for systems with layered, defense-in-depth approaches to security, an attacker can still cause damage. We need to know how many “open doors” we have left for attackers.

One answer to the problem is to have a central correlation engine that is fed with inputs from IACS to the business/corporate network.

8.3.1 Correlation Benefits

While there are many sources of security data available, the amount of data is substantial and often in incompatible formats. Both of these factors hinder transforming the raw data into useful information [5]. A best-in-class correlation system can help by gathering data from all sources and analyzing it for trends.

Some benefits of implementing such a correlation system are

- Event and log aggregation;
- Normalizing of events into a standard format;
- Categorizing and prioritizing events;
- Filtering extraneous events;
- Grouping similar events;
- Discovering relationships between events;
- Health monitoring from many small data points;
- Building big picture of the IACS health.

Awareness of a problem is the first step to implementing preventive or corrective measures.

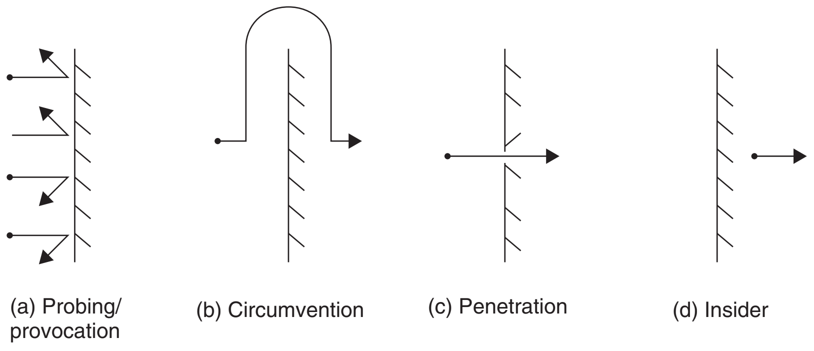


FIGURE 8.2 IDS event triggered responses [6, p. 7].

8.3.2 Detection

There are four types of security events that should be detected.

In Figure 8.2, we let the depicted barrier abstractly to represent the perimeter defense. The four categories of events that we want to detect apply to the physical world as well as to computer systems and networks.

The probing/provocation category represents the case when attackers attempt to penetrate the defense but are unsuccessful. Examples in the cyber realm include port scanning and repeated authentication or authorization failures, such as password-guessing or file system browsing. Even though the perimeter defense works as intended, we still want to detect this kind of event because we are under attack and the attackers could eventually succeed.

Circumvention occurs when attackers find a way to reach their goal without confronting the perimeter defense. As an example, a corporation could have a strictly configured firewall protecting its corporate network from the Internet, but a badly configured wireless access point on the corporate network can allow an attacker parked on the street outside to get to the network without even going through a firewall.

Penetration occurs when vulnerability in the perimeter defense allows attackers to get through. An example of penetration is when an attacker with knowledge of software bugs can compromise the system using access that allows through well-configured firewall.

Finally, Insiders are attackers already inside the perimeter. For example, a firewall between the corporate network and the Internet does nothing to stop a disgruntled employee from stealing data from an internal database and hand-carrying it out of the building on a CD-ROM or other portable storage device. It should be noted that an attacker who has used circumvention or penetration to get inside the perimeter could also be considered an insider, from a detection perspective.

8.3.3 Technical Challenges

8.3.3.1 Typical IACS Environment. A test bed model (Fig. 8.3) in LOGIIC-1 project was developed using generic DCS and SCADA system with field devices to describe typical IACS environment. Some trade-offs and assumptions were taken into account in this testing model.

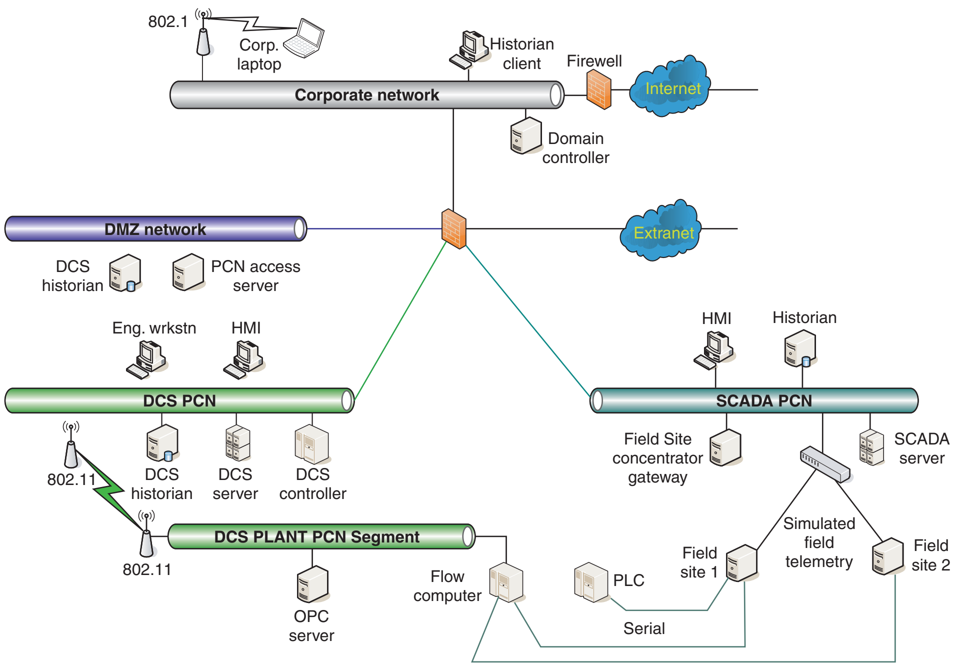


FIGURE 8.3 LOGIIC-1 Baseline O&G lab environment (courtesy of DHS LOGIIC brochure).

8.3.3.2 IACS Abnormal Events. There is a technical challenge in understanding the abnormal events that can be caused by an adversary in a PCSs [3]. IACS are vulnerable to the same kind of attacks experienced in a standard IT environment, but have the added vulnerability of attacks that are unique to IACS.

8.3.3.3 Detecting IACS Abnormal Events. Another challenge is in understanding how to detect the abnormal events that can be caused by an adversary in a PCSs [3]. Standard information technology defenses can detect and defend against the same types of attacks in PCSs.

8.3.4 Implementing Defense and Detection in-Depth

The next technical challenge is to identify the layers that need to be instrumented to achieve a defensive in-depth detection [3]. The following layers were identified:

- Network Boundary
- Host Network Connection
- Host Operating System
- Process Control Application.

The final challenge is to show that IT network devices (e.g. IDSs) can be used with IACS, as well as with their field devices such as flow computers or PLCs. Security alerts from the devices must be able to be correlated to provide the proper intrusion detection in a realistic control system environment.

8.3.5 Test Bed Operating Model

The LOGIIC-1 test bed included four individual networks: a Corporate Network, a DMZ Network, a DCS Network, and a SCADA Network. The test environment includes both a SCADA application typically used to manage pipelines as well as a DCS application used to run refineries. These applications reside on process control networks (PCNs) with other IACS-specific equipment.

The standard IT defenses selected as event sources include the following:

- Network segment firewalls (in reporting, not blocking modes);
- Host firewalls (again, in reporting, not blocking modes);
- Network IDSs;
- Network devices (wired and wireless routers).

Three sources specific to control systems are

- PCS-protocol aware IDSs on the PCNs;
- Alarms from the DCS and SCADA;
- Alarms from flow computers.

A suite of sensors was selected to implement this defense-in-depth strategy. These sensors are triggered by abnormal activity and produce security events that are collected and correlated by an Enterprise Security Management (ESM) application. It is critical to relate security events in the IT network with IACS events to provide situational awareness. This allows IACS operators to identify threats that would previously go unnoticed. These threats can now be mitigated before potentially serious process disruptions occur.

Three sets of correlation rules were developed to enable this awareness:

1. Rules that identify steps of the critical attack scenarios (e.g. moving from network segment to another).
2. Rules that implement common IACS policies. IACS is quite static compared to business/corporate networks, so violation alerts can include rogue systems, IACS configuration changes, and port scans.
3. Rules that apply a data dictionary for IACS-specific security events. This dictionary would map proprietary logged IACS events to standardized security events.

8.4 SUMMARY

In the LOGIIC-1 Project, the team was able to implement ESM application (correlation engine) in generic O&G DCS & SCADA systems within a laboratory environment and integrated them with a simulated business network [5]. As a result, the project

- Successfully developed, implemented, and tested four attack scenarios, which model new threats to IACS brought by standardization and interconnectivity;
- Implemented a PCS security data dictionary;
- Identified, correlated, and alerted the compromises to environment at and across all levels;

- Provided enhanced situational awareness;
- Built an in-depth solution for industry deployment.

IT-type sensors were placed to detect events on the IACS generated information, which was combined with events extracted from the control system applications. Attack pictures were created using events from both sources.

The IT types of sensors provided events generated by their standard IT signature set, as well as events generated by a Modbus signature set to detect PCS-specific attacks. The control system applications were also able to provide unique control system alarm events for correlation.

On the basis of the results, it was predicted that there would be a reduction in workload for a security analyst looking for attacks, since filtering reduced the number of events an analyst would need to examine. One of the attack scenarios used created over 7,000,000 low-level events from the system sensors, which were reduced to about 1000 correlated events and then further prioritized to only 130 high-priority alerts.

The LOGIIC-1 results have now been implemented by several companies in their real-world environment, proving that this LOGIIC collaboration/partnership works very effectively.

8.5 NEXT STEPS

The LOGIIC model was developed to have broad applicability within the O&G industry as well as other IACS-dependent industries and government, and the synergy from such a private–public partnership results in higher quality results, reduced R&D, and lower costs. Addressing IACS cyber security risks within any critical infrastructure environment is a shared problem and needs to be addressed through a collaborative effort. The LOGIIC model has proven to be a vehicle that provide the necessary collaborative results.

In addition to the LOGIIC model, industries can improve PCSs security by supporting other industry collaboration such as the following:

- ISA-99 Committee that establishes standards, recommended practices, technical reports, and related information that will define procedures for implementing electronically secure manufacturing and control systems and security practices and assessing electronic security performance. The Committee's focus is to improve the confidentiality, integrity, and availability of components or systems used for manufacturing or control, and to provide criteria for procuring and implementing secure control systems. Compliance with the Committee's guidance will improve manufacturing and control system electronic security, and will help identify vulnerabilities and address them, thereby reducing the risk of compromising confidential information or causing manufacturing control systems degradation or failure [7].
- ISCI, which is an industry consortium that facilitates an efficient forum of asset owners and suppliers for proposing, reviewing, and approving security conformance requirements for products in the automation controls industry. The resulting requirements form the basis for the *ISASecure*™ compliance designation, enabling suppliers

to develop secure automation control products based on industry consensus security standards (security compliance "out of the box"). The *ISASecure*™ designation creates instant recognition of automation control products and systems that comply with *ISASecure*™ technical specifications. As a result, asset owners are able to efficiently procure and deploy *ISASecure*™ products with well-known security characteristics that are in conformance with industry consensus security standards such as ISA99. [8]

- Other security collaboration/partnerships such as API and NPRA.

8.6 CONCLUSION

The Event Correlation research conducted by the LOGIIC program addresses the need for coordination at many levels if our nation's critical PCSs are going to be secure. At the technology level, security data from many disparate sources must be collected and analyzed as an integrated resource. Otherwise, a potential avalanche of events can result in valuable security information being overlooked or misinterpreted, increasing the probability of a successful attack.

At the same time, coordination at the organizational and national level is also critical. Without it, each company would be forced to proceed on its own, achieving far less in the end. Instead, the synergy generated by the private–public partnership in LOGIIC resulted in a security project with higher quality results, reduced research time, and lower costs. We believe it stands as a model for industry and government cooperation in critical infrastructure security going forward.

ACKNOWLEDGMENTS

We would like to thank:

- Chevron Corporation, for supporting cyber security activities such as LOGIIC, ISCI, and ISA-99 in the O&G industry, and also for assistance in publishing this chapter.
- The members of LOGIIC-1(Correlation Project) for their participation in the project and their significant contributions to the solution (http://www.cyber.st.dhs.gov/logiic.html).
- Ulf Lindqvist, Dale Peterson, Thomas Culling, Eric J. Byres, and Linda Shaltz for their special contributions to the completion of this paper.
- The Department of Homeland Security for providing valuable information via the LOGIIC website http://www.cyber.st.dhs.gov/docs/LOGIICbrochure.pdf.

REFERENCES

1. ANSI/ISA-99.00.01-2007 (2007). *Security for Industrial Automation and Control Systems, Part1: Terminology, Concepts, and Models*. p. 24, used with permission, ISA, www.isa.org.
2. GAO (2004). *Challenges and Efforts to Secure Control Systems* March, 2004.
3. LOGIIC-1 Team (2005). *Project Framing Document for DHS LOGIIC Project*, July, 2005.

4. ISA Security Compliance Institute (2007). *Membership Prospectus*, June, 2007.
5. Aubuchon, T. (2006). The LOGIIC correlation project. *Presented at DHS LOGIIC Cyber Security Project Conference*, Houston, September 11, 2006.
6. Ulf Lindqvist (1999). *On the Fundamentals of Analysis and Detection of Computer Misuse*. PhD Thesis, School of Electrical and Computer Engineering, Chalmers University of Technology, Göteborg, Sweden Copyright 1999 by Ulf Lindqvist, figure reprinted with permission.
7. ISA99 Purpose (1995-2007). *ISA Website*, http://www.isa.org/MSTemplate.cfm?MicrositeID=988&CommitteeID=6821, used with permission, ISA, www.isa.org.
8. ISA Insights (2008). *The ISA Security Compliance Institute, 2008 Edition, used with permission—ISA Security Compliance Institutee*.

FURTHER READING

ANSI/ISA-TR99.00.02-2004 (2004). *Integrating Electronic Security into the Manufacturing and Control Systems Environment*.

ANSI/ISA-TR99.00.01-2007, (2007). *Security Technologies for Industrial Automation and Control Systems*.

Byres E.J., Leversage D, and Kube N. (2007). Security incidents and trends in SCADA and process industries. *Industrial Ethernet Book issue* **39**: 2.

Byres E.J. and Lowe J.. (2004). The myths and facts behind cyber security risks for industrial control systems, *VDE 2004 Congress*, VDE, Berlin, October.

http://www.us-cert.gov/control_systems/csdocuments.html#docs.

Kuipers D. and Fabro Mark. *Control Systems Cyber Security Defense-in-Depth Strategies*. (2006). Idaho National Lab, Idaho State.

NIST SP-800-53, Revision 2, NIST Recommended Security Controls for Federal Information Systems. http://csrc.nist.gov/publications/nistpubs/800-53-Rev2/sp800-53-rev2-final.pdf, 2007.

Permann M., Hammer J., Lee K., and Rohde K.. (2006). *"Mitigations for Security Vulnerabilities Found in Control System Networks"*, ISA.

Securing your SCADA and Industrial Control System, (2005). *U.S. DHS, ISBN 0-16-075115-2*.

Using Operational Security (OPSEC) to Support a Cyber Security Culture in Control Systems Environments version 1.0, Recommended Practice, February (2007).

US-CERT Informational Focus Paper, *Control Systems Cyber Security Awareness*, United States Computer Emergency Readiness Team, July.

9

CYBER SECURITY FOR THE BANKING AND FINANCE SECTOR

Valerie Abend and Brian Peretti
Department of Treasury, Washington, D.C.

C. Warren Axelrod
Bank of America, Charlotte, North Carolina

Andrew Bach
NYSE Euronext, New York, New York

Kevin Barry, Don Donahue, and Ken Wright
Depository Trust and Clearing Corporation, New York, New York

John Carlson
BITS, Washington, D.C.

Frank Castelluccio, Dan DeWaal, David Engaldo, and George Hender
Options Clearing Corporation, Chicago, Ilinois

David LaFalce
The Clearing House, New York, New York

Mark Merkow
American Express Company, New York, New York

William Nelson
FS-ISAC, Dulles, Virginia

John Panchery
Securities Industry Financial Market Association, New York, New York

Dan Schutzer
Financial Services Technology Consortium, New York, New York

David Solo
Corporate Technology Office, Citigroup Inc., New York, New York

Jennifer L. Bayuk
Consultant, Towaco, New Jersey

9.1 HISTORY OF COOPERATION

The US government and financial institutions have a long history of cooperation. The government recognized financial institutions as an integral part of the nation's critical infrastructure. As such, financial institutions are highly regulated and constantly supervised by regulatory agencies to ensure that they are able to withstand the various and increasing threats they face.

Examples of cooperation between the public and private sector in the late 1990s include preparations for the Century Date Change or "Y2K", *Preliminary Research and Development Roadmap for Protecting and Assuring Critical National Infrastructures* (July 1998) by the President's Commission on Critical Infrastructure Protection (PCCIP) and the Critical Infrastructure Assurance Office (CIAO),[1] and Presidential Decision Directive (PDD) 63 on Critical Infrastructure Protection (CIP, May 1998). PDD 63 established the first governmental approach to protecting the nation's critical infrastructures, assigning responsibility for protecting infrastructures in different economic segments to different governmental agencies, provided each responsible agency would appoint a private sector "Sector Coordinator" to work with the agency to pursue infrastructure protection in the sector, and encouraging the sharing of infrastructure protection information between government and private industry through the formation of information sharing and analysis centers (ISACs). It also supported research and development, outreach, and vulnerability assessment. PDD 63 described "A National Goal" as follows:

> "No later than the year 2000, the United States shall have achieved an initial operating capability and no later than five years from today [i.e. by May 22, 2003] *the United States shall have achieved and shall maintain the ability to protect the nation's critical infrastructures form intentional acts that would significantly diminish the abilities of*
>
> - the Federal Government to perform essential national security missions and to ensure the general public health and safety;
> - state and local governments to maintain order and to deliver minimal essential public services;
> - *the private sector to ensure the orderly functioning of the economy and the delivery of essential* telecommunications, energy, *financial* and transportation *services*." [emphasis added]

Under PDD 63, the Department of the Treasury ("Treasury") was assigned the responsibility for the banking and finance sector, and appointed Steve Katz, then Chief Information Security Officer for Citibank, as the first private sector "Sector Coordinator".

In the following years, the US Congress focused on cyber security issues as it related to privacy protection. Two significant laws governing privacy and security protections were enacted in the 1990s, the Health Insurance Portability and Accountability Act of (1996) also known as (HIPPA) and the Financial Services Modernization Act of 1999,[2] also known as the Gramm–Leach–Bliley Act (GLBA) (1999).

[1]The *Preliminary Research and Development Roadmap for Protecting and Assuring Critical National Infrastructures* is available at http://cipp.gmu.edu/archive/190_PCCIPCIAORandDRoadmap_0798.pdf Other pertinent documents can be found in the CIP Digital Archive in the George Mason University School of Law Critical Infrastructure Protection Program website at http://cipp.gmu.edu/clib/CIPDigitalArchive.php.
[2]Public Law No. 106–102.

HIPAA[3] was enacted to restrict control of and access to patients' information and GLBA includes a provision requiring financial institutions to safeguard personal information. In 2001, regulators finalized regulations requiring financial institutions to establish appropriate safeguards for the use, disclosure, privacy, and security of personal information, including Social Security Numbers (SSNs). The regulators applied strong enforcement tools to ensure that financial institutions complied with these security requirements. In addition, the Federal Financial Institutions Examination Council (FFIEC),[4] issued several Information Technology booklets on topics including information security, business continuity planning (BCP), and outsourcing.[5]

In January 2000, the Clinton Administration released *Defending America's Cyberspace: National Plan for Information Systems Protection, Version 1.0: An Invitation to a Dialogue*. This report urged the creation of public private partnerships to address cyber security issues

Shortly after the 9/11 attacks of September 11, 2001, the government and financial services industry responded. Executive Order (EO) 13228[6] *Establishing the Office of Homeland Security (HLS) and the Homeland Security Council* created the present structure for the protection of the homeland and EO 13231[7] *Critical Infrastructure Protection in the Information Age*, outlined, *inter alia*, the public partnerships context for the protection of the critical infrastructure. Private sector advisory councils were formed, including the Homeland Security Advisory Council(HSAC) (EO 13228) and the National Infrastructure Advisory Council (NIAC) (EO 13231). The Office of HLS, first headed by former Pennsylvania Governor Thomas Ridge, was formed. In addition, the President's Critical Infrastructure Protection Board (PCIPB), based on the Clinton administration's *Defending America's Cyberspace* plan, was established. The PCIPB coordinated an effort to draft a national infrastructure protection strategy that included contributions from both public and private participants. All participants were asked to comment on how this effort should evolve. In particular, the goal was to avoid legislation and regulation by means of proactive collaborative measures. Each of the critical sectors was directed to publish its own strategy.[8]

Several financial services industry organizations supported these efforts, including the Securities Industry Association (formerly SIA, now Securities Industry and Financial Markets Association [SIFMA]), BITS (the Financial Services Roundtable's technology and operations division), and the Financial Services Information Sharing and Analysis Center (FS-ISAC). This support was intended to foster closer working relationships between government and the finance sector.

The US financial regulators and the US Treasury Department were also looking at these issues. Following a series of organizational meetings in 2001, the US Treasury and financial regulators developed a process to coordinate the activities of federal and

[3]Public Law 104–191, 42 U.S.C. 1301 et seq.

[4]An interagency body with representation from the Board of Governors of the Federal Reserve System, Federal Deposit Insurance Corporation (FDIC), the National Credit Union Administration (NCUA), Office of the Comptroller of the Currency (OCC), and Office of Thrift Supervision (OTS).

[5]These Booklets are available at www.ffiec.gov/guides.htm.

[6]http://frwebgate.access.gpo.gov/cgi-bin/getdoc.cgi?dbname=2001_register&docid=fr10oc01-144.pdf.

[7]http://frwebgate.access.gpo.gov/cgi-bin/getdoc.cgi?dbname=2001_register&docid=fr18oc01-139.pdf.

[8]The entire list of sector plans, as well as copies of the plans, are available at the website of the Partnership for Critical Infrastructure Security (PCIS) at www.pcis.org.

state financial services regulators by establishing the Financial and Banking Information Infrastructure Committee (FBIIC).[9]

The FBIIC, originally a standing committee of the PCIPB, but currently chartered under the President's Working Group on Financial Markets, is charged with improving coordination and communication among financial regulators, enhancing the resiliency of the financial sector, and promoting the public–private partnership. Treasury's Assistant Secretary for Financial Institutions chairs the committee.

In fulfilling its mission, the FBIIC set out to:

- identify critical infrastructure assets, their locations, potential vulnerabilities, and prioritize their importance to the financial system of the US;
- establish secure communications capability among the financial regulators and protocols for communicating during an emergency; and
- ensure sufficient staff at each member agency with appropriate security clearances to handle classified information and to coordinate in the event of an emergency.

Working with appropriate members of financial institution regulatory agencies, the FBIIC has accomplished the following:

- provided key federal and state financial regulators with secure telecommunications equipment for use in a crisis, and we adding a capacity for encrypted e-mail;
- written emergency communications procedures allowing communication between financial regulators and Federal, state, and local stakeholders;
- worked to systematically identify critical financial infrastructures, assess vulnerabilities within the critical financial infrastructure, address vulnerabilities, and evaluate progress; and
- identified the infrastructure that is critical to the retail payments system, the insurance industry, and the housing finance industry.

On May 10, 2002, key leaders from the financial services industry, with the encouragement of the Treasury, established the Financial Services Sector Coordinating Council (FSSCC).[10] Rhonda MacLean, then Chief Information Security Officer at Bank of America Corporation, was appointed the second Sector Coordinator for Financial Services by Treasury, and served as the founding Chairman of the FSSCC. The banking and finance sector published its first version of the sector's critical infrastructure protection plan in May 2002. The "National Strategy for Critical Infrastructure Protection" was jointly drafted by several associations including BITS, SIA, FS-ISAC, AbA, and in consultation with the financial regulators.[11]

Members of the FSSCC and FBIIC meet three times a year for discussions and briefings.

On September 18, 2002, the Bush administration released a draft of *The National Strategy to Secure Cyberspace*. The *National Strategy* outlined the "preferred" means of interaction between the public and private sectors. After incorporating comments, the Bush administration released the final *National Strategy to Secure Cyberspace* in February

[9]Membership information can be found at www.fbiic.gov.

[10]Details about the FSSCC and its activities can be found at the FSSCC website at www.fsscc.org.

[11]A 2004 update of this strategy and other publications about the FSSCC's activities can be found at the FSSCC website.

2003.[12] On March 1, 2003, the Department of Homeland Security (DHS) was formally established and many of the responsibilities of the PCIPB were transferred to DHS.

In September 2002, several regulatory agencies released a draft paper outlining more stringent BCP requirements for certain types of large financial institutions. The *Draft Interagency White Paper on Sound Practices to Strengthen the Resilience of the US Financial System* was released for public comment by the Federal Reserve Board (FRB), Office of the Comptroller of the Currency (OCC), Securities and Exchange Commission(SEC), and the New York State Banking Department. Several financial institutions and associations submitted detailed comment letters on the proposal and objected to several onerous proposed requirements. In April 2003, three of the original agencies (the FRB, OCC, and SEC) released the final *Sound Practices White Paper* after considering 90 comment letters from industry participants.[13] The revised final paper did not insist on a minimum distance between primary and backup sites (e.g., 300 mile mission distance between primary and backup sites). However, it does require that institutions have staff, located outside their primary sites, which can conduct business if those at the primary site cannot get to the backup facilities. This became a good precedent for how meaningful, respectful discussion can lead to a proposal that meets requirements but is not overly burdensome on industry members.

In 2003, the President released the *National Strategy to Secure Cyberspace* and *National Strategy for Physical Protection of Critical Infrastructures and Key Assets*. These documents called for Treasury, as the lead agency for the banking and finance sector, to develop a research and development agenda for the sector. Treasury, working with the FBIIC and the FSSCC, published an agenda for the sector entitled "Closing the Gap". The driving force behind the document was a desire to identify key areas where additional research dollars could be spent to make the sector more secure. This document was socialized among Federal departments and agencies, academics, and financial services participants.

On March 7 and 8, 2005, Treasury, in conjunction with the National Science Foundation (NSF), hosted a workshop entitled "Resilient Financial Information Systems". Participants from academia and the public and private sectors worked to discuss and identify research priorities to advance the resilience of the financial sector and protect the nation's critical financial infrastructure. As the issue of research and development (R&D) for the financial services sector matured, the FSSCC developed a working group to focus specifically on the issue for R&D and to coordinate its activities with respect to critical infrastructure and key resources (CI/KR) R&D. At Treasury's request, the FSSCC joined DHS in a May 2005 workshop focused on R&D priorities.

DHS published an updated version of the National Infrastructure Protection Plan (NIPP) in 2005. The role of the sector-specific agencies in coordinating the activities of the sector was again reaffirmed in the document. As DHS was finalizing the NIPP R&D plans and programs, the FSSCC formed an R&D Committee to focus on those plans and programs that would provide the most significant benefits with respect to the specific CI/KR requirements of the financial services industry. In May 2006, this committee issued a list of priority research projects. The FSSCC Research and Development

[12]*The National Strategy to Secure Cyberspace*, The White House, February 2003, is available at www.whitehouse.gov/pcipb/cyberspace_strategy.pdf. This document implements a component of *The National Strategy for Homeland Security* and is complemented by *The National Strategy for the Physical Protection of Critical Infrastructures and Key Assets*, which are available at www.whitehouse.gov/pcipb/physical_strategy.pdf.
[13]The Interagency Paper is available at www.sec.gov/news/studies/34-47638.htm.

Committee Research Challenges and the FSSCC Research and Development Research Agenda were issued to assist researchers in focussing research on top concerns.[14] In February 2008, the FSSCC R&D Committee began to "beta test" the Subject Matter Advisory Response Team (SMART) program. The SMART program assists research and development organizations working on Critical Infrastructure Protection Projects by providing subject matter expertise for financial institutions necessary to facilitate their R&D endeavors.

9.2 ORGANIZATIONAL ROLES

9.2.1 FSSCC

The Financial Services Sector Coordinating Council (FSSCC) for critical infrastructure protection and homeland security (CIP/HLS) is a group of more than 30 private sector firms and financial trade associations that works to help reinforce the financial services sector's resilience against terrorist attacks and other threats to the nation's financial infrastructure. Formed in 2002, FSSCC works with Treasury, which has direct responsibility for infrastructure protection and HLS efforts for the financial services sector.

The mission of the FSSCC is to foster and facilitate the coordination of financial services sector-wide voluntary activities and initiatives designed to improve CIP/HLS. Its objectives are to:

- provide broad industry representation for CIP/HLS and related matters for the financial services sector and for voluntary sector-wide partnership efforts;
- foster and promote coordination and cooperation among participating sector constituencies on CIP/HLS related activities and initiatives;
- identify voluntary efforts where improvements in coordination can foster sector preparedness for CIP/HLS;
- establish and promote broad sector activities and initiatives that improve CIP/HLS;
- identify barriers and recommend initiatives to improve sector-wide voluntary CIP/HLS information and knowledge sharing and the timely dissemination processes for critical information sharing among all sector constituencies; and
- improve sector awareness of CIP/HLS issues, available information, sector activities/initiatives, and opportunities for improved coordination.

As described above, the FSSCC is the private side of the public–private partnership which supports the National Infrastructure Protection Plan (NIPP). The other organizations listed in this section are all members of the FSSCC. Each organization has strengths in different areas, allowing the FSSCC to coordinate efforts of various members in support of overall infrastructure protection goals. Since the FSSCC was established, it has been chaired by distinguished and prominent members of the financial community Rhonda MacLean of Bank of America from 2002–2004, Donald Donahue of The Depository Trust and Clearing Corporation from 2004 through 2006 and George S. Hender of The Options Clearing Corporation from 2006 to 2008, and Shawn Johnson of State Street Global Advisors in 2008.

[14]Both of these documents are available at www.fsscc.org.

9.2.2 FSSCC Member Organizations

All FSSCC member organizations have contributed to industry goals for CIP. The organizations described below have provided the most direct focus on collaboration with respect to cyber security issues in the Banking and Finance Sector.

9.2.2.1 BITS. In 1996, members of Bankers Roundtable (now The Financial Services Roundtable) created BITS in order to respond to significant technological changes facing the banking industry. BITS initially focused on changes in electronic commerce and the payments system, but evolved over time to focus on new threats that emerged in the areas of Internet security, fraud reduction, and CIP. Before 9/11, BITS helped to create the FS-ISAC. After 9/11, BITS helped to create the FSSCC and ChicagoFIRST.[15]

In 2001, BITS established the BITS Crisis Management Coordination Working Group (CMC-WG). This working group implemented The BITS and Financial Services Roundtable Crisis Communicator, a high-speed communications programs, that allowed the organization to connect all the key players—member CEOs and government and other business leaders—who might need to convene and determine how to address a crisis. The *BITS and Financial Services Roundtable (FSR) Crisis Management Process: Members' Manual of Procedures* was developed to provide BITS' members with the ability to communicate and coordinate with each other, government agencies, and other sectors in order to implement the emergency response and recovery process for the financial services sector.

One of the greatest lessons learned from 9/11 was the extent of the financial services sector's interdependencies and reliance on other critical sectors, specifically telecommunications and power. With the help of the Board of Governors of the Federal Reserve System, notably Steve Malphrus, BITS convened a conference in New York City in July 2002. The conference focused on ways to get tangible progress from other critical infrastructure sectors toward the goal of cooperation between government and the private sector.

One tool that resulted from the BITS Telecommunications Working Group efforts is the *BITS Guide to Business—Critical Telecommunications Services*. Completed in 2004[16], the Guide is based on extensive work by BITS members, participation by major telecommunications companies, and involvement by the National Communications System (NCS) and the President's National Security Telecommunications Advisory Council (NSTAC). The Guide is a comprehensive tool used by BITS' member institutions to better understand the risks of telecommunications interdependencies and achieve greater resiliency.

9.2.2.2 ChicagoFIRST. Another clear lesson from 9/11 was the stunning impact an event could have on critical financial services operations that are heavily located in one regional area. Louis Rosenthal, ABN AMRO, and Ro Kumar, The Options Clearing Corporation, saw the potential risks in the Chicago area and energized their peers and a set of partners. BITS facilitated the process of forming the regional coalition. In 2003–04 the US Treasury Department founded an evaluation and guide for establishing regional

[15]ChicagoFIRST is a nonprofit association dedicated to addressing HLS and emergency management issues affecting financial institutions and requiring a coordinated response.

[16]The BITS Telecommunications Working Group, led by John DiNuzzo (formerly of FleetBoston/Bank of America Corporation) was a subgroup of the BITS CMC-WG.

coalition through the Boston Consulting Group and BITS. ChicagoFIRST, the result of these efforts, is a free-standing nonprofit organization that provides robust coordination services to maintain the resilience of the critical financial services that reside in the area. It continues to serve as a model for others, including FloridaFIRST and other regional coalitions.[17]

9.2.2.3 Financial Services Information Sharing and Analysis Center (FS-ISAC). The FS-ISAC was conceived at a meeting of Financial Industry leaders with the Treasury at the White House Conference Center in March 1999. An Information Sharing Working Group was established. The financial services industry members participating in the original Information Sharing Working Group appointed a Board of Managers, who formed FS-ISAC limited liability corporation (LLC). It was officially launched by US Treasury Secretary Lawrence A. Summers at a ceremony in the Treasury building on October 1, 1999, as a means of meeting the finance sector's information-sharing obligation under PDD 63 on CIP.

On December 9, 2003, the Treasury announced that it would purchase $2 million in services from the FS-ISAC. Treasury's contract with the FS-ISAC resulted in a new, next-generation FS-ISAC that is intended to benefit the Treasury, other financial regulators, and the private sector. In the press release, the Treasury indicated the purposes for the funding were as follows[18]

- Transform the FS-ISAC from a technology platform that serves approximately 80 financial institutions to one that serves the entire 30,000 institution financial sector, including banks, credit unions, securities firms, insurance companies, commodity futures merchants, exchanges, and others.
- Provide a secure, confidential forum for financial institutions to share information among each other as they respond in real time to particular threats.
- Add information about physical threats to the cyber threat information that the FS-ISAC currently disseminates.
- Include an advance notification service that will notify member financial institutions of threats. The primary means of notification will be by Internet. If, however, Internet traffic is disrupted, the notification will be by other means, including telephone calls and faxes.
- Include over 16 quantitative measures of the FS-ISAC's effectiveness that will enable the leadership of the FS-ISAC and Treasury to assess both the FS-ISAC's performance and the aggregate state of information sharing within the industry in response to particular threats.

The FS-ISAC was able to arrange with a managed security service provider to fund the initial development and implementation of the FS-ISAC systems and networks in return for the right to reuse the technology developed. The FS-ISAC thus succeeded in meeting its original goal of becoming a viable means for the banking and finance sector to share information about security threats, vulnerabilities, incidents, and remedies. E-mail alerts and notifications sent by the FS-ISAC give financial firms advanced notice of threats,

[17]Improving Business Continuity in the Financial Services Sector: A Model for Starting Regional Coalitions (US Treasury: November, 2004). http://www.treas.gov/press/releases/reports/chicagofirst_handbook.pdf
[18]http://www.ustreas.gov/press/releases/reports/factsheet_js1048.pdf.

vulnerabilities, and events so that they can proactively protect themselves. The FS-ISAC also hosts an information-sharing website, conference calls, and conferences that allow its members more interactive sharing opportunities.

In 2006, the FS-ISAC established a Survey Review Committee to provide oversight of the process of member-submitted surveys of the FS-ISAC membership. The FS-ISAC survey process allows for one live poll at a time to ensure maximum participation. The primary contact at each member organization is asked to complete each survey or route it to the appropriate area within their company to have it answered by the most qualified individual. Surveys conducted in 2007 included *Employee Access to HR Information*, *Data Transfer Methods*, and *Information Security Program Organization*. Once the survey is completed, a Poll Results Report is created that includes a brief summary and the final poll results. Using the survey tool link provided, members can also conduct their own detailed analysis of survey results to meet their unique needs.

Through the personal involvement of members of the FS-ISAC's Board of Managers and the FS-ISAC membership at large, the reach of the FS-ISAC members[19] quickly spread well beyond the original mandate. Early on, board members were involved in efforts such as

- participating, through the FSSCC, in drafting the finance sector's segment of Version 2.0 of the NIPP;
- assisting in, and being supportive of, the establishment of the BITS laboratory for testing and certifying security software relevant to financial services institutions;
- working with Treasury to develop an outreach and education program to increase awareness of sector security threats, vulnerabilities, and best practices, and to indicate how the FS-ISAC might assist them in these areas;
- briefing Federal agencies as to the workings of the FS-ISAC; and
- testifying before congressional committees and otherwise representing the views of the banking and finance sector on cyber security and CIP.

The FS-ISAC has been a model for a number of other ISACs in critical US sectors, such as transportation, energy and information technology, as well as ISACs in foreign countries (e.g. Canada) and in individual corporate organizations (e.g. the Worldwide ISAC). Its October 2007 biannual conference was recently coordinated in conjunction with the CIP Congress, carrying the theme "When Failure is Not an Option" and was accordingly attended by members of other ISACs.

9.2.2.4 FSTC. The Financial Services Technology Consortium (FSTC) was established in 1993 at the dawn of the commercialization of the Internet. FSTC is a nonprofit organization with members from the financial services industry (financial services providers and vendors), government agencies, and academia, who collaborate on projects to explore and solve strategic business–technology issues through concept validation, prototype and piloting, and development of standards. Its mission is to harness technology advances and innovative thinking to help solve the problems of the financial services industry.

Early projects dealt with paper check imaging, the convergence of the payments products, and securing electronic banking, commerce, and payments over the Internet. These

[19]The Board of Managers and members of the FS-ISAC are not restricted from other industry activities beyond the work of the FS-ISAC.

projects helped spur the growth of electronic commerce and paved the way for Check 21 and the electronification of the paper check through the development of important new standards and industry utilities and collaborations.

After September 11, FSTC's focus expanded to include addressing business continuity issues in addition to security, fraud management, and payments, leading to a partnership with Carnegie Mellon that developed a Resiliency Framework. FSTC also initiated a focus on enterprise architecture aimed at helping financial services firms to streamline and consolidate their siloed systems and processes, enabling the reduction of redundant processes and systems, to provide a more efficient and flexible organization, able to more rapidly and easily accommodate new products, services, and processes needed to meet new business opportunities and threats.

FSTC thrives when the knowledge of members comes together through the formation of initiatives and projects that will better the industry as a whole. FSTC projects are its core activity and one of the key benefits of FSTC membership.

9.2.2.5 SIFMA. SIFMA provides a forum for securities firms, exchanges, industry utilities, and regulators to share knowledge, plans, and information. It is responsible for developing and promoting industry-specific practice guidelines, for providing liaison between the securities industry and regulators and legislators, and for coordinating industry-wide initiatives. SIFMA has standing committees to coordinate industry-wide initiatives for various types of securities industry trading and operations activities.

The SIFMA BCP Committee was established as the SIA BCP in November 2001 to address and coordinate business continuity issues for the securities industry. In conjunction with the BCP Committee mission, SIFMA (and its predecessors, the SIA and the Bond Markets Association) has led an extensive on-going industry-wide business continuity testing initiative since 2002. The effort allows the industry as a whole to verify and demonstrate the resilience of the securities markets and to provide individual firms with opportunities to test their procedures with other industry participants in a way they could not do on their own. Industry tests include tabletop exercises, connectivity tests, communications tests, participation in national disaster recovery tests, and pandemic flu exercises. SIFMA in conjunction with the BCP Committee operates the Securities Industry Emergency Command Center that functions as the industry's central point of emergency communications and coordination during significant emergencies.

Initial testing efforts in 2002, 2003, and 2004 involved basic connectivity tests between individual firms and exchanges. Much more robust business continuity tests were conducted in 2005 and 2006. Over 250 firms, exchanges and industry utilities participated in these tests, which involved transmission of dummy transactions from firms' and exchanges' backup sites using backup communications links. The industry demonstrated a 95% pass rate on these tests. SIFMA also coordinates securities industry participation in the national TopOff emergency exercises and focuses heavily on planning for a potential flu pandemic and on conducting pandemic planning exercises.

SIFMA's Information Security Subcommittee, which was established in 2003, addresses and coordinates information security issues from an industry perspective and facilitates information sharing among SIFMA member firms. The Subcommittee provides comments to regulatory authorities on proposed information security rules and regulations and develops industry initiatives. The Subcommittee has focused on a variety of issues including developing guidance on the design and testing of Sarbanes Oxley controls, working with legislators on proposed Security Breach Legislation, tracking and assessing Microsoft security releases, and establishing guidance on effective means of dealing with phishing attempts.

In 2007, SIFMA formed the Information Risk Advisory Council to provide advice to SIFMA's Technology, Information Security, BCP, and Privacy Committees. The Council identifies issues of significant importance to securities firms and works with SIFMA Committee to integrate these into the committees' annual goals.

9.3 SAMPLE SIGNIFICANT EVENTS

Although cyber security-related events are a daily occurrence in the financial industry, some events are more significant than the others with respect to collaborative information sharing. The events listed below were significant in that the collaboration that occurred during the event served to strengthen the bonds of communication between public and private sector CIP organizations.

9.3.1 Russian Hacker Case

In June 1994, a Russian crime ring managed to get inside the Citibank computer system and transfer $140,000 from the Philippine National Bank to a bank in Finland. The bank in the Philippines called to complain that the transaction had not been authorized. Citibank realized something was amiss and set up a special team to start looking into transactions of similar circumstance. However, it was not given that the unauthorized transfer was the first discovery of a chain of illegal activity. By the middle of July, the team identified a similar transfer had taken place and yet a third by the end of the month. By this time, Citibank had called in the Federal Bureau of Investigation (FBI) and the investigation was in full swing. Transactions were being illegally transferred from cities as far away as Djakarta and Buenos Aires to banks in San Francisco and Israel. In total, fraudulent transactions amounted to more than $3 million; though in the end, the gang of thieves managed to abscond with only $400,000.

The system breached was called the Citibank Cash Management system. This system allowed corporate customers to transfer money automatically from their accounts to whoever they are paying. And it handled approximately 100,000 transactions a day, totaling $500 billion. The Citibank system relied on static passwords, which they intend for users to memorize. The passwords remain the same each time a user enters the system, and although they are encrypted, the crime ring was somehow able to get a password and identification numbers of some of these corporate customers. The investigation team realized that the passwords traversed through many network links that were not necessarily fully owned and operated by the bank, but many were leased from telecommunication companies in various countries which provided the bank with network links between its offices. The question the investigators faced was did the perpetrator have an insider in Citibank or was he able to get them using conventional "network-sniffing" software.

On August 5, a fraudster transferred $218,000 from a Citibank account in Djakarta and another $304,000 from a bank in Argentina to Bank of America accounts in San Francisco that had been set up by a Russian couple. They would go to the bank after the money was transferred and attempt to withdraw it. At that point, investigators identified the perpetrators. They were kept under observation by both the public and private sector through October, transferring money from and to more accounts.

The idea of computer control of funds was new to the media at that time. It was a new idea to reporters that a person could be sitting at a computer in Russia in the middle of the night keying in passwords and watching money move across a screen. The Internet was still young at the time and largely unused commercially. The transfers were done through a proprietary network managed by Citibank. But, like the Internet, these proprietary networks cross over other proprietary networks and it is at these points that passwords become most vulnerable. Yet cooperation between the bank investigators, telecommunications administrators, and law enforcement led eventually to Vladimir Levin, a young Russian hacker. He was trapped through a traced telecommunications line performing a fraudulent transaction and was imprisoned. In the course of the investigation, several people were arrested (including half a dozen Russian citizens, for which this story is known as the “Russian Hacker Case”). Immediately after, Citibank ended the use of static passwords over its Funds Transfer networks and started issuing One Time Password tokens to customers using those networks (these tokens were a form of two factor authentication from a small company named RSA from its founders, Rivest, Shamir, and Adelman, then infrequently encountered).

9.3.2 Slammer Worm

On January 23, 2003, a structured query language (SQL) injection dubbed the “slammer worm” started to infect rapidly through computer systems throughout the world. Although a patch was released for the vulnerability, many organizations had not installed it. As a result, the worm spread very quickly, infecting, by one account, 75,000 victims within 10 min after its release.

Although financial institutions were not greatly affected by the worm, Treasury, in coordination with the FBIIC and FSSCC, convened a meeting on February 25, 2003, to discuss issues related to the worm. In addition to members of the FBIIC and FSSCC, several private sector groups attended, including Microsoft and electronic data system (EDS). At the meeting, communications protocols were developed to aid in the sharing of information in the event of another incident. The protocols were exercised during several other virus/worm attacks, including SoBig.F and BugBear.b.

9.3.3 2003 Power Outage

At approximately 4:11 pm Eastern Daylight Time (EDT) on August 14, 2003, a power outage affected a large portion of the Northeastern United States, roughly from Detroit to New York City. Although there was minimal disruption to delivery of financial services in the affected area, the incident did expose a greater need to continue to examine the backup systems institutions. For example, the American Stock Exchange had relied upon steam power to cool their trading floor. Upon reaching out to the SEC and the Treasury, a backup steam generator was located and the exchange was able to open and close on Friday, August 15, 2003.[20] Many lessons learned from that set of events. One lesson led to the *BITS Guide to Business—Critical Power*, developed in cooperation with the

[20]The report, *Impact of the Recent Power Blackout and Hurricane Isabel on the Financial Services Sector*, can be found at http://www.treas.gov/offices/domestic-finance/financial-institution/cip.

TABLE 9.1 Publications and Events

Date	Name of Publication/Event	Comments
February 1996	CIWG (Critical Infrastructure Working Group) Report	Suggested establishing PCCIP (President's Commission on Critical Infrastructure Protection) for the longer-term view and the IPTF (Infrastructure Protection Task Force) for coordination of then existing infrastructure protection efforts.
July 1996	EO (Executive Order) 13010	Formed PCCIP, IPTF and CIAO (Critical Infrastructure Assurance Office) Available at www.fas.org/irp/offdocs/eo13010.htm
October 1997	Critical Foundations: Protecting America's Infrastructures	Report issued by PCCIP suggesting a strategy incorporating research and development, information sharing, education, and awareness
May 1998	PDD-63 (Presidential Decision Directive Number 63) for Critical Infrastructure Protection	By May 2003: The Federal Government to perform essential national security missions and to ensure the general public health and safety State and local governments to maintain order and to deliver minimum essential public services The private sector to ensure the orderly functioning of the economy and the delivery of essential telecommunications, energy, financial, and transportation services.
July 1998	Preliminary Research and Development Roadmap for Protecting and Assuring Critical National Infrastructures	Report issued by PCCIP and CIAO as a follow-up of *Critical Foundations: Protecting America's Infrastructure*. Section 2.1 addresses the Banking and Finance sector
October 1999	Official launch of the FS-ISAC (Financial Services Information Sharing and Analysis Center)	Launched by US Treasury Secretary Laurence P. Summers—available at www.fsisac.com
January 2000	Defending America's Cyberspace: National Plan for Information Systems Protection, Version 1: An Invitation to a Dialog	This report urged the creation of public private partnerships to address cyber security issues
January 2001	Report of the President of the United States on the Status of Federal Critical Infrastructure Protection Activities	Available at www.fas.org

TABLE 9.1 (*Continued*)

Date	Name of Publication/Event	Comments
March 2002	Banking and Finance Sector: The National Strategy for Critical Infrastructure Protection	Available at www.pcis.org
May 2002	Banking and Finance Sector National Strategy for Critical Infrastructure Assurance	Available at www.pcis.org
July 2002	National Strategy for Homeland Security	Available at www.whitehouse.gov/homeland/book/nat strat hls.pdf
February 2003	The National Strategy for the Physical Protection of Critical Infrastructures and Key Assets	Available at www.whitehouse.gov/pcipb/physical.html
February 2003	The National Strategy to Secure Cyberspace	Available at http://www.whitehouse.gov/pcipb/
March 2003	FFIEC IT Examination Handbook: Business Continuity Planning	Available at www.ffiec.com
2003	PCIS Industry Compendium to the National Strategy to Secure Cyberspace	Analysis of plans and summary of commonalities. Available at www.pcis.org
July 2003	Risk Management Principles for Electronic Banking, Basel Committee on Banking Supervision, Bank for International Settlements	Available at www.bis.org/publ/bcbs98.pdf
December 2003	Homeland Security Presidential Directive (HSPD)—7 on Critical Infrastructure Identification, Prioritization, and Protection	Covers policy, roles and responsibilities of Secretary of Homeland Security, other offices, and so on, coordination with the private sector. Note: Consistent with Homeland Security Act of 2002, produce "National Plan for Critical Infrastructure and Key Resources Protection" within one year, that is, by December 2004. www.whitehouse.gov/news/releases/2003/12/print/20031217-5.html
May 2004	Homeland Security Strategy for Critical Infrastructure Protection in the Financial Services Sector: Version 2	Objectives of Financial Services Strategy: Identifying and reducing vulnerabilities in the financial services infrastructure to such attacks Ensuring the resiliency of the nation's financial services infrastructure to minimize the damage and expedite the recovery from attacks that do occur, and

(*continued*)

TABLE 9.1 (*Continued*)

Date	Name of Publication/Event	Comments
		Promoting public trust and confidence in the financial services sector's ability to withstand and recover from attacks that do occur. Available at www.fsscc.org
February 2005	National Infrastructure Protection Plan (Interim)	Superseded by June 2006 NIPP http://cipp.gmu.edu/archive/ Interim NIPP Feb 05.pdf
2005	FFIEC IT Examination Handbook: Information Security	Available at www.ffiec.com
April 2003	Interagency Sound Practices to Strengthen the Resilience of the US Financial System	Available at www.sec.gov/news/studies/ 34-47638.htm
April 2006	FSSCC Research Challenges Booklet	Available at www.fsscc.org
June 2006	National Infrastructure Protection Plan	Available at www.dhs.gov
October 2006	FSSCC R & D Agenda	Available at www.fsscc.org
December 2006	FSSCC Annual Report	FSSCC published the Banking and Finance Sector-Specific Plan as their annual report. Available at www.fsscc.org
May 2007	Sector-Specific Plan: Banking and Finance Sector for Critical Infrastructure Protection	http://www.dhs.gov/xlibrary/assets/nipp-ssp-banking.pdf
2005 (–2007)	Protecting the US Critical Infrastructure: 2004 (–2006) in Review	Annual reports, expected to continue, available at www.fsscc.org

Critical Power Coalition and Power Management Concepts, and published in 2006. It provides financial institutions with industry business practices for understanding, evaluating, and managing the associated risks, when the predicted reliability and availability of the electrical system are disrupted—and it outlines ways by which financial institutions can enhance reliability and ensure uninterrupted backup power.

The following table, Table 1 describes a series of publications and events related to information sharing and coordination within the finance and banking sectors.

9.3.4 Pandemic Planning

In September and October 2007, SIFMA, in partnership with the FSSCC, the FBIIC, and the Treasury, conducted a multiweek pandemic flu exercise for the full financial services sector. This was the largest most ambitious financial services exercise to date that addressed business process recovery as a sector in communication with its sector-specific agency. The exercise offered a realistic simulation of the spread of a pandemic wave in the

United States. It was designed to identify how a pandemic could affect the financial markets and to provide participants with an opportunity to examine their pandemic business recovery plans under a demanding scenario. Over 2700 financial services organizations participated.

9.3.5 Operation Firewall

On October 28, 2004, the US Department of Justice, in coordination with the United States Secret Service (USSS), executed over 28 search and arrest warrants in connect with Operation Firewall,[21] an undercover investigation designed to stop the flow of stolen credit card numbers and other personal information. This operation lured criminals into a false sense of security by creating a fake website for buying and selling purloined credit card information. The main target was a group that called itself Shadowcrew, whose sole purpose was to defraud the financial services sector.

The operation, which lasted over an 18 month period, ended with the seizure of over 100 computers and the arrest of 28 individuals—21 in the United States and seven in Europe and Russia. Through the cooperation of several major financial services sector entities, the underground "carding" scene was dealt a major blow from which it is still attempting to recover.

9.4 FUTURE CHALLENGES

The examples above demonstrate high levels of collaboration among dedicated individuals representatives financial institutions, associations, and government agencies. For this collaboration to continue, it will require proactive engagement, open communications, and trust. The industry needs to cooperatively work with the respective agencies to develop rules and regulations that best meet the requirements of government while maintaining a strong finance sector and not overburdening financial institutions.

Since 9/11, government has proven its willingness to reach out and ensure the consensus of the financial community in its efforts to strengthen the infrastructure. It has also demonstrated increased trust on the part of the private side of the financial sector of government's intent and a willingness to work with the various agencies, and to persuade others that cooperation is ultimately the best approach where each side can achieve its goals.

FURTHER READING

The FSSCC Research and Development Committee. (2006). *The FSSCC Research and Development Committee Research Challenges*, April 2006, http://www.fsscc.org.

The FSSCC Research and Development Committee. (2006). *The FSSCC Research and Development Committee Research Agenda*, October 2006, http://www.fsscc.org.

[21] http://www.secretservice.gov/press/pub2304.pdf.

INDEX

access control lists (ACLs), 13
Ad hoc On-Demand Distance Vector (AODV), 63
Advanced Encryption Standard (AES), 30
antiphishing efforts, 45
audit, 16
automated security policy, 13

banking and finance sector cyber security
  cooperation history, 98–102
  future challenges, 112
  organizational roles
    FSSCC, 102–107
  sample significant events, 107
    2003 power outage, 108–111
    Operation Firewall, 112
    pandemic planning, 111–112
    Russian hacker case, 107–108
    slammer worm, 108
Basic Security Profile (BSP), 83
Bell and LaPadula model, 15
benchmarks, 2
Biba model, 15
biometric devices, as physical interface, 34
BITS, 99, 103
  BITS and Financial Services Roundtable (FSR) Crisis Management Process: Members' Manual of Procedure, 103s
  Crisis Management Coordination Working Group (CMC-WG), 103
  Guide to Business—Critical Telecommunications Services, 103
blacker, 25

ChicagoFIRST, 103–104
Citibank Cash Management system, 107
commercial-grade security, 30
Common Vulnerability Scoring System (CVSS), 6
compartmented mode system, 22
configuration errors, 48
control systems and commercial off the shelf (COTS) technology, 88
covert channels, 21–22
Critical Infrastructure Assurance Office (CIAO), 98
Critical Infrastructure Protection (CIP), 98
Critical Infrastructure Protection And Homeland Security (CIP/HLS), 102

Data Capture, 72
dedicated system, 10
Department of Homeland Security (DHS), 101
developmental threats, 18
digital certificates, 46
directed diffusion, 62
discretionary access control policies, 10
distributed control systems (DCSs), 89, 91, 93
Dynamic Source Routing (DSR) protocol, 63

Enterprise Security Management (ESM), 93
European ARTIST2 Consortium, 65

Federal Financial Institutions Examination Council (FFIEC), 99
Federal Reserve Board (FRB), 101
Financial and Banking Information Infrastructure Committee (FBIIC), 100, 108

Financial Services Information Sharing and Analysis Center (FS-ISAC), 99, 104–105
Financial Services Modernization Act (1999), 98
Financial Services Sector Coordinating Council (FSSCC), 100, 101–102, 108
  member organizations, 103–107
Financial Services Technology Consortium (FSTC), 105–106
flaw hypothesis methodology, 21
FloridaFIRST, 104

Global Distributed Honeynet (GDH), 71, 73–74
global mandatory policies, 12
Gramm–Leach–Bliley Act (GLBA) (1999). *See* Financial Services Modernization Act (1999)
group key management protocol (GKMP), 62

hammering, 31
HCISEC (interface between the fields of human—computer interaction and security), 41
Health Insurance Portability and Accountability Act (HIPPA) (1996), 98, 99
‘‘high-assurance program’’ (HAP), 36
Homeland Security Advisory Council (HSAC), 99
HoneyMonkey project, 75
honeynet technologies, 69
  Generation III, 71–73
  Global Distributed Honeynet (GDH), 73–74
  honeyclients, 74–75
  honeypot
    essentials, 69–70
    risk, 70–71
  low-interaction malware collectors, 76
honeypot
  essentials, 69–70
  risk, 70–71
honeywall, 71

Identity Metasystem, 85
industrial automation and control systems (IACS), 87
  background, 87–88
    new threats, 88–89
    problem, 88
    solution, 89
  scientific study, 89–90
    correlation benefit, 90
    detection, 91
    implementing defense and detection in-depth, 92
    technical challenges, 91–92
    test bed operating model, 93
information sharing and analysis centers (ISACs), 98, 105
integrity Trojan Horse, 14
intrusion detection systems (IDS), 3, 89
  event triggered responses, 91
INtrusion-tolerant routing protocol for wireless SEnsor NetworkS (INSENS), 64

ISA-99 Committee, 94
ISASecure™, 94–95

key continuity management (KCM), 47

LKHW, 62, 64
Localized Encryption and Authentication Protocol (LEAP), 63
LOGIC-I project, 89, 91, 92, 93, 94, 95

mandatory policies, 10–15
  enforcement, 13
message-based security, 83–84
metrics and measures, for cyber security, 1
  Common Vulnerability Scoring System (CVSS), 6
  contrasts, 1–2
  measures selection for metrics support, 2
  problems
    with accuracy of measures, 2–4
    with measures selection, 4–5
    with use of measures, 5
  research directions, 6
multicore processors, 22
multilevel mode system, 23
multilevel security (MLS), 9
  background, 9–10
  enforcement of, 17
    assurance, 20
    cascade problem, 23
    covert channels, 21–22
    design approaches, 17–18
    object reuse considerations, 22
    secure system development, 20–21
    target environment, 22–23
    threats to MLS systems, 18–19
  platforms and architectures, 24–25
    use of applications, 25
  policies, 10–13
    confinement, 13–15
    supporting policies, 15–16
    trusted subjects, 16–17
multiple independent levels of security. *See* partitioning kernels

National Communications System (NCS), 103
National Infrastructure Advisory Council (NIAC), 99
National Infrastructure Protection Plan (NIPP), 101, 102
National Institute of Standards and Technology (NIST), 6
National Science Foundation (NSF), 101
  Cyber Trust (CT) Program, 65
  Embedded and Hybrid Systems (EHS) Program, 65
National Security Telecommunications Advisory Council (NSTAC), 103
nCipher algorithm, 31
Nepenthes, 76

Network in a Box (NiaB), 46–47
Network of Excellence on Embedded Systems Design, 65
New York State Banking Department, 101
nondiscretionary policies. *See* mandatory policies
NoSEBrEak, 73

Office of the Comptroller of the Currency (OCC), 101
operational mode system, 22
operational threats, 18
Operation Firewall, 112
organizational security policy, 13

pandemic planning, 111–112
partitioning kernels, 24
password, 42–43
  management, 35
periods processing, 10
phishing, 44–45
power outage (2003), 108–111
President’s Commission on Critical Infrastructure Protection (PCCIP), 98
President’s Critical Infrastructure Protection Board (PCIPB), 99, 101
Presidential Decision Directive (PDD), 98
process control networks (PCNs), 93
process control systems (PCSs), 92, 93, 94, 95
psychological acceptability, 41
Public Key Infrastructure (PKI), 30, 46

radio frequency identification (RFID) TAGS, 57, 58, 60, 61, 62, 63
regrading, of selected intelligence information, 12
Request Security Token (RST), 84
Request Security Token Response (RSTR), 84
Russian hacker case, 107–108

sanitization, 17
Sea View model, 15
Sebek, 72–73
Secure Ad hoc On-Demand Distance Vector (SAODV), 63
secure attention key, 16
Secure Code Update By Attestation (SCUBA) protocol, 65
Secure Efficient Ad hoc Distance-vector (SEAD), 63
Secure Implicit Geographic Forwarding (SIGF), 64
secure information aggregation (SIA), 64
secure sockets layer (SSL), 82–83
secure system development, 20–21
Securities and Exchange Commission (SEC), 101
Securities Industry and Financial Markets Association (SIFMA), 99, 106–107
security, of distributed, ubiquitous, and embedded computing platforms
  critical need analysis, 65
  global research and funding, 65
  research directions, 66
  scientific overview, 57–58
    design space constraints, 59–60
    middleware and applications, 64–65
    networking layers, 61–64
    physical layer, 61
    security properties, 58–59
    solution approaches, 60
Security Compliance Institute (ISCI), 94
Security Context Token (SCT), 84
Security Token Service (STS), 83, 85
sensitivity levels, 11
service oriented architectures (SOA), 79
  secure web services, 82
    message-based security, 83–84
    transport-based security, 82–83
  security challenges, 80–81
  web service security directions
    federations, 84
    identity, 85
Sesame, 44
shared-resource matrix method, 21
side channels, 22
SiteKey, 45
slammer worm, 108
Snort-inline process, 72
SOAP (Simple Object Access Protocol), 82, 83
Social Security Numbers (SSNs), 99
SPARCLE system, 44
Spinning Cube of Potential Doom, 44
SPINS, 62, 64
ssh program, 47
statistical en-route filtering (SEF), 64
steganography, 17
storage channels, 22
structured query language (SQL), 108
Subject Matter Advisory Response Team (SMART) program, 102
subversion, 19
supervisory control and data acquisition (SCADA), 89, 91, 93
system high operation, 22

technology usability and management
  current research, 41–42
    existing security technologies improvement, 42–43
  open challenges and take-aways, 48–50
  systems management, 43–44
  web security and phishing, 44–46
    design guidelines for building better systems, 47–48
    new technologies designing with usability in mind, 46–47
test bed operating model, 93
threshold cryptography, 62

Timed Efficient Stream Loss-tolerant Authentication (TESLA), 62
timing channels, 22
TinyPK, 62
TinySec, 62
tranquility, 17
transitivity, 33
transport-based security, 82–83
Triple AES, 30
Trojan Horse, 13–14, 19, 25
trusted computing base (TCB), 24
Trusted Network Connect (TNC), 36–37
trusted path, 16
Trusted Platform Module (TPM), 31–37
trusted platforms, 29
  challenges, 38–39
  expansion of ring of trust, 37–38
  international scope, 35
    integration, 35–36
    networking trusted platforms, 36–37
    trusted software, 36
  trusted computing, 29–31
    biometric devices as physical interface, 34
    essentials, 31–24
    extension to storage, 34
    hardware security, 31
    usage model, 35

United States Secret Service (USSS), 112
US Department of Justice, 112
uTESLA, 62, 64

virtual machine monitor (VMM), 25
virtual private network (VPN), 25

Web Services Description Language (WSDL), 82
Web Services Security (WSS). *See* service oriented architectures (SOA)
wireless local area network (WLAN), 44, 46–47
wireless sensor networks (WSNs), 58, 59, 60, 62, 63, 64, 66
WS-Federation, 84
WS-MetadataExchange, 85
WS-Policy, 84
WS-SecureConversation, 84
WS-Trust, 83, 85

XML, 82
XML Schema, 82