CERIAS Tech Report 2004-38

WHAT SHOULD A GOOD SECURITY MODEL BE?

by Marina Bykova

Center for Education and Research in
Information Assurance and Security,
Purdue University, West Lafayette, IN 47907-2086

What Should a Good Security Model Be?

Marina Bykova
Computer Sciences Department and CERIAS
Purdue University
mbykova@cs.purdue.edu

Abstract

In this article we survey a number of security models – which range from the first models to newly proposed approaches – in an attempt to answer the question of what we want our security model to be. The emphasis of this work is not on past or current security models, but rather on new approaches that have been proposed in the literature but have not yet found their way to the end user. The models described in this work provide unusual ways of addressing security needs and may be difficult to employ due to drastic differences from the currently accepted norms. These models, however, may have useful properties that the current systems do not possess, and might provide more efficient ways of securing our systems.

1 Introduction

In this work we focus on one of the most fundamental topics of security: How security can be defined and what security model or security policy can adequately reflect our expectation of a secure system. We present a number of security models starting from the earliest models, covering approaches in current use, and describing a number of solutions that have been recently proposed in the literature. The goal is to give a wide range of security models and identify their distinctive features and capabilities. This will help the reader to compare the models and make conclusions about the applicability of one or another model to a particular domain.

Over the last decade, the question of an adequate security model has attracted the attention of many researchers. Many new solutions have been proposed in the literature, some of which are in wide use today, while others, possibly more striking and effective, are not broadly known even to the academic community. Here we survey new approaches that have been developed in recent years with an emphasis on solutions that provide an unusual way of addressing security needs, or in one way or another differ from the security standards that we use today.

This work should be not viewed as an attempt to survey all of the security models that have been proposed in the past, which by itself might be an impossible task. In fact, selection of the past and present models to cover in this article was challenging because of the large number of existing models. This work also should not be viewed as a survey of all new models that have been recently proposed in the research literature. The focus of this work is on new directions that provide a new angle of approach to existing security problems. The models presented in this survey might never find their way to the end user, but they still deserve our attention, if not as models to blindly adopt, then at least as models that possess useful properties and from which we can learn.

The rest of this article is organized as follows: In section 2 we briefly describe a few early security models developed in the 1970s. Section 3 gives a quick overview of selected security models that are in wide use today. In section 4, the core of this work, we describe a number of security approaches proposed in the last decade that deviate from the commonly accepted view of security mechanisms. Lastly, section 5 concludes the survey.

2 Where It All Started

In the beginning there was a matrix. The earliest and possibly the most influential security model in the form of an access control matrix was proposed by Lampson [1], and later refined by Graham and Denning [2, 3]. In the matrix, each column corresponds to an object that we want to protect; and each row corresponds to a subject — an active entity, such as a user or process, which might possess rights over objects. A cell that corresponds to the intersection of a subject and object lists all of the rights that the subject is allowed to have over that object. Harrison, Ruzzo, and Ullman in their widely known HRU model [4] provide a number of fundamental results about the expressive power of models built on access control matrices.

Another early model that influenced the development of a large number of security models is the Bell-LaPadula model [5, 6]. This approach is based on the military-style classification scheme where all objects are placed in different compartments based on the level of their sensitivity. The model enforces confidentiality policies by controlling information flow. More precisely, it uses a number of rules which guarantee that no information can flow from a more sensitive to a less sensitive object, i.e., from a more restricted set of users to a larger community.

The Bell-LaPadula model stimulated the development of other types of models which use the same idea of data compartments. For example, the Biba integrity model [7], which is more suitable for commercial rather than military use, utilizes the same idea of (integrity) levels and information flow rules.

We will not go deeper in describing these models, but instead would like the reader to have a quick snapshot of the early theoretical models that were at the roots of security as a science. Security was primarily a concern of military and governmental institutions at the time these systems were developed, and they were designed to be well suited for such environments.

3 What We Have Now

In this section we review a few selected approaches from a large body of security models available today. Our selection is not accidental and consists of security solutions that, in our view, correspond to the most active current areas of research, are in wide practical use, or both. There are many other security models that are interesting from theoretical and practical points of view, but they are out of scope of this work.

3.1 Access Control Lists

Access Control Lists (ACLs) are not a security model on their own, nor they are an access control model. Instead, ACLs can be viewed as a mechanism for realizing permissions which are theoretically modeled as an access control matrix. Using matrices is impractical in many environments because of their large sizes, scarce space utilization, and poor scalability. ACLs can solve many of these problems and at the same time are conceptually simple, which has caused this mechanism to be widely adopted.

An access control list is defined as a set of pairs that specify subjects and their rights over a specific object or resource. The fact that there is no need to list all possible subject–object pairs in ACLs, and also possible aggregation of subjects or objects into groups (or use of wildcards to match a set of them) significantly reduces the space usage required to store these permissions. The rights normally can be positive or negative (i.e., grant or deny permissions), which makes it possible to use default permissions and further reduce the size of the lists. Different levels of aggregations and default permissions, however, make the ACL writing and maintenance process error-prone because of the possibility of conflicts and unforeseen rule interference.

Because access control lists can implement a large variety of access rules, they are well suited for many domains. This has made them a popular access control mechanism widely used, in particular, in operating systems and in network-based access (e.g., firewalls).

3.2 The era of RBAC

While early models relied on either Discretionary Access Control (DAC) or Mandatory Access Control (MAC), a more recent Role-Based Access Control (RBAC) model [8] has rapidly gained in popularity. The attractiveness of this concept is in its ability to naturally model many environments and simplify user management.

The model is built on the notion of a role — a collection of permissions suited for a particular task within the organization. A role can be assigned to many users; and a user can assume one or more roles, depending on his job function in the organization. Adding a user is then as easy as assigning appropriate roles to him, without explicitly granting a potentially long list of permissions that the user must have to perform his job function. User deletion is also as easy as disabling the ability of the user to assume certain roles.

The RBAC model is well suited for multiuser environments, but a role is more than just a group of users. A role ties a group of users on one side and a set of permissions on the other. The notion of role does not depend on the users that currently can assume that role.

RBAC has many variations. RBAC models form a family, each member of which differs in their capabilities of handling role hierarchies and constraints. More recent extensions to the model include temporal role-based access control (TRBAC) [9], generalized temporal role-based access control (GTRBAC) [10], and others. RBAC is still an active area of research.

To see how popular RBAC is, we only need to mention that it is used in many applications and different domains such as banking, health care, in operating system administration, on the Web, in DBMS systems, peer-to-peer collaborative environments, and more. An entire conference — the ACM Workshop on Role-Based Access Control — was dedicated to RBAC to stimulate development of this technology.

3.3 Trust Management systems

Decentralized Trust Management (TM) models were introduced only recently [11], but this topic has rapidly become an active area of research. Trust management was introduced as a unified and flexible mechanism for negotiating trust between strangers. It involves defining security credentials and formulating security policies. Authorization works by determining whether a particular set of credentials satisfies the security policy and granting trust to the owner of the credentials.

Similar to RBAC, trust management systems comprise a family of models that differ in their expressive power; like RBAC, these models are propagating in many domains. TM triggers new problems such as distributed credential chain discovery [12], automated trust negotiation [13, 14], protection of sensitive credentials in trust negotiation [15, 16, 17], and others that are under active current development.

4 What We Envision

With the introduction of the ACM New Security Paradigms Workshop (NSPW) in 1993, the research community was encouraged to develop new and possibly risky approaches to managing computer security. This conference was founded on the premise of providing a stimulating and interactive forum for exchanging innovative ideas, and is successfully working in this direction. The

research community continues to generate new approaches, and most topics covered in this section were introduced at NSPW.

It is widely known that security requirements evolve over time. In its early days, security was required only for military and national security applications, while nowadays it has come into the realm of the ordinary computer user. Therefore, the old models do not work and do not scale well for diverse environments and needs. In particular, Blakley [18] points out that:

- Policies do not scale well and their complexity quickly increases as systems grow and diverge, which makes them unmanageable.
- Strong secrecy is not achievable because it depends heavily on people, who generally cannot keep secrets well. Also, in many cases perfect security is not needed.
- Achieving system integrity is hard, very expensive, and requires trade-offs.

Thus the goal of this section is to describe the various paths we can take in addressing security needs that have been proposed in the literature. Some of them might appear to be difficult or unrealistic to adopt, but the goal of this survey is to list potential solutions in as complete a manner as possible.

4.1 Security as an inherent property

The first aspect that we are going to cover is security as an inherent vs. imposed property. Many sources agree on the fact that security should be explicitly built-in to applications at design time, rather than being added as an afterthought to otherwise fully developed systems. Smetters and Grinter [19] give examples of existing applications where security mechanisms are transparent to the user and are integral parts of the applications. For example, they mention identity-based encryption (IBE) [20], which is already being utilized by recent security designs. Identity-based encryption allows any string (e.g., the user's email address) to be used as a public key, which significantly simplifies the key management process. An interested user can refer to [19] for more examples of secure applications where security is not a burden to the user.

Next, we describe new directions that have been proposed in recent years but have not yet found use in existing security applications.

- **Size of data made proportional to its value.** Blakley [18] suggests mimicking security experiences that exist in the world outside of computers today. The goal of computing so far has been to make things faster and more convenient, while in real life very valuable items are designed to be inconvenient and cumbersome to handle. For example, $1 billion in $100 bills requires a massive amount of storage and cannot be carried by a single person, while a digital representation of $1 billion occupies only a few bytes. If we apply the principle of inconvenience to computing and make the size of digital cash proportional to its value, then stealing large sums of digital money is no longer as easy as stealing a few dollars.

  Another example mentioned in [18] is copyrighted material. If we make the size of representation of music, movies, and/or books proportional to their monetary value, then the user can be simply charged for the connect time that is needed to download the copyrighted material.

- **Information retrieval made slow.** Nelson [21] introduced the concept of unhelpfulness as a security mechanism. The author suggests controlling the rate at which information is released to the user (or whether it is released at all) as a way of managing security and ensuring compliance with the principle of least privilege. When it is not critical that the

system does not release large amounts of data in a timely manner, the application can be made intentionally slow and unhelpful in allowing access to more information. Systems that employ this principle already exist outside the digital world, e.g, release of unclassified information by the National Security Agency, where small portions of the telephone book are not classified while the entire book is considered to be classified information. This model, however, has not been yet applied to computing, and might allow us to build more secure and flexible systems.

- **Co-evolution of programs and data.** Blakley [18] points out another way of making security an integral part of applications. He suggests creating copies of the same program that look different by co-evolving program code and data. For example, we might partition an application's code and all the files that a user creates using this application into the same number of files as were provided as input. We store these output files, encrypted, in place of the original files, and when the application is invoked both the code and the data are reconstructed by using a shared secret.

  This strategy has a number of advantages. First, it forces different copies of the same application to look different, thus possibly reducing the number of computer viruses to which the application will be vulnerable. Second, it guarantees that any change to either the code or user data will result in major modifications of a large number of files, and thus cannot go undetected. The application, however, is now personalized to a particular user, which might require changes to the application update procedures that we currently employ.

There are many ways of integrating security into the applications that we use on a daily basis, where these security mechanisms might range from the obvious to rather unusual, and from very straightforward to implement to those that require major shifts in the way applications currently operate. But regardless of what mechanism we use, it can be effective only if we, from the beginning, design our applications with this security mechanism in mind.

4.2 Survivability approach to security

The survivability approach to security described below ties monetary values and security together. It might be considered by the reader to be related to the economic approach discussed in the next section. We, however, describe the two approaches separately because of drastic differences in their goals and means for achieving these goals.

An argument that we might hear today is that businesses are not going to invest resources in security unless they believe the money is well utilized. Lipson and Fisher [22] proposed a so-called survivability approach to security. Survivability is viewed as a merge of computer security and business risk management for the purpose of protecting valuable assets. Because no single system can be completely secure against attacks or fully tolerant to failures, the goal of this approach is to ensure that critical functionality is maintained even in the presence of attacks, failures, or accidents.

Survivability is defined in [22] from two perspectives: technical and business. The technical side of survivability allows us to ensure the availability of information and continuity of services by building survivable systems from unsurvivable components. From the business perspective, survivability, unlike traditional security, has a very strong emphasis on mission survival, but not on the survival of system components or even the system itself. From this point of view, survivability is more similar to risk management than traditional technical solutions that are not customized for a particular task. The survivability model is designed to protect the most valuable aspect of the system, its business mission, and tolerate failures of nonessential components, which we view as a reasonable path to security.

4.3 Economic and incentive-based approaches

The economic approach to security discussed here was suggested by Blakley [18]. In his work, the author gives an example of a system that makes privacy of medical records dependent on economics, instead of using the traditional protection via secrecy. More precisely, the model is defined as the following: Before any user is permitted to look at a patient's record, a pre-defined amount of money is transferred from the user's account to the patient's account. The model works because if the patient looks at his own record, the money is transferred from his own account to his own account. If a legitimate person, such as a doctor, looks at the record, then that person will bill the patient for the amount of money that was originally transferred to the patient's account. And finally, if an unauthorized user accessed the record, then the patient at least has some monetary compensation for the loss of privacy. The idea is to punish the user first, and if later the access is considered legitimate, reverse the action.

While the economic approach described above provides economic motivation strictly for individuals, as Blakley [18] further points out, we can design systems that provide incentives for cooperation or, in other words, build incentive-compatible security. By providing a set of incentive and punitive mechanisms, we can encourage users to act in a certain way and to not substantially deviate from the desired behavior, thus making the system more secure. For example, this strategy can prove very effective against social engineering attacks. Weirich and Sasse [23] investigate persuasive mechanisms that can be employed in order to make users cautious about their passwords and protection of the passwords. An example of such mechanisms is advertising that a break-in into an employee's account might result in personal embarrassment. Therefore if we build incentive-compatible security mechanisms into our systems, we will be dealing with more secure and stable systems.

4.4 Immune-system based approach

This section is based on the discussion of immune-system defense mechanisms by Somayaji et al. [24] and Williams [25] that draw appealing parallels between the human body defense system and computer security mechanisms[1]. The main argument behind immune-system based proposals is that human defense mechanisms have been evolving and developing for hundreds of thousands of years, and we should be able to learn from this experience and possibly adopt the methods to the domain of computing.

Immune systems have a number of similarities with computer security. Nowadays security deals with uncontrolled, imperfect, and open systems. Similarly, the human immune system is composed of many imperfect, unreliable, and short-lived components. The human immune system is not perfect, it makes mistakes. Likewise, we do not attempt to talk about perfect detection mechanisms, perfect protection, or perfect security. And both human health and computer security are impossible to measure: how can you tell that you are healthy? how do you know that your system is secure? Despite a large number of similarities, the immune system design principles differ dramatically from the security mechanisms that we currently employ. Below we briefly describe some of the immune system features that can be applied to the computing world to solve our security problems in a non-traditional way.

- **Multiple levels of protection.** No single protection mechanism constitutes the overall security. Instead, many (possibly more specific) protection mechanisms at different levels are used to protect the system, resulting in high quality defense of the system at large.

[1]This selection of publications is by no means an exhaustive list of work in the direction of the human immune system as applied to computer security. An interested reader is encouraged to take a look at other publications such as [26, 27, 28, 29, 30].

- **Distributed control.** No central coordination takes place, which means that there is no single point of failure. Different components of the systems might be responsible for conducting a certain task or performing a task at their local level, which makes the system very robust.

- **Diversity.** Diversity makes it harder for vulnerabilities to spread from one system to another. Diversity can be achieved by making sure that no single system is exactly the same as another, every system is unique. Alternatively, diversity can be achieved by making the protection mechanisms unique.

- **Adaptability.** The system learns to detect new threats, as well as remembering previously detected threats and applying its knowledge to recognize these.

- **Disposability.** No system component is vital — every cell can be replaced. This concept might be difficult to implement at the hardware level, but is definitely possible at the level of processes or agents.

- **Autonomy.** Every system is standalone; it does not need external control or maintenance. Such a system is capable of detecting problems and repairing them on its own. This level of independence might not be required or might be too difficult to achieve for all computer systems, but we may still benefit from the ability of systems to manage some of their security problems locally.

- **No trusted components.** No system component is considered secure and trusted; any component could be compromised. The protection mechanisms themselves are evaluated the same as any other component of the system and, if compromised, replaced.

- **Identity via behavior.** Unlike conventional security mechanisms, where identification depends on possession of a secret, in the immune system, identity is verified through behavior. When this concept is applied to computer security, we might, for example, consider the system calls that an application makes to be a metrics that we can use to evaluate its behavior.

If we go beyond these design principles in applying human health principles to the realm of computer security, we can develop an entire healthy style of living for computers. For example, Williams [25] suggests providing software consumers with detailed information about the products they buy, similar to the labeling system currently in place for food items. As food is marked according to its ingredients (vitamins, nutrition, etc.), we similarly label software packages and information placed on the web according to their content and the amount of danger they might present. Another healthy habit that can be borrowed from our health care system is to give vaccinations to a computer before it is allowed to be connected to the Internet.

There is, however, a significant difference between human and computer immune systems. The goal of human defense mechanisms is purely the survival of the system; there are no such notions as confidentiality or correctness. In the world of computers, however, we are concerned with many other issues apart from availability. All of confidentiality, integrity, availability, accountability, and correctness are essential to the operation of information systems, and we need to take this requirement into account if we decide to adopt some of the immune system defense mechanisms to computer security.

4.5 Optimistic security

Povey [31] gives a new security paradigm for mission-critical systems, called optimistic security. In this model, users are permitted to exceed their normal privileges under exceptional circumstances.

The motivation for this approach is that during the handling of disasters, medical emergencies, or other time-critical events, it might be crucially important to relax static authorization rules and allow users to take actions in reaction to a dynamically changing environment.

The optimistic approach is built on the grounds that, regardless of how flexible or expressive our access control mechanism is, it will not be able to take into account the dynamic nature of many mission-critical environments. There could be unforeseen situations that are not built into our security model but which need to be effectively handled should they occur. Thus, the proposed solution is to allow users to exceed their normal privileges under the assumption that most accesses will be legitimate and that users will take advantage of the additional privileges on rare occasions. The system also relies on external controls to ensure that the system's integrity is preserved, as well as on compensating mechanisms to correct undesired consequences.

Povey provides a number of requirements that any optimistic model must meet. They are: constrained entry points, accountability, auditability, recoverability, and the presence of deterrents. Optimistic security models must allow for secure auditing and reliable rollback. If an unreasonable access is detected, an administrator should be able to undo illegitimate modifications, take punitive actions, and remove privileges. Also, such systems should be employed only in environments where the risk of failure or the cost of recovery are low compared to the cost of not having the privileges to adequately deal with a situation. That is, optimistic security is not well suited for financial institutions, but can rather be used, for example, to sandbox semi-trusted software or to create emergency "break-glass" tools.

4.6 "Strike back" approach

Another security approach, which can be viewed as the opposite of the constraint-relaxed optimistic model, was proposed by Welch et al. [32]. The authors design their model in the context of information warfare, where a cyber attacker is treated as enemy who presents a serious threat to national security. They work from the assumption that information wars threaten the nation as much as conventional wars through disruption of financial, health, and other vital and quality of life infrastructures, and should be treated accordingly. Thus, the goal of computer security is to take appropriate actions to win this information war.

All of the actions that we currently take to protect our systems are defensive. We allow the enemy to attack our system on their own time, then we recover and take measures to prevent similar types of attacks from recurring. All the attacker needs to do is to develop a new type of attack to make us suffer losses again. The suggested approach is to change this strategy to an offensive one. This does not mean that we must become aggressive, but that once attacked we should take offensive measures to combat the threat. These measures include identifying the enemy's center of gravity (which might range from personal freedom or the wealth of individuals to knowledge and financing of foreign governments), and fighting to destroy it.

The authors argue that currently information warfare neglects key principles of war that are proven to work in conventional wars over time. Key principles of war are the offensive, maneuver, surprise, security, and economy of force. They tell us to place the enemy in a position of disadvantage, never permit the enemy to acquire unexpected advantage, strike the enemy when he is unprepared, and combat in the most effective way possible.

Although these measures might seem justified when we are dealing with serious threats and when our goal is to protect national security, we must first ensure that we are dealing with an enemy and that the aggression is well justified before any offensive measures becomes a possibility. Furthermore, it is important to take into account the role of deception in computer attacks, because offensive actions directed towards the wrong target invalidate the usefulness of this approach and, more importantly, they harm innocent civilians.

4.7 Functional approach

Another approach to computer security that questions the adequacy of the existing security mechanisms was proposed by Nelson [33]. We call this approach functional because it attempts to define security in the context of an application, with respect to its functions and needs, rather than universally for all applicable contexts.

Nelson points out that we do not very often ask ourselves what a secret is, or whether our assumptions about secrets and security are true. We attempt to define formal models that will capture the notion of security, but we do not necessarily question the adequacy of applying such models to our systems. Secrets cannot be kept absolute: we might keep them secret for some time, we might make their discovery hard, or we might prevent them from leaking quickly. What is and is not acceptable, and what is considered secure, must be decided on an individual basis, with respect to the application context. It is not surprising that general, application-independent security models that attempt to preserve perfect secrecy are usually too strict and expensive to build.

There are two main conclusions that the author makes in her work. The first is that security cannot be expressed accurately using only syntactic information. Any algorithmic solution that does not take into account the specifics of a system will capture only part of the requirements, and will miss important aspects of the system's security.

The second conclusion is that functional models that take into account the application's domain as well as the data capture the required behavior more precisely and can be very useful. Such systems look at the semantics of the application, and cannot normally be expressed algorithmically as general models. Thus, if we take formal models and enrich them with application semantics, our definition of security becomes more complex but our systems become more effective.

5 What We Should Learn

There are two main conclusions that we would like to draw as a result of this survey. The first conclusion, perhaps already known by many professionals in the field, is that there is no silver bullet solution to all security problems. We witness a growing number of solutions that specialize on narrower and more specific systems, in contrast to older, more general models. As security has become a universal concern that applies equally to applications from drastically differing domains, the "one size fits all" approach does not work. For instance, the optimistic security and strike back approaches described above have contradicting goals and assumptions and are intended to be used in different environments. Other models (e.g., immune-system based and functional approaches) can coexist, and we might utilize one or more of the compatible solutions for securing our system. Furthermore, all of the models given in this work possess useful properties only when they are used in the intended context and may become useless or even harmful in other settings.

The number of methods available to manage security issues will continue to grow. We will see more customized solutions that are sufficiently secure only in the domain in question, and solutions that take into account semantics and the context of the application. The main challenge now is to make an educated choice of a solution or multiple solutions that reflect our security needs and which will give the best possible results for our job function.

The second conclusion, which again is not new, is that any model that we employ should interoperate with the existing infrastructure; at the very least during the transition period. Meadows [34] describes three different ways of managing computer security in relation to the existing security mechanisms. They are the Live With It, Replace It, and Extend It paradigms.

The Live With It paradigm uses the existing infrastructure without any modifications to produce quick, ad hoc fixes to improve security. Both the underlying system and the environment are taken

as given, and we apply patches to the system to provide coarse-grained, imperfect security. For example, we use virus checkers and firewalls instead of designing more effective solutions that will result in more secure systems (e.g., disallowing modifications to the system to make it invulnerable to viruses). This approach can be viewed as a cheap and popular alternative that provides a reasonable, but not optimal, level of security.

In the Replace It paradigm, we replace the current version of the system or application with a secure version. This approach might seem very attractive because it allows us to design the most secure version of the system possible, but it may be very attractive only in theory. As we attempt to realize such a system, we will run into problems with its users and other applications or systems that now have to be made compatible with our new system. Due to wide spread and interdependence of security solutions and components, the Replace It approach becomes harder to apply, and any solution that we employ should take into consideration the existing infrastructure and the environment.

Lastly, the Extend It paradigm works by either extending the existing infrastructure or replacing small portions of it. The idea is to extend the capabilities of the system to function securely, while keeping the number of components that have to be replaced at the minimum level. This paradigm can be employed in multiple steps or the system can support multilevel security. We can transition to secure solutions through a chain of incremental changes, and need to keep the existing infrastructure in mind, regardless of how brilliant a new solution might seem.

Acknowledgments

We would like to thank Eugene Spafford and Ethan Blanton for their useful comments.

References

[1] B. Lampson, "Protection," Princeton Symposium of Information Science and Systems, pp. 437–443, Mar. 1971.

[2] P. Denning, "Third Generation Computer Systems," Computing Surveys, Vol. 3, Issue 4, pp. 175–216, Dec. 1971.

[3] G. Graham and P. Denning, "Protection – Principles and Practice," AFIPS Spring Joint Computer Conference, Vol. 40, pp. 417–429, 1972.

[4] M. Harrison, W. Ruzzo, and J. Ullman, "Protection in Operating Systems," Communications of the ACM, Vol. 19, Issue 8, pp. 461–471, Aug. 1976.

[5] D. Bell and L. LaPadula, "Secure Computer Systems: Mathematical Foundations," Technical Report MTR–2547, Vol. 1, MITRE Corporation, Mar. 1973.

[6] D. Bell and L. LaPadula, "Secure Computer System: Unified Exposition and Multics Interpretation," Technical Report MTR–2997 Rev. 1, MITRE Corporation, Mar. 1975.

[7] K. Biba, "Integrity Considerations for Secure Computer Systems," Technical Report MTR–3153, MITRE Corporation, Apr. 1977.

[8] R. Sandhu, E. Coyne, H. Feinstein, and C. Youman, "Role-Based Access Control Models," IEEE Computer, Vol. 29(2), pp. 38–47, 1996.

[9] E. Bertino, P. Bonatti, and E. Ferrari, "TRBAC: A Temporal Role-based Access Control Model," ACM Workshop on Role-based Access Control, pp. 21–30, Jul. 2000.

[10] J. Joshi, E. Bertino, U. Latif, and A. Ghafoor, "Generalized Temporal Role Based Access Control Model (GTRBAC) (Part I) Specification and Modeling," Technical Report CERIAS TR 2001–47, Purdue University, 2001.

[11] M. Blaze, J. Feigenbaum, and J. Lacy, "Decentralized Trust Management," IEEE Symposium on Security and Privacy, pp. 164–173, May 1996.

[12] N. Li, W. Winsborough, and J. Mitchell, "Distributed Credential Chain Discovery in Trust Management," ACM Conference on Computer and Communications Security, pp. 156–165, 2001.

[13] W. Winsborough, and N. Li, "Towards Practical Automated Trust Negotiation," IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY 2002), pp. 92–103, Jun. 2002.

[14] W. Winsborough, K. Seamons, and V. Jones, "Automated Trust Negotiation," DARPA Information Survivability Conference and Exposition (DISCEX 2000), Jan. 2000.

[15] J. Holt, R. Bradshaw, K. Seamons, and H. Orman, "Hidden Credentials," ACM Workshop on Privacy in the Electronic Society, Oct. 2003.

[16] N. Li, W. Du, and D. Boneh, "Oblivious Signature-Based Envelope," ACM Symposium on Principles of Distributed Computing (PODC 2003), pp. 182–189, Jul. 2003.

[17] W. Winsborough and N. Li, "Protecting Sensitive Attributes in Automated Trust Negotiation," ACM Workshop on Privacy in the Electronic Society, pp. 92–103, Nov. 2002.

[18] B. Blakley, "The Emperor's Old Armor," ACM New Security Paradigms Workshop (NSPW), pp. 2–16, Sep. 1996.

[19] D. Smetters and R. Grinter, "Moving from the Design of Usable Security Technologies to the Design of Useful Secure Applications," ACM New Security Paradigms Workshop (NSPW), pp. 82–89, Sep. 2002.

[20] D. Boneh and M. Franklin, "Identity-base Encryption from the Weil Pairing," CRYPTO 2001, pp. 213–229, Springer-Verlag, LNCS 2139.

[21] R. Nelson, "Unhelpfulness as a Security Policy or It's About Time," ACM New Security Paradigms Workshop (NSPW), pp. 29–32, Aug. 1995.

[22] H. Lipson and D. Fisher, "Survivability — A New Technical and Business Perspective on Security," ACM New Security Paradigms Workshop (NSPW), pp. 33–39, Sep. 1999.

[23] D. Weirich and M. Sasse, "Pretty Good Persuasion: A First Step towards Effective Password Security in the Real World," ACM New Security Paradigms Workshop (NSPW), pp. 137–143, Sep. 2001.

[24] A. Somayaji, S. Hofmeyr, and S. Forrest, "Principles of a Computer Immune System," ACM New Security Paradigms Workshop (NSPW), pp. 75–82, Sep. 1997.

[25] J. Williams, "Just Sick About Security," ACM New Security Paradigms Workshop (NSPW), pp. 139–146, Sep. 1996.

[26] P. D'haeseleer, S. Forrest, and P. Helman, "An Immunological Approach to Change Detection: Algorithms, Analysis, and Implications," IEEE Symposium on Security and Privacy, pp. 110–119, May 1996.

[27] S. Forrest, S. Hofmeyr, and A. Somayaji, "Computer Immunology," Communications of the ACM, Vol. 40, Issue 10, pp. 88–96, Oct. 1997.

[28] S. Forrest, S. Hofmeyr, A. Somayaji, and T. Longstaff, "A Sense of Self for UNIX Processes," IEEE Symposium on Computer Security and Privacy, pp. 120–128, May 1996.

[29] S. Forrest, A. Perelson, L. Allen, and R. Cherukuri, "Self-nonself Discrimination in a Computer," IEEE Symposium on Research in Security and Privacy, pp. 202–212, May 1994.

[30] J. Kephart, "A Biologically Inspired Immune System for Computers," International Workshop on the Synthesis and Simulation of Living Systems, pp. 130–139, 1994.

[31] D. Povey, "Optimistic Security: A New Access Control Paradigm," ACM New Security Paradigms Workshop (NSPW), pp. 40–45, Sep. 1999.

[32] D. Welch, N. Buchheit, and A. Ruocco, "Strike Back: Offensive Actions in Information Warfare," ACM New Security Paradigms Workshop (NSPW), pp. 47–52, Sep. 1999.

[33] R. Nelson, "What is a Secret – and – What does it have to do with Computer Security?" ACM New Security Paradigms Workshop (NSPW), pp. 74–79, 1994.

[34] C. Meadows, "Three Paradigms in Computer Security," ACM New Security Paradigms Workshop (NSPW), pp. 34–37, Sep. 1997.