Vulnerabilities and Threats in Distributed Systems

Bharat Bhargava and Leszek Lilien

Department of Computer Sciences and Center for Education
and Research in Information Assurance and Security (CERIAS),
Purdue University, West Lafayette, IN 47907, USA
`{bb, llilien}@cs.purdue.edu`

Abstract. We discuss research issues and models for vulnerabilities and threats in distributed computing systems. We present four diverse approaches to reducing system vulnerabilities and threats. They are: using fault tolerance and reliability principles for security, enhancing role-based access control with trust ratings, protecting privacy during data dissemination and collaboration, and applying fraud countermeasures for reducing threats.

1 Introduction

Security vulnerabilities dormant in a distributed system can be intentionally exploited or inadvertently triggered. The threats of exploitation or triggering are only potential, and materialize as an attack or an accident. Efficient elimination and masking of vulnerabilities and threats requires cost-based risk analysis.

Vulnerabilities exist in hardware, networks, operating systems, database systems, and applications. New ones are being discovered every day. Information about identified vulnerabilities and threats can be obtained from the well-known security incident databases, or *metabases*, such as ICAT, CERT, vdb, or CVE, from notification systems such as Cassandra [22], or from other sources of information on security incidents.

After discussing vulnerabilities and threats, this paper presents briefly four different ideas or mechanisms for reducing them:

Applying Reliability and Fault Tolerance Principles to Security Research. Many ideas or algorithms from research in reliability and fault tolerance provide useful analogies to research in security. Examples include disabling quorums to deny access, use of checkpointing for intrusion detection, and adaptability to timing, severity, duration, and extent of attacks.
Using Trust in Role-Based Access Control. Trust is needed for access control in open systems. There are problems with identity-based approaches and use of digital credentials. Ongoing research can produce credible trust ratings for a user based on multiple types of evidence, including credentials, observed behavior, recommendations, and reputations. Trust ratings are used to enhance the role-based access control (RBAC) mechanism. We are building a testbed for experiments to validate the process of trust, and study privacy and fraud.

Privacy-Preserving Data Dissemination. Trust and privacy are closely intertwined in interactions among cooperating entities. Preserving data privacy is essential. Objects can encapsulate privacy policies, owner's preferences, and other metadata along with owner's data. They can include mechanisms such as *apoptosis*—that leads to a clean self-destruction whenever this object feels threatened, and *evaporation*—that allows gradual and adaptive object distortion and erasure in proportion to perceived misuse.

Fraud Countermeasure Mechanisms. Vulnerabilities can be identified via studies of fraud. Fraud can be detected by identifying patterns of deceiving behavior. We identified three types of fraudulent user behavior, and developed schemes to evaluate threats and detect fraud.

2 Vulnerabilities

Modeling Vulnerabilities. A *vulnerability* can be defined as a flaw or weakness in system security procedures, design, implementation, or internal controls. A vulnerability can be accidentally triggered or intentionally exploited, causing security breaches [27].

Modeling vulnerabilities includes analyzing their features, classifying them and building their taxonomies, and providing formalized models. Many diverse models of vulnerabilities in various environments and under varied assumptions are available in the literature. A detailed analysis of four common computer vulnerabilities in [17] identifies their characteristics, the expected policies violated by their exploitation, and the steps needed for the eradication of such vulnerabilities in future software releases. A vulnerability lifecycle model has been applied in [4] to three case studies, which show how systems remain vulnerable long after security fixes. During its lifetime, vulnerability can be in any of the following six states: birth, discovery, disclosure, correction, publicity, and death.

A model-based analysis technique to identify configuration vulnerabilities in distributed systems [23] involves formal specification of desired security properties, an abstract model of the system that captures its security-related behaviors, and verification techniques to check whether the abstract model satisfies the security properties.

Two kinds of vulnerabilities can be distinguished: operational and information-based. The former include an unexpected broken linkage in a distributed database, and the latter include unauthorized access (secrecy/privacy), unauthorized modification (integrity), traffic analysis (inference problem), and Byzantine input [3].

Vulnerabilities do not have to be exhaustively removed since they only create a potential for attack. Feeling threatened by vulnerabilities all the time is not desirable. Vulnerabilities exist due to not only mistakes or omissions, but can be a side effect of a legitimate system feature, as was the case with the `setuid` UNIX command [14]. Some vulnerabilities exist in systems and cause no harm in its life cycle. Some known ones have to be tolerated due to economic or technological limitations. Removal of others may reduce usability. To require passwords not only for logging in, but also for any significant resource request may make it secure but lowers usability. The system design should not let an adversary know vulnerabilities unknown to the system owner.

Fraud Vulnerabilities. A *fraud* can be defined as a deception deliberately practiced in order to secure unfair or unlawful gain [2]. Disclosing confidential information to unauthorized people or unauthorized selling of customer lists to telemarketers constitutes fraud. This shows an overlap of fraud with privacy breaches.

Fraud can make systems more vulnerable to subsequent fraud. This requires protection mechanisms to avoid future damage.

Fraudsters can be classified into two categories: impersonators and swindlers [13]. An *impersonator* is an illegitimate user who steals resources from victims, for instance by taking over their accounts. A *swindler* is, in contrast, a legitimate user who intentionally benefits from the system or other users by deception. For instance, swindlers obtain legitimate telecommunications accounts and use the services without intention to pay the bills.

Fraud involves abuse of trust [12, 29]. A fraudster strives to present himself as a trustworthy individual and friend. In a clear way, the more trust one places in others the more vulnerable one tends to become.

Vulnerability Research Issues. Vulnerabilities, analogously to faults, enable failures and attacks. They could be characterized as flaws in design, implementation, or deployment. The severity of a flaw and its impact on an application need analysis. Qualitative impact may be expressed as a low/medium/high degree of degradation in terms of performance and availability. Quantitative impact is in terms of economic loss, measurable cascade effects, and time needed to recover. It could include quantification of reoccurrences of failures or attacks.

Procedures and methods are needed for efficient extraction of the characteristics and properties of the known vulnerabilities. This is analogous to understanding how faults occur. Tools that search for known vulnerabilities in the metabases have limitations. Security mechanisms that add or modify entries in the metabases can only follow, not anticipate, the steps of an attacker. Characteristics can be learnt from the behavior of the attacker or using ideas such as honeypots.

A comprehensive taxonomy of vulnerabilities for different application areas need be constructed. Medical systems may have critical privacy vulnerabilities, whereas vulnerabilities in defense systems might destroy or distort resources and capabilities. A good taxonomy will facilitate both prevention and elimination of vulnerabilities. A metabase of vulnerabilities reveals characteristics in flaws for preventing not only identical but also similar vulnerabilities. It also contributes to identification of related vulnerabilities, including dangerous synergistic ones. Characterization of and a model for a set of synergistic vulnerabilities can lead to uncovering gang attack threats or incidents. It should be noted that the characteristics for a set are, in general, more than a simple "sum" of individual characteristics.

Formalisms to represent vulnerabilities and their contexts are needed. The challenge is to investigate how vulnerability in one context propagates to another. Different kinds of vulnerabilities might be emphasized in different contexts.

Quantitative lifecycle models for vulnerabilities should be built after a thorough analysis of vulnerabilities for a given type of application or system, exploiting their unique characteristics. In each lifecycle phase, the cumulative system vulnerability should be determined, and the most dangerous or the most common types of vulnerabilities recognized. Knowledge of the degree of system vulnerability, the duration of the lifecycle phases, and the prominent types of vulnerabilities for a given phase will be helpful in protecting the system against these types of vulnerabilities. The best defensive procedures can be adaptively selected from a predefined set.

The lifecycle models should help solving a few problems. First, they should help avoid vulnerabilities in a deployed system most efficiently by discovering and eliminating them at the design and implementation stages. Second, they should facilitate evaluations and measurements of vulnerabilities in system components and subsystems and of the system as a whole at each lifecycle stage. Third, the models would assist in most efficient discovery of vulnerabilities in a deployed system before they are exploited by an attacker or a failure. They would assist in most efficient elimination or masking of these vulnerabilities, e.g. based on principles analogous to fault-tolerance. Alternatively, an attacker can be kept unaware or uncertain of important system parameters by, for example, non-deterministic or deceptive system behavior, increased component diversity, or multiple lines of defense.

Research should provide methods of assessing the impact of vulnerabilities on security in applications and systems. It should create formal descriptions of the impact of vulnerabilities, and develop quantitative vulnerability impact evaluation methods. Resulting ranking will help in risk analysis. Investigators can identify the fundamental design principles and guidelines for dealing with system vulnerabilities at any system lifecycle stage. Based on these principles and guidelines, the best practices for reducing vulnerabilities at different lifecycle stages should be developed. Finally, interactive or fully automatic tools and infrastructures—encouraging or enforcing use of these best practices—at each lifecycle stage should be developed.

Research is also needed on vulnerabilities in security mechanisms themselves, and on vulnerabilities due to non-malicious but threat-enabling uses of information [21].

3 Threats

3.1 Models of Threats

We define *threats* against systems as entities that can intentionally exploit or inadvertently trigger specific system vulnerabilities to cause security breaches [16, 27]. An *attack* is an intentional exploitation of vulnerabilities, and an *accident* is an inadvertent triggering of vulnerabilities. Both materialize threats, changing them from potential to actual.

Threats can be classified according to actions and consequences [26]. Actions can be of the following types: observe, destroy, modify, and emulate threats. Consequences include disclose, execute, misrepresent, and repudiate threats, integrity threats. A threat can be tolerated or eliminated based on the degree of risk acceptable to an application. Threat to human life may require complete elimination. Threat to redundant software or hardware can be tolerated briefly.

Threats can be countered by their avoidance (prevention) or tolerance.

Threat Avoidance. The analogy between fault avoidance in the reliability area [24, 5, 21] and threat avoidance should be considered in the system design. Once the system is deployed, the designers cannot change the basic system structures and mechanisms. The threat avoidance methods, petrified in the system, are effective only against less sophisticated attacks. Executors of the most sophisticated attacks have motivation, resources, and the whole system lifetime to discover its vulnerabilities. Such attacks need to be approached from the threat tolerance side [20], and knowledge of fault avoidance in the reliability area can be leveraged.

Understanding different threat sources is necessary for effective threat avoidance. Different human threats, their motivation and potential attack modes are described in [27]. Attacks can be classified as target-of-opportunity attacks, intermediate attacks, or sophisticated attacks [20].

Several research efforts focus on providing guidelines for better designs that prevent threats. A model for secure protocols is proposed in [15]. Formal models for the analysis of authentication protocols are proposed in [25] and in our paper [10]. Security models for statistical databases useful to prevent data disclosures are discussed in [1], and a detailed comparative analysis of the most promising methods for protecting dynamic-online statistical databases is presented there.

Threat Tolerance. Fault-tolerant schemes are neither concerned with each individual failure nor spend all resources in dealing with them. Transient and non-catastrophic errors and failures are ignored if this can benefit the system. In the same way, we need to conduct research on using a form of intrusion tolerance for dealing with lesser security breaches, which are common in daily activities. Applying the fault tolerance approach to security attacks on database systems [3], we can list the following phases: attack avoidance (a.k.a. prevention), attack detection, damage confinement, damage assessment, reconfiguration, repair, fault treatment to prevent a recurrence of similar attacks, and continuation of service.

Fraud Threat Detection for Threat Tolerance. Fraud threats can be viewed as a special category of general security threats, and as the first step in some threat tolerant solutions (majority voting is an example of threat tolerance without threat detection). Fraud detection systems are widely used in telecommunication, online transactions, computer and network security, and insurance. Effective fraud detection uses both fraud rules and pattern analysis. Due to the skewed distribution of fraud occurrences, one challenge in fraud detection is a very high false alarm rate.

3.2 Fraud Threats

Fraud threats can be viewed as a special category of general security threats that should be analyzed considering salient features of fraud [9]. It should be noted that fraud often occurs as a malicious opportunistic reaction, triggered by a careless action. Threat analysis should also consider that fraud escalation seems to be a natural phenomenon. Gang fraud can be especially damaging since gang fraudsters can cooperate in misdirecting suspicion on others.

Individuals or gangs planning fraud thrive in an environment with fuzzy assignment of responsibilities between participating entities, be they human or artificial [9].

Very powerful fraudsters might be able to create environments that facilitate fraud that they plan. Examples include CEO's involved in insider trading.

3.3 Threat Research Issues

Since threats are context-dependent, an analysis of threats already present in the security incident metabases has to start with identifying threats relevant for the context. The analysis needs to find salient features of these threats, as well as indirect associations between threats—also via their links to related vulnerabilities. Next, a threat taxonomy, specialized for the considered context, should be defined.

Formal models of threats, including their context-dependent aspects, are needed. Quantifying the notion of a threat calls for measures to determine threat levels. Avoiding/tolerating threats via unpredictability or non-determinism should be tried.

The formal qualitative and quantitative models—such as a lifecycle threat model—can provide a solid basis for detecting known and discovering unknown threats, and for establishing threat measures. Since threat analysis is strongly linked to the analysis of vulnerabilities, this should result in identifying characteristic features of related vulnerabilities that link them to specific threats. Similarly, one can investigate the links from threats to vulnerabilities. The results of this reverse link analysis may necessitate correcting our vulnerability analysis models and methods.

Development of quantitative threat models can use analogies to the reliability models. An example is a Markov chain model to compute security measures. Two variables *time* and *effort* can be used to rate different threats or attacks. By investigating the nature and properties of attacks, threats, and vulnerabilities, one can formulate the distribution of their random behavior. The security measure named the Mean Effort To security Failure (METF), which is analogous to the Mean Time To Failure (MTTF) reliability measure, could be used. New security measures can be introduced, starting with an evaluation of the suitability of two measures, namely the Mean Time To Patch and Mean Effort To Patch. They are analogous to the Mean Time To Repair (MTTR) reliability measure, and the METF security measure.

An evaluation a specific threat impact can start with the relevant threat properties, such as direct damage, indirect damage, recovery cost, prevention overhead, and interaction with other threats and defensive mechanisms.

Research must include inventing algorithms, methods, and design guidelines to reduce the number and the severity of threats. Injection of unpredictability or uncertainty may increase system security. As an example, one can enhance data transfer security in a distributed system by sending portions of critical data through different routes. Research is also needed on threats to security mechanisms themselves.

Finally, since threat detection is needed for threat tolerance, it should be studied. This includes investigation of fraud threat detection for fraud threat tolerance.

4 Mechanism to Reduce Vulnerabilities and Threats

4.1 Applying Reliability and Fault Tolerance Principles to Security Research

We have been conducting research in reliable distributed systems for a very long time. We have worked on the development of concepts such as consistency, atomicity, durability, availability, rollback, checkpoints, adaptability, etc. [8, 10].

We perceive that the ideas, concepts, or algorithms known from reliability area can have analogies in the security area. We need to apply the science and engineering of reliability research to the research in security and vice versa [6].

The analogies start with basic notions used in security and reliability. Vulnerability corresponds to a fault, a threat corresponds to an error, and a security breach corresponds to a failure/crash [6, 7].

We perceive an analogy between fault tolerance and threat tolerance. The approaches to handling a threat are: *threat disregarding* (ignores a potential threat), *threat avoidance* (avoids a threat by eliminating it, its cause, or its consequences), and *threat tolerance* (gracefully adapts to threats that have materialized) [27].

The analogy between the notion of time for accidental failures and the notion of expended effort for intentional security breaches can be exploited [18]. The effort-to-breach distribution of security is analogous to time-to-failure distribution of reliability. There are differences between seemingly identical notions in reliability and security areas, such as the notion of system boundaries—narrower for reliability and more open for security. Further, reliability analogies are not helpful in some situations, including the instance of intentional breaches arising from intentional malicious faults, and the instance when expenditure of effort is instantaneous. In this case, analogy to time in the area of reliability is meaningless, due to the sequential nature of time. The security function $R(e)$, analogous to the reliability function, can be defined to address some quantitative aspects of operational security.

The following examples of solutions illustrate reliability-security analogies. To increase reliability in distributed systems, a quorum of replicas can be formed in the presence of failures. To make systems secure against unauthorized access, one can use the reverse strategy of making it difficult to form quorums. Research on checkpointing can be applied to intrusion detection. The checkpoints ensure that the systems can be brought back to a secure status. To deal with failures, we build systems that are fault tolerant. We must build systems *attack tolerant* to security attacks. We need to deal with common and less severe security violations as we have learned to deal with every-day and relatively benign reliability failures.

4.2 Using Trust in Role-Based Access Control

The traditional, identity-based approaches to access control are inadequate or even inapplicable to open computing, including Internet-based computing [28]. In addition, the common user authorization approach of granting access privileges to users based solely on user's ownership of digital credentials (evidence), presented directly to the system, has its share of problems. First of all, holding credentials does not certify that the user will not carry out harmful actions [12].

Authorization based on both credentials and trust is more credible than one based on credentials alone, since it makes access control adaptable to users' behavior. This is the reason why we included trust in access control mechanisms in open computing. Existing computational trust management models can be classified as authorization-based or reputation-based. Our design integrates them into one framework.

In our model of trust [12], we have incorporated comprehensive aspects of trust in social systems and computer science applications. One challenge was to select carefully all and only useful trust aspects needed for our system design in a way preventing adverse affects on the flexibility or performance.

We developed algorithms for automating evaluation of trust, or inference of trust. They produce trust ratings for a user based on: (a) dynamic, continuously updated system's own view of user's behavior in interactions with the system, (b) system's own evidence records, (c) evidence records obtained from "foreign" reputation servers, and (d) system security policies. It is important to note that in producing the trust ratings the algorithm considers credibility of the evidence provider.

Good trust inference algorithms needs to accommodate multiple types of evidence. They should be adaptive, and able to tolerate uncertainty, incompleteness or inaccuracy of evidence (especially in case of subjective evidence). Before the algorithm is able to infer trust for a specific application, available and acceptable evidence must be identified. Examples of pieces of evidence include credentials, observed user behavior, recommendations, and reputations. The credibility, availability, and volatility of different types of evidences differ, and they are all affected by societal value, privacy concerns, relevant legislation, and other factors.

The capability to use trust ratings for users was applied for enhancing the well-known role-based access control (RBAC) mechanism. Trust management is performed in this system by a *trust-enhanced role-mapping (TERM)* server, which interacts with RBAC and a reputation server in the process of user authorization.

TERM uses two kinds of evidence for producing trust ratings: (a) direct, first-hand experiences reported to TERM by RBAC, and (b) recommendations of users about others users. The TERM server does not accept recommendations at a face value but assigns to them its credibility rating. TERM server interacts with a reputation server, which is a dynamic trust information repository, and evaluates reputation—based on trust information—by using algorithms specified by the TERM server. We have built a testbed prototype system, named Trust Enhanced Role Assignment (TERA), for experiments verifying the system's process of producing trust ratings for its users, and studying trust, privacy, and fraud.

4.3 Privacy-Preserving Data Dissemination

Trust and privacy are closely intertwined. For any collaboration—or even any interaction—a level of trust must be established. Even just perceived threats to users' privacy by a collaborator may result in substantial lowering of trust. This could result in rejection of collaboration between prospective partners, a loss to all of them. Therefore, protecting and ensuring privacy of sensitive information are necessary components of mechanisms for reducing vulnerabilities and threats.

We briefly sketch our approach [11]. A *guardian* is either the original owner, or a subsequent stakeholder of sensitive data. A guardian may pass private data to another guardian in a data dissemination chain (actually, a cyclic graph). The risk of privacy violations grows with the chain length and milieu fallibility and hostility.

Traditionally, owner's privacy preferences or policies are *not* transmitted due to neglect or failure. If a privacy policy is not included with data, even an honest receiving guardian is unable to honor them. A simple solution is encapsulation of policies and other metadata including owner's privacy preferences with owner's sensitive data and ensuring that owner's metadata are never decoupled from his data.

Suppose that a customer "deposits" his data in a bank. The bank immediately encapsulates data within an atomic private object, which includes private metadata with customer's privacy preferences. Obviously, transmitting complete metadata is inefficient. They are extensive, describing all foreseeable aspects of data privacy that can be needed to address privacy issues under any circumstances. For efficiency reasons, based on the application semantics, only some metadata are carried along.

With atomic self-descriptive objects, there is no way that a sending guardian can transmit to a receiving guardian an incomplete object. This solution solves the problem for friendly environments.

The solution must be extended to embrace hostile and unfamiliar environments. In the first step, the extension will involve an atomic *apoptosis*, that is a clean self-destruction, whenever the object feels threatened. A private object is here a binary-state or atomic entity, which can be either intact or safely destroyed. In the second step, we generalize the notion of apoptosis with the idea of *evaporation*. Object's private data are not destroyed all at once but evaporate gradually, adaptively and in proportion to the object's distrust towards its current milieu.

Perfect passing of objects is not always desirable. When data are captured by spyware embedded in browser extensions, owners want to see them distorted once they leave their computer. Owners are often willing to share their data locally, e.g., with colleagues in their lab, but want to prevent any wider dissemination. This suggests that private objects should be evaporating in proportion to their "distance" from their source. Owners generally trust their original guardians more than subsequent and more distant ones. Unauthorized data disclosures become more probable further away. Different context-dependent proximity metrics can be used.

4.4 Fraud Countermeasure Mechanisms

We have concentrated on swindler detection. The major challenge is to react to a suspicious action or cooperation that may lead to a fraud. Three approaches were considered: (1) detecting an entity's activities that deviate from legitimate patterns; (2) constructing state transition graphs for existing fraud scenarios and detecting frauds similar to the known ones; and (3) discovering an entity's intention based on past behavior. An architecture incorporating all three approaches is proposed in [13].

The deceiving intention prediction (DIP) algorithm is the critical element of the architecture. Its role is discovery of deceiving intention of an entity, based on entity's history and current behavior.

We have identifies three types of deceiving user behavior: (a) *uncovered deceiving intentions,* where swindler's trust ratings are stably low and vary in a small range over time, (b) *trapping intentions,* where a swindler first exhibits intentionally blameless behavior to gain trust, and then commits a fraud, and (c) *illusive intentions,* where a swindler exhibits cycles of blameless behavior followed by intervals of fraudulent actions. We see *cycles* of preparation and entrapment in Case (c), in contrast to Case (b) where one preparation interval precedes one entrapment period.

We have experimentally evaluated the DIP algorithm [13] investigating its performance for different types of user behavior, including the deceiving behaviors defined above. Given a user behavior sequence, DIP calculates for it the value of the *DI-confidence indicator*, which is a real number ranging over [0,1] with the higher values indicating higher chances of an illegitimate behavior.

Our experimental results can be summarized as follows [13]:

- For a swindler with uncovered deceiving intentions: Since the probability of fraud is high, the swindler is put under system supervision most of the time. The final trust values are at 0.1, close to the minimum. The DI-confidence is around 0.9.
- For a swindler with trapping intentions: DIP responds very quickly with a drop in trust ratings when a swindler ends preparation and enters the entrapment phase: increasing DI-confidence from 0.22 to 0.76 takes only a sequence of 6 ratings.
- For a swindler with illusive intentions: DI-confidence increases (trust falls) when the swindler ends the preparation phase of a cycle and starts an entrapment. DI-confidence decreases (trust grows) when the swindler ends the entrapment phase and reenters the preparation phase. Still, DIP is able to catch this smart swindler because her DI-confidence eventually increases to about 0.9. This demonstrates that an effort to hide periods of fraudulent activities with periods of good behavior is less and less effective with each repetition of the preparation-entrapment cycle.

5 Conclusions

Investigation of vulnerabilities and threats and devising countermeasures is an important research area with a high potential for practical impact. Our contributions of four different ideas and mechanisms for reducing system vulnerabilities and threats, presented in the paper, show a few of the possible directions for research.

We are using the presented mechanisms in our experimental testbed for investigation of new solutions for security and privacy in distributed systems. (More information is available at: http://raidlab.cs.purdue.edu//NSFtrust//.html.).

Acknowledgements. We are grateful for contributions made by Ms. Anjali Bhargava (fault tolerance and security) and Ms. Yuhui Zhong (trust in RBAC and fraud). Research is supported in part by NSF grants IIS-0209059 and IIS-0242840.

References

1. N.R. Adam and J.C. Wortmann, "Security-Control Methods for Statistical Databases: A Comparative Study," *ACM Computing Surveys*, Vol. 21, No. 4, Dec. 1989.
2. *The American Heritage Dictionary of the English Language, Fourth Edition,* Houghton Mifflin, 2000.
3. P. Ammann, S. Jajodia, and P. Liu, "A Fault Tolerance Approach to Survivability," in *Computer Security, Dependability, and Assurance: From Needs to Solutions*, IEEE Computer Society Press, Los Alamitos, CA, 1999.
4. W.A. Arbaugh, et al., "Windows of Vulnerability: A Case Study Analysis," *IEEE Computer,* pp. 52-59, Vol. 33 (12), Dec. 2000.
5. A. Avizienis, J.C. Laprie, and B. Randell, "Fundamental Concepts of Dependability," *Research Report N01145*, LAAS-CNRS, Apr. 2001.
6. A. Bhargava and B. Bhargava, "Applying fault-tolerance principles to security research," in *Proc. of IEEE Symposium on Reliable Distributed Systems*, New Orleans, Oct. 2001.
7. B. Bhargava, "Security in Mobile Networks," in *NSF Workshop on Context-Aware Mobile Database Management (CAMM),* Brown University, Jan. 2002.
8. B. Bhargava (ed.), *Concurrency Control and Reliability in Distributed Systems*, Van Nostrand Reinhold, 1987.
9. B. Bhargava, "Vulnerabilities and Fraud in Computing Systems," *Proc. Intl. Conf. IPSI*, Sv. Stefan, Serbia and Montenegro, Oct. 2003.
10. B. Bhargava, S. Kamisetty and S. Madria, "Fault-tolerant authentication and group key management in mobile computing," *Intl. Conf. on Internet Comp.*, Las Vegas, June 2000.
11. B. Bhargava and L. Lilien, "Private and Trusted Collaborations," *Proc. Secure Knowledge Management (SKM 2004): A Workshop*, Amherst, NY, Sep. 2004.
12. B. Bhargava and Y. Zhong, "Authorization Based on Evidence and Trust," *Proc. Intl. Conf. on Data Warehousing and Knowledge Discovery DaWaK-2002,* Aix-en-Provence, France, Sep. 2002.
13. B. Bhargava, Y. Zhong, and Y. Lu, "Fraud Formalization and Detection," *Proc. Intl. Conf. on Data Warehousing and Knowledge Discovery DaWaK-2003,* Prague, Czechia, Sep. 2003.
14. M. Dacier, Y. Deswarte, and M. Kaâniche, "Quantitative Assessment of Operational Security: Models and Tools," *Technical Report, LAAS Report 96493*, May 1996.
15. N. Heintze and J.D. Tygar, "A Model for Secure Protocols and Their Compositions," *IEEE Transactions on Software Engineering*, Vol. 22, No. 1, 1996, pp. 16-30.
16. E. Jonsson *et al.*, "On the Functional Relation Between Security and Dependability Impairments," *Proc. 1999 Workshop on New Security Paradigms*, Sep. 1999, pp. 104-111.
17. I. Krsul, E.H. Spafford, and M. Tripunitara, "Computer Vulnerability Analysis," *Technical Report, COAST TR 98-07,* Dept. of Computer Sciences, Purdue University, 1998.
18. B. Littlewood *at al.*, "Towards Operational Measures of Computer Security", *Journal of Computer Security*, Vol. 2, 1993, pp. 211-229.
19. F. Maymir-Ducharme, P.C. Clements, K. Wallnau, and R. W. Krut, "The Unified Information Security Architecture," *Technical Report, CMU/SEI-95-TR-015*, Oct. 1995.
20. N.R. Mead, R.J. Ellison, R.C. Linger, T. Longstaff, and J. McHugh, "Survivable Network Analysis Method," *Tech. Rep. CMU/SEI-2000-TR-013*, Pittsburgh, PA, Sep. 2000.
21. C. Meadows, "Applying the Dependability Paradigm to Computer Security," *Proc. Workshop on New Security Paradigms*, Sep. 1995, pp. 75-81.

22. P.C. Meunier and E.H. Spafford, "Running the free vulnerability notification system Cassandra," *Proc. 14th Annual Computer Security Incident Handling Conference*, Hawaii, Jan. 2002.
23. C. R. Ramakrishnan and R. Sekar, "Model-Based Analysis of Configuration Vulnerabilities," *Proc. Second Intl. Workshop on Verification, Model Checking, and Abstract Interpretation (VMCAI'98)*, Pisa, Italy, 2000.
24. B. Randell, "Dependability—a Unifying Concept," in: *Computer Security, Dependability, and Assurance: From Needs to Solutions*, IEEE Computer Society Press, Los Alamitos, CA, 1999.
25. A.D. Rubin and P. Honeyman, "Formal Methods for the Analysis of Authentication Protocols," Tech. Rep. 93-7, Dept. of Electrical Engineering and Computer Science, University of Michigan, Nov. 1993.
26. G. Song *et al.*, "CERIAS Classic Vulnerability Database User Manual," Technical Report 2000-17, CERIAS, Purdue University, West Lafayette, IN, 2000.
27. G. Stoneburner, A. Goguen, and A. Feringa, "Risk Management Guide for Information Technology Systems," *NIST Special Publication 800-30*, Washington, DC, 2001.
28. M. Winslett *et al.*, "Negotiating trust on the web," *IEEE Internet Computing Spec. Issue on Trust Management*, 6(6), Nov. 2002.
29. Y. Zhong, Y. Lu, and B. Bhargava, "Dynamic Trust Production Based on Interaction Sequence," Tech. Rep. CSD-TR 03-006, Dept. Comp. Sciences, Purdue Univ., Mar.2003.