2009 International Conference on Computational Science and Engineering

# The Insider Threat Security Architecture:

**A framework for an integrated, inseparable, and uninterrupted self-protection mechanism**

Ghassan "Gus" Jabbour
*The Volgenau School of IT & Engineering*
*George Mason University*
*Fairfax, VA 22030, USA*
*Email: gjabbour@gmu.edu*

Daniel A. Menascé
*Department of Computer Science*
*George Mason University*
*Fairfax, VA 22030, USA*
*Email: menasce@gmu.edu*

***Abstract*—Providing an uninterruptable self-protection mechanism that is totally integrated into and inseparable from the computing system that is being protected ensures a complete, affordable, and assured compliance with system security audits. This paper presents the Insider Threat Security Architecture (ITSA) and describes its various components. It presents a security scenario where privileged users can compromise the system that they protect and how that same scenario can be mitigated under the ITSA framework. It also describes the foundational premise that this framework is built upon. It draws the distinction between the proposed approach and the traditional most common approach to providing system protection. It emphasizes the unquestionable importance of making the self-protection mechanism as an integral part of the core components of the system that is being protected.**

***Keywords*-Insider threat; security policy; autonomic systems; database systems**

## I. Introduction

Information is one of the most important assets of an organization. Protecting such an asset is critical for establishing and maintaining a trustworthy relationship between the organization and its clients or user community. However, securing the information asset means that the organization must implement a process that protects and secures its computing infrastructure including networks, applications, databases, storage media, and communication channels. It is the primary responsibility of the guardian of sensitive, mission-critical, or personally identifiable information to protect it against intentional or unintentional compromises. But, despite all the damaging security breaches of the past decade, organizations are still lacking in instituting a comprehensive security framework that ensures a fully integrated and uninterrupted self-protection. As the paradigm of building computing environments continuously shifts towards global, interconnected, internet-based deployment of applications and exchange of business transactions, the impact of an attack becomes much more damaging [1]–[3]. It could wreck havoc in a business' financial standing, damage its reputation, and shake its customer's loyalty. Therefore, securing an organization's information asset becomes not only a technology challenge but also a critical business issue.

The focus of major security efforts in recent years has been on hackers and those perceived as foreign to the organization. But, information security concerns have changed dramatically over the past few years. What used to be the work of amateurs and hobby hackers has now evolved into the work of criminals trying to profit from online fraud and those trying to compromise mission-critical systems. In addition, the source of the most damaging threat to information systems is increasingly becoming the work of insiders [4], i.e., those whom the organization usually considers trustworthy and loyal. Threats to computer security or information systems may originate from either external sources or from within an organization. Detecting any threat is critical to the security of any information management system or the health of any organization; however, insider threats are the hardest to tackle and to mitigate [4]–[10]. It has been a common practice to detect external threats through the use of software tools and technologies such as password enforcement, firewalls, encryption, two-factor authentication, access-control system audits, patch management, network traffic monitoring, and penetration testing. However, internal threats are much harder to address since there is no way to monitor and decipher the insiders intent.

Managing and mitigating security risks in today's globally interconnected computing environment requires creating and implementing a security framework that extends beyond the traditional approach of protecting against the outsider threat while trusting that all employees will always be ethically motivated and do what is in the best interest of their employer. Our research focuses on protecting computing and information-based systems against the insider threat. We argue that while many methods (e.g., whistle blower policies, embedded or hidden cameras, clean desk policies, and software tools) have been used to detect insider threat, this problem remains a main concern, and addressing it is essential to the survivability of any organization or system that deals with sensitive, private, or mission-critical information.

The rest of this paper is organized as follows. Section 2 describes a security scenario in which privileged users can

978-0-7695-3823-5/09 $26.00 © 2009 IEEE
DOI 10.1109/CSE.2009.278

compromise the system they protect and shows how that same scenario can be mitigated under the ITSA framework; this scenario lays the foundation for Section 3, which presents the Insider Threat Security Architecture (ITSA) and describes the major components of ITSA and the guiding principles that run throughout the framework. Section 4 provides a conclusion and future direction for self-protection mechanisms in computing infrastructures.

## II. Problem Description

This section describes the deficiencies in self protection mechanisms in today's computing environments and defines the terms that are critical to understanding the foundation of the ITSA framework. The section then presents an example of how the framework could be used to provide a defense mechanism that surpasses traditional approaches in protecting computer systems, specifically against the insider threat. To begin, we define two important terms: the *security policy* and the *defense mechanism*. The security policy consists of a set of system parameters and their allowable values as defined by the system owner or owners. The defense mechanism consists of the logic, encoded in a set of stored procedures, which protects the system configuration settings from being changed in a way that renders them in non-compliance with the security policy. The defense mechanism is able to query the security policy to verify if changes to the security settings can be allowed. All attempts to change the configuration settings of the system must be mediated by the defense mechanism. The defense mechanism can only be changed by the system owner or a set of system owners that we refer to in this paper as the *Super System Owner*. The security policy itself may or may not reside within the system that is being protected. However, the defense mechanism must be an integral and inseparable part of the core components of the system that is being protected. When the communication between the security policy and the defense mechanism is interrupted, the system protects itself by maintaining its operational state. No changes can be made to the configuration parameters of the system until that communication is restored.

Figures 1 and 2 contrast the traditional approach versus the ITSA approach for protecting a database against the insider threat. Figure 1 shows the traditional approach where the database administrator (DBA) has unfettered access to the system and is able to make any kind of changes without any restrictions or limitations. A DBA can compromise the system in a very subtle way or in a very obvious way. The former case includes changing some configuration parameters that change the behavior of the system in a way that is not very obvious to the system owner or user community. An example would be changing system audit criteria to overlook certain audit conditions. The latter involves deleting system objects or even dropping the entire database.

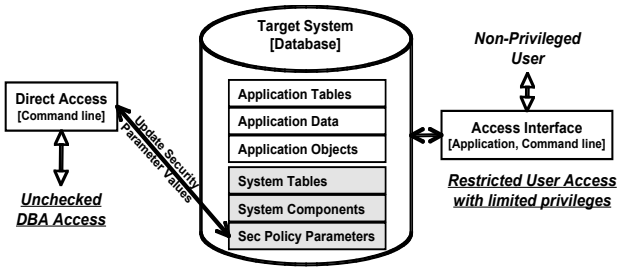


Figure 1. Traditional approach to system protection: no protection against insiders

Figure 2, which is based on the ITSA framework, shows that no privileged user, including the DBA, can make changes to the system without going through the defense mechanism. The actions of any privileged user are verified against a security policy created, maintained, and collectively owned by the Super System Owner. The ITSA framework mandates that more than one system owner be involved in affecting the security policy. Each system owner has a partial password to the system but when combined with the passwords of all other system owners it constitutes a complete and valid password to access and modify the security policy and the system. This ensures that no individual system owner can modify the policy alone. As shown in Fig. 2, any access to the target system by any user must go through the defense mechanism which is embedded inside the target system. The defense mechanism queries the security policy to verify the actions submitted against the system. The security policy may or may not be embedded inside the target system. The security policy can only be modified by the super system owner. On the other hand, the super system owner can access and modify the security policy (after being authenticated) thus affecting the defense mechanism.

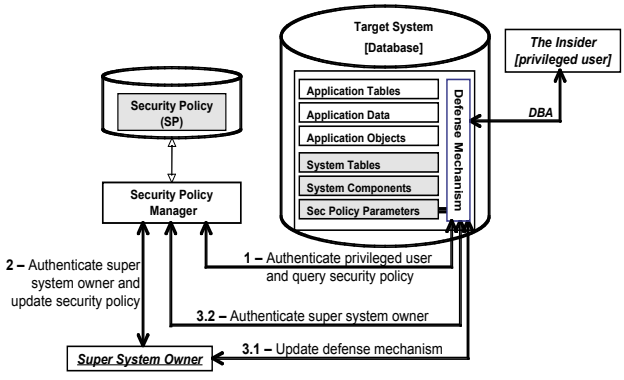


Figure 2. ITSA Enabled protection against the insider threat

## III. Implementation Scenario

This section presents an implementation scenario where the ITSA framework is utilized to enforce security configurations in databases thus enabling them to self-protect. To demonstrate the usability of the proposed framework we first consider a scenario where a DBA can compromise a database in a traditional database management system. A user who has privileged access can normally alter any system configuration in order to change the way the system reacts to system commands. In an Oracle database for example, a DBA can change a wide variety of system parameters in a way that renders the database in non-compliance with secure baseline configuration guidelines as set by the system owners. Among the ones that have high impact on the security of a database are:

- Proper authentication of clients
- Limiting the number of operating system users
- The principle of least privilege
- Revoking execute privileges on PL/SQL packages that establish network connections by the database to any receiving (or waiting) network service
- Revoking execute privileges on PL/SQL packages that allow the database server to request and retrieve data via HTTP, which may permit data to be sent via HTML forms to a malicious web site
- Granting admin privileges to other non-privileged users
- Changing password management parameter settings (e.g., password lifetime and expiration time, password strength, and password reuse)
- Enabling, disabling, and modifying audit criteria which could hinder individual accountability, reconstruction of events, intrusion detection, and problem identification.

We now consider an example that shows how the ITSA framework can protect the system even if the DBA intends to compromise it. We specifically discuss database auditing. Tracking and auditing database activities is essential to any organization that has to comply (and document its compliance) with a wide range of federal laws and regulations such as the Sarbanes-Oxley Act, FISMA, Basel II, HIPAA and other regulations. Database auditing enables the collection and review of database activities and security related events that might have adverse effects on the state of the database. In this example we consider the Fine-Grained Auditing (FGA) capability that is built into Oracle 10g databases. When enabled, FGA records the activities of users based on the conditions set in the FGA policy. It supports privacy and accountability policies in an Oracle database and since auditing occurs inside the database, not in an application, actions are audited regardless of the access methods employed by users (i.e., through tools such as SQL*Plus or applications), allowing for a fail-safe environment. However, that fail-safe environment is obviously not safe when a DBA can act as an insider threat.

Consider a scenario where an insurance company utilizes database auditing to track and consequently monitor payments that are made to its customers. A DBA creates an auditing policy that monitors all claim payments that are $100,000 or more. Any payments made in the amount specified are recorded in the audit trail table of the database and the payment's department may be alerted to verify the action. However, the DBA can change the policy in a very subtle way and in an unchecked or unnoticed manner to allow payments to go through for certain people (e.g., friends and family).

To set up the environment let us create the schema that owns the CLAIMS table and grant it the minimum privileges it needs:

```
SQL>
CREATE USER INSURANCE IDENTIFIED BY
                        INSURE_001
DEFAULT TABLESPACE USERS
TEMPORARY TABLESPACE TEMP
QUOTA UNLIMITED ON USERS;

User created.
SQL> GRANT CREATE SESSION TO INSURANCE;
Grant succeeded.
SQL> GRANT CREATE TABLE TO INSURANCE;
Grant succeeded.
SQL> GRANT DROP ANY TABLE TO INSURANCE;
Grant succeeded.
```

We now create the CLAIMS table:

```
SQL> CONNECT INSURANCE/INSURE_001
Connected.

SQL> CREATE TABLE CLAIMS (
CLAIM_ID NUMBER          NOT NULL,
CUST_ID NUMBER           NOT NULL,
CUST_NAME VARCHAR2(50),
CLAIM_VALUE NUMBER(15,2),
CLAIM_PAYMENT NUMBER,
CLAIM_DATE DATE DEFAULT SYSDATE NOT NULL,
PAY VARCHAR2(1),
CONSTRAINT CLAIM_ID_PK PRIMARY
                     KEY (CLAIM_ID));
Table created.
```

Let us now insert data into the CLAIMS table:

```
INSERT INTO CLAIMS
 (CLAIM_ID, CUST_ID, CUST_NAME,
  CLAIM_PAYMENT, PAY)
  VALUES (101, 1001,
     'JANE DOE', 500, 'Y');
1 row created.
```

```
INSERT INTO CLAIMS
 (CLAIM_ID, CUST_ID, CUST_NAME,
  CLAIM_PAYMENT, PAY)
  VALUES (102, 1002,
    'CANDY STOHR', 1500, 'Y');
1 row created.

INSERT INTO CLAIMS
 (CLAIM_ID, CUST_ID, CUST_NAME,
  CLAIM_PAYMENT, PAY)
  VALUES (103, 1003,
    'JOHN SMITH', 50000, 'Y');
1 row created.

INSERT INTO CLAIMS
 (CLAIM_ID, CUST_ID, CUST_NAME,
  CLAIM_PAYMENT, PAY)
  VALUES (104, 1004,
    'JOHN DOE', 100000, 'N');
1 row created.

INSERT INTO CLAIMS
 (CLAIM_ID, CUST_ID, CUST_NAME,
  CLAIM_PAYMENT, PAY)
  VALUES (105, 1005,
    'TAYLOR HAM', 150000, 'N');
1 row created.

INSERT INTO CLAIMS
 (CLAIM_ID, CUST_ID, CUST_NAME,
  CLAIM_PAYMENT, PAY)
  VALUES (106, 1006,
    'DUSTY ROADS', 200000, 'N');
1 row created.
```

Notice that customer John Doe has a claim payment of $100,000 in the system and that the field PAY is set to N (i.e., no). Now let us create the FGA policy as follows:

```
SQL> connect sys/password as sysdba
Connected.
SQL>
BEGIN
DBMS_FGA.ADD_POLICY (
OBJECT_SCHEMA =>'INSURANCE',
OBJECT_NAME =>'CLAIMS',
POLICY_NAME =>'MONITOR_CLAIMS',
AUDIT_COLUMN => 'CLAIM_VALUE,
CLAIM_PAYMENT, CLAIM_DATE, PAY',
AUDIT_CONDITION =>
     'CLAIM_PAYMENT >= 100000',
STATEMENT_TYPES =>
'SELECT, INSERT, DELETE, UPDATE'
);
END;
PL/SQL procedure successfully completed.
```

From here on, any command that is issued to the database that checks or modifies the CLAIM_PAYMENT column is audited. Suppose the DBA (or any user) issues the following command:

```
SELECT CLAIM_ID, CUST_NAME, CLAIM_PAYMENT,
  PAY
FROM CLAIMS
WHERE CUST_NAME = 'JOHN DOE'
AND CLAIM_PAYMENT >= 100000;
```

The result would be:

```
CLAIM_ID CUST_NAME CLAIM_PAYMENT PAY
-------- --------- ------------- -----
104      JOHN DOE  100000        N
```

To view that this action has been recorded we issue the following command with DBA privileges:

```
SELECT OS_USER, OBJECT_NAME, SQL_TEXT
FROM DBA_FGA_AUDIT_TRAIL;
```

and get the following result:

```
OS_USER OBJECT_NAME SQL_TEXT
------- ----------- --------------------
jabbour CLAIMS      SELECT CLAIM_ID,
-PC\                CUST_NAME,
jabbour             CLAIM_PAYMENT,
                    PAY FROM CLAIMS
                    WHERE CUST_NAME =
                    'JOHN DOE' AND
                    CLAIM_PAYMENT >=
                    100000
```

But the DBA can change all that in one of two ways. The DBA can change the audit criteria of the FGA policy, or change the amount of the payment in the CLAIMS table and then delete all related audit records in the audit trail table. To do the latter the DBA would do the following. First, change the payment amount from $100,000 to $99,999 so that the audit condition 'CLAIM_PAYMENT >= 100000' would not apply. This is done as follows:

```
SQL> UPDATE CLAIMS
      SET CLAIM_PAYMENT = 99999
      WHERE CUST_NAME = 'JOHN DOE'
      AND CLAIM_PAYMENT >= 100000;
1 row updated.
```

Second, the DBA would change the value of the PAY column from N to Y so that the payment may be issued. This is done as follows:

```
SQL> UPDATE CLAIMS SET PAY = 'Y'
     WHERE CUST_NAME = 'JOHN DOE'
     AND CLAIM_PAYMENT = 99999;
```

```
1 row updated.
```

Finally, the DBA would delete any trace of the above statements from the audit trail table as follows. First, view the statements and get the session ID for each related statement as follows:

```
SQL> SELECT SESSION_ID, OS_USER, SQL_TEXT
     FROM DBA_FGA_AUDIT_TRAIL;

SESSION_ID: 82
OS_USER: jabbour-PC\jabbour
SQL_TEXT: SELECT CLAIM_ID, CUST_NAME,
CLAIM_PAYMENT, PAY FROM CLAIMS
WHERE CUST_NAME = 'JOHN DOE'
 AND CLAIM_PAYMENT >= 100000

SESSION_ID: 83
OS_USER: jabbour-PC\jabbour
SQL_TEXT: UPDATE CLAIMS
SET CLAIM_PAYMENT = 99999
WHERE CUST_NAME = 'JOHN DOE'
AND CLAIM_PAYMENT >= 100000
```

Second, delete the statements that have session ID 82 and 83 as follows:

```
SQL> DELETE FROM DBA_FGA_AUDIT_TRAIL
WHERE SESSION_ID = 82;
1 row deleted.

SQL> DELETE FROM DBA_FGA_AUDIT_TRAIL
WHERE SESSION_ID = 83;
1 row deleted.
```

Checking to find out if the statements were deleted:

```
SQL> SELECT SESSION_ID, OS_USER, SQL_TEXT
      FROM DBA_FGA_AUDIT_TRAIL;
no rows selected.
```

This example shows how a privileged user can change critical data in the database that could result in compromising the organization's information system. In the above example, when the organization's next payment batch process runs to issue claim payments, John Doe would get issued a check for $99,999 that has not been cleared by the accounts payable department. Following the ITSA framework however, the DBA would not be able to make any of the changes if they were not allowed by the security policy. The DBA's actions would have failed and would have alerted the system owners of the failed attempt. Under the ITSA framework, the DBA would call a stored procedure named AlterAuditPolicy entering the values of the CLAIM_ID, CUST_NAME, CLAIM_PAYMENT, and PAY columns as such:

```
EXEC AlterAuditPolicy('104', 'JOHN DOE',
                      '99999','Y');
```

The DBA here is requesting to change the CLAIM_PAYMENT from $100,000 to $99,999 and the PAY from N to Y. The AlterAuditPolicy stored procedure would then verify the requested change against the security policy and abort the transaction alerting the system owner of the failed attempt and informing the DBA of the outcome. The stored procedure is embedded in the system components of the database and is not accessible to the DBA since under the ITSA framework all access to the database goes through the defense mechanism and never directly to the database objects.

## IV. The Architecture

To protect against the insider threat, organizations should implement an enterprise security strategy that encompasses a holistic approach based on a framework that extends beyond protecting against traditional hackers and the outsider threat. The framework must be based on the notion that a system must protect itself against any action that contradicts its security policies or compromises its established security configuration settings regardless of who the initiator of that action is. The framework must also be based on the notion that no unchecked access is granted to any trusted entity including the system and database administrators. All actions are verified before deemed safe. According to the proposed framework, the defense mechanism is embedded into the system by the system owners in such a way that allows the system to self-protect against any undesirable outcome. As mentioned before, each system owner has a partial password to the system but when combined with the passwords of all the other system owners it constitutes a complete and valid password to the system.

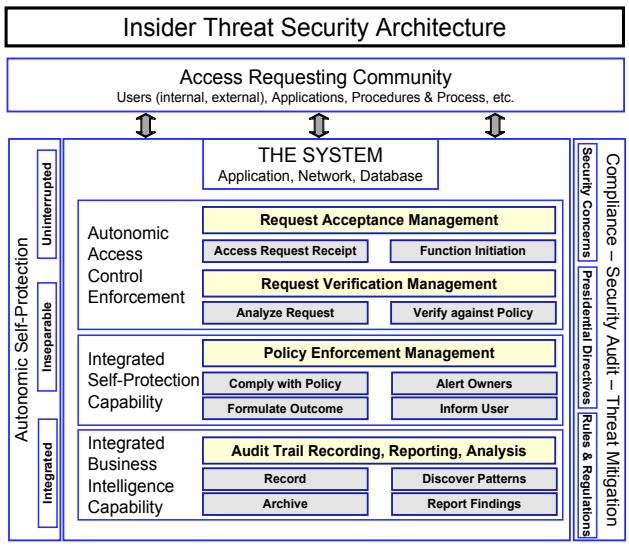


Figure 3. The Insider Threat Security Architecture framework

The Insider Threat Security Architecture (ITSA) framework (see Fig. 3) is built on the foundational notion that any protection mechanism must be totally integrated into, and inseparable from the system that is being protected in such a way that the protection process cannot be interrupted. The major components of the ITSA framework may be categorized into three primary areas: Autonomic Access Control Enforcement, Integrated Self-Protection Capability, and Integrated Business Intelligence Capability. These three areas make up the core defense mechanism of any computing system that is equipped with the ITSA framework. Any communication with an ITSA-equipped computing system by the user community is routed through the defense mechanism before it is processed. In addition, there are three threads that run throughout the framework and act as the guiding principles for implementing self protection: compliance, security audit, and threat mitigation.

### A. Autonomic Control Enforcement

All requests are received by the Autonomic Access Control Enforcement module of the ITSA framework. This module has the built-in logic to receive, verify, accept, or reject any command-based request based on a repository of all acceptable procedural calls for all the integrated built-in functions of the targeted system. The access enforcement process is initiated by the submission of a request through a procedural call to the built-in logic. Request specifics and conditions are submitted in the form of parameter variables that are checked and verified against the security policy. The outcome of the verification process is communicated to the requestor as well as the system owners, and the request is either applied or rejected based on its compliance with the security policy. While this general process of enforcement might be common, the fact that it is totally integrated into and inseparable from the core components of the system that is being protected is what makes it powerful and distinguishable from other protection mechanisms. Figure 4 shows the steps involved in determining the outcome of a request. All requests are received by the Autonomic Access Control Enforcement module of the ITSA framework and are carefully examined for ensuring compliance with security policies. Each action that is supported by the system has a built-in logic interface mechanism that a user can initiate to request certain action or change to the system. These supportable actions (SA) are founded on the basis of predetermined allowable actions and/or changes that the target system can safely accommodate. This is obviously controlled and mandated by the built-in security policy that the system owners put in place.

### B. Integrated Self-Protection Capability

The method for detecting an illegal operation uses a set of rules that are created by the super system owner or under their direct supervision and involvement. The collection of these rules and the processes that enforce them is what makes up the security policy. The policy enforcement mechanism and/or its outcome is transparent to the user in the absence of security violations. Also, the security policy is never modifiable by a privileged user (e.g., system or database administrator) without the collective approval of the super system owner (i.e., all individual system owners).

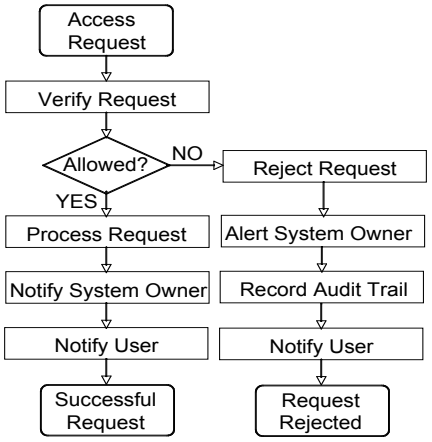


Figure 4. Access request and verification process.

Figure 5 shows the user hierarchy and classification. At the top of the hierarchy, a User is classified as either a Regular User or a System Owner. Within the System Owner class, a user is at least an Individual System Owner. By association then, an Individual System Owner is part of the Super System Owner class. A Regular User is either a Privileged User (e.g., system administrator or a database administrator) or Non-Privileged User.

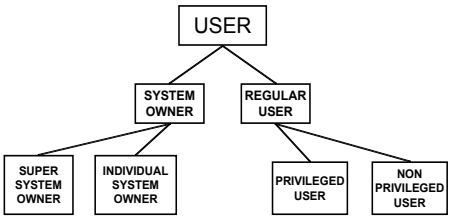


Figure 5. ITSA Framework User Hierarchy.

In summary, the self-protection mechanism ensures that all requests to the system are verified against the security policy. Every request submitted to the system must comply with the security policy for it to be applied. Requests that are intended to affect the security policy itself can only come from the super system owner. This ensures that a system owner alone cannot make changes to the vital security policy of the system. It has to be a collective agreement between the system owners. Based on the nature of the request submitted to the system, one of the following processes is carried out:

- Verify Security Policy: marks a request as safe to execute or reject.
- Process Request: formulates an execution plan and prepares the appropriate system processes or code commands and executes them against the system in order to process a users request.
- Reject Request: ensures that all requests marked as reject are not applied to the system.
- Alert System Owner: alerts the system owners of suspicious requests or activities.
- Record Audit Trail: records all actions submitted to the system in an audit trail repository for use by the integrated business intelligence capability of the ITSA framework.
- Inform User: sends informative messages to the user regarding the status of their requests.

### C. Integrated Business Intelligence Capability

One of the main components of the Integrated Business Intelligence Capability (IBIC) is the Comparison & Pattern Discovery module. The role of this module is to discover patterns, associate actions with users, and alert system owners of risks and potential threats that may be posed by users of the system. The purpose of this module is to discover, identify, and be aware of: 1) people or processes who have access to the system, 2) people or processes who are requesting access to the system, 3) types of requests and/or actions intended to execute against the system, 4) requests that have been rejected and their associated users (requestors), 5) requests that have been approved and successfully executed against the system, and 6) requests that have been approved but have not yet been executed (or failed to successfully execute).

The other two components of IBIC are the Audit Trail Recording module and the Discovery Reporting module. The former is concerned with keeping an archive of the audit trail, while the latter is concerned with reporting any discoveries or useful patterns and/or associations to the system owners. In addition to the three major components (or capabilities) of the ITSA framework, there are three threads that run throughout the framework and act as the guiding principles for implementing self-protection. These are the following.

**Compliance.** The pressure to tighten up security of information systems and organizational data has been growing with the increasing number of regulations over the past years. This is especially applicable to governmental agencies and their computing infrastructure. Regulations such as FISMA [11], Sarbanes-Oxley [12], HIPAA [13], Gramm-Leach-Bliley [14], SB 1386 [15], and HSPD-12 [16] are just some of these regulations that organizations have to continuously comply with. Faced with the requirement to comply, and the reality of being unprepared, many organizations had to implement manual processes in order to pass security checks and audits. But, since these processes are cumbersome and time consuming, the consequence was a depletion in budget and resources in a way that organizations had to divert resources from high stakes projects in order to meet the audit deadlines. As a result, businesses started going after solutions that would be effective in automating the compliance process and meeting the security challenges. What has been missing, however, is the attention to the insider threat. Companies had entrusted their system and database administrators with ensuring compliance. They gave them full control over that task, and in many cases provided minimal or no supervision. The result, as well documented in [1], has been catastrophic. The main contribution of the ITSA framework is to remove the threat posed by insiders by delegating the protection mechanism completely to the system itself. It creates and integrates a strong inseparable self-protection mechanism into the target system, thus removing the potential of insiders from being able to compromise the system.

**Security Audit.** Maintaining system security is an ongoing process that requires a commitment from management to provide the needed resources to secure the organization's computing infrastructure and comply with government imposed rules and regulations. However, while this undertaking may seem costly and unnecessary to many executives, the fact is that the cost of a serious security breach can be very high in two ways. The cost could be specifically high for fixing the problem caused by the breach. It could be even higher when including the indirect cost of the loss of reputation and confidence in the organization by its user community [1].

Many organizations devote significant amount of resources to keeping their computing infrastructure safe, however, many others simply ignore it because they perceive other tasks as being more important to the organizations mission, objective, and/or the bottom line. But, with the rapid advancement in the development of new innovative and powerful information technology comes more responsibility and cautiousness. Such advancements have unquestionably produced tremendous benefits to almost every industry, but at the same time they have "created significant, unprecedented risks to government operations" [17] and many other information technology infrastructures.

A system security audit is the act of assessing the security status of a system and consequently verifying its compliance with the security rules, regulations, and/or policies that apply to its function and role. This audit process is usually feared by organizations who in many cases try to postpone it or even escape it. It is a resource consuming process and the outcome of it might not be something that executives want to face. This is where the ITSA framework becomes valuable for an organization. Building an embedded and integrated self-protection mechanism that utilizes a security policy that is completely owned and affected by the super system owner

alone greatly lessens the burden and the cost of an audit on the organization.

**Threat Mitigation.** Insiders have been shown to be the biggest and most damaging threats to the computing infrastructure and information system of any organization [1], [4], [6], [18]–[20]. Detecting the insider threat is hard to tackle and to mitigate. The ITSA framework guarantees that even the most trusted employee is treated as a potential hacker and is checked against a security policy, which may not be altered even by the insiders themselves.

## V. Conclusion

This paper presented the ITSA framework and made the case for designing, developing, and implementing integrated, inseparable, and uninterrupted self-protection mechanisms into the core components of a computing system infrastructure. We believe that instituting the approach of the ITSA framework provides computing and information systems with a superior self-protection capability that surpasses, in its effectiveness, existing system security tools and approaches. It also relieves organizations from the financial and resource intensive burden that the compliance with stringent security rules and regulations imposes.

A test environment is under development to demonstrate the functionality and effectiveness of the ITSA framework in a commercial database management system environment. This allows us to clearly articulate the direct benefits of the framework to system security. As for future research, we believe that it would be a valid and productive effort to apply and demonstrate the capability of the ITSA framework to other computing environments such as networks, operating systems, and interoperable systems and services.

## References


[1] Jabbour, G. and D.A. Menascé, Stopping the Insider Threat: the case for implementing autonomic defense mechanisms in computing systems. Proc. 2009 Intl. Conf. Information Security and Privacy (ISP-09), 2009.

[2] Jabbour, G. and D.A. Menascé, Policy-Based Enforcement of Database Security Configuration through Autonomic Capabilities. The Fourth Intl. Conf. Autonomic and Autonomous Systems (ICAS 2008), March 16-21, 2008.

[3] Bertino, E. and R. Sandhu, Database Security - Concepts, Approaches, and Challenges. IEEE Tr. Dependable and Secure Computing, 2005. 2(1).

[4] Mallery, J., Hackers Are Not the Biggest Threat to Data: Employees Are. Information Systems Security, www.infosectoday.com,2007.

[5] Carroll, M.D., Information Security: Examining and Managing the Insider Threat. InfoSecCD Conference'06 - ACM, 2006.

[6] Bishop, M. and D.A. Frincke, Combating the Insider Cyber Threat. IEEE Security & Privacy, 2007, pp. 61–64.

[7] Conry-Murray, A., The Threat From Within. www.networkcomputing.com, 2005.

[8] Dai, Y.-S., et al., Autonomic Security and Self-Protection based on Feature-Recognition with Virtual Neurons. Proc. 2nd IEEE Intl. Symp. Dependable and Secure Computing (DASC'06), 2006.

[9] Iyer, A. and H.Q. Ngo, Towards A Theory of Insider Threat Assessment. Proc. 2005 Intl. Conf. Dependable Systems and Networks, 2005: p. 108-117.

[10] Stiennon, R., Your DBA has his/her hand in the till. www.zdnet.com, 2007.

[11] FISMA, Federal Information Security Management Act of 2002. FISMA", Title III of E-Government Act passed by the 107th Congress and signed into law by the president, 44 U.S.C. 3541, et seq., 2002.

[12] Sarbanes-Oxley, The Sarbanes-Oxley Act of 2002. www.soxlaw.com/index.htm, 2002.

[13] 104th-Congress, The Health Insurance Portability and Accountability Act of 1996 (HIPAA) Privacy Rule, www.hipaa.org/, 1996.

[14] Gramm-Leach, The Gramm-Leach Bliley Act. www.ftc.gov/privacy/privacyinitiatives/glbact.html, 1999.

[15] Peace, S., SB 1386. http://info.sen.ca.gov/pub/01-02/bill/sen/sb_1351-1400/sb_1386_bill_20020926_chaptered.html, 2003.

[16] USDA, Homeland Security Presidential Directive 12 (HSPD-12). http://hspd12.usda.gov/, 2004.

[17] USGAO, Management Planning Guide for Information Systems Security Auditing. www.gao.gov/special.pubs/mgmtpln.pdf, 2001.

[18] RSA, The Insider Security Threat in I.T. and Financial Services: Survey Shows Employees' Everyday Behavior Puts Sensitive Business Information at Risk. www.rsa.com/press_release.aspx?id=9703, 2008.

[19] Tan, L., Asia worried about insider threat. ZDNet Asia, www.zdnetasia.com/insight/specialreports/it-priorities/2008/0,3800016949,62047738,00.htm, 2008.

[20] Witting, M., The Biggest Security Threat for 2008 and Beyond: End Users. TechNewsWorld, 2008.