Reasoning about Policy Noncompliance

R. Baird  
University of Tulsa  
800 S Tucker Drive  
Tulsa, OK 74104  
918.631.3283  
robert-baird@utulsa.edu

R. Gamble  
University of Tulsa  
800 S Tucker Drive  
Tulsa, OK 74104  
918.631.2988  
gamble@utulsa.edu

ABSTRACT
In this paper, we introduce a tuple notation for noncompliance that represents certification problems when meeting security controls in distributed, multi-component software systems. The security controls are adopted from NIST SP800-53 and DoD 8500.2 documents. We derive tuples from component policies and interactions, along with the risks associated with violating the security controls. Tuples can be clustered from different perspectives, reasoned about to target the cause and strength of noncompliance. They can also be mapped to specific security concerns and weaknesses in the multi-component architecture.

Categories and Subject Descriptors
D.2.11 [Software Engineering]: Software Architectures – patterns, Service Oriented Architectures (SOA), security

General Terms
Security, Verification.

Keywords
Systems of systems design, security policy interactions

1. INTRODUCTION
Distributed systems, such as systems of systems (SoSs) and service-oriented architectures (SOAs), have come under increasing scrutiny to meet federal security requirements. If studied in a vacuum, components and services can be shown to meet the security policies accompanying their operation. These policies may be segregated into different, yet interdependent, types, such as authorization, data protection, audit, and contingency planning. However, when combined into a larger SoS or SOA, conflicts emerge that are attributed to interactions among entities, environment constraints, and the global system security requirements. Thus, compliance with security certification criteria needs to be more exacting to understand potential violations within distributed system architectures.

A 2005 report from the INFOSEC Research Council [1] outlines the hardest and most critical research problems for generating trustworthy systems. For example, Building Secure Scalable Systems stipulates that high assurance systems should be built from the ground up to ensure security controls are in place to protect against attacks. However, current development strategies leverage commercial and government off the shelf systems (COTS/GOTS) creating new systems that scale well from a development viewpoint, but not from the viewpoint of security. Problems with policy interactions and system configuration issues can cloud system assessment. Attempts to analyze the security of a system at an architectural level, prior to system implementation, have been shown to detect security failures [2]. Our research seeks to target security concerns related to distributed component interactions and express them in a form for compliance assessment against a set of security controls.

As per the Federal Information Systems and Defense Information Systems guidelines, documents such as the NIST SP800-53 [3] and DoD 8500.2 [4] state certification guidelines for security controls that organizations choose to govern system. The security certification process ensures the resulting system complies with those controls. A variety of security taxonomies and ontological specifications [5-6] exist to direct the evaluation of security controls. Researchers currently examine SOA security issues by focusing on mitigation strategies such as determining the existence of a violation by employing models to understand how assets traverse services [7]. Security patterns offer a way to match detected problems with mitigation design strategies. Complex security problems may require the application of multiple security patterns. Previous research maps the text of the controls to risk components (assets, criticality, etc.) to correlate with control categories [8]. Other research creates a UML meta-model to represent component policies in the form of attributes, behaviors, and mechanisms to express controls and find conflicts [9].

Understanding the cause and effect of noncompliance can aid in altering the distributed system design to mitigate the risk of being unable to adequately secure the system. Mitre has developed the CVE and CWE [10] to point to potential resolutions of noted security problems. However, policy related problems must first be understood in terms of functional interaction among components before functional intervention can be prescribed. The objective of this work is to classify the conflicts in terms of their affect on compliance verification against prescribed security controls. We derive noncompliance tuples from meta-knowledge acquired during security control verification experimentation. Once classified, we embed the functional interaction properties into a tuple. The existence and combination of tuples infer other compliance problems, many of which can be mapped to weaknesses that can be mitigated.

2. ISSUES OF NONCOMPLIANCE
Our representation of noncompliance in a multi-component system
We define a noncompliance tuple, \( \rho = ((c, t), sc, p, \sigma) \) where
- \( c \in \text{Scope} \), where the system is affected architecturally
- \( t \in \text{Type} \), how the system is affected in general
- \( sc \subseteq \text{Security control requirements} \), why compliance is an issue
- \( p \subseteq \text{Components} \), what interactions cause problems
- \( \sigma \subseteq \text{Policies} \), which policy types are not being satisfied

We divide the scope into the categories: local, global, configuration, communication, trust, failure, direct, indirect, interference, and dependency. Each scope provides an understanding of the policy’s major affect of noncompliance. Table 1 defines each scope. A scope is paired with an influence. We identify three types of influence on compliance: positional, behavioral, and comparative. Positional influences designate structural or configuration problems. Behavioral shows functional expectations of interactions that cause noncompliance. Comparative expresses policy mismatches among interacting components. This abstraction level yields immediate architectural assessment of the potential security certification issues.

### Table 1. Noncompliance Scope and Type Pairings

<table>
<thead>
<tr>
<th>Scope</th>
<th>Type</th>
<th>Rationale</th>
</tr>
</thead>
<tbody>
<tr>
<td>Local</td>
<td>Positional</td>
<td>Policy expectations local to a single component or service</td>
</tr>
<tr>
<td>Global</td>
<td>Positional</td>
<td>System level policy expectations</td>
</tr>
<tr>
<td>Configuration</td>
<td>Positional</td>
<td>Architectural arrangement expectations of interaction partners at their interfaces</td>
</tr>
<tr>
<td>Communication</td>
<td>Behavioral</td>
<td>Mechanism by which the component exchanges information at an interface</td>
</tr>
<tr>
<td>Trust</td>
<td>Behavioral</td>
<td>If an interaction partner is trusted at an interface port for a particular communication style</td>
</tr>
<tr>
<td>Failure</td>
<td>Behavioral</td>
<td>Policy interactions inhibits component or service functions</td>
</tr>
<tr>
<td>Direct</td>
<td>Comparative</td>
<td>Applicable to a single policy group</td>
</tr>
<tr>
<td>Indirect</td>
<td>Comparative</td>
<td>Applicable to a trace through a communication path among</td>
</tr>
<tr>
<td>Interference</td>
<td>Comparative</td>
<td>A component inhibits the satisfaction of a system policy or function</td>
</tr>
<tr>
<td>Dependency</td>
<td>Comparative</td>
<td>Applicable to multiple policy groups requiring reasoning about dependencies among policy instantiations</td>
</tr>
</tbody>
</table>

In our representation we let Components represent all possible interactions between components or services in the system. Then each element of Components is an ordered pair of exposed interfaces of components or services that interact in the direction of the ordered elements. If \((A, B) \in p\), then A interacts with B.

In this paper we primarily focus on audit security controls from NIST SP800-53 and the DoD 8500.2 since they offer a broad range of policy information. Complying with audit controls cannot be performed in a vacuum, but requires knowledge of data protection, contingency planning and non-repudiation policies. Each control can contain multiple statements that provide additional specification and guidance. Each individual statement can pertain to multiple functional requirements for a distributed system. The set Policies represents the different policy types that govern multi-component systems as prescribed by the organization. Table 2 shows a partial listing of relevant audit security controls in SP 800-53 to distributed systems and their mapping to policy types of authentication (AC), audit and logging (AU), authorization (AZ), contingency planning (CP), and data protection (DP). This table has direct correspondence to the set Policies that populate \( \sigma \) in the noncompliance tuple.

### Table 2. Partial Listing of Control to Policy Mappings

<table>
<thead>
<tr>
<th>Control</th>
<th>AC</th>
<th>AU</th>
<th>AZ</th>
<th>CP</th>
<th>DP</th>
</tr>
</thead>
<tbody>
<tr>
<td>AU-3(1)(2)</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AU-5(1)(2)/(3)(4)</td>
<td>√</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AU-6(1)(4)</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AU-6(7)</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AU-7(1)</td>
<td></td>
<td>√</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AU-8(1)</td>
<td></td>
<td></td>
<td>√</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AU-9(1)(3)</td>
<td></td>
<td></td>
<td></td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>AU-9(2)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>√</td>
</tr>
<tr>
<td>AU-9(4)(a)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>√</td>
</tr>
<tr>
<td>AU-9(4)(b)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>√</td>
</tr>
<tr>
<td>AU-10(1)(2)/(3)(4)(5)</td>
<td>√</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AU-12(1)(2)</td>
<td></td>
<td></td>
<td></td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>AU-13(1)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>√</td>
</tr>
</tbody>
</table>

### 3. REPRESENTING NONCOMPLIANCE

Figure 1 shows a high level view of our process for formulating security controls into noncompliance tuples. The concept of using a tuple space allows for inference and derivation of new tuples that show noncompliance, clustering tuples across various aspects or properties to examine them in groups to indicate where the system may be strongly noncompliant, and mapping certain tuples to system weaknesses as indicated by the CWE.

![Figure 1. Noncompliance Representation Process](image)

We start the process through the analysis of compliance statements from security control documentation. NIST SP800-53 AU-12(1) states “the information system compiles audit records...”
(from components) ... into a system-wide audit trail that is time correlated ...”. Following the process in Figure 1, we state the following functional problems for multi-component system assessment and their architectural interactions.

**s1:** A component does not generate audit records
**s2:** A component lacks functionality to transfer audit records
**s3:** Incorrect non-audit policy inhibits audit record transfer
**s4:** No single, central audit record collector forms the audit trail
**s5:** A sent audit record never arrives at the central collector

Figure 2 maps the statements above to their scope and type according to Table 1. For simplicity, we show the most prominent mappings. However, there are secondary mappings which would be represented by distinct tuples. For example, incorrect non-audit policy inhibiting audit record transfer also maps to dependency by definition. Similarly, no single central audit record collector maps to configuration problems.

Figure 2. Connecting Controls to Scope and Type

The security control AU-9(2) states “the information system backs up audit records onto a different system or media than the system being audited” and AU-9(3) stipulates that “the information system uses cryptographic mechanisms to protect the integrity of audit information and audit tools.” Requirements manifested in distributed systems by AU-9(2) and AU-9(3) are stated below. We preface each with their ordered pair (scope, influence).

**t1:** (local, positional) a component in the system does not encrypt audit records
**t2:** (local, positional) a component in the system does not digitally sign audit records
**t3:** (direct, comparative) between two components in the system an encryption incompatibility problem exists
**t4:** (direct, comparative) between two components in the system a digital signature conflict exists
**t5:** (local, positional) the central audit collector does not encrypt the time correlated audit trail
**t6:** (configuration, positional) the central audit collector does not transfer the audit trail to a remote system

**4. ANALYZING TUPLES**

Figure 3 is a multi-component system architecture with components A, B, D, and E. The organization expects compliance with the audit security controls. Summarily, components A, B, and D must communicate audit records according to the functional requirements outlined in the prior section. In addition, there must be a flow of audit records from all components to sink E, making E the central audit collector. Otherwise, the system configuration must be changed to add a central audit collector and communication links for the component transmissions of their individual records. The central audit collector must be responsive to the needs of the audit trail formation and protection.

4.1 Manipulating the Tuple Space

Upon examining the relevant component policies, we generate the following initial tuples. For space, we focus on specific compliance concerns. Assume that B lacks functionality to directly transfer audit records to D. This observation would be represented as:

\[(\text{interference, comparative}, \{\text{AU-12(1)-s2}\}, \{(B, D)\}, \{\text{AU}\})\]

Additionally, assume that no components are configured as the central audit collector. As a global property, individual tuples cannot be formed to express it. Therefore, this tuple is expressed by referring to the system name, W, where W is the set of all ordered pair interactions between components in Figure 3.

\[(\text{global, positional}, \{\text{AU-12(1)-s4}\}, \{W\}, \{\text{AU}\})\]

Together these two tuples infer that B is not the central audit record collector. Therefore, new noncompliance tuples are derived to express that A’s audit records are sent, but do not reach to D or E, because they are stopped by B, given the system configuration description in Figure 3.

\[(\text{indirect, comparative}, \{\text{AU-12(1)-s5}\}, \{(A, D)\}, \{\text{AU}\})\]

\[(\text{indirect, comparative}, \{\text{AU-12(1)-s5}\}, \{(A, E)\}, \{\text{AU}\})\]

We observe that D and E have conflicting encryption standards

\[(\text{direct, comparative}, \{\text{AU-9(3)-s3}\}, \{(D, E)\}, \{\text{DP}\})\]

From this tuple, we infer that the system will not comply with DoD 8500.2 ECCT-2 statement

**u1:** (interference, comparative) data being transmitted are separately encrypted

\[(\text{interference, comparative}, \{\text{ECCT-2-u1}\}, \{(D, E)\}, \{\text{DP}\})\]

We deduce that D cannot transfer audit records to E because of conflicting data protection policy (a non-audit policy).

\[(\text{communication, behavioral}, \{\text{AU-12(1)-s3}\}, \{(D, E)\}, \{\text{AU, DP}\})\]

Note the tuple above introduces two policy types that are affected. Further derivation may include policy type dependencies as well as addition data protection compliance concerns.

**4.2 Finding Weaknesses**

Certain noncompliance tuples link to system security weaknesses as expressed in CWE entries [10]. These entries provide resolution guidance. Weaknesses in the CWE are organized according to a taxonomy of classification with:

- a **class** representing an abstract type
- a **base** which is a more specific type of weakness yet still independent of specific software architectures
• weakness variants that are tied to specific architectures, technologies, or implementations.

Resolution strategies for weaknesses in the CWE typically occur at the variant level of specification where direct actions can be taken to mitigate risk. These actions can be specific configuration directives or the application of software patches for example. Higher level weaknesses at the class or base levels present mitigation strategies at the architecture and design phase of the software development lifecycle. Operational mitigations can be presented if the weakness is detailed enough to allow it.

Although it is continually evolving with respect to weakness entries, the CWE has weaknesses related to the security control statements in Section 3 including: Insufficient Logging (778), Missing Encryption of Sensitive Data (311), Insufficient Verification of Data Authenticity (345), Inadequate Encryption Strength (326), and Improper Authentication (287). Hierarchies exist within the CWE to assist in finding similar weaknesses to a specific violation of compliance. However, care must be taken to select the appropriate depth in the hierarchy of the CWE.

Determining the correct technique to map the tuples to weaknesses requires preknowledge in selecting an appropriately related set of CWE entries. For instance, the extended description of the Insufficient Logging (778) states “When security-critical events are not logged properly, such as a failed login attempt, this can make malicious behavior more difficult to detect and may hinder forensic analysis after an attack succeeds.” This entry indicates there is a direct mapping between the above central audit collector noncompliance tuple and CWE 778, as notated below:

\[
((\text{global}, \text{positional}), [\text{AU-12}(1)\text{-}s4], [W], [\text{AU}]) \Rightarrow \text{CWE}_{778}
\]

The statement of class weaknesses in the CWE, specifically 778, occurs at a high level reasoning about single black box systems. To generate more specific mappings for weaknesses an investigation of different types of noncompliance tuples is required. Knowing that component D cannot transfer audit records to component E due to a conflicting data protection policy (for example due to different encryption algorithms or key sizes), we return to the previous tuple:

\[
((\text{communication}, \text{behavioral}), [\text{AU-12}(1)\text{-}s3], ([D,E], [\text{AU}, \text{DP}]))
\]

Following the deduction and inference process outlined earlier in section 4.1 new statements of incompatibility for AU-9(3) can be found which can map towards more specific base level weaknesses:

\[
((\text{communication}, \text{behavioral}), [\text{AU-12}(1)\text{-}s3], ([D,E], [\text{AU}, \text{DP}]))
\]

\[
\Rightarrow \text{CWE}_{311}
\]

Weakness 311 presents several potential mitigation strategies to combat the missing encryption. Many design strategies the weakness lists include concepts such as threat modeling, system compartmentalization, and proper naming conventions for data structures. One particularly useful strategy that the CWE suggests is turning to the FIPS 140-2 certification standard for certifying cryptographic modules. Relationships to other CWE entries such as Cleartext Transmission of Sensitive Information (312) can assist the developer in maintaining a secure system.

Although only a few mappings are presented, the tuple clustering approach and mapping can be expanded to reason about all of the named certification criteria, enabling a system designer to investigate weaknesses and vulnerabilities during system design.

5. Conclusion
The establishment of the tuple space informs the mechanisms for reasoning about system organization, vulnerability mapping, and mitigation strategies that are needed to reduce certification noncompliance problems in a SoS. As rules are formalized, they may require specifying additional tuple types with varying degrees of complexity. For instance, we need to represent the SoS architecture for proper inference about component interaction. As more tuples are mapped to the CWE, mitigation strategies will also be expressed in the space to simulate potential resolution and its impact across the system.

Acknowledgement: This material is based on research sponsored by the Air Force Office of Scientific Research, under agreement number FA-9550-09-1-0409. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon.

6. REFERENCES