Defusing Intrusion Capabilities by Collaborative Anomalous Trust Management

Khalil Abuosba¹, Clemens Martin²
Philadelphia University, University of Ontario Institute of Technology
kh_abuosba@philadelphia.edu.jo, clemens.martin@uoit.ca

Abstract
From a computer security perspective, services provided by distributed information systems may be organized based on their security attributes goals and requirements; these processes and services are categorized as anonymous, registered, encrypted and trusted. In this research, we propose a solution for operational trust assurance problems where vulnerabilities reduction is implicitly observed. Collaborative Anomalous Trust Management (CATM) is a methodology that may be utilized for the purpose of affirming trust between communications endpoints. In conjunction with Trusted Computing Base, Zero Knowledge Protocol, and Layered Trust, CATM is defined. CATM builds its trust credentials based on computing environment variables. Ideally this methodology is suited for Service Oriented Architectures such as web services where service providers and consumers interact at different levels of security requirements. This methodology is best optimized for use as a risk management utility. In this approach vulnerabilities are implicitly reduced, hence intrusion capabilities are defused.

1. Introduction
In distributed systems environments, avoiding potential loss of information security attributes is a major goal in B2B processes. Distributed systems are characterized as autonomous computers that might be built and deployed based on different architectures and platforms; securing these heterogeneous systems is considered to be a major challenge which systems designers and administrators strive to achieve. These systems are characterized as entities that operate in their own independent computing environments which are threatened by remote exploitation of vulnerabilities targeted on them. Our approach aims to assist in the reduction of such exploitations. We view distributed systems as sets of end-to-end computational environments that must be dynamically configured and audited prior to transaction execution. The industry lacks methods of trust assurance that utilize computing environment variables as the trust assurance credentials.

2. Business Processes Organization
Services may be categorized as anonymous, registered, encrypted, or audited processes. Each of these process layers is mapped according to security requirements/goals/attributes; anonymous processes require the functional availability and integrity attributes, while registered processes may require one or more of the functional authorization, authentication, identification, and non-repudiation attributes; on the
other hand, encrypted processes require the functional confidentiality attribute, and finally audited processes require the nonfunctional trust requirement.

3. Operational Environment Trust Decomposition

In order to classify distributed systems as reliable and survivable, systems must be operational and capable of resisting attacks. Operational means systems status reflects the degree [1] to which the system can be expected to remain free of security breaches under particular conditions of operations. Operational environments are classified as either threatening or nonthreatening environments [1]; depending on the intruders’ capabilities and environment profile definitions, the environment state may fluctuate between being threatening or nonthreatening. Operational trust mechanisms shall enforce characteristics of these environments based on operational profile configuration compliance. In Intrusion Detection Systems (IDSs) capabilities are defined as the basic units that describe the access that an attacker has obtained during an intrusion. Based on our Distributed Processes Security Organization Reference Model in Figure 1; we view trust as the top layer of the security decomposition attributes; trust may be defined and built based on properties defined in lower layers depending on process requirements.

3.1 Operational Trust Management

Operational trust may be defined as reliance [2]; reliance is interpreted as the act of relying, or the condition or quality of being reliant; dependence; confidence. Reliance is built on a rational assessment of risks and justifications [2]. We emphasize that our work primarily deals with operational trust management. Due to the fact that transactions within distributed systems occur in heterogeneous environments (usually) of different ownerships, operational trust is achieved by deploying flexible unified mechanisms, autonomous of local control where the mechanism is separated from an access control policy. Our approach is based on profiling the endpoint’s computational environment based on identifying resources that may not compromise any of the system’s security attributes (i.e. building an ideal system profile specified based on minimum computational resources required). “Operational trust is the trust that is required from every person and earned from every entity to accomplish an endeavor. Complex operations involve several entities that require a level of interdependence. Each relationship requires a level of trust in order to complete the entire mission. By sharing knowledge and dividing workload; we become more efficient at accomplishing the mission”[3]. If deployed; operational trust is considered to be an operational requirement where purposes of deployment are usually targeted to achieve a major goal usually as a decision making utility “by discovering ways to make trust assessments more precise, we can make decision-making easier and more correct.” [2]. Trust Management is defined as “A unified approach to specifying and interpreting security policies, credentials, and relationships which allow direct authorization of security-critical actions”[4]. We also focus on a second definition “The activity of collecting, encoding, analyzing and presenting evidence relating to competence, honesty, security or dependability with the purpose of making assessments and decisions regarding trust relationships” [5]. We view operational trust management as a mechanism that must conform to three properties. The first is that it is built based on unified mechanisms [4] where policies, credentials, and trust relations are treated as separate entities encapsulated in a common transportation wrapper leading to the handling of security in a comprehensive, consistent, and largely transparent manner. The second, it supports autonomy; distributed systems, by definition, are autonomous nodes that are connected by a network. Any trust management solution must conform to this property; participating endpoints in a trust management mechanism are usually of different ownership and control. The third it supports separation of concern such as mechanism for verifying credentials are separated from policy. Also, we emphasize that this mechanism must be interoperable in a loose coupling operational environment.

3.2 Profile-based Trust Management

In distributed systems, trust management is classified as Policy-based Trust Management, Reputation-based Trust Management [6], Capability-based Trust Management [6], and Certificate-based Trust Management [7]. In this work we introduce new type of trust management; Profile-based Trust Management is trust utility in which its credentials are built based on the computing environment configuration variables. We view profile-based trust management as an approach that addresses computing environment configuration compliance based on the process of defining a computing profile of identified attributes abstracted from the environment resources lists and logs. An alternative wider classification of such trust is the Knowledge-based trust management which is also based on policies definition and deployment [8]. This method might incorporate other
methods such as policy-based trust management and capability-based trust management approaches in its deployment as policy-based trust is typically involved in access control decisions and capabilities are defined as verification identifiers [6] where access control conditions that are eventually meant to yield a definite decision. We introduce this method in order to resolve the operational communication authorization trust problem in which vulnerabilities reduction is implicitly achieved; reducing the system surface of attack reduces the exposure to some possible vulnerabilities; reduction of the attack surface may be achieved by [9] reducing the number of system actions, removing a known or potential system vulnerability by strengthening the pre- and post-conditions of a system action, eliminate an entire attack class, and reducing number of instances of an attack class. Our approach is based on generating authorization and possibly authentication credentials based on the host environment variables profile in which collaboration occurs for the purpose of fulfilling operational trust between communications endpoints. Profile-based trust management has the advantage to reducing operational risk based on the size of the active preconfigured resources; it is considered to be an enforcement utility that shall provide an assurance vehicle to operational trust. What distinguishes this approach is rather than building the trust credentials based on authentication elements such as public key infrastructure, this approach builds the trust definition based on the profile attributes; these attributes are abstracted from the endpoint environment variables.

4. Operational Configuration Profile (OCP)

Based on classifications of operational environments as threatening or non-threatening, we define an Operational Configuration Profile as a facility to be used for specifying environment boundaries. Strict and restricted boundaries are used to operate in a nonthreatening environment; but loose and un-restricted boundaries are deployed within threatening environments. OCPs are specified based on abstraction of specified operational computing environment resources. Attributes of the OCP may be viewed as environment beliefs. An operational profile is characterized as an abstracted entity of the operational environment derived from the operational environment attributes memberships. The main goal of introducing Operational Configuration Profile is to specify environment boundaries and enforce compliance within an input space of a computing environment. An “input space is the totality of all inputs that might ever be encountered…Seeing security conditionally as a system attribute (informally its ability to resist attacks involving a specified effort environment) seems more natural than seeing it marginally, that is averaging over all the different efforts (environments) that attackers might expend” [10]. We assume that OCPs are not forgeable and immune to insertion and modification attacks. Attacking effort is defined as an indirect measure of a whole range of attributes, including financial cost, elapsed time, experience and ability of attacker, etc. We aim to introduce a simple mechanism that might be deployed as a service within an operational environment possibly to be integrated within the operating system environment rather than being built as a separate application. As a guideline for generating OCPs, we recommend the following:

1. Elements/attributes of the Operational Configuration Profile neither compromise the system itself nor cause exposure to system resources.
2. The OCP elements must not reveal information that may be used by intruders as capabilities.
3. Encrypted-OCPs may be used in secure environments where PKIs are deployed and where the OCP contains information that may be used by intruders as capabilities; however, PKIs must not be a requirement for OCP implementations.
4. Implementations must be simple and do not consume computational resources.
5. For the purpose of lowering remote traffic overheads and addressing problems associated with intermediaries such man-in-the-middle attacks, OCP-Digests may be exchanged between endpoints rather than the entire OCP message.
6. The Proof of compliance tokens are generated from the OCPs by using a hash function and may be combined with an identifier by using a combining function followed by an exchange of messages between endpoints.
7. OCPs must be unified and extensible in order to guarantee interoperability.
8. OCPs must conform to the separation of concern paradigm in away where OCP is separated from the verification algorithm. We classify concerns as basic concerns or special purpose concerns where basic concern projects functional concerns that might be mapped to functional requirements. Special purpose concerns may be grouped as aspects related to management and optimization of computational algorithms and a set of nonfunctional concerns that might be mapped to non-functional requirements. Benefits gained from deployment include high level of abstraction, better and ease of understanding of the functionalities, as well as weak coupling.
9. Lightweight implementations may be deployed based on environment configuration inspection using system calls.
10. The configuration profile may be generated and validated by using a state-detection algorithm. Generation of such a profile is performed while the system is classified as a stable state/property (stable property example is a computation has terminated) [12]. Proof of compliance of a system state (system configuration) may be obtained by recording a system snapshot of the local systems variables defined within an Operational Configuration Profile.
11. In order to achieve confusion/diffusion [13] and compression an ad-hoc hash function may be deployed.
12. Ideally, we view the Operational Configuration Profile as an abstraction of a computing environment of a reflective system. Reflective systems are defined as ones, that are able “to inspect and to adapt their own internal behavior” [14]. We identify the OCP as a configuration profile of a reflective system in which reconfiguring the environment assert modes of operation. Reflective systems are characterized as being able to support architectural refinement including ”the ability to modify network attributes using dynamic plug-ins”[15]; an example of such a system is the Genesis Kernel, these types of kernels are able to automate the creation, deployment and management of networks architectures.

5. Collaborative Anomalous Trust Management (CATM)

Collaborative Anomalous Trust Management is built based on intrusion capabilities diffusion. Proper [16] argues that science should adopt a methodology based on falsification, because no number of experiments can ever prove a theory, but a single experiment can contradict one. Thus, theories should be accepted as scientific only if they show the essential characteristic of falsifiability. From a computer security perspective, services and features provided by distributed information systems may be organized as various types of processes built based on their security attributes, goals, and requirements. These processes are categorized as anonymous, registered, encrypted and/or trusted processes. In this section, we propose a solution to operational trust management and vulnerabilities management. Collaborative Anomalous Trust Management (CATM) is a methodology that may be utilized for the purpose of affirming trust between communications endpoints. In conjunction with a trusted computing base, zero knowledge protocol, a security policy enforcement mechanism, a layered trust credential validation process, and profile definition and encapsulation mechanism, CATM is defined. This methodology considers trust as a cumulative entity in which degrees of trust are asserted based on the process security goals. Ideally this methodology is suited for Service Oriented Architectures such as web services where service providers and consumers interact at different levels of security requirements. We view Collaborative Anomalous Trust Management as a methodology that is deployed within a dynamic computing environment; its modes of operation are loose, strict, or disabled. CATM is composed of four phases. The first phase is the computing base definition phase. The second phase is building and exchanging a valid environment variables identifier (system footprint) using a hashing function. The third phase is validating the identifier. The fourth is interpreting the environment status based on the identifier. We emphasize here that building the environment variables identifiers is based on the Zero Knowledge Protocol [17]. There is no need to switch systems to administrator mode. Systems snap shots can be performed offline and messages may be exchanged asynchronously. Marsh [18] emphasizes “that there is a possibility that we may be able to observe anomalies in trusting behavior which, until now, have not been observed or noted. That this is a possibility rests on the fact that we have a formalism which enables implementations of trusting agents to be developed, thus allowing experimentation and possible refutation”. As per Propper’s observation [16]; science can never prove things to be true, but it can prove them to be false.

6. Operational trust assertion based on profile verification

Operational assurance is achieved based on analysis of systems architecture and systems integrity defined within a Trusted Computing Base (TCP). “Protection would be enhanced if a user could restrict access to only those objects a program requires for its execution,..., Conventional architectures support a single privileged mode of operation. This structure leads to monolithic design; any module needing protection must be part of the single operating system kernel. If, instead, any module could execute within a protected domain, systems could be built as a collection of independent modules extensible by any user.” [3]. The base shall maintain a domain for its own execution. “Operational definitions of trust require a party to make a rational decision based on knowledge of possible rewards for trusting and not trusting. Trust enables higher gains while distrust avoids potential loss. Therefore risk aversion is a critical parameter in
defining trust. In the case of trust on the Internet, operational trust must include both evaluation of the users’ intention and the users' competence,..., an operational approach arguably supports a focus on the types of harms resulting from trust betrayal” [3]; thus it is clear that there is a great benefit gained from operational trust fulfillment. The main goal of our work is to achieve operational trust assertion based on compliance of endpoints operational profiles. Our work's main goal is to authorize services based on assurance of compliance of operational trust between endpoints, any divergence from the norm is interpreted as a threat. We introduce the Asynchronous Computing Environment Profile Unification Method (ACEPUM) as a method for trust assurance. ACEPUM is built based on the CATM guidelines. ACEPUM is built for the purpose of achieving profile compliance. Due to operational cost and requirements constraints of solutions such as Virtual Private Networks and private Electronic Data Interchange (EDI), this method is introduced for the purpose of restricting operational computing environment based on configuration management of systems resources. The Peer-to-Peer (P2P) communication model is characterized by dynamic symmetry: each party exposes a set of comparable functionality and any party can initiate a communication session at any time [19]. Securing web services computing environment involves securing endpoints as well as the communication channel. Current existing protection technologies and mechanisms such as firewalls and information encryption deployments are minimal list for securing any computing environment. However, the standard technologies lack the ability to protect the environment from unknown threats. Regardless of the deployment architecture; web services computing environments are characterized as being multi-tiers end-to-end computational channels.

Figure 2. ACEPUM Collaboration Diagram
On each endpoint of communications reventative measures must be taken at each endpoint. These measures may be defined in security policies. Threats may arise from unpatched software as well as mis-configuring devices and systems. We propose deployment of a new hybrid message based method that acts as an operational trust conveyance utility. Computing resources are audited and computing environment boundaries are minimized to their lowest enabled number of resources prior to services activation. Environments may be defined on an ideal minimum requirements basis. Several modes of operations may be defined dynamically based on the environment profiles configuration definitions. Asynchronous Computing Environment Profile Unification Method is based on generating an environment identifier to be exchanged between services peers which is calculated based on running resources and environment identifier specified by services provider. ACEPUM may be deployed in three different modes of operation; dedicated mode (operates based on minimum functional requirements), strict mode (operates based on a negotiated user preference profile), and loose mode (profile elements are not audited, but only explicitly defined attributes of the trusted computing base are verified for compliance, in implementations, bounded number of threads and handles are examples of such attributes). Figure 2 demonstrates a collaboration diagram of a proposed implementation of ACEPUM where two different endpoints of communications collaborate to achieve an OCP compliance. In this implementation an endpoint of communication verifies that a requester's computing environment is initialized to an Ideal pre-published OPC, based on the OCP's hashed token and a session key; a computing base digest is generated, compared with the verifier's ideal digest, and verified for compliance based on binary decision making operator. We recommend ACEPUM implementations for critical resources and environment identifier specified by services provider. ACEPUM may be deployed in three different modes of operation; dedicated mode (operates based on minimum functional requirements), strict mode (operates based on a negotiated user preference profile), and loose mode (profile elements are not audited, but only explicitly defined attributes of the trusted computing base are verified for compliance, in implementations, bounded number of threads and handles are examples of such attributes). Figure 2 demonstrates a collaboration diagram of a proposed implementation of ACEPUM where two different endpoints of communications collaborate to achieve an OCP compliance. In this implementation an endpoint of communication verifies that a requester's computing environment is initialized to an Ideal pre-published OPC, based on the OCP's hashed token and a session key; a computing base digest is generated, compared with the verifier's ideal digest, and verified for compliance based on binary decision making operator. We recommend ACEPUM implementations for critical applications implemented based on SOA that conform to the properties loose-coupling, interoperability, extensibility, and scalability.

7. Conclusion and Future Work

In this work we have introduced a new trust management method in which vulnerability reduction is implicitly observed leading to defusing intrusions capabilities. The unique side of this approach is that the trust credentials are based on the computing environment variables. In our future work we will address extensively environment definition methods, profiles verification algorithms; we will also be exploring implementing solutions based on autonomic multiagent systems in compliance with the CATM properties. More research ought to be conducted for addressing problems associated with forging profiles.

8. References