Adaptive Trust Negotiation and Access Control

Tatyana Ryutov, Li Zhou, and Clifford Neuman
Information Sciences Institute
University of Southern California
Marina del Rey, CA 90292-6695
{tryutov, zhou, bcn}@isi.edu

Travis Leithead, Kent E. Seamons
Internet Security Research Lab
Brigham Young University
Provo, UT 84602-6576
{tleithea, seamons}@cs.byu.edu

ABSTRACT
Electronic transactions regularly occur between business partners in separate security domains. Trust negotiation is an approach that provides an open authentication and access-control environment for such transactions, but it is vulnerable to malicious attacks leading to denial of service or leakage of sensitive information. This paper introduces an Adaptive Trust Negotiation and Access Control (ATNAC) framework to solve these problems. The framework combines two existing systems, TrustBuilder and GAA-API, to create a system with more flexibility and responsiveness to attack than either system currently provides.

Categories and Subject Descriptors
D.4.6 [**Operating Systems**]: Security and protection – *access controls*, *authentication*. K.6.5 [**Management of Computing and Information Systems**]: Security and protection – *authentication*.

General Terms
Security.

Keywords
Access control, trust negotiation, adaptive systems, denial of service.

1. INTRODUCTION
Electronic business transactions often take place between entities that are strangers to one another. This presents a challenge for establishing trust between service requesters and providers that are not in the same security domain. Buyers need to trust that sellers will provide the services they advertise and will not disclose private or sensitive buyer information. Sellers need to trust that the buyers will pay for services or are not underage for accessing services or purchasing certain goods.

In these situations, the challenges of establishing trust are made more complex because participants conduct business transactions that extend beyond their local security domain. Since online communications provide a nearly anonymous medium, people are more inclined to cheat than in face-to-face interactions. The participants can conduct fraudulent and malicious actions in order to damage a competitor and steal money or sensitive information.

To solve these problems, we developed an **Adaptive Trust Negotiation and Access Control (ATNAC)** framework. The framework is based on two well established systems. The GAA-API provides adaptive access control that captures dynamically changing system security requirements. The TrustBuilder system regulates when and how sensitive information is disclosed to other parties. In this paper we describe the synthesis of these two systems into one access control architecture for electronic business services. This combination extends the capabilities of each system. In particular, the framework allows us to:

- Detect and thwart attacks on electronic business transactions.
- Adapt information disclosure and resource access policies according to a level of suspicion.
- Support cost effective trust negotiation, such that TrustBuilder is invoked only when negotiation is required by access control policies.

The framework adapts the security policies according to the sensitivity of the access request and a suspicion level associated with the requester. When sensitivity of the request is low, the server may allow a customer to purchase goods with weaker proofs of payment (e.g., one-time credit card). This provides better protection for the customer but increases the financial risk of the online store. When transaction sensitivity becomes higher (e.g., a customer attempts to make a large purchase), the server requires more trust-worthy credentials, such as a valid user identity or user credit rating. This provides better protection for the server and guards against potential attackers by requiring additional credentials with stricter access control policies before granting the request.

The ATNAC framework detects malicious activity by examining failure types and behavior patterns of the access control and trust negotiation. If any suspicious event is identified (e.g., a user repeatedly presents forged credit card credentials with incorrect card expiration dates), security policies adapt in real time to dynamically protect the system from potential threats. For

SACMAT'05, June 1–3, 2005, Stockholm, Sweden.

instance, the system can react by denying the access, increasing the suspicion level for the misbehaving user, or banning the user.

Traditional online security measures involve identity-based access controls that assume that service providers and requesters are known to each other. Publicly available access control policies specify which users have access to what resources. Users are typically required to pre-register with a service provider before requesting a service. This can often discourage online commerce.

Parties without pre-existing relationships cannot perform sensitive transactions based on identity. Thus, to conduct business there must be a means of establishing a sufficient level of mutual trust. Since neither party is known to the other, both parties may have sensitive information that they are reluctant to reveal until the other party authenticates to a certain level. Therefore, secure electronic business transactions must support access control policies that regulate not only the granting of resources, but also the disclosure of sensitive user information to strangers.

Trust negotiation (TN) [12, 13, 14] is a recent approach to access control and authentication that enables resource requesters and providers in open systems to establish trust based on attributes other than identity. One approach to trust negotiation is to establish trust through the gradual, iterative, and mutual disclosure of credentials and access control policies.

The focus of this paper is to support adaptive, fine-grained access control and trust negotiation in the presence of malicious service requesters. Malicious requests may come in the form of denial of service (DoS) attacks against trust negotiation using its underlying protocols or attempts to obtain sensitive information. The techniques outlined in this paper may be used to protect requesters from malicious service providers; however, this paper focuses on protecting the server provider—as such, our examples exhibit some degree of asymmetry in the trust placed on the participating parties.

The paper discusses weaknesses and limitations in current access control and trust negotiation systems in section 2, followed by a presentation of ATNAC framework in section 3 that addresses many of these concerns. Section 4 covers the details of the framework's suspicion level and discusses how suspicion levels are adjusted based on feedback from the ATNAC framework. Section 5 illustrates the usage of the framework through an online purchasing example. Related work and conclusions are presented in sections 6 and 7.

2. LIMITATIONS OF ADAPTIVE ACCESS CONTROL AND TRUST NEGOTIATION

The Generic Authorization and Access-control API (GAA-API) [8, 9, 10] is a middleware API that supports fine-grained access control and application level intrusion detection and response. The GAA-API allows dynamic adaptation to network threat conditions communicated by an Intrusion Detection System (IDS). The GAA-API can also detect some intrusions by evaluating access requests and determining whether the requests are allowed and if they represent a threat according to a policy. The GAA-API employs a policy evaluation mechanism extended with the ability to generate real time actions, such as checking a current system threat level, generating audit records and updating firewall rules. However, the API supports neither trust negotiation nor protection of sensitive policies. Therefore, the GAA-API/TrustBuilder integration described in this paper leverages the best features of the two systems in order to support adaptive trust negotiation.

While the GAA-API can be used by a number of different applications[1] with no modification to the API code, this paper focuses on a web server implementation because it is a common environment and illustrates the framework well.

Trust negotiation has been explored by several groups of researchers over the past five years (e.g., [1, 2, 3, 5, 6, 7, 11]). The previous research has primarily focused on the details of performing a single trust negotiation. In this paper, we consider the behavior of TrustBuilder, a trust negotiation system developed jointly by researchers at BYU and UIUC. TrustBuilder is vulnerable to DoS attacks, a threat that has not been previously explored in the context of trust negotiation, as well as sensitive information leaks. Both forms of attack pose a significant threat to a trust negotiation system in operation. Information about the TrustBuilder project is available at http://isrl.cs.byu.edu.

There are several ways that an adversary might launch a DoS attack against a trust negotiation system at the application layer. Attacks could result from initiating a large number of TN sessions with the server, disclosing a very complex policy that the server must evaluate, and disclosing many credentials to the server that are invalid or irrelevant to the negotiation. The net result of such attacks is that the server expends an excessive amount of computational resources on illegitimate requests and is hampered in its ability to service requests from legitimate clients.

Another type of attack that an adversary can launch against trust negotiation is to negotiate with the intent of collecting or inferring sensitive information instead of establishing trust to proceed with a transaction. An attacker might send policies that require the other party to reveal sensitive credentials. This technique is a variant of common "phishing" attacks that could easily exploit trust negotiation systems.

TN systems must be adaptive to detect and thwart such attacks. They must support the monitoring of conditions on which trust evaluation is based, dynamically update those trust ratings, and modify system protection levels as a result. Trust is not static, but changes over time because of provider experience or past interactions with the requestor. Multiple successful past interactions increase trust, whereas suspicious provider or requester behavior (e.g., a large number of failed negotiations) negatively influences perceived trust and increases suspicion.

The novelty of this work lies in the cooperation between the access control and TN systems that allow ATNAC to:

[1] The GAA-API can provide enhanced security for applications with different security requirements. We have integrated the GAA-API with the Apache Web server, SOCKS5, sshd, and FreeS/WAN IPsec for Linux.

- **Support fine-grained adaptive policies that protect the system from a number of attacks by adjusting its behavior based on the perceived suspicion levels and feedback from Intrusion Detection Systems (IDS).** ATNAC makes use of dynamic run-time system suspicion levels to adjust access control and sensitive information release policies; limit the receivable number of credentials and policies, and tighten session timeout, minimum turnaround time, and round-count values.
- **Jointly calculate the system suspicion levels based on the feedback from access control and TN.** Suspicion levels are calculated by examining failure types and behavior patterns at the access control and trust negotiation levels. For example, credentials with improper attributes may refer to misuses of a user's identity. Repeated failures in identity verification may result from attempts at impersonation, and consecutive timeouts may be the result of a DoS attack.
- **Support cost effective TN.** Trust negotiation can be quite expensive in terms of computation, network bandwidth. ATNAC can reduce the overhead in certain circumstances by associating less restrictive policies (which require a fewer number of credentials) with lower suspicion levels. This will result in a higher probability of finding a mutually satisfactory policy and improving the success rate of negotiations. Fine-grained access control can also reduce the overhead by initiating the TN only when it is required by the access control policies. For example, a policy may allow access to a sensitive resource for any requesters connecting from a trusted domain, without evaluating credentials that are subject to trust negotiation. Requests originating from outside of the trusted domain must first invoke a TN process used to elevate the level of trust required before proving possession of those credentials.

3. THE ATNAC FRAMEWORK

To protect their resources, e-business providers maintain the following security entities:

- **Public resources, services and credentials** that are accessible to clients, e.g., database of products and VeriSign certified company credentials.
- **Public security policies** that govern resources, services, operations and credentials and can be freely available to clients.
- **Sensitive resources** that can be accessed online only by a limited number of users. For example, customer lists, marketing strategies, custom computer programs, manufacturing techniques, and business forms.
- **Sensitive services** such as a company's online internal services: e-library, payroll, etc.
- **Sensitive credentials** such as the number of customer complaints filed with the service provider.
- **Sensitive operations** (access rights) such as database queries, etc. Some operations (even those on public resources) that leak information to competitors can be restricted.
- **Sensitive security policies** may include:
  - *Access control policies* that govern sensitive resources, services, and operations.
  - *Credential release policies* that specify which credentials to disclose at a given state of the trust negotiation and the conditions of their release.

Because the requester security entities are just a subset of those used by the provider (requesters may not have service operations), we focused on the provider security entities discussed above.

The ATNAC framework, shown in Figure 1, is based on two separate software components: TrustBuilder and the GAA-API. Access control policies that govern public and sensitive resources, services, operations, and policies are expressed in EACL format [10] and are enforced by the GAA-API. Sensitive security policies are enforced by TrustBuilder, which employs X.509v3 digital certificates and TPL policies [4] as attribute credentials and sensitive security policies, respectively.

We integrated the GAA-API into an Apache web server, replacing original access control (*mod_access*) and authentication (*mod_auth*) modules with calls to the GAA-API. The *check_dir_access* function as also modified. This "glue" code extracts the information about requests from the Apache core modules, initializes the GAA-API, calls the API functions to evaluate policies and, finally, returns an access control decision and status values to the Apache modules.

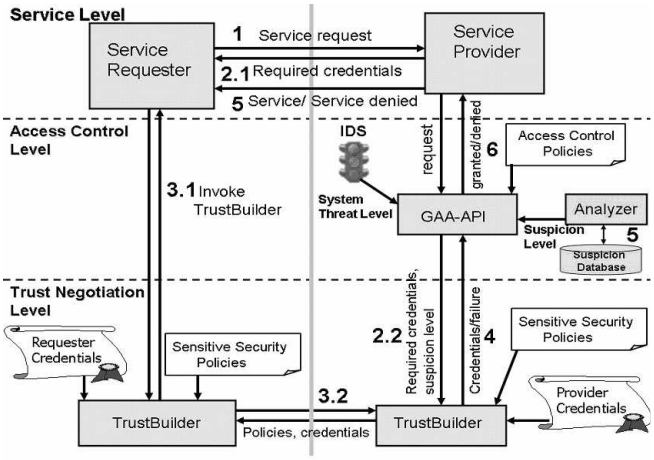

Figure 1. Integrated TrustBuilder/GAA-API framework.

In the GAA-API/TrustBuilder integration, a typical session involving trust negotiation proceeds as follows: The service requester sends a request to the service provider (1). The request is forwarded to the GAA-API. The API first evaluates the access control policies that govern access to the requested service. The policies specify which credentials are necessary to satisfy the request. The set of required credentials depend on the sensitivity

of the request, the service or operation, a system threat level, and the suspicion level (SL) for a particular requester.

The GAA-API uses the credentials to evaluate the policy and either grant or deny the request (6). If the required credentials are not available, or the system threat level is high, the API submits the request for required credentials to the client (2.1). At the same time, the Analyzer calculates the suspicion level for the particular requestor/connection. Next, the GAA-API passes the suspicion level and a set of required credentials to TrustBuilder (2.2).

The requester receives a response requiring further proof (2.1), and initiates trust negotiation (3.1) with the TrustBuilder server. The TrustBuilder server and client then engage in a negotiation to obtain the requested client credentials (3.2). During the negotiation, the TrustBuilder server selects credential release policies with the help of the current suspicion level.

If the TrustBuilder server's policy is satisfied and all the client's credentials are verified successfully, the negotiation succeeds, and the verified credentials are passed to the GAA-API (4). Feedback from TrustBuilder is also used to adjust the current suspicion level. The Analyzer checks the session history in order to identify active attacks or potential threats and calculates the suspicion level for each requester. The Analyzer stores this information in the Suspicion Database using the IP address or username as the index key (5). Increasing the suspicion level tightens the access control and sensitive security policies. The updated suspicion level is used by the GAA-API to process future requests from the same requester. The GAA-API uses TN results (4) to evaluate the access control policy and either grants or denies the requested service (6).

4. SUSPICION LEVEL

The ATNAC framework provides monitoring and policy adaptation at the access control and trust negotiation levels to detect and respond to attacks. Our approach is based on assigning each monitored entity a Suspicion Level (SL). The SL indicates how likely it is that the requester is acting improperly.

ATNAC maintains a separate SL for each requester (host or user) of a service. In order to protect the system against spoofed IP addresses, ATNAC either verifies the host identity (e.g., using host based authentication via X.509 certificates) or verifies the user identity (if the access control policy requires one) of the required attribute in order to determine the SL. The ATNAC stores this information in the *Suspicion Database*.

The SL consists of three components: SL=<S_{DoS}, S_{IL}, S_O>. S_{DoS} indicates a probability of DoS attack on behalf of the requester. S_{IL} is attributed to sensitive information leakage attempts. Finely, S_O indicates other suspicious behavior (e.g., misuses of a user's identity or impersonation attempts). All three values range from 0.0 to 1.0. The Analyzer increases the SL relative to the number of occurrences of suspicious events and decreases the SL when a "positive" event happens (e.g., successful trust negotiation and/or successfully completed business transaction). The value by which the SL is increased depends on the confidence level that the repeated event indicates malicious activity. To calculate a SL, ATNAC uses information reported by TrustBuilder and GAA-API. Alert events thrown by the ATNAC are summarized below:

- *CERTIFICATE_INTEGRITY_ERROR* is reported when TrustBuilder receives an expired, damaged, or revoked certificate or an invalid signature on a certificate. This event may indicate tampering with the certificate or an attempt to use an invalid certificate. This may also be a result of certificate damage during transmission. We attribute this event to S_O, however we moderately increase the suspicion by 0.1 for each reported error of this type.
- *CERTIFICATE_OWNERSHIP_ERROR* occurs when a requester can not prove ownership of the corresponding private key. Possibly, this is an attempt to use a stolen certificate. We consider this an intrusive event and attribute it to S_O by increasing the S_O by 0.25, 0.25, 0.5 on the first, second, and third consecutive errors.
- *ROLE_EXPRESSION_UNSATISFIABLE* occurs when the requester can not satisfy the TrustBuilder policy in the course of a trust negotiation. If this alert event appears a few times in the row, it is likely to be an attempt to probe the server policy that may lead to intrusion or information leakage. ATNAC responds by increasing S_{IL}.
- *POLICY_LIMIT_EXCEEDED* is reported by TrustBuilder when the user sends abnormally large numbers of policies. This behavior may indicate an attempt to gain information about sensitive server credentials. In this case, the Analyzer increases S_{IL}.
- *CREDENTIAL_LIMIT_EXCEEDED* is reported by TrustBuilder when the user sends an abnormally large number of credentials. Since this may be an attempt to exhaust the server resources via public/private key operations, the Analyzer increases S_{DoS}.

In some cases, an attacker can use a simple request to invoke expensive operations on the server that involve, for example, public key cryptographic operations. Thus, a large number of such requests can greatly degrade performance or even disable the server. Similarly, when a client does not respond to a trust negotiation, the server resources allocated for this negotiation persist until a timeout failure occurs. If attackers open many negotiations without giving further responses, it can exhaust the server resources. The TrustBuilder server is especially vulnerable to such attacks, because the trust negotiation process is very expensive. TrustBuilder alone does not have the ability to detect such attacks.

However, such timeout failures may also be a result of accidental network partition. From a single timeout failure it is impossible to distinguish a DoS attack from a network failure. The Analyzer identifies that the requester has frequently generated an unusually high number of similar requests. The user S_{DoS} will increase (slowing down his ability to make rapid requests) or the user will be banned outright. To mitigate the server's burden under such attacks, the ATNAC reduces the server timeout value or denies the user further access.

During a trust negotiation process, the credentials ultimately sent by the client must match the credentials originally requested by the access control policies for this session. If the requester sends irrelevant credentials, the TN process aborts immediately and the S_{DoS} is set to 1. This regulation prevents attackers from sending a great number of irrelevant credentials for the TrustBuilder server to verify, which can degrade the server performance significantly. To counteract such DoS attacks, once the TrustBuilder server identifies an abnormal number of client credentials within one trust negotiation, it generates an alert event, which bans this requester.

If either of the S_{DoS}, S_{IL}, or S_O is greater than 0.9, the ATNAC will ban this requester at the firewall. Other countermeasures include: notifying network servers that are monitoring security relevant events in the system, tightening local policies (e.g., restricting access to local users only or requesting extra credentials), and modifying overall system protection. Examples include: terminating the session, logging the user off the system, disabling the user's account, blocking connections from particular parts of the network, or stopping selected services (e.g., disable SSH connections). These actions would be followed by an alert to the security administrator, who can then assess the situation and take the appropriate corrective actions. This step is important since an automated response to attacks can be used by an intruder impersonating a host or a user in order to stage a DoS attack. These policies are evaluated and enforced by the GAA-API.

If S_{IL} is greater than a policy defined threshold, TrustBuilder will impose stricter sensitive credential release policies asking for more proofs from the requester, or refuse to release sensitive information altogether. If the system SL continues to increase, the GAA-API tightens its access control policies (e.g., by providing service only to users with local accounts and proper authentication, or requiring encryption on all connections).

Before starting a trust negotiation, TrustBuilder first verifies that the GAA-API asked the client for credentials. If not, the trust negotiation request is considered irrelevant to the services to be authorized; TrustBuilder regards it as an invalid trust negotiation request and immediately rejects the connection. With this regulation imposed, the attacker has to compromise the GAA-API before it can gain any access to the more vulnerable TrustBuilder server. Otherwise, the attacker can easily exhaust the server resources by launching excessive trust negotiations.

To protect against some DoS attacks, TrustBuilder adjusts the timeout for each requester by:

$$ServerTimout = DefaultServerTimeout * (1 - S_{DoS})$$

On the GAA-Apache server, we restrict the minimum time span between two consecutive requests from a same user by:

$$MinSpan = 1 \text{ sec} * S_{DoS}$$

5. ONLINE STORE EXAMPLE

To demonstrate the adaptive GAA-API/TrustBuilder architecture, we present an online purchasing example. An online store is selling goods to the public. Customers can access the store web page and submit purchase requests that specify the goods they want to buy. Before the store server grants a purchase order, it requires the customer to present various proofs of payment. Figure 2 illustrates a high level view of server and client credentials, and the server and client trust negotiation policies for this example. Figure 3 shows a table of access control policies evaluated by the GAA-API. The access control policy selected for a given customer depends on two factors:

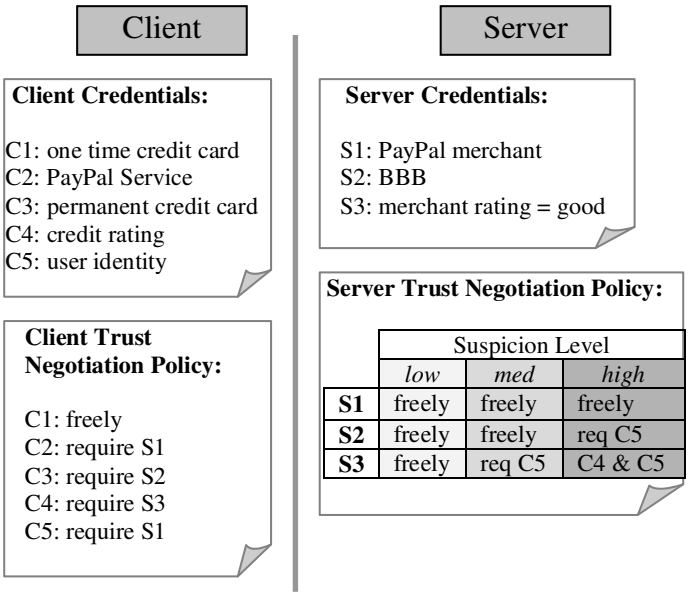

Figure 2: Policies and credentials evaluated by TrustBuilder in the online purchasing example.

1. **Purchase amount.** Larger purchase amounts mean higher transaction sensitivity.
2. **Suspicion level for current customer**, as determined by the Analyzer. If a user has established a good record in his/her past transactions, lower suspicion level is assigned. However, if a user's record is bad, the transaction sensitivity increases. Figure 3 depicts three arbitrary quanta chosen from the range of suspicion levels for the purposes of this example.

When the suspicion level is *low* ($0 < SL \leq 0.33$), the server allows a customer to purchase with a one time credit card or PayPal service. This provides better protection for the customer, but increases the financial risk of the online store. As transaction sensitivity increases, the server requires additional trustworthy credentials with stricter access control policies (e.g., a permanent credit card credential, valid user identity, or even a user credit rating).

The credential requirements described above are defined by an access control policy. The table in Figure 3 shows that the required credentials depend on the purchase amount and the suspicion level quantum. For example, when the suspicion level for a customer is *medium* ($0.33 < SL \leq 0.66$), the server asks for the following credentials depending on the purchase amount of the current transaction:

Access control policy

Required credentials		Amount of purchase		
		<$50	**≥$50, $500≤**	**>$500**
Suspicion Level	***low*** <0.33	C1|C2|C3	C2|C3	C3
	medium ≥0.33 <0.66	C2|C3	C3	C3&C5
	high ≥0.66 <1.0	(C1|C2) &C5	C3&C5	C3&C4 &C5

Figure 3: Access control policy describing credentials required and evaluated by the GAA-API in the online purchasing example.

- If the purchase amount is less than $50, require a PayPal service payment (C2) or a permanent credit card credential (C3).
- If the purchase amount is between $50 and $500, the only credential accepted is a permanent credit card (C3).
- If the purchase amount is greater than $500, require both the permanent credit card (C3) and the user identity (C5).

TrustBuilder is responsible for obtaining and verifying the required credentials. Based on the credential requirements and the suspicion level, TrustBuilder adapts trust negotiation policies, exchanges the credentials, and builds trust with the customer's TrustBuilder. To illustrate this process, assume that a requester wants to make a small purchase (<$50). Also assume that the Analyzer associates the requester with a suspicion level of 0.5 (i.e., *medium*). When the GAA-API receives the purchase request, it consults the access control policy and determines that in order to grant this request, the client has to provide a permanent credit card or PayPal credential (i.e., C2 | C3). The GAA-API invokes TrustBuilder and passes it the suspicion level and the request for the credentials.

The TrustBuilder server begins the first round of negotiation by sending a request to the TrustBuilder client stating that it must present the required credentials. The client evaluates the trust negotiation policy and concludes that in order to release his permanent credit card credential (C3), he must have proof that the online store belongs to the Better Business Bureau (S2); and in order to release his PayPal service credential (C2), he must be sure that the merchant will accept his PayPal service credential (S1).

The first round of negotiation concludes when the TrustBuilder client sends a policy back to the server requiring S1 and S2. The TrustBuilder server determines that the PayPal merchant credential (S1) can be given out to anyone, independent of the current suspicion level. Its Better Business Bureau credential (S2) may also be sent freely, but only to those clients who have a suspicion level less than *high*. Both credentials are then returned to the client in the beginning of the second round of negotiation.

S1 and S2 unlock the client's credentials C2 and C3, respectively. The client chooses to send his permanent credit card (C3) back to the server to conclude the trust negotiation by satisfying the server's original policy requirements. After the client supplies the necessary credential and it is successfully verified at the server, the purchase request is granted. If the client tries to cheat and sends a fake credit card credential that fails verification by TrustBuilder, the request will be denied and this client's suspicion level will be elevated resulting in later application of more restrictive policies. Similarly, if the purchase amount were higher, then the required credentials and policy requirements would also be stricter (e.g., compare the columns in the table of Figure 3).

Therefore, with trust negotiation feedback, security policies adapt in real time and guard against potential attackers by requiring more credential proofs and stricter policy requirements before an attacker can succeed in compromising the system.

6. RELATED WORK

Several systems supporting trust negotiation have been discussed previously in the literature.

Bonatti and Samarati [3] proposed a framework based on a policy language and an interaction model for regulating access to network services. Their trust establishment framework uses logical rules for accessing services and avoiding the unnecessary disclosure of sensitive information. Each party maintains either interaction-specific or persistent state (e.g., credentials and provided services).

Winsborough and Li [12] introduced the Trust Target Graph (TTG) protocol for conducting trust negotiation. A particular emphasis of their work was protection against leaking sensitive information during a trust negotiation. They developed the role-based trust management framework (RT) [6] that consists of a policy language and runtime engine supporting trust negotiation.

PeerTrust [7] is a trust management system that uses a simple and expressive policy language based on distributed logic programs. PeerTrust agents perform automated trust negotiation to obtain access to sensitive resources. The underlying trust negotiation methods are similar to those explained in this paper. PeerTrust policies are evaluated by a Datalog-based logic engine that allows delegation of authority, signed rules, expression of complex conditions, and sensitive policies.

Bertino et al. [2] present X-TNL, an XML-based language for conducting trust negotiation. X-TNL provides a medium to transport information about the negotiating parties called a certificate. A certificate can be either a credential or a declaration. A credential is a list of properties of a negotiating party certified by a Certificate Authority. A declaration contains helpful information (e.g., policies) for the negotiation process.

Cassandra [1], a rule-based policy specification language and evaluation engine, combines features of RT and is closely related

to PeerTrust. Based on DatalogC, Cassandra provides a complete trust management framework with primitives for performing actions (is X permitted to view resource Y), activating and deactivating roles on services (entity A may change role R on a specific service), and requesting credentials. Cassandra contains the primitives necessary for bilateral credential exchange and supports distributed trust negotiation. An implementation of Cassandra was used in a case study for managing access-control to Electron Health Records in England's National Health Service data-spine.

The Secure Dynamically Distributed Datalog (SD3) trust management system [5] has a policy language that allows users to specify high level security policies. SD3 handles policy evaluation and certificate verification. Since the policy language in SD3 is an extension of Datalog, security policies are a set of assumptions and inference rules.

The threats to trust negotiation discussed in this paper are relevant to the trust negotiation systems described in this section. The ATNAC framework presented in this paper is the first example of a system that supports both fine-grained adaptive access control and adaptive trust negotiation. The approaches used to develop the framework have the potential to enhance similar systems that support trust negotiation.

7. CONCLUSIONS AND FUTURE WORK

We have presented the ATNAC framework for protecting sensitive resources in electronic commerce. This framework combines adaptive trust negotiation and fine-grained access control. Trust negotiation is a new paradigm for access control and authentication, which builds trust by accumulating attributes contained in credentials. Adaptive TN responds to requester sensitivity according to a suspicion level by dynamically adjusting its policies, message-accept window, and round count. It also provides feedback in the form of alerts when policies are not satisfied, credential verifications fail, or proof of credential ownership is invalid. Adaptive access control monitors requester activity and calculates suspicion levels based on feedback from the trust negotiation software and an Analyzer. The access control policies are updated to reflect changes in suspicion level and only invoke trust negotiation as needed. Our framework integrates TrustBuilder, an implementation of trust negotiation, with the GAA-API, an adaptive access control API. The GAA-API/TrustBuilder implementation is available at http://gaaapi.sysproject.info**.**

Using our framework, we are able to support the dynamic adjustment of security policies based on suspicion level and general system threat level. The system can detect and protect against some DoS attacks on the service provider. We also guard against sensitive information leak using dynamic policies both in the GAA-API and TrustBuilder. Finally, we support cost-effective trust negotiation by only invoking trust negotiation as directed by the access control policies. Our framework brings adaptive trust negotiation and access-control security services to e-commerce in open systems, eliminating the need for identity registration.

Several areas of future work include the analysis of our method for calculating suspicion levels, tightening the integration of the trust negotiation and access control systems, and identifying alternate methods for detecting release of sensitive information. The current implementation of adaptive trust negotiation uses a simple algorithm for calculating the SL. In the future, we will consider the application of anomaly detection techniques that determine normal user activity and then flag all behavior that falls outside the scope of normal as anomalous. We also plan to analyze the current framework and define abstract interfaces for the communication between GAA-API and TrustBuilder. A well-defined interface will allow integration with other trust negotiation systems.

Determining exactly when sensitive information is being disclosed is a difficult task. One preventative method was presented in the paper: tightening the policy requirements based on a suspicion level and probability of a DoS attack. Another method for detecting release of sensitive information is to maintain a specific context for each trust negotiation. By observing the context surrounding each credential disclosure, irrelevant or sensitive credential releases may be detected, and feedback provided. Many other possible detection techniques are possible including statistical learning and ontology methods.

While this work was specifically targeted as a useful system for electronic commerce applications, it will also be useful in other contexts. Typically, access control in computational grids is accomplished by a combination of identity certificates and local accounts. Resource providers have to maintain a local account for every potential user. This approach does not scale well as the numbers of potential users and resources increase. Users will also need a paradigm that is more flexible and is able to gradually establish trust, so that they do not have to present their most sensitive certificates to another party before first establishing a minimum level of trust in the other party.

8. ACKNOWLEDGEMENTS

This research was supported by funding from DARPA through SSC-SD grant number N66001-01-1-8908 and AFRL grant number F30602-00-2-0595 (DEFCN), the National Science Foundation under grant no. CCR-0325951 and ACI-0325409, as well as prime cooperative agreement no. IIS-0331707, and The Regents of the University of California. The views and conclusions contained herein are those of the authors and should not be interpreted as representing the official policies or endorsement of the funding agencies. Figures and descriptions were provided by the authors and are used with permission

9. REFERENCES

[1] Becker, M. Y. and Sewell, P. Cassandra: distributed access control policies with tunable expressiveness. In *Policies in Distributed Systems and Networks*, June 2004.

[2] Bertino, B., Ferrari, E., and Squicciarini, A.C. Trust-X: A Peer-to-Peer Framework for Trust Establishment. In *IEEE Transactions on Knowledge and Data Engineering, 16,* 7 (July 2004).

[3] Bonatti, P. and Samarati, P. A Unified Framework for Regulating Access and Information Release on the Web. In *Journal of Computer Security, 10,* 3, (2002), 241-271.

[4] Herzberg, A., Mass, Y., Mihaeli, J., Naor, D., and Ravid, Y. Access control meets public key infrastructure, or: Assigning roles to strangers. In *Proceedings of the 2000 IEEE Symposium on Security and Privacy*, May 2000, 2-14.

[5] Jim, T. SD3: A Trust Management System With Certified Evaluation. In *IEEE Symposium on Security and Privacy*, Oakland, CA, May 2001.

[6] Li, N., Mitchell, J., and Winsborough, W. RT: A role-based trust-management framework. In *Proceedings of The Third DARPA Information Survivability Conference and Exposition (DISCEX III)*, April 2003.

[7] Nejdl, W., Olmedilla, D., and Winslett, M. PeerTrust: Automated Trust Negotiation for Peers on the Semantic Web. In *Proceedings of the Workshop on Secure Data Management in a Connected World (SDM '04)* in conjunction with 30th International Conference on Very Large Databases, Aug./Sept. 2004.

[8] Ryutov, T. and Neuman, C. The Specification and Enforcement of Advanced Security Policies. In *Third International Workshop on Policies for Distributed Systems and Networks (POLICY 2002)*, Monterey, CA, June 2002.

[9] Ryutov, T., Neuman, C., and Kim, D. Dynamic Authorization and Intrusion Response in Distributed Systems. In *Proceedings fo the 3rd DARPA Information Survivability Conference and Exposition (DISCEX III)*, Washington, D.C., Apr. 2003.

[10] Ryutov, T., Neuman, C., Kim, D., and Zhou, L. Integrated Access Control and Intrusion Detection for Web Servers. In *IEEE Transactions on Parallel and Distributed Systems, 14,* 9 (Sept. 2003), 841-850.

[11] Skogsrud, H., Benatallah, B., and Casati, F. Model-driven trust negotiation for Web services. *IEEE Internet Computing, 7,* 6 (Nov./Dec. 2003).

[12] Winsborough, W. and Li, N. Towards Practical Automated Trust Negotiation. In *Third International Workshop on Policies for Distributed Systems and Networks (POLICY 2002)*, Monterey, CA, June 2002.

[13] Winsborough, W.H., Seamons, K.E., and Jones, V.E. Automated trust negotiation. In *DARPA Information Survivability Conference and Exposition*, Volume 1, Hilton Head, SC, 2000, 88-102.

[14] Winslett, M., Yu, T., Seamons, K. E., Hess, A., Jacobson, J., Jarvis, R., Smith, B., and Yu, L. Negotiating Trust on the Web. *IEEE Internet Computing,* 6, 6 (Nov./Dec. 2002).