An Access Control Model for Dynamic Client-Side Content

Adam Hess and Kent E. Seamons
Internet Security Research Lab
Department of Computer Science
Brigham Young University
Provo, UT 84602 USA
{ahess,seamons}@cs.byu.edu

ABSTRACT

The focus of access control in client/server environments is on protecting sensitive server resources by determining whether or not a client is authorized to access those resources. The set of resources are usually static, and an access control policy associated with each resource specifies who is authorized to access the resource. In this paper, we turn the traditional client/server access control model on its head, and address how to protect the sensitive content that clients disclose to servers. Since client content is dynamically generated at runtime, the usual approach of associating a policy with the resource (content) a priori does not work. In this paper, we propose an access control model for protecting client-side content that is dynamically generated and disclosed at runtime. Our model identifies sensitive content, maps the sensitive content to an access control policy, and establishes the trustworthiness of the server before disclosing the sensitive content to the server. The model targets open systems, where clients and servers do not have preexisting trust relationships. We have implemented the model within TrustBuilder, an architecture for negotiating trust between strangers based on properties other than identity. The implementation is the first example of *content-triggered trust negotiation* and currently supports access control for sensitive content disclosed by web and email clients.

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.* C.2.2 [**Computer-Communication Networks**]: Network protocols – *applications.*

General Terms

Security, Design, Experimentation.

Keywords

Trust Negotiation, Access Control, Authentication, Credentials

SACMAT'03, June 2-3, 2003, Como, Italy.

1. INTRODUCTION

While performing Internet transactions, users often disclose sensitive information about themselves and their affiliated organizations. The disclosure of this sensitive information can pose serious threats to individual privacy and the confidentiality of business secrets. If communication between the client and server is not confidential, an eavesdropper can access the information. Moreover, the intended recipients of the information may misuse it or fail to protect it appropriately.

To illustrate the problem, consider the following example. Alice is ready to make an online purchase from Bob, a stranger who operates an online store. The purchase request form requires that the purchaser include a valid credit card number and shipping address. Alice deems these pieces of information to be sensitive. If Alice were to send the information over plain HTTP, an eavesdropper could easily access the data, since the underlying communication protocol lacks confidentiality. Using HTTP over SSL (HTTPS) is one way to achieve data integrity and confidentiality. However, this only protects Alice from a malicious man-in-the-middle attacker and not from potential misuse by Bob. To achieve some level of assurance in this regard, Alice needs to first authenticate Bob to a role commensurate with the receipt of this information. For example, Alice may want to verify that Bob is a member of the Better Business Bureau and is authorized to perform credit card transactions. SSL is incapable of this level of authentication, and only provides assurance that a client is connected to the web server on the host specified in the URL. Thus, given the current model, how can Alice trust Bob to handle her sensitive data appropriately?

The disclosure of sensitive information is not limited to web forms alone. As another example, suppose Alice emails Bob her address for shipping information and credit card number for billing purposes. Like HTTP, email has no built-in mechanism for Alice to determine whether Bob is trustworthy.

Other types of sensitive information that Alice may disclose include financial information, health records, Social Security number, or confidential work related data. The disclosure of this information is not limited to web forms. Alice can also send her sensitive information to Bob via email, chat programs, or file transfers. Like SSL, these applications lack the capability to determine the sensitivity of Alice's outgoing network content and Bob's trustworthiness. Without this capability, Alice must complete these tasks manually. In this paper, we explore how to complete these tasks automatically.

In order to equip clients with automated support for protecting client content, two parts to the problem must be solved: authentication and authorization. The client must be able to authenticate the unfamiliar server and then determine whether the server is authorized to receive the sensitive content.

In an open system like the Internet, clients and servers are usually strangers. They have no preexisting relationship and are not in the same security domain. Thus, identity-based authentication is not appropriate in this setting. The identity of the server may be irrelevant to the problem of whether or not the client should trust the server. Instead, attributes other than identity are useful in determining the server's trustworthiness. It is infeasible to use password-based authentication in this scenario, for two reasons: First, the client and server have not met previously to establish a shared secret (i.e., password), and second, it is not practical to require that the server register a username and password with each client that desires to access the server.

The standard model for access control consists of the triple (S, O, M) where S is the set of subjects, O is a set of objects, and M is an access control matrix. The rows and columns of the matrix correspond to distinct subjects and objects, respectively. An entry in the access control matrix, denoted $M[s, o]$ where $s \in S$ and $o \in O$, represents the privileges of subject s to access object o. The columns of M are access control lists for objects in the system. The rows of M are capability lists for the subjects in the system. This access control model works well in a closed system with a known set of subjects and static objects. The environment we are targeting in our research is an open system where subjects are characterized by their role or other attributes. This requires authentication techniques based on attributes other than identity. In addition, the objects that require protection are dynamically generated by the client, and must be recognized on-the-fly so that an appropriate access control policy, known as a disclosure policy, can be dynamically associated with the sensitive object before it is disclosed to the unfamiliar server. For example, a client's credit card number is not statically stored on its local machine for servers to access. Rather, the client dynamically enters this information into web forms, email messages, etc.

The remainder of this paper is organized as follows. In section 2, we discuss trust negotiation, an approach to establishing trust between strangers in open systems. In section 3, we describe our access control model for providing access control for dynamically generated client-side content. In section 4, we describe two implementations of the model in the TrustBuilder architecture that support web and email client applications. Section 5 includes a discussion of related work and section 6 contains conclusions and future work.

2. TRUST NEGOTIATION

A recent approach to establishing trust between strangers is trust negotiation [23], the process of iteratively disclosing digital credentials that describe attributes of the negotiation participants. This approach relies on access control policies that govern access to protected resources by specifying credential combinations that must be submitted to obtain authorization.

Credentials must be verifiable and unforgeable, so we adopt public key certificates, such as X.509v3 certificates, to obtain the necessary guarantees. Digital credentials contain digitally signed assertions by a credential issuer about a credential owner. A credential uses name/value pairs to describe attributes of the owner. Each credential may also contain the public key of the credential owner. The owner can answer challenges and otherwise demonstrate ownership of the credentials. Other approaches are also possible [12].

A digital credential may contain sensitive information whose disclosure must be carefully managed in accordance with an access control policy that specifies which credentials must be received before it can be disclosed. Such policies can govern access to all sensitive resources, including credentials, roles, capabilities, policies, and services.

Access control policies can also contain sensitive information, requiring protection in the form of additional policy specifications. Earlier work in trust negotiation introduced support for sensitive policies using policy graphs [18]. The presence of sensitive policies requires that trust be established gradually. For example, suppose a client begins an interaction with an unfamiliar web server. Before sending a sensitive request for credentials to the server that would reveal information regarding the nature of the client's business, the client may request credentials attesting to how the server handles private information and whether the server conforms to certified security practices. Once the client has established this initial level of trust, the client can continue by sending the sensitive request for further credentials from the server.

The delivery of policies and credentials is defined by a trust negotiation protocol. Trust negotiation strategies control the content of the messages, determining which credentials and policies to disclose, when to disclose them, in what order, and when a negotiation should be terminated.

An example of a trust negotiation is shown in figure 1, illustrating an on-line bookstore that offers discounts to students at accredited universities. Alice requests a student discount, but she has no prior knowledge of the bookstore's requirements for proof of student status since she is a first-time customer. One approach is for the bookstore to transmit a policy to Alice that specifies that she must submit a student ID and a credit card number in order to make an online purchase and receive a student discount. Alice is willing to disclose her credit card number only to a business that is a member of the Better Business Bureau (BBB). In accordance with her policy, her trust negotiation agent discloses her student ID and requests that the bookstore disclose a BBB member credential to her. The bookstore then sends Alice a BBB member credential. Finally, Alice submits a valid digital credit card number and receives the student discount.

Examples of our earlier work in trust negotiation include support for sensitive credentials and access control policies [18], the definition and interoperability of trust negotiation strategies [25], a trust negotiation protocol based on an extension to TLS [8], protecting privacy during trust negotiation [20], an analysis of policy languages for trust negotiation [19], and the development of the TrustBuilder architecture for trust negotiation [24]. Additional information about our trust negotiation research and the TrustBuilder prototype can be found at http://isrl.cs.byu.edu.

To date, the focus of trust negotiation has been on the protection of sensitive server resources. The server initiates trust negotiation

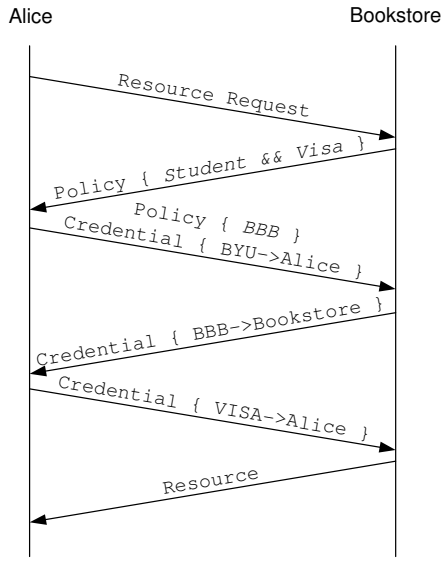

Figure 1 - Simple trust negotiation scenario between a student, Alice, and a bookstore

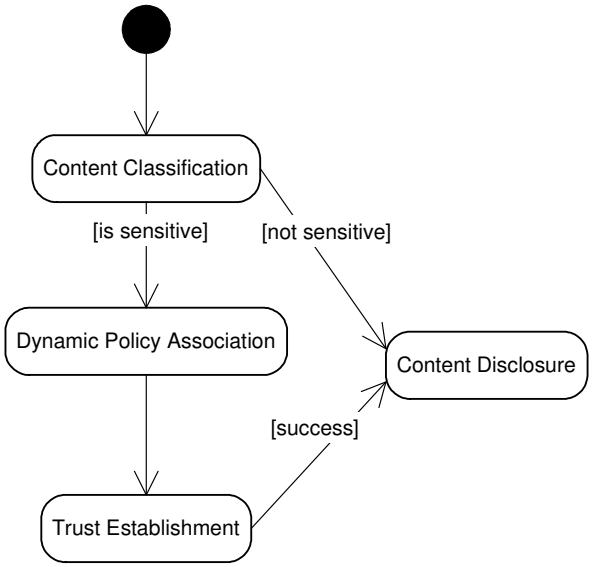

Figure 2 - State diagram for dynamic client content access control system

whenever a client requests a sensitive resource. During trust negotiation, client protection has been limited to the authorized disclosure of sensitive credentials and policies. In order to protect dynamically generated sensitive client content, this paper introduces *content-triggered trust negotiation*, permitting a client to initiate a trust negotiation with a server prior to requesting a service whenever the client discloses sensitive information as part of the service request. This form of protection has not been available to clients in any previous work on trust negotiation.

3. ACCESS CONTROL MODEL

In this section, we present an access control model for dynamically generated client-side content. The proposed model turns the traditional client/server access control model on its head, and addresses how to protect the sensitive content that clients disclose to servers. Since client content is dynamically generated, the usual approach of associating a policy with the resource a priori does not work. The model has provisions for determining the sensitivity of dynamic client content, mapping the content to an access control policy, and establishing the trustworthiness of the server before disclosing the content to the server.

The state machine shown in figure 2 illustrates the flow of information in the model. The start state corresponds to a client who is about to disclose information to a server. All outbound content is filtered to determine whether it is sensitive according to predefined rules. If the content is not sensitive, it is immediately disclosed to the server. If the content is sensitive, an access control policy is dynamically generated based on the types of sensitive information found in the content. Finally, the client must establish trust in the server by determining whether the server satisfies the access control policy governing disclosure of the sensitive content. If the server is deemed trustworthy, the sensitive content is disclosed.

The three primary components of the access control model for sensitive content are content classification, dynamic policy association, and trust establishment. The remainder of this section includes a detailed discussion of these three components.

3.1 Content Classification

In our model, document classification methods are employed to determine whether client content is sensitive. A variety of document classification techniques exist, each directed at a specific classification domain. Our model is general purpose to support a wide range of techniques, in order to recognize sensitive data in a variety of contexts.

Traditional classification models assume dynamic queries that operate on a static document base; however, our system uses a filtering approach in which the queries are static and the content is dynamic. A set of these queries, denoted as Q, is created by a user or administrator to describe a type of sensitive content. For example, a user who deems his bank account number to be sensitive may create a query that detects its disclosure. While this example can be accomplished using simple text pattern matching, our model can also extend to more sophisticated algorithms, such as algebraic and probabilistic models that may require training on relevant documents.

Each query has a name that represents the associated relevant content type, such as `bank_account_query` in the previous example. Associated with each sensitive content type is a disclosure policy created by a user or administrator that specifies what roles the recipient must authenticate to in order to receive the sensitive content

The content classification engine filters all outbound messages from the client according to the following definitions.

$$classify(m,Q) = \bigcup_{\forall q \in Q} sim(m,q) = T$$

$$sim(m,q) = \begin{cases} typeName & \text{if } m \text{ is similar to } q \\ \emptyset & \text{otherwise} \end{cases}$$

The engine invokes the classify function, which takes as input the outbound content m and a set of queries Q, and returns a set of

sensitive content type names T. Each $q \in Q$ is a query that looks for a specific sensitive content type. The classify function is defined as the union of the results of a similarity function *sim* applied to all elements of Q. The *sim* function filters the content m with respect to query q, according to the classification method in use.

3.2 Dynamic Policy Association

When dynamic client content is determined to be sensitive, an access control policy must be dynamically associated with the content in order to control its disclosure. This can be accomplished using the following approach. The dynamic policy association phase utilizes a database of one-to-one mappings between each sensitive content type and a corresponding access control policy. Suppose the classification engine generates a set T of sensitive content types associated with a message m. Let P be the set of all policies in the system such that for every $t \in T$, there is an associated policy $p \in P$. For a given message m, a final disclosure policy p' can be constructed by conjoining each policy p associated with every $t \in T$.

To provide administrative scalability, the model will leverage a role-based access control model [16]. Roles can be thought of as the intensional analogue to the extensional groups widely used for access control in environments such as file systems. The policy language used should allow users to define new roles and their semantics.

In our model, access control policies for sensitive content types are represented as role expressions, according to the grammar in figure 3. The language for describing role-based access control requirements is built on propositional logic without negation. Because the model targets open systems, negation is excluded to ensure monotonicity. If it were included, then parties could establish trust by not admitting that they possessed certain credentials or attributes. By use of logic, role expression policies are capable of capturing intricate trust requirements by combining roles with Boolean operators. Role expressions are considered satisfiable if there exists some interpretation for which the logic evaluates to true.

Expression	→	*Role*
	|	*ComplexExpression*
ComplexExpression	→	(*Expression*)
	|	*Expression Op Expression*
Op	→	∧ | ∨

Figure 3 - Role expression grammar in BNF

The following is a simple example of the use of role expressions in the dynamic policy association process based on the purchasing scenario presented in the introduction. Suppose Alice is completing an online purchase from Bob, and she is required to disclose her credit card number and mailing address. Once Alice's content classification engine determines that these items are sensitive, her local policy database is consulted to obtain the access control policy associated with each type of sensitive information. The conjunction of the relevant role expressions creates the logical role expression for the final disclosure policy. These generated policies can be rewritten as more precise expressions without a loss of semantics by applying Boolean algebra rules, such as the absorption, distributive, and idempotent laws. For instance, if her credit card number has an associated access control policy of `AuthorizedVisaMerchant` ∧ `BBBMember` and her mailing address has an associated access control policy of `BBBMember`, the simplified conjunction of these two policies is `AuthorizedVisaMerchant` ∧ `BBBMember`. The final disclosure policy is then forwarded to Alice's security agent, to be used in establishing trust with the server.

3.3 Trust Establishment

Once an access control policy has been dynamically associated with sensitive client content, it is the job of the security agent to authenticate the message recipient according to the policy. This section details the requirements and environment in which the security agent operates.

In today's highly networked world, the security agent must be able to authenticate strangers, i.e., entities that do not have a common security domain or shared secret. Hence, identity based authentication is inadequate due to the lack of any previous relationships that would give meaning to identity. To overcome this limitation, the security agent should define the roles in the disclosure policy in terms of attributes which the other party may possess rather than using a traditional identity based access control model.

An important design decision is to specify how the client determines the trust agent with which to initiate a negotiation. Previous work in trust negotiation has not addressed this issue. Traditionally, the server simply reacts to a client request for service and initiates a trust negotiation in-band with that client. With access control for dynamic client content, the server cannot rely on an existing connection with the client because the negotiation may take place before the client makes first contact with the service. Standard conventions need to be adopted to prescribe the relationship between a service and the associated trust establishment agent that represents the service. The solution cannot require clients to disclose, even inadvertently, sensitive information about the service request they intend to submit.

For possible solutions, consider a client's security agent that desires to contact a web server's security agent. A web server could support trust establishment in advance of a sensitive request in several ways. First, a new HTTP header could be introduced to indicate the desire to establish trust in the web server prior to making a specific request of the server. Second, the TLS/SSL protocol [5][15] could be extended to support a more sophisticated version of server authentication through trust negotiation rather than the limited client/server authentication it supports today. Trust Negotiation over TLS (TNT) is an early implementation of this concept, which assumes that servers will enforce access control policies on requested resources after the secure connection has been established [8]. An extension to TNT supporting content-triggered trust negotiation would allow clients to initiate trust negotiation during the establishment of a TLS connection before any sensitive data is transmitted. Third, web servers could employ an external security agent to handle trust negotiation by redirecting the client to the agent when trust negotiation is requested. This decouples the server from having to make trust decisions.

Other technologies besides web servers could adopt similar or alternative approaches. Consider a client using distributed object systems technologies such as SOAP and RMI to invoke a method that includes a sensitive input parameter. Some approaches to permit the client to establish trust in the service before completing the remote method invocation could be to provide a standard method interface with each object for establishing trust or to provide a standard service for establishing trust.

4. IMPLEMENTATION

The access control model for dynamic client content is general-purpose and suitable for integration into a wide range of applications and protocols. Application-level protocols are particularly relevant because most sensitive content originates at that level and the full semantics of the data are available to the content analysis procedure for determining sensitive content. Table 1 lists common application-level protocols and typical sources of potentially sensitive data in those protocols. Applications using these protocols could adopt the access control model for dynamic client content in order to safeguard client data.

Table 1 - Common application-level protocols and typical sources of potentially sensitive content

Protocol	Potential Sensitive Data in Protocol
HTTP	form data, headers, cookies, URLs
SMTP	email messages, attachments
FTP	Transferred files
SOAP	method parameters and names
NNTP	uploaded news posting
CORBA	method parameters and names

We implemented a prototype of the access control model for dynamically generated client-side content that supports two kinds of applications in order to demonstrate that the model is general-purpose. The prototype supports web applications that use the Hypertext Transfer Protocol (HTTP) [6] and email applications that use the Simple Mail Transfer Protocol (SMTP) [14]. Both protocols are common and frequently carry sensitive client data. The protocols differ significantly. For instance, HTTP is a synchronous protocol used for direct communication between web clients and servers. In contrast, email clients use SMTP to send messages asynchronously to the email server of the recipient. The email message is indirectly relayed through Message Transfer Agents (MTAs) rather than through a direct TCP connection between the sender and the receiver.

Our goal in developing the prototypes was to determine whether the access control model was flexible enough to accommodate the differences in these two application domains. In the remainder of this section, we discuss the major design issues faced in building a prototype for web and email applications.

4.1 TrustBuilder

Both prototypes make use of the same classification routines and security agent, TrustBuilder [24]. Since TrustBuilder was designed to protect static server resources, minor modifications were made so that it could perform content-triggered trust negotiation. A new method was added to the interface that allows clients to give TrustBuilder dynamically created policies for evaluation by its compliance engine. These policies are the role expressions described earlier. TrustBuilder utilizes the XML-based Trust Policy Language [7] developed at IBM Research to define the mapping of strangers to roles based on credentials issues by third parties. Each role contained in the final role expression generated from the policy association phase has a corresponding TPL policy that TrustBuilder uses to authenticate the server. TrustBuilder uses similar policies to govern the disclosure of sensitive credentials during trust negotiation.

The creation of these policies requires the successful classification of the client's content. For our implementation, we experimented with three different classification models: Boolean, vector space [17], and fuzzy set [11]. These were chosen because each has a relatively high accuracy within its application domain.

The Boolean method is based on set theory, and its queries are formulated using Boolean algebra. These queries are composed of keywords and determine set membership of a given document. Set membership is strict and is determined by the binary result of a query. The strengths of this method are its simplicity, speed, and guaranteed accuracy; however, its weakness lies in its inability to perform partial matches.

The fuzzy set model extends the Boolean model by allowing approximate document matching. Instead of enforcing strict set membership, this model returns a degree of membership bound between 0 and 1. Traditionally, this model is very accurate, with only minor cases of false positive results. We chose to implement the fuzzy set model described in [11], which simulates a thesaurus made up of keywords from existing documents that provide partial matching for query terms.

The vector space model, like the fuzzy set, also allows for partial matching. This model operates by transforming the terms in queries and documents into vectors of numeric weights. The angle between a query vector and document vector is then measured to determine the degree of similarity between the query and document, with smaller angles indicating higher relevancy. While this approach is more flexible than the Boolean method, a small percentage of misclassifications occur, resulting in false positives and negatives.

There are two possibilities for misclassifications in our system, false positives and false negatives. A false positive occurs when content is classified as sensitive when it actually is not. When this overprotection happens, trust negotiation transpires even though it is not necessary. If the recipient fails to authenticate, the client will be notified and can override TrustBuilder's decision and send the content regardless. This is analogous to how web browser implementations respond when they receive an invalid server certificate in SSL. In such cases the user can choose to bypass the browser's security settings and accept the invalid certificate. A false negative occurs when sensitive content is deemed unrestricted and inappropriately disclosed. One approach to reducing false negatives is to classify more aggressively. However, this will likely result in an increase in false positives.

To minimize the overhead of content classification at runtime, our implementation of the fuzzy set and vector space model support off-line training on relevant sensitive documents to create the queries used to identify sensitive content. Our initial experiments using all three content classification approaches demonstrate that

the overhead for classification can be acceptable for interactive applications like web browsing.

4.2 HTTP

A web browser requests resources from a web server through HTTP, a lightweight, stateless, application-level protocol (see figure 4). HTTP is able to transport a variety of data types. There are several instances where a web client could intentionally or inadvertently disclose sensitive data. The most prevalent example occurs when users fill out a web form and submit it to a server. The client's data is transferred in the body of the HTTP request or appended to the URL, depending on whether the web form uses the POST method or GET method, respectively. When a user uploads a file to the server, it is transferred in the body of the request message.

Figure 4 - Traditional HTTP client and server

A web request could inadvertently disclose sensitive information whenever the HTTP Referrer header is present. This header contains the entire URL of the referring web page. This presents a serious privacy vulnerability, since any HTML form data embedded in the previous URL is included.

Even if the URL does not contain embedded client content, the web request could be sensitive. The resource requested in the URL may reveal information about a client's interests or intentions. For example, a user requesting a web page on certain health issues invites the server to construe certain assumptions about the client.

HTTP cookies also present a risk for the inadvertent disclosure of sensitive information. In essence, cookies are pieces of information that a web server can store and retrieve from a client's machine. Even if cookies contain sensitive content, it is usually not a problem to send them back to the original server that generated them. The danger comes from intra-domain cookies, or cookies that are sent to all requesting servers within a given domain. This occurs when the cookie's creator adds the domain tag to the cookie header and indicates that other machines within a given subdomain can access the cookie. For example, a cookie sent from "www.example.com" with its domain set to "example.com" would be accessible to servers such as "shipping.example.com" and "order.store.example.com". This is a problem whenever the user trusts the cookie's originator but does not trust the other servers within the originator's domain. Figure 5 illustrates an HTTP request with several pieces of sensitive information included in the referrer header, cookie, and content body.

In order to perform trust negotiation over HTTP, the protocol must be extended. The current HTTP specification provides for two forms of authentication, Basic and Digest, that occur in-band with HTTP requests and responses. For the purposes of trust negotiation, we have created a new authentication type called `TrustNegotiation` that carries credentials and policies between client and server within HTTP headers. Since credentials and policies may be sensitive, SSL is used to ensure confidentiality and integrity.

There are several options to add the functionality of trust negotiation to a web browser. First, a proxy server on the client side could intercept client requests and determine if the content is sensitive so that additional trust in the server can be established before forwarding the request. Alternatively, a proxy server could be configured to sit at the edge of the network inside a firewall and provide consistent, mandatory content disclosure policy association for an entire organization. This eases administrative overhead, because it does not rely on clients configuring their own local environment, nor does it require the installation of additional software on each client machine. It also permits transparent interception of all outgoing requests, with potential to improve dramatically the control that organizations and households have over their sensitive content. However, the overhead of examining each request could be prohibitive. A browser plug-in offers similar functionality, but at the individual browser level. This would allow individual users fine-grained control over their personal content at the cost of more administrative overhead.

For our implementation, we built a proxy server that provides a flexible test platform for all types of browsers and scenarios (see figure 6). The proxy server indicates the desire to negotiate trust by adding a new trust negotiation request HTTP header,

```
POST /convert.cgi HTTP/1.0
Accept: */*
Referer: http://www.somesite.com/login.html?username=bob&password=ez2guess
Content-Type: multipart/form-data; boundary=---------------------------7d22b6030346
User-Agent: Mozilla/4.0 (compatible; MSIE 5.5; Windows NT 5.0)
Host: example.com
Content-Length: 264
Cookie: ssn=123-45-6789; name=Joe%20Smith

-----------------------------7d22b6030346
Content-Disposition: form-data; name="inputfile"; filename="C:\test.txt"
Content-Type: text/plain

My phone number is 801-123-4567.
-----------------------------7d22b6030346--
```

Figure 5 - HTTP request with sensitive information in the referer header, cookie, and body

signifying to the server that it should return a trust negotiation response. Since a resource is not returned, the client can simply indicate the server's root as the requested resource portion of the URL, since it will be ignored in this case. In this way, the client does not reveal what resource it intended to request. When sensitive content is detected and trust is established with the web server, the proxy forwards the HTTP request to the web server and relays the subsequent response to the client. In the case of a failed trust negotiation, the proxy terminates all communication with the web server and sends a web page back to the client's browser, indicating trust negotiation failure.

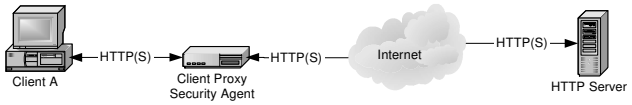

Figure 6 - Traditional HTTP client and server with security agent proxy

Traditional web clients make SSL connections through proxies by sending a HTTP CONNECT request to the proxy. This command indicates that the proxy should connect to a specified server and act as a tunnel between the two hosts. The proxy is required to respond and indicate success or failure, after which the client begins to negotiate an SSL session directly with the web server through the proxy. Even though the proxy can see all information relayed between the two hosts, it cannot determine the message's content because of the cryptographic properties of SSL. This feature is detrimental to our proposed system since the proxy must be able to view the content in plaintext to determine its sensitivity.

This problem is addressed by creating a man-in-the-middle proxy [15]. When the client requests that a pipe be created through the proxy to the destination web server, the proxy does not make the connection, but signals to the client that it has been made. Believing that the tunnel is established, the client then begins the SSL handshake with what it believes to be the web server, but is actually the proxy server. The client starts by sending the `ClientHello` message and expects in return the `ServerHello` message from the web server. This message is expected to include the web server's public key identity certificate; however, this is not feasible for the proxy to send, since the proxy has no access to the server's private key to answer key challenges. Instead, the proxy replies with its own certificate with the common name indicated as "*" instead of the web server's domain name. This common name is used by the browser to verify that the certificate belongs to the website that is being browsed and not some man-in-the-middle attacker. These names can be wildcard matched, thus allowing the proxy to send the name of "*" to match all websites. Since no respected certificate authority would issue such a certificate, the proxy's administrator must create a local certificate authority that issues the certificate. Additionally, the administrator must add the local authority to the browser's list of trusted certificate authorities. When the client receives the proxy certificate, it accepts it as the valid certificate, since domains are wildcard matched. Once the SSL session is established between the client and the proxy, the proxy will be able to accept the client's data in plain text and determine its sensitivity. The proxy server will then make an SSL connection to the intended web server and either release the content or negotiate trust based on sensitivity.

4.3 Email

Simple Mail Transfer Protocol (SMTP) is the principal means by which email is transported through the Internet. Clients use this protocol to forward mail to the end recipient via Mail Transfer Agents (MTA). Email messages are forwarded by these MTAs to the recipient's local mail server mailbox. To retrieve the messages from the mailbox, the recipient uses a protocol, such as the Post Office Protocol v3 (POP3) or the Internet Message Access Protocol (IMAP). Figure 7 illustrates how SMTP and POP3 are used to relay email messages.

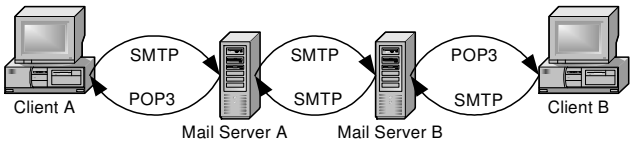

Figure 7 - Traditional SMTP / POP3 message relaying

Typical electronic mail is composed of an envelope, header, and a body. The message body may consist of a single message or a multipart collection of messages and attachments. Clients may disclose sensitive content in an email's subject line, message body, or attachments. Figure 9 illustrates a multipart message with sensitive content included in the email body and the attachment.

Much like the previous HTTP example, we have inserted a security agent proxy to examine relevant portions of outgoing SMTP messages for sensitivity. When an outgoing email message is deemed sensitive, trust negotiation is initiated by withholding the sensitive message and sending a trust negotiation email message. These messages are MIME multipart email messages containing credentials and policies as attachments. The security agent proxy on the recipient side scans incoming emails when messages are retrieved from the mail server via POP3. To distinguish these trust negotiation mail messages from normal email messages, we introduced a new mail extension header, `X-TrustNegotiation`, which is ignored by mail agents that do not support trust negotiation. When a trust negotiation message is received, the recipient responds by sending a relevant trust negotiation response. Messages are exchanged between security agent proxies using SMTP and POP3 until the message originator deems success or failure. If successful, the original message is sent to the authenticated recipient; otherwise, an error response is sent to the sender in the form of an email. Figure 8 illustrates message relaying with our security agent proxies that support trust negotiation.

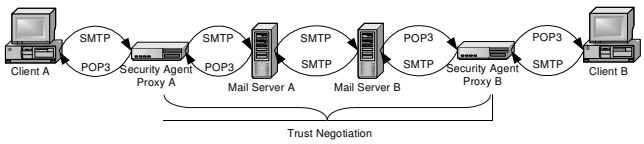

Figure 8 - Traditional SMTP / POP3 message relaying with security agent proxies

The major differences between the HTTP and SMTP scenarios are the trust negotiation message formats and message transport security procedures. Trust negotiation messages in HTTP are sent inline with HTTP requests and responses. SMTP trust negotiation information, including credentials and policies, differs by being sent as multipart email messages, allowing backwards

compatibility with existing mail systems. Our implementation works with standard email clients, such as Outlook and Mozilla.

The HTTP scenario provides confidentiality and integrity for trust negotiation messages by using SSL; however, this mechanism is not available within electronic mail due to the lack of point-to-point connections between email senders and recipients. Our current implementation does not provide confidentiality. Since the end-to-end communication channel cannot be secured, the next option is to encrypt parts of the message. Potential solutions include S/MIME, PGP/MIME, and identity based encryption. Both S/MIME and PGP require that the email sender obtain the public key certificate of the recipient to encrypt the session symmetric key. This requires that the sender obtain the certificate in advance or that the security agent has access to an infrastructure to automatically perform a lookup for the certificate at the time that trust must be established. On the other hand, identity based encryption (IBE) [3] does not require that the sender obtain an identity certificate from the recipient. Instead, the public key is generated from publicly available information such as the recipient's email address and other identifying items. While IBE technology is not as widespread as S/MIME or PGP, a reference implementation has been created by the Stanford Applied Cryptography Group (see http://crypto.stanford.edu).

HTTP messages have a single sender and recipient. In contrast, SMTP allows a single sender to specify multiple recipients. This point to multipoint communication occurs when additional email addresses are supplied in the TO, CC, or BCC fields. In this case, the sender would negotiate trust with each receiving party before disclosing a message containing sensitive content.

Since email is an asynchronous store-and-forward protocol, it does not demand interactive response time. A more intensive content classification effort can be incorporated in an email system than may be practical in interactive applications.

5. RELATED WORK

Previous work has incorporated X.509v3 certificates that store attributes or roles of subjects for use in access control decisions (e.g., [4][7][21]). Research in trust negotiation treats certificates themselves as potentially sensitive, requiring an access control framework to regulate their disclosure. There are several recent research projects that explore trust negotiation using digital credentials. RT [22] is a role-based trust management framework that has been used for trust negotiation. Bonatti and Samarati [2] introduced a trust establishment framework. Their system includes a portfolio and service protection language (PSPL) that expresses rules for accessing services and disclosing user portfolio objects, including credentials. X-Sec [1] is an XML-based language for specifying credentials and security policies for Web document protection. It was not originally designed for establishing trust between strangers, but it can easily be extended to do this. Our work presented in this paper compliments the previous work on trust negotiation, and is the first example of content-triggered trust negotiation to permit a client to proactively establish trust in a server prior to sending a sensitive service request.

There are identity management efforts underway to better handle users' sensitive personal information, such as the Liberty Alliance [10]. One focus of that project is to provide single sign-on across a federation of trusted web sites and securely manage personal information. This is an example of a closed system, whereas our focus for content-triggered trust negotiation is aimed at first-time interactions between strangers. The approach presented in this paper may be suitable during the initial trust establishment process between a user and a Liberty-enabled website or to automatically manage introductions between affinity group members under Liberty Alliance. Trust negotiation is aimed at handling introductions between strangers and is not a replacement for authentication in closed systems.

```
From: "Adam Hess" <ahess@cs.byu.edu>
To: <stranger@example.com>
Subject: Top Secret email!!!
Date: Wed, 11 Sep 2002 15:33:07 -0700
MIME-Version: 1.0
Content-Type: multipart/mixed;
       boundary="----=_NextPart_000_0000_01C259A8.87404870"

------=_NextPart_000_0000_01C259A8.87404870
Content-Type: text/plain; charset="iso-8859-1"
Content-Transfer-Encoding: 7bit

My phone number is (801)123-4567 and ssn is 123-45-6789.
------=_NextPart_000_0000_01C259A8.87404870
Content-Type: text/plain; name="password.txt"
Content-Transfer-Encoding: 7bit
Content-Disposition: attachment; filename="password.txt"

My username is bob and password is ez2guess.
------=_NextPart_000_0000_01C259A8.87404870--
.
```

Figure 9 - Example of email message with sensitive content

The Platform for Privacy Preferences (P3P) [13] is a project developed by the World Wide Web Consortium, which allows websites to express their privacy policies for handling sensitive client information. Such policies inform the client regarding who is collecting the data, what is being collected, why it is being collected, and what will be shared with others. The privacy policies are created by the web site, and clients must trust that the policy reflects the web site's practices. In our research, we focus on attributes that are asserted by trusted third parties. We support a more general notion of attributes, and view privacy practices as one of many attributes of potential interest to a client.

Another self-regulatory privacy method uses privacy seals that are placed on websites to give end users a sense of security. Some existing trust label companies are TRUSTe, Verisign, BBBOnline, and webtrust.org. While the privacy seals approach attempts to assure clients that the website can be trusted, the weakness of this system lies in the fact that clients rarely verify the validity of the seal on the website, because it is typically time consuming and confusing for the average user.

Some commercial firewalls exist that assist web clients in preserving their privacy by not allowing user-defined keywords to be transmitted in HTTP requests unless the communication channel is secure. However, this only guarantees confidentiality and integrity while the message is in transit and says nothing of the server's trustworthiness.

Another form of preventing unauthorized message delivery is SPAM filtering software for email. Like the proxies presented in this paper, SPAM blocking software can perform header and text analysis; however, many messages are blocked using blacklists or collaborative signature based spam-tracking databases instead of user-defined queries. SPAM filters differ from the proxies used in this paper by operating on incoming messages instead of outbound content.

6. CONLUSIONS AND FUTURE WORK

Conventional access control mechanisms are primarily designed to protect the release of known resources to known users. This paper has explored the scenario when neither the resource nor user has been introduced into the system. Specifically, the research presented has focused on content that is generated dynamically by a client and disclosed to another party, such as web forms and emails. Since the content does not persist on the user's hard drive prior to disclosure, it is hard to affix an access control policy in advance. Instead, the association between the sensitive content and a suitable access control policy must be determined at runtime. In addition, the sender and receiver lack a pre-existing trust relationship, and thus they are not able to authorize a sensitive transaction based on identity, as is normally done in closed systems. Instead, they rely on trust negotiation to establish trust between strangers in open systems.

To address these problems, this paper presents an access control model for regulating the disclosure of dynamic client content and describes its implementation, content-triggered trust negotiation. The goal of this model is to provide greater protection to sensitive information that clients may intentionally or inadvertently disclose to servers outside of their security domain, where no preexisting trust relationship exists. When this new model is applied, an attempt to transmit sensitive data generates appropriate access control policies dynamically and initiates trust negotiation. The iterative disclosure of access control policies and associated attribute-based credentials gradually builds necessary trust, providing the client with greater assurance to disclose the sensitive content. This new model is composed of three phases: identifying sensitive dynamic client content, generating access control policies for such content, and establishing the trustworthiness of the server prior to the disclosure of the sensitive content.

To implement this model, we have extended TrustBuilder to perform content-triggered trust negotiation, expanding the use of trust negotiation beyond the protection of static sensitive server resources. Our implementation supports two rather diverse protocols, HTTP and SMTP. HTTP is a point-to-point synchronous communication protocol used to access resources on the World Wide Web. In contrast, SMTP is an asynchronous, message-routed protocol used in the delivery of MIME-formatted email messages. The underlying differences of these protocols motivate disparate approaches to establishing trust when applying access control for dynamic client content. HTTP mediates trust negotiation within its synchronous request and response messages, while SMTP operates asynchronously through trust negotiation messages embedded in email.

The current implementation of the model is not intended to provide a foolproof mechanism for protecting disclosures a client might make; nevertheless, it does serve as additional assurance in trusting the intended recipient via the guarantees of successful trust negotiation. The current implementation does not identify sensitive content that is encrypted, hidden using steganographic techniques, or otherwise obfuscated at the application layer of the networking protocol stack. Additionally, while our implementation makes use of three accurate classification techniques, false positives and false negatives may occur.

Our proposed model provides a new application area for document classification research. Further experimentation and research within that area may lead to greater assurance to clients who use our proposed model.

In the future, our implementation has the potential to be deployed as a plug-in for a variety of Internet-aware applications. Current potential target platforms include email clients, web browsers, newsgroup programs, chat clients, and RPC clients, among others.

This paper focused on classifying outbound client content and authenticating the recipient when the content is sensitive. In the future, we plan to apply the client's content classifier to content received from the server. This will explore situations in which dynamic content from the server is questionable or when the client requires a high degree of trust in the content. For example, when the client content classifier detects that the inbound content refers to stock quotes or medical advice, the client may require assurance that the server provides trustworthy content. Even though the model appears to be well suited for this problem, analyzing dynamic server content will place greater demands on system scalability in cases where inbound content is proportionally greater than outbound, such as client HTTP traffic.

We are currently exploring approaches to confidential trust negotiation using SMTP. Even if a negotiation is confidential, there are elements inherent to the communication protocol, such

as email and IP addresses, which prevent total anonymity. Email addresses are especially revealing and can be used to derive much information about an individual. An area for future research is the use of anonymous remailers within the email prototype. Anonymous remailers have the capability of hiding the true identity of the sending party. In essence, they operate by removing any identifying headers and then forwarding the message to the recipient with an anonymous email address for the sender inserted by the remailer. Several remailers can be chained together to protect the anonymity of the sending party. The use of remailer technology could provide anonymity during trust negotiation when a user attempts to send a sensitive email message.

7. ACKNOWLEDGEMENTS

This research was supported by funding from Zone Labs, Inc. and DARPA through AFRL contract number F33615-01-C-0336 and the SSCSD grant number N66001-01-18908. The authors would like to thank Marianne Winslett for her helpful discussions on content-triggered trust negotiation and her feedback on earlier versions of this paper. They also thank the anonymous reviewers for their suggestions that helped to improve the paper.

8. REFERENCES

[1] E. Bertino, S. Castano, and E. Ferrari, "On Specifying Security Policies for Web Documents with an XML-based Language," Sixth ACM Symposium on Access Control Models and Technologies, Chantilly, Virginia, May 2001.

[2] P. Bonatti and P. Samarati, "Regulating Service Access and Information Release on the Web," Seventh ACM Conference on Computer and Communications Security, Athens, Greece, November 2000.

[3] D. Boneh and M. Franklin, "Identity Based Encryption from the Weil Pairing," Crypto 2001, Santa Barbara, California, August 2001.

[4] D. Chadwick and A. Otenko, "The PERMIS X.509 role based privilege management infrastructure," Seventh ACM Symposium on Access Control Models and Technologies, Monterey, California, June 2002.

[5] T. Dierks and C. Allen, The TLS protocol version 1.0, RFC 2246, January 1999.

[6] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1," RFC 2616, June 1999.

[7] A. Herzberg, J. Mihaeli, Y. Mass, D. Naor, and Y. Ravid, "Access Control Meets Public Key Infrastructure, Or: Assigning Roles to Strangers," IEEE Symposium on Security and Privacy, Oakland, California, May 2000.

[8] A. Hess, J. Jacobson, H. Mills, R. Wamsley, K. E. Seamons, and B. Smith, "Advanced Client/Server Authentication in TLS," Network and Distributed System Security Symposium, San Diego, California, February 2002.

[9] R. Housley, W. Polk, W. Ford, and D. Solo, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile," RFC 3280, April 2002.

[10] Liberty Alliance Project, http://www.projectliberty.org, December 2002.

[11] Y. Ogawa, T. Morita, and K. Kobayashi. "A Fuzzy Document Retrieval System Using the Keyword Connection Matrix and its Learning Method," Fuzzy Sets and Systems, Vol. 38, pp. 17-41, 1991.

[12] J. Park and R. Sandhu, "Binding Identities and Attributes Using Digitally Signed Certificates," Proc. 16th Ann. Computer Security Applications Conference, IEEE CS Press, Los Alamitos, California, 2000, pp. 120-127.

[13] The Platform for Privacy Preferences 1.0 (P3P1.0) Specification. W3C Candidate Recommendation. 16 April 2002, http://www.w3.org/TR/P3P/.

[14] J. Postel, "Simple Mail Transfer Protocol," RFC 821, August 1982.

[15] E. Rescorla, SSL and TLS: Designing and Building Secure Systems, Addison-Wesley, 2001.

[16] R. S. Sandhu and E. J. Coyne, "Role-Based Access Control Models," IEEE Computer, Volume 29, Number 2, February 1996, pp. 38-47.

[17] G. Salton and M.E Lesk, "Computer Evaluation of Indexing and Text Processing," Journal of the ACM 15(1):8-36, 1968.

[18] K. E. Seamons, M. Winslett, and T. Yu, "Limiting the Disclosure of Access Control Policies During Automated Trust Negotiation," Network and Distributed System Security Symposium, San Diego, California, February 2001.

[19] K. E. Seamons, M. Winslett, T. Yu, B. Smith, E. Child, J. Jacobson, H. Mills, and L. Yu, "Requirements for Policy Languages for Trust Negotiation," Third International Workshop on Policies for Distributed Systems and Networks (POLICY 2002), Monterey, California, June 2002.

[20] K. E. Seamons, M. Winslett, T. Yu, L. Yu, and R. Jarvis. "Protecting Privacy during On-line Trust Negotiation," 2nd Workshop on Privacy Enhancing Technologies, San Francisco, California, April 2002.

[21] D. Shin, G. Ahn, and S. Cho, "Role-based EAM Using X.509 Attribute Certificate," 16th Annual IFIP WG 11.3 Working Conference on Data and Application Security, University of Cambridge, United Kingdom, July 2002.

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

[23] W. Winsborough, K. E. Seamons, and V. E. Jones, "Automated Trust Negotiation," DARPA Information Survivability Conference and Exposition, Hilton Head, South Carolina, January 2000.

[24] M. Winslett, T. Yu, K. E. Seamons, A. Hess, J. Jacobson, R. Jarvis, B. Smith, and L. Yu, "Negotiating Trust on the Web," IEEE Internet Computing, Vol. 6, No. 6, November/December 2002.

[25] T. Yu, M. Winslett, and K. E. Seamons, "Supporting Structured Credentials and Sensitive Policies through Interoperable Strategies for Automated Trust Negotiation," ACM Transactions on Information and System Security, volume 6, number 1, February 2003