CONTENT-TRIGGERED TRUST NEGOTIATION

by

Adam Timothy Hess

A thesis submitted to the faculty of

Brigham Young University

in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computer Science

Brigham Young University

February 2003

BRIGHAM YOUNG UNIVERSITY

GRADUATE COMMITTEE APPROVAL

of a thesis submitted by

Adam Timothy Hess

This thesis has been read by each member of the following graduate committee and by majority vote has been found to be satisfactory.

_____	_____
Date	Kent E. Seamons, Chair
_____	_____
Date	Mark J. Clement
_____	_____
Date	Parris Egbert

BRIGHAM YOUNG UNIVERSITY

As chair of the candidate's graduate committee, I have read the thesis of Adam Timothy Hess in its final form and have found that (1) its format, citations, and bibliographical style are consistent and acceptable and fulfill university and department style requirements; (2) its illustrative materials including figures, tables, and charts are in place; and (3) the final manuscript is satisfactory to the graduate committee and is ready for submission to the university library.

_____
Date

_____
Kent E. Seamons
Chair, Graduate Committee

Accepted for the Department

_____
David W. Embley
Graduate Coordinator

Accepted for the College

_____
G. Rex Bryce
Associate Dean,
College of Physical and
Mathematical Sciences

ABSTRACT

CONTENT-TRIGGERED TRUST NEGOTIATION

Adam Timothy Hess

Department of Computer Science

Master of Science

Traditionally, 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 resources are usually static, and an access control policy associated with each resource specifies the identity of who is authorized to access the resource. In this thesis, I turn the conventional 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 generated at the time of disclosure, the usual approach of associating a policy with the resource a priori does not work. This thesis proposes an access control model for protecting dynamic client-side content that identifies sensitive content, maps such content to an access control policy, and establishes the trustworthiness of the server before the content is disclosed to the server. The model targets open systems, where clients and servers do not have preexisting trust

relationships. I 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.

ACKNOWLEDGMENTS

I would like to thank my wife Emily, family, Dr. Kent Seamons, and members of the Internet Security Research Lab for their help and support throughout my education. Additionally, this research was supported by funding from Zone Labs, Inc. and DARPA through AFRL contract number F33615-01-C-0336 and through the Space and Naval Warfare Systems Center San Diego grant number N66001-01-18908.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

Introduction

While performing Internet transactions, users often reveal sensitive information about themselves and their affiliated organizations. The disclosure of such information poses serious threats to individual privacy and the confidentiality of business or government 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.

1.1. Problem Description

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 eavesdropper 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. SSL is incapable of this level of authentication. Thus, given the current model, how can Alice trust Bob to handle her sensitive data appropriately?

Other types of sensitive content that Alice may disclose include financial information, health records, Social Security Number, source code, confidential government data and confidential business related data. The disclosure of this information is not limited to HTML forms on the World Wide Web. Alice can also send her sensitive information to Bob via email, chat programs, file transfers, or other networked applications, thus creating dynamic contexts for sensitive content. 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 offline before disclosing her sensitive content.

To further illustrate the problem at hand, consider the results from research

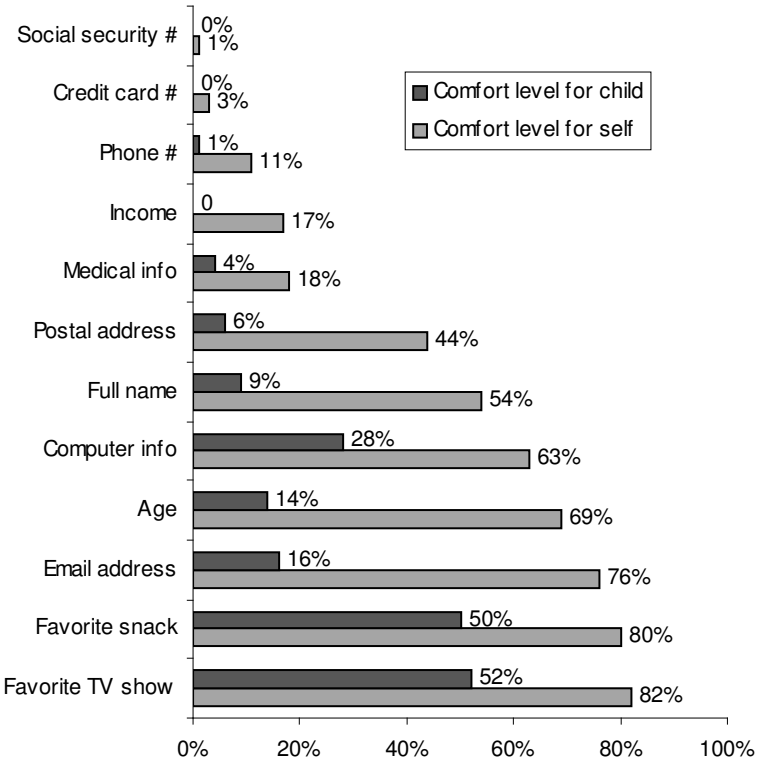


Figure 1 – Percentage of respondents in [1] who were always or usually comfortable providing information to web sites

published in [1] where approximately 400 various Internet users were asked their opinion on the disclosure of personal data in online situations. Figure 1 gives the percentage of respondents who were always or usually comfortable providing different types of personal information online for themselves or for a child. If this graph truly represents the attitudes of Internet users at large, then the general public is very uneasy in releasing private information given the current state of technology. The work of this thesis aims to strengthen user confidence when such disclosures are necessary.

1.2. Current Obstacles

In order to equip clients with automated support for protecting dynamic outbound content, there are two problems that must be solved: authentication and authorization. The authentication issue is that parties without pre-existing relationships can't perform a sensitive transaction based on identity alone. Additionally, authorization is complicated because it is difficult to associate access control policies with transient content. A solution to the authentication and authorization problem requires that the client must be able to authenticate an unfamiliar server and then determine whether the server is authorized to receive the sensitive content.

1.2.1 Authentication

Standard authentication mechanisms are not adaptable to the proposed problem. To illustrate this, consider a password based system. It is infeasible to use such 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 traditional username and password with each client that

desires server authentication. 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.

1.2.2 Authorization

Traditional authorization is also unsuitable to perform access control in environments where dynamically generated resources are to be disclosed to strangers. The standard model for access control consists of the triple (S, O, M) where S is a set of subjects requesting objects, O is a set of objects to be protected, 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 individual objects in the system. The rows of M are capability lists for the unique subjects in the system. This access control model works well in a closed system with a known set of subjects and static objects. The target environment in my 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 generated for 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.

1.3. Thesis Statement

Early research has shown that trust negotiation is a viable solution for establishing trust between strangers. Existing trust negotiation methods are server-based and protect static resources. My hypothesis is that clients can successfully employ trust negotiation to protect sensitive content by generating an appropriate access control policy that can be used during trust negotiation to authenticate a server.

The principal contribution of this thesis is the design and implementation of a system that automatically establishes trust during client-initiated transactions with strangers when sensitive content is disclosed. The benefits of this system are the ability to communicate information whose disclosure would have previously been prohibited and the prevention of unauthorized disclosures of sensitive information. The system has the potential to impact private personal transactions, sensitive business negotiations, and confidential government communications. The primary contributions of this thesis will appear in [11].

The remainder of this thesis is organized as follows. In chapter 2, I discuss trust negotiation, an approach to establishing trust between strangers in open systems. In chapter 3, I describe my access control model for providing access control for dynamic client-side content. In chapter 4, I describe an implementation of the proposed model built into the TrustBuilder architecture that supports web and email client applications. Chapter 5 includes a discussion of related work and chapter 6 contains conclusions and future work.

Chapter 2

Trust Negotiation

A large majority of the transactions on the Internet occur between hosts that have no relationship of trust, i.e. – strangers. In order for strangers to conduct sensitive transactions, a sufficient level of mutual trust must be established. Current authentication systems that are based on identity are ill suited to stranger authentication, principally because identity is irrelevant without prior context or a pre-existing relationship. Instead, verifiable attributes of the participants are more relevant. Examples of such attributes could be employment status, citizenship, age, credit ratings, accreditations, certifications, security clearances, etc. A system using such attributes for authentication has been developed and is called *trust negotiation* [26]. This method incrementally builds trust by iteratively disclosing digital credentials that contain attributes of the negotiating participants. This approach relies on access control policies that govern access to protected resources by specifying combinations of attribute credentials that must be submitted to obtain authorization. The research in this thesis builds on prior work in trust negotiation. The remainder of this section discusses the fundamentals of trust negotiation.

2.1. Digital Credentials

Digital credentials are analogous to the paper credentials that people carry in their wallets. Both carry attributes about the owner, are verifiable, and were issued by some third party. However, due to their cryptographic properties, digital credentials have the

advantage of being unforgeable. Digital credentials contain digitally signed assertions by a credential issuer about a credential owner. The credential is signed with the issuer's private key and can be verified with the issuer's public key. 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 credentials by encrypting data with its private key that others can decrypt with the public key contained within the credential. Other approaches are also possible [15].

Credentials are a more general name for certificates, such as X.509 certificates. The work for this thesis makes use of X.509 version 3 certificates. These are ASN.1 encoded records that are the standard for digital certificates today. In their latest version,

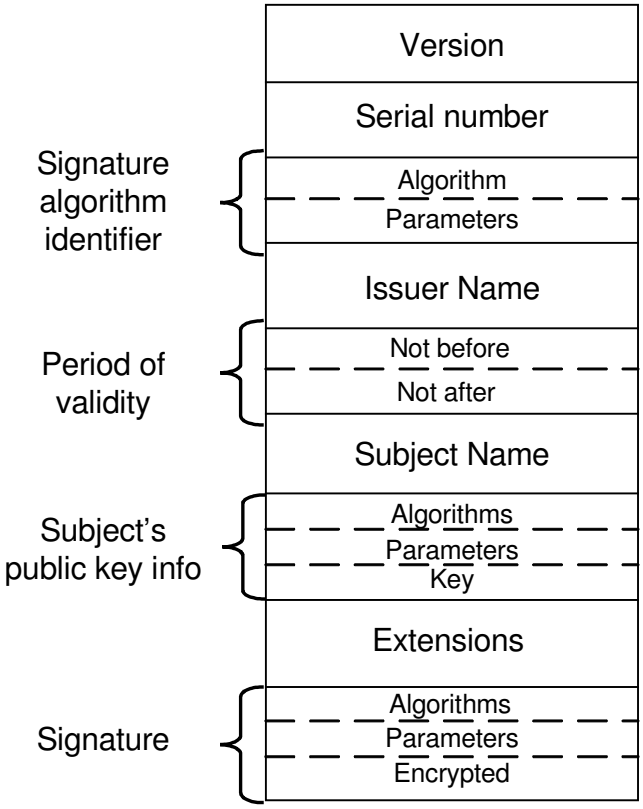


Figure 2 - X.509 version 3 certificate format

attributes are expressed as a set of extensions. The general format of a X.509 version 3 certificate is shown in figure 2. It should be noted that it carries all of the previous properties mentioned for digital credentials such as verifiability and authenticity.

2.2. Policies

The content of many digital credentials makes them sensitive. For example, a credential may contain nonpublic attributes about an individual. Additionally, the sheer fact that a certain organization has issued some subject a credential may convey confidential affiliations. Because of their sensitive nature, the disclosure of digital credentials must be carefully managed in accordance with an access control policy that specifies which credentials must be received before it can be disclosed. A credential can only be disclosed when its access control policy has been satisfied. 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 according to additional policy specification. Earlier work in trust negotiation introduced support for sensitive policies using policy graphs [21]. 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 and privacy 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. This protocol is executed by TrustBuilder [26], an architecture for trust negotiation. Within TrustBuilder, 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.

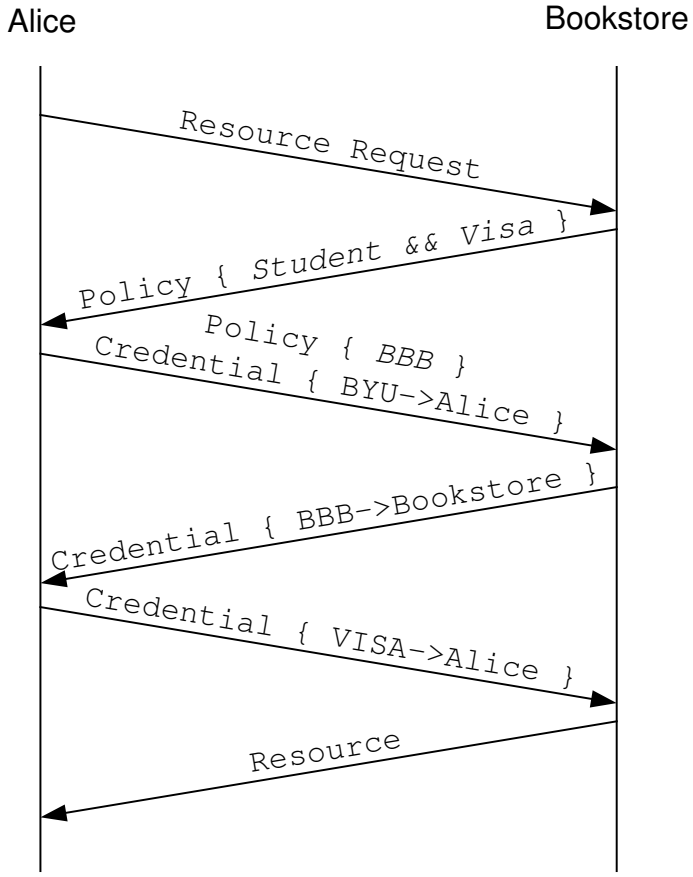


Figure 3 - Simple trust negotiation scenario between a student, Alice, and a bookstore

2.3. Trust Negotiation Example

Figure 3 shows an example of a trust negotiation where an on-line bookstore offers discounts to students of accredited universities. When a first-time customer requests a student discount, he or she will not have prior knowledge of the bookstore's

requirements for proof of student status. One approach is for the server to transmit a policy to the client. Such a policy could specify that the customer must submit a student ID and a credit card number in order to make an online purchase and receive a student discount. The customer (for example, 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 server return a BBB member credential to the client. The server then sends the client a BBB member credential. Finally, the client submits a valid digital credit card number and receives the student discount.

2.4. Previous Trust Negotiation Research

Examples of earlier work in trust negotiation include support for sensitive credentials and access control policies [21], the definition and interoperability of trust negotiation strategies [27], a trust negotiation protocol based on an extension to TLS [10], protecting privacy during trust negotiation [23], an analysis of policy languages for trust negotiation [22], and the development of the TrustBuilder architecture for trust negotiation [26]. To date, the focus of trust negotiation has been on the protection of sensitive static server resources. The server initiates trust negotiation 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 sensitive dynamic client content, this work of this thesis 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 earlier work on trust negotiation.

Chapter 3

Access Control Model for Dynamic Client-Side Content

In this section, I present an access control model for dynamic 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 created dynamically, 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.

There are two design goals for the model. First, the model needs to be flexible and general purpose so that it is independent from any particular classification method, networking protocol, or content format. Second, an implementation of the model should introduce minimal performance overhead.

The state machine shown in figure 4 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 generation, and trust establishment. The remainder of this section includes a detailed discussion of these three components.

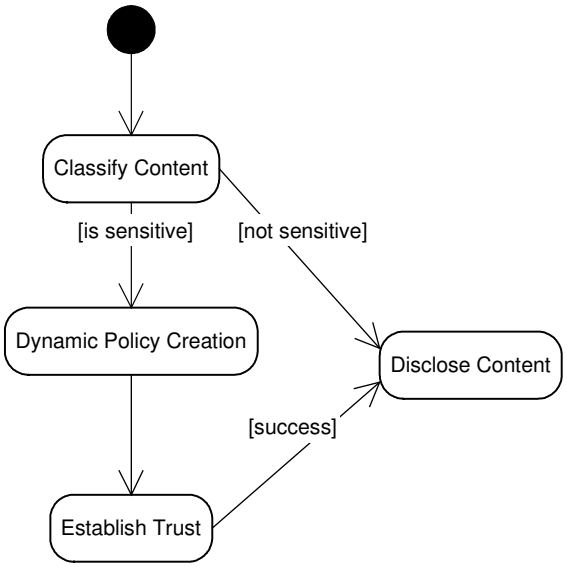


Figure 4 - State diagram for proposed dynamic client content access control system

3.1. Content classification

In this 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. This model is designed to support a wide range of classification techniques, allowing sensitive data to be recognized in a variety of contexts.

Traditional classification models assume dynamic queries that operate on a static document base; however, this 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 specific type of sensitive content. For example, a user

that 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, this model can also extend to more sophisticated algorithms, such as algebraic, probabilistic, or other machine learning 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" from the previous example. Associated with each sensitive content type is a disclosure policy created by the user that specifies what roles must be satisfied in order to release the content to the recipient.

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 specifies 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 an individual query q, according to the classification method in use. The predefined sensitive content type name is returned if the content matches the query; otherwise, null is returned.

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 module maintains 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 performing the conjunction of each policy p associated with every $t \in T$.

To provide administrative scalability, the model will leverage a role-based access control model [19]. 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 associated semantics.

In my model, access control policies for sensitive content types are represented as role expressions, according to the grammar in figure 5. 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 the logical Boolean operators $\wedge$ (AND) and $\vee$ (OR). Role expressions are considered satisfiable if there exists some interpretation for which the logic evaluates to true.

$$
\begin{aligned}
\textit{Expression} &\rightarrow \textit{Role} \\
&\mid \textit{ComplexExpression} \\
\textit{ComplexExpression} &\rightarrow (\ \textit{Expression}\ ) \\
&\mid \textit{Expression Op Expression} \\
\textit{Op} &\rightarrow \wedge \mid \vee
\end{aligned}
$$

Figure 5 - Role expression grammar in BNF

The conjunction of each relevant sensitive content type's role expression creates the logical role expression for the final disclosure policy; nevertheless, the policy may be further simplified. These generated policies can be entailed into more precise expressions without a loss of semantics by applying Boolean algebra rules, such as the absorption, distributive, and idempotent laws. For example, the absorption law states that $a \wedge (a \vee b) = a \vee (a \wedge b) = a$, the distributive law allows $(a \wedge b) \vee (b \wedge c) = b \wedge (a \vee c)$, and the idempotent law implies $a \wedge a = a$ and $a \vee a = a$. Once the final disclosure policy has been generated and simplified, it is forwarded to the client's security agent, to be used in establishing trust with the server.

3.3. Trust establishment

Once an access control policy has been dynamically generated and 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, pre-existing relationship, 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 that the other party may possess instead of methods consistent with a traditional identity based access control model.

An important design decision is determining how the client security agent discovers and interfaces with the server's agent. 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, neither party can rely on an existing connection because the negotiation may take place before the client makes first contact with the server. Standard conventions need to be adopted to prescribe the relationship between clients and servers and their associated representative trust establishment agents. The solution cannot require clients to disclose, even inadvertently, sensitive information about the service request they intend to submit.

There exist many possible solutions for security agent discovery. To illustrate these, 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 SSL/TLS protocol [7][18] 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 [10].

This can be expanded to allow clients to initiate trust negotiation in the establishment of a TLS connection before any sensitive data is transmitted. Third, web servers could run a separate web service or employ a third party to provide trust negotiation by redirecting the client to a suitable security agent when trust negotiation is requested. This decouples the web server from having to execute trust negotiation responsibilities.

Chapter 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.

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

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

In order to demonstrate that the proposed system is general-purpose and extendible, I have implemented a prototype of the access control model for dynamic client-side content that supports two different usage models. The prototype supports web applications that use the Hypertext Transfer Protocol (HTTP) and email applications that use the Simple Mail Transfer Protocol (SMTP) and Post Office Protocol (POP3). These protocols are the most prevalent on the Internet and frequently carry sensitive client data. Despite their similarities, these protocols differ significantly with regard to

their communication paradigms. 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.

My 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, I 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 [26]. TrustBuilder has been developed by the Internet Security Research Lab at Brigham Young University as a standalone module for trust negotiation. It acts as a security agent for a single entity and runs as a SOAP service.

Since TrustBuilder authorization mechanisms were originally designed to protect static server resources, minor modifications were made so that it could perform content-triggered trust negotiation. To enable this, a new method was added to the TrustBuilder interface that allows clients to pass dynamically created policies for evaluation by its compliance engine. These policies represent the disclosure requirements for a sensitive message.

4.1.1 Content Classification

The creation of these policies requires the successful classification of the client's content. For my implementation, I experimented with three different classification models: Boolean, vector space [20], and fuzzy set [14]. 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. I chose to implement the fuzzy set model described in [14]. This method simulates a thesaurus made up of keywords from existing documents that provide partial matching for query terms. Traditionally, this model is very accurate, with only minor cases of false positive results. The disadvantage of my implementation was the requirement for large amounts of training data.

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. My system forms queries by training on

relevant sensitive documents. While this approach is more flexible than the Boolean method, a small percentage of misclassifications occur, resulting in false positives and negatives.

4.1.2 Content Misclassification

There are two possibilities for misclassifications in this 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 occurs 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.

4.2. HTTP

On the World Wide Web, web browsers request resources from web servers through the Hyptertext Transfer Protocol (HTTP), a lightweight, stateless, text-based, application-level protocol. HTTP is built on a request / reply model where the client is always the initiator of a transaction. Though HTTP is a simple text based protocol, it is capable of transporting a variety of data types and formats.

4.2.1 Sensitive Content in HTTP

There are several instances where a web client could intentionally or inadvertently disclose sensitive data. The most prevalent and easily recognizable example occurs when a user fills out a web form and submits it to a server. The World Wide Web is replete with such forms in web pages, and though not all request potentially sensitive client information, all have the capability to receive and mishandle it. When these forms are submitted, the client's data is transferred in the body of the HTTP request or embedded in the URL, depending on whether the web form uses the POST method or GET method, respectively. In addition to text entry, file upload is also allowed in these web forms. When a user uploads a file to the server, it is transferred in the body of the request message. URLs with embedded sensitive data are of special concern because they can be disclosed in other media beside traditional HTTP. For example, a user could forward a URL with sensitive information to someone via an email.

Even if the URL does not contain embedded client content, the web request could be sensitive. The actual 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 make certain assumptions about the client.

A web request could inadvertently disclose sensitive information whenever the HTTP `Referer` header is present. This header contains the entire URL of the referring web page. Though browser specific, the header is typically sent when requesting an inline web page object, such as an image, or when a link on a web page is clicked. This header presents a serious privacy vulnerability, since the entire URL from the referring page is included, which for reasons previously discussed, may be sensitive. In the case of

```
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
From: joesmith@byu.edu
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 6 - HTTP request with sensitive information in the referer header, cookie, and body

inline web page objects, requests may be made transparently to untrusted servers without the consent or acknowledgement of the end user.

Another HTTP header that presents a possibility for sensitive disclosure is the `From` header. This header gives the email address of the user, which the user may want to keep private. Though it is included in the HTTP specification, this header has seen little use and is rarely implemented in any browser.

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. Typically, cookies hold keywords or tokens that alone mean nothing, but denote something about a specific user to the server, such as a session identifier. 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 other servers within the originator's domain. Figure 6 illustrates an HTTP request with several pieces of sensitive information included in the referer header, cookie, and content body.

4.2.2 HTTP Extensions for Trust Negotiation

In order to perform trust negotiation within HTTP, the protocol must be extended. The current HTTP specification provides two forms of authentication, Basic and Digest. Both occur in-band within HTTP requests and responses, using the `Authorization` and `WWW-Authenticate` headers respectively. Within traditional HTTP, the server always initiates authentication by setting the response code to 401 (Unauthorized) and including the `WWW-Authenticate` header. The field values of this header include the authentication scheme used by the server, such as Basic or Digest, and any other parameters applicable to the requested URL, such as security realm. In response, the client sends its original web request with the addition of the `Authorization` header, which includes the authentication scheme used by the client and its username and password in plaintext or message digest form. If the client successfully authenticates, then the resource is released; otherwise, the server repeats its previous unauthorized message.

For the purposes of trust negotiation, I have created a new authentication type, in addition to Basic and Digest, called TrustNegotiation that carries credentials and policies

between client and server instead of usernames and passwords. To ensure compatibility with existing software, both the `Authorization` and `WWW-Authenticate` headers are still used for the client and server respectively; however, their field values are supplemented with credential and policy sections. These allow a party to reveal data necessary to carry out trust negotiation.

While HTTP authentication is traditionally server initiated, content-triggered trust negotiation assumes that clients can instigate a request for authentication. This requires that the web server be prepared to examine all incoming requests to determine if trust negotiation is required by the client. In my implementation, this is simply done by adding a module to a web server that filters all incoming requests, looking for the `Authorization` header. If the authorization method is Basic or Digest, then the web server is allowed to continue processing the incoming request; however, if its method is TrustNegotiation, the module will reply with a trust negotiation message within a `WWW-Authenticate` header or redirect the client to a URL on the server that provides trust negotiation.

It is important to note that the resource request that the client ultimately wants is not disclosed when it performs trust negotiation in HTTP since the request may contain sensitive information. Rather, the client simply indicates the server's document root "/" as the requested resource portion of the HTTP request, since it will be ignored when a server supports trust negotiation. This method provides backward compatibility with existing HTTP servers. For instance, when a client sends the `Authorization` header with a list of credentials and policies to a particular server that does not support trust

negotiation, the server will simply ignore the header and reply in context to what it perceives as a normal HTTP request for the server's document root.

Because of the potentially sensitive nature of credentials and policies, SSL is used to ensure their confidentiality and integrity. If the client's initial request for trust negotiation occurs on an unsecured channel, the web server should indicate by HTTP redirection where the client can connect for an encrypted session. If the client is not redirected to a secure location, then it terminates the trust negotiation.

4.2.3 HTTP Proxy

There are several options to add the functionality of trust negotiation to a web browser. First, a proxy server on the client's machine could intercept outgoing 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. When protecting an 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 machine. It also permits transparent interception of all outgoing requests, with potential to dramatically improve 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 my implementation, I built a proxy server that provides a flexible test platform for all types

of browsers and scenarios. I ran this proxy locally on the client's machine as this provided a much simpler experimentation tool than a browser plug-in.

Figure 7 - Traditional HTTP client and server

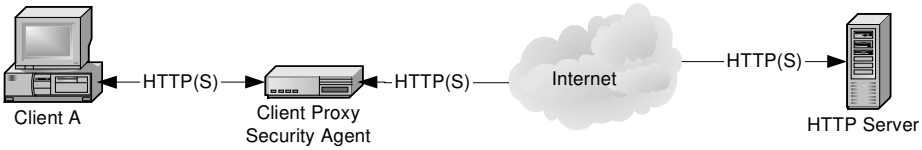


Figure 8 - Traditional HTTP client and server with security agent proxy

When a sensitive request is detected by the proxy, it withholds the original request and generates a new trust negotiation request including the appropriate headers previously discussed along with any credentials or policies that may be applicable to the first round of negotiation. After this is sent, the proxy prepares to receive the server's response. When the response is received, the proxy parses out any credentials or policies included by the server. This information is then sent to TrustBuilder which indicates whether the original content disclosure policy has been fulfilled or what other credentials and policies should be sent from the client to the server. This process continues iteratively until the server successfully negotiates or the proxy deems failure. When trust is established with the web server, the proxy forwards the original HTTP request to the web server and relays the subsequent response to the client. In the case of a failed trust negotiation, the proxy alerts the user with a message in a dialog box which explains the problem and gives the user the option of overriding TrustBuilder's decision and forwarding the sensitive content regardless. This capability is analogous to SSL when a

user receives a prompt indicating that the server's certificate in invalid but is given the option to continue the session anyway.

4.2.4 Security Considerations

Traditional web browsers make SSL connections with web servers through proxies by sending a HTTP CONNECT request to the proxy. This command indicates that the proxy should connect to a specified server and merely act as a tunnel between the two hosts, relaying each of their generated messages. The proxy is required to respond to the client indicating connection 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 content because of the cryptographic properties of SSL. The relay of these messages is shown in figure 9. This feature is detrimental to the system proposed here, since the proxy must be able to view and parse the content in plaintext to determine its sensitivity.

This problem is addressed by creating a man-in-the-middle proxy [18]. 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

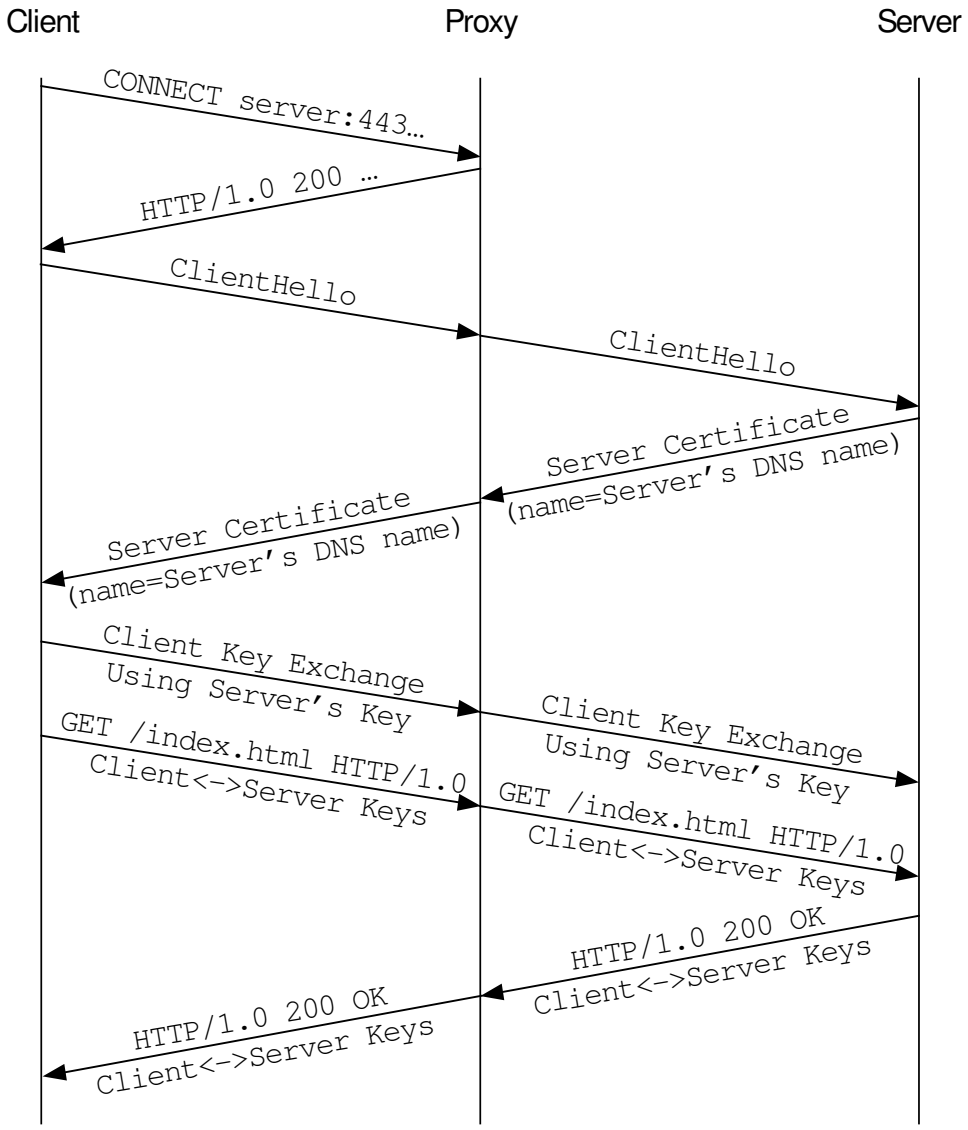


Figure 9 - Normal SSL tunneling proxy in action

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. This is done by including the web server's domain name as the common name in a certificate that is digitally signed by a trust certificate authority. These names can be

wildcard matched, thus allowing the proxy to send the name of "*" to match any website. Since no respected certificate authority would issue such a certificate, the proxy's administrator must create a local certificate authority that issues the wildcard 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 for any website, 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. After examining the content, 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. The exchange of messages in a man-in-the-middle proxy is shown in figure 10.

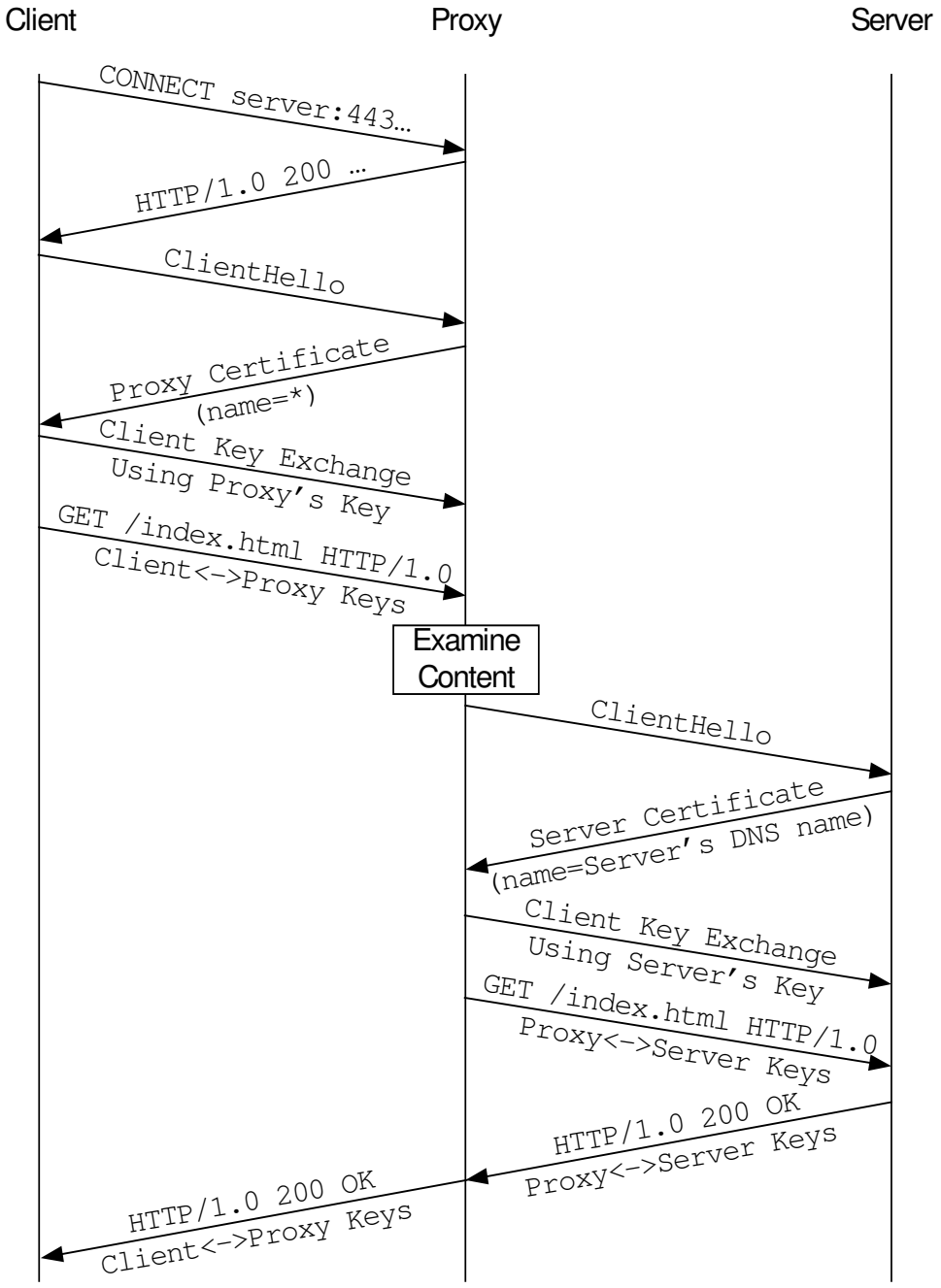


Figure 10 - Man-in-the-middle proxy with content classification in action

4.2.5 Performance Considerations

While my implementation was meant to serve as a proof-of-concept and was not tuned for performance, it is still useful to note certain performance characteristics to determine the feasibility of such a system in a production environment. To assist in determining the characteristics of the system under varying loads, queuing analysis was used. For the HTTP model, I assumed random arrivals of incoming HTTP requests with exponential service times or an M/M/1 queue in Kendall notation. The main parameters involved are average service time and average arrival rate. For the average arrival rate, I assumed that a user would spend about 30 seconds on a given web page, making the average arrival rate 0.03 requests per second. In order to calculate average service times, I created three different categories: web page request without a proxy, web page request with the proxy, and a web page request that required a two round trust negotiation. The web page request without a proxy was used to give a baseline measurement. I then began to measure the response time for receiving a given web page using all three techniques. Additionally, in order to fully test out the content classifiers in all scenarios, I used varying sizes of web requests and created a Boolean, fuzzy, and vector space model query. The size of the content in the web requests ranged from 47 to 8362 bytes.

Table 2 shows the average response times observed for one requested web page. Table 3 considers the fact that a web page is actually made up of several requests for inline objects, such as images. For this table, I estimated that each web page contained an average of 5 additional items that would require web requests. The service time for the proxy with trust negotiation in table 3 assumes that only one of the five requests will require trust negotiation. When no trust negotiation is present the proxy introduces very

minimal overhead, even though the incoming content is being parsed and classified by the three different classification queries.

Service Type	Average Service Time
No proxy (direct connection)	0.08 sec
Proxy – no negotiation	0.1 sec
Proxy – negotiation (2 step negotiation)	2.1 sec

Table 2 - Average response times for 1 requested object

Service Type	Average Service Time	Server Utilization
No proxy (direct connection)	0.4 sec	1.2%
Proxy – no negotiation	0.5 sec	1.5%
Proxy – negotiation (2 step negotiation)	2.5 sec	7.5%

Table 3 - Average response times for 5 requested objects

As can be observed from the table above, the server reaches the highest utilization when the proxy performs a trust negotiation. Figure 11 below shows the division of time within a two-step trust negotiation. The largest amount of time is spent in the strategy

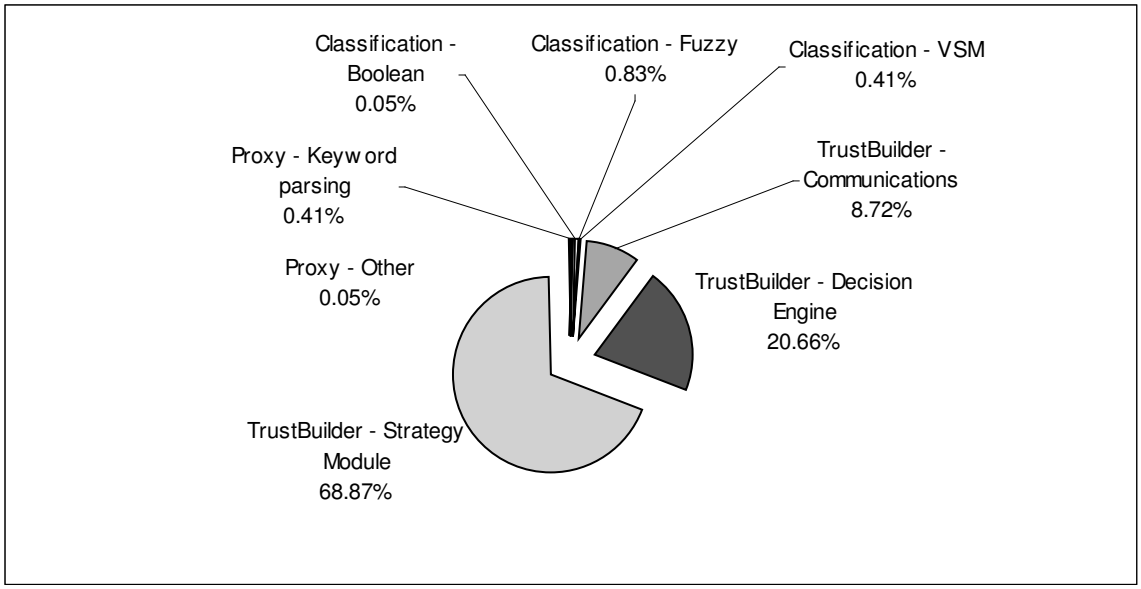


Figure 11 - Distribution of time spent in different parts of HTTP proxy and TrustBuilder during a two-step trust negotiation

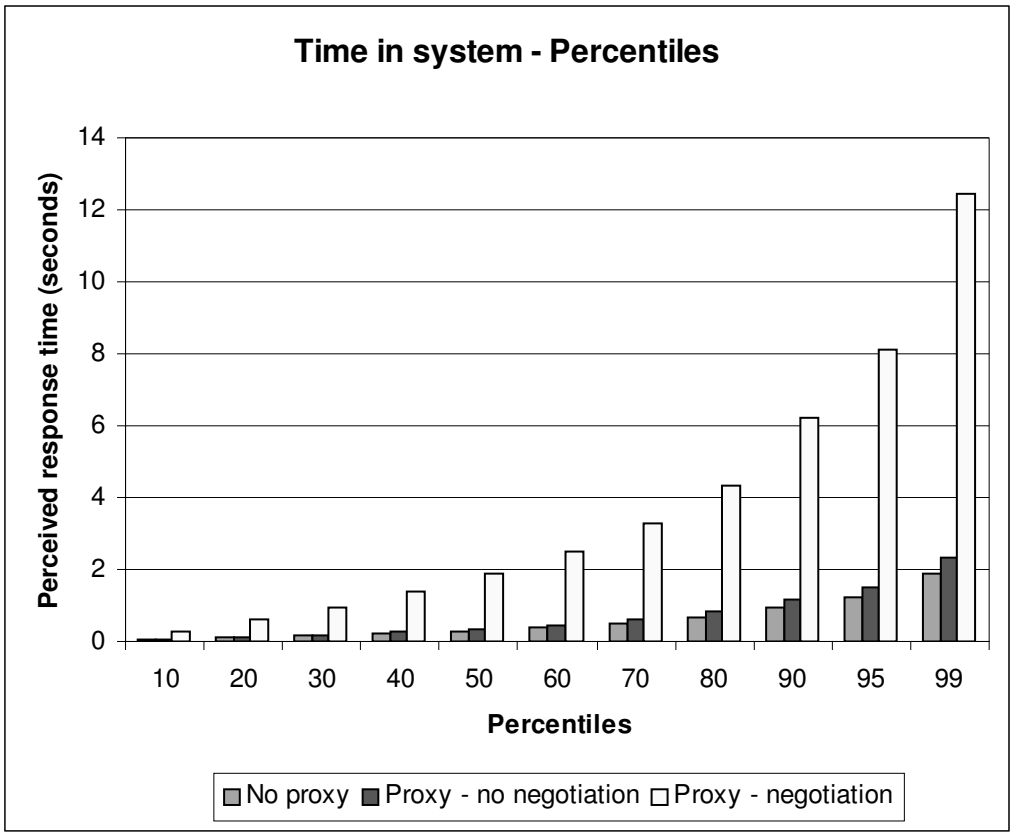


Figure 12 – Percentiles depicting the maximum response time to service a web request

module where TrustBuilder relies on the IBM Trust Establishment package. Specifically, most of the time is spent in the routines that format X.509 certificates to a printable format. The package is purely a research prototype and not optimized for performance. An industry release implementation would be expected to significantly reduce this overhead.

From queuing analysis other statistics can be derived that assist in understanding the dynamics of the system. For example, web requests that require trust negotiation are serviced almost instantly since they have a mean waiting time of 0.2 seconds. Additionally, the mean response time for a web request with trust negotiation is 2.7 seconds. The probability that such a request will be no more than the average service

time of 2.5 seconds is 60.3%. To compare the throughput of trust negotiation web requests, consider the graph in figure 12. This graph demonstrates the perceived response times given for a range of percentiles for the three test groups. For example, 50% of the web requests will have a response time no greater than 0.281 seconds when no proxy exists, 0.352 when the proxy is used, and 1.873 seconds when trust negotiation is performed. It is interesting to note the nonlinear increase in response times when trust negotiation occurs. Improving the overall trust negotiation performance is an important area for future work. Appendix A has additional information on the queuing theory used.

4.3. SMTP and POP3

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. This forwarding does not have to happen in real time. SMTP is a "store and forward" protocol, meaning that each MTA will store received email messages and then forward and delete them at some future time. 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). POP3 simply retrieves and deletes email messages from mail servers while IMAP offers more advanced email retrieval options.

4.3.1 Sensitive Content in Email

Typical email messages are composed of a header and a body. The header is a collection of fields that give information about the message's delivery and contents. The

```
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 13 - Example of email with sensitive content

message body may consist of a single message or a multipart collection of messages and attachments. Because of the wide variety of content that can be sent via email, ample opportunities exist for clients to disclose sensitive content. The most obvious location of this information is in the message body or attachments, where clients can input any type of information. Email message can also include sensitive content within its headers. For example, the subject header may state something private. Figure 13 illustrates a multipart message with sensitive content included in the email body and the attachment.

4.3.2 Email Proxy

Much like the previous HTTP example, I have inserted a security agent proxy to examine outgoing SMTP messages for sensitivity. Like the HTTP proxy, only the common places for sensitive content are examined and most other headers are ignored. When an outgoing email message is deemed sensitive, trust negotiation is initiated by

withholding the sensitive message and sending a trust negotiation mail message. These messages are MIME multipart email messages containing credentials and policies. In other words, credentials and policies are transported in email message as attachments. To distinguish these trust negotiation mail messages from normal email messages, a new mail header, `X-trustnegotiation`, is added to the outgoing message. Since this header is an extension header beginning with `X-*`, mail agents that do not support trust negotiation will treat the message just as a traditional email. The security agent proxy on the recipient side scans incoming emails when messages are retrieved from the mail server via POP3. When a trust negotiation message is received, the recipient's security agent 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, as in the HTTP example, the message is not sent and the sender is given the option of overriding TrustBuilder's decision.

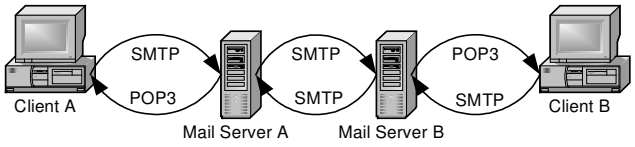


Figure 14 - Traditional SMTP / POP3 message relaying

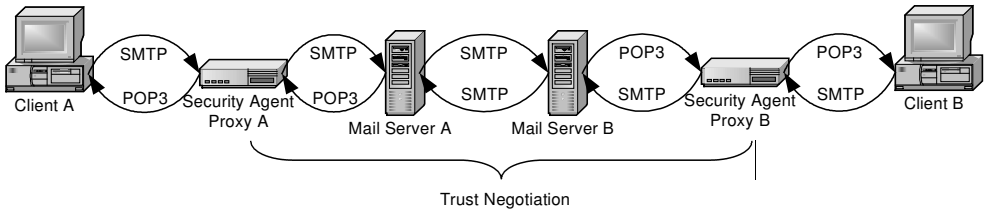


Figure 15 - Traditional SMTP / POP3 message relaying with security agent proxies

While HTTP messages have a single sender and recipient, SMTP allows single users to indicate multiple recipients. This point to multipoint communication occurs when additional email addresses are supplied in the TO, CC, or BCC fields. In this case, each receiving party must authenticate individually to the sender before the sensitive message will be sent to them.

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. Backwards compatibility with existing mail systems is thus enabled by inserting each credential and policy as attachments within the email message. Additionally, a brief text message is included in the body of the email explaining that the received message is a trust negotiation message and that some party would like to negotiate trust with the recipient.

4.3.3 Security Considerations

Due to the sensitive nature of a user's credentials and policies, care should be taken to ensure their safe transport throughout a network. 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. A variation on this can be created by using SSL between each mail-forwarding host. This is not an attractive alternative since mail may reside unencrypted on the mail servers while waiting to be forwarded or retrieved making them vulnerable to attack. A more effective method is to encrypt the

email message body instead of the communication channel. Since only the sender and receiver have to perform cryptographic operations, server overhead is diminished along with a minimized reliance on infrastructure to provide security.

Current methods for email message security include S/MIME, PGP/MIME, and identity based encryption, all of which provide confidentiality with encryption and integrity and authenticity with digital signatures. Both S/MIME and PGP require that the email sender obtain the public key certificate of the recipient to encrypt the session symmetric key that will be used to encrypt the message. For this to happen in an automated fashion requires an established infrastructure such that the sender can perform the associated lookup. In contrast, identity based encryption 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 this technology is not as widespread as S/MIME or PGP, the Stanford Applied Cryptography Group has created a reference implementation called Identicrypt [5].

4.3.4 Performance Considerations

Because of its communication characteristics, the email proxy faces different performance challenges than the HTTP proxy. Foremost, email is not a performance-critical application since email transport does not have to happen in real time. The throughput of email messages is typically a function of the MTAs that will relay the mail messages. Because of the "store and forward" delivery paradigm, there are no guarantees of when the message will be delivered.

While the HTTP and email proxies differ in several aspects, the do have in common the content classification libraries and routines. Thus, any increase in content classification performance will bring benefits to both implementations.

Chapter 5

Related Work

There are several recent research projects that explore trust establishment using digital credentials. RT [25] is a role-based trust management framework that has been used for trust negotiation. Additionally, Bonatti and Samarati [4] have 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. X-Sec [3] 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. The work presented in this paper is the first use of content-triggered trust negotiation to permit a client to initiate a trust negotiation with a server.

There are identity management efforts underway to better handle user's sensitive personal information, such as the Liberty Alliance [13]. 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 my 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. My approach may also be useful to automatically manage introductions between affinity group members under the Liberty Alliance. Trust negotiation is aimed at handling introductions between strangers and is not a replacement for authentication in closed systems.

The Platform for Privacy Preferences (P3P) [16] is a project developed by the World Wide Web Consortium, which allows websites to express their privacy policies in an automated fashion to clients. Such policies allow the client to know who is collecting the data, what is being collected, why it is being collected, and what will be shared with others. The disadvantage is that a trusted third party does not assert the websites' information practices.

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, which 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. Like the proxies presented in this thesis, 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 thesis by operating on incoming messages instead of outbound content.

Chapter 6

Conclusions and Future Work

Conventional access control mechanisms are primarily designed to protect the release of known resources to known users. This thesis has looked at the scenario when neither the resource nor user has been introduced into the system. Specifically, the research presented has focused on transient content that is generated by a client and released 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 must be made at runtime between a policy and the transient content. In addition, the sender and receiver lack a pre-existing relationship, and thus they are not able to authorize a sensitive transaction based on identity.

6.1. Contributions

To solve these problems, this thesis presents an access control model for regulating disclosure of dynamic client content and 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 policies dynamically and initiates trust negotiation. The iterative disclosure of access control policies and associated attribute-based credentials gradually builds necessary trust, which allows the client to confidently 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, I have extended TrustBuilder [26] to perform content-triggered trust negotiation, expanding the use of trust negotiation beyond the protection of static sensitive server resources. My 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. On the other hand, SMTP is an asynchronous, message-routed protocol used in the delivery of MIME-formatted email messages. The underlying differences of these distinct 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 primary contributions of this thesis will appear in [11].

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 will also 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 my implementation makes use of three accurate classification techniques, false positives and false negatives may occur.

6.2. Future Work

My proposed model provides a new application area for document classification research. Further experimentation and research within this area may lead to greater security for clients who use my proposed model. Examples of possible future work in document classification that would impact my proposed model include improved filtering algorithms and error reporting.

In the future, my 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 thesis focused on classifying outbound client content and authenticating the recipient when the content is sensitive. In the future, the client's content classifier can be applied 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 the volume of inbound content is proportionally greater than outbound, such as client HTTP traffic.

Though measures have been taken to ensure confidentiality within trust negotiation, there are items inherent to the communication protocol, such as email and IP addresses, that prevent total anonymity. Email addresses are especially revealing and can be used to contrive much information about an individual. Another future development

for the system would be 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 a new from address created by the remailer. Several of these remailers can be chained together to increase the anonymity of the sending party. By using this technology, the initiator of a trust negotiation using email could perform trust negotiation under total anonymity.

References

[1] M. Ackerman, L. Cranor, and J. Reagle. Privacy in E-commerce: Examining User Scenarios and Privacy Preferences. *1st ACM Conference on Electronic Commerce*, Denver, Colorado, November 1999.

[2] T. Barlow, A. Hess, and K. E. Seamons. Trust Negotiation in Electronic Markets. *8th Research Symposium in Emerging Electronic Markets*, Maastricht, Netherlands, September 2001.

[3] E. Bertino, S. Castano, and E. Ferrari. On Specifying Security Policies for Web Documents with an XML-based Language. *6th ACM Symposium on Access Control Models and Technologies*, Chantilly, Virginia, May 2001.

[4] P. Bonatti and P. Samarati. Regulating Service Access and Information Release on the Web. *7th ACM Conference on Computer and Communications Security*, Athens, Greece, November 2000.

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

[6] Y. Chu, J. Feigenbaum, B. A. LaMacchia, P. Resnick, and M. Strauss. REFEREE: Trust Management for Web Applications. *Computer Networks and ISDN Systems*, Vol. 29, 1997.

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

[8] 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.

[9] 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.

[10] 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.

[11] A. Hess and K. E. Seamons. An Access Control Model for Dynamic Client Content. *To appear in the 8th ACM Symposium on Access Control Models and Technologies*, Como, Italy, June 2003.

[12] 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.

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

[14] 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, 1991.

[15] J. Park and R. Sandhu. Binding Identities and Attributes Using Digitally Signed Certificates. *16th Annual Computer Security Applications Conference*, Los Alamitos, California, 2000.

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

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

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

[19] R. S. Sandhu and E. J. Coyne. Role-Based Access Control Models. *IEEE Computer*, Vol. 29, No. 2, February 1996.

[20] G. Salton and M.E Lesk. Computer Evaluation of Indexing and Text Processing. *Journal of the ACM* , Vol. 15, No. 1, 1968.

[21] 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.

[22] 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. *3rd International Workshop on Policies for Distributed Systems and Networks*, Monterey, California, June 2002.

[23] 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.

[24] M. Tanner. Practical Queueing Analysis. McGraw-Hill, 1995.

[25] W. Winsborough and N. Li. Towards Practical Automated Trust Negotiation. *3rd International Workshop on Policies for Distributed Systems and Networks*, Monterey, California, June 2002.

[26] 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.

[27] 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*, Vol. 6, No. 1, February 2003.

Appendix A - Queuing Analysis and Notation

The work of this thesis assumes that HTTP requests will have a random arrival rate with service times that are exponentially distributed. Incoming requests have equal priority and are serviced in a first-come-first-serve manner. Additionally, there is no limit to how many requests may be waiting to be serviced. The number of these waiting requests has no effect on the average rate at which other requests arrive. These assumptions follow precisely the requirements of a M/M/1 model in queuing theory. Listed below are the parameters and formulas that were used in this thesis for performance analysis [24].

λ	Mean arrival rate
T_s	Mean service time
ρ	Server utilization
T_w	Mean waiting time to be serviced
T_Q	Mean total time in system
$\pi(r)_{T_Q}$	rth percentile of total time in system

$$T_w = \frac{\rho T_s}{1-\rho}$$

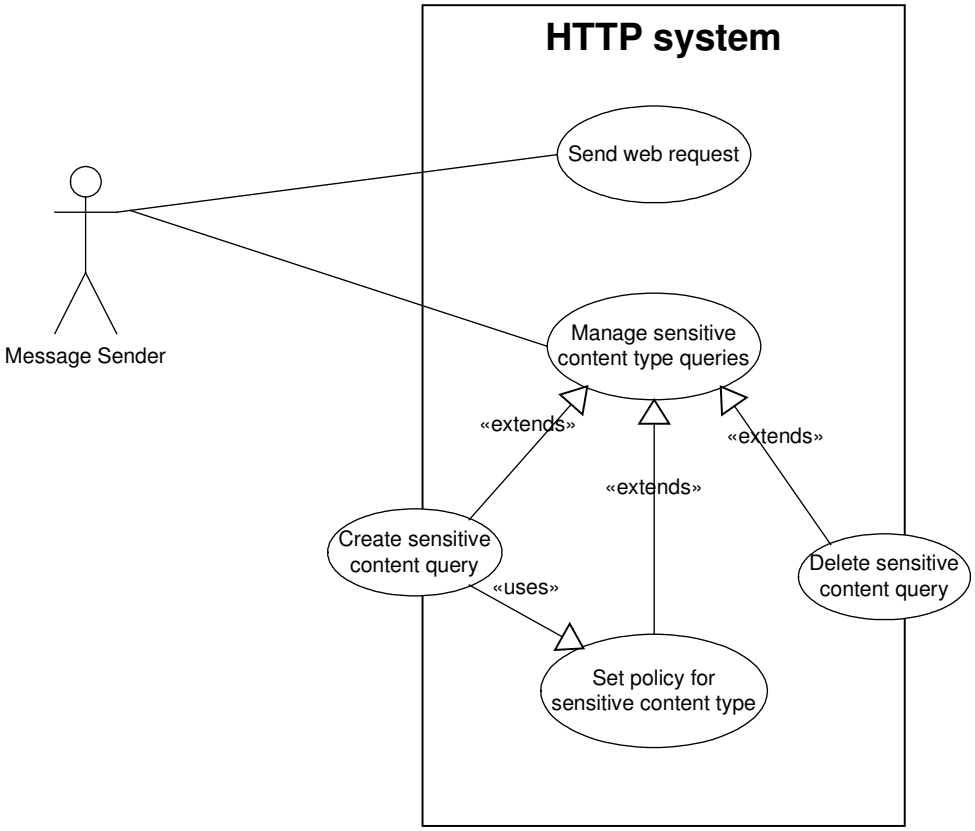
$$T_Q = \frac{T_s}{1-\rho}$$

$$\pi(r)_{T_Q} = \ln\left(\frac{100}{100-r}\right) T_Q$$

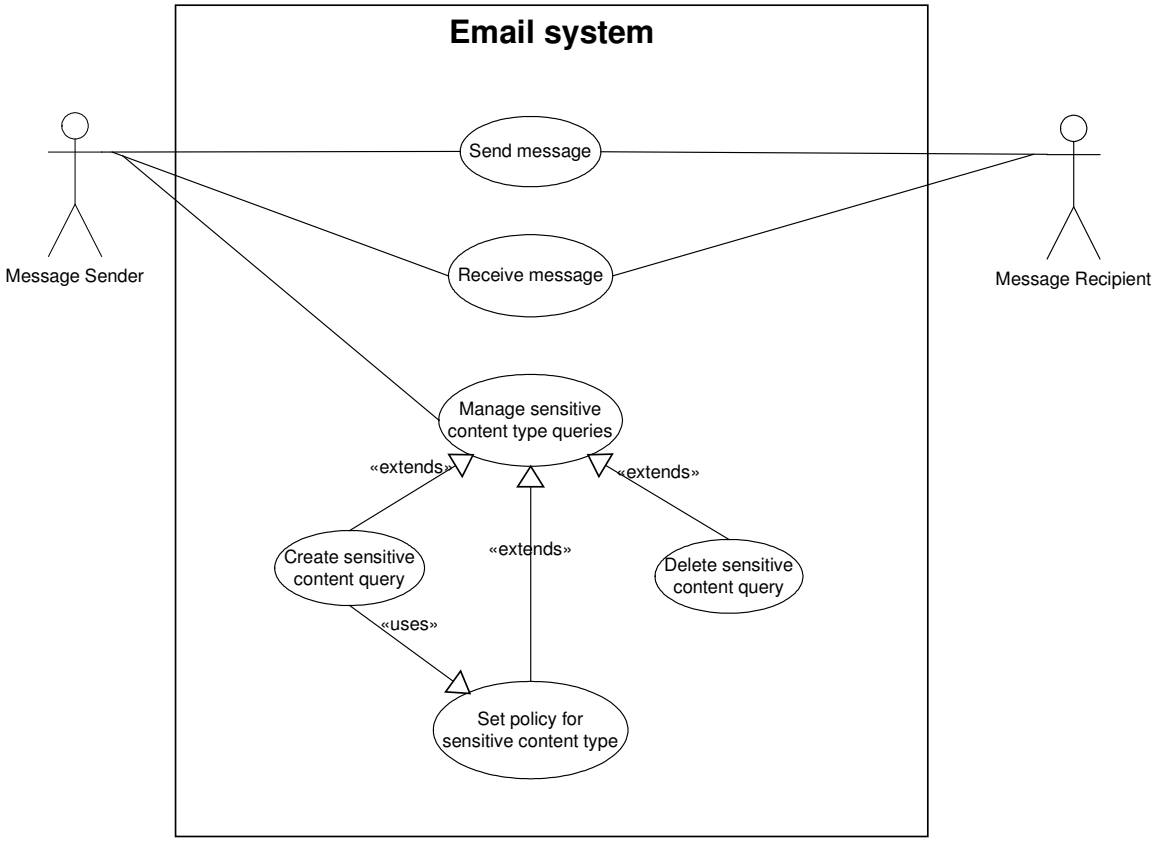
$$\text{Prob(time in system} \le t) = 1 - \exp\left(-\frac{t}{T_Q}\right) = 1 - \exp\left(-\frac{(1-\rho)t}{T_s}\right)$$

Appendix B - UML Diagrams

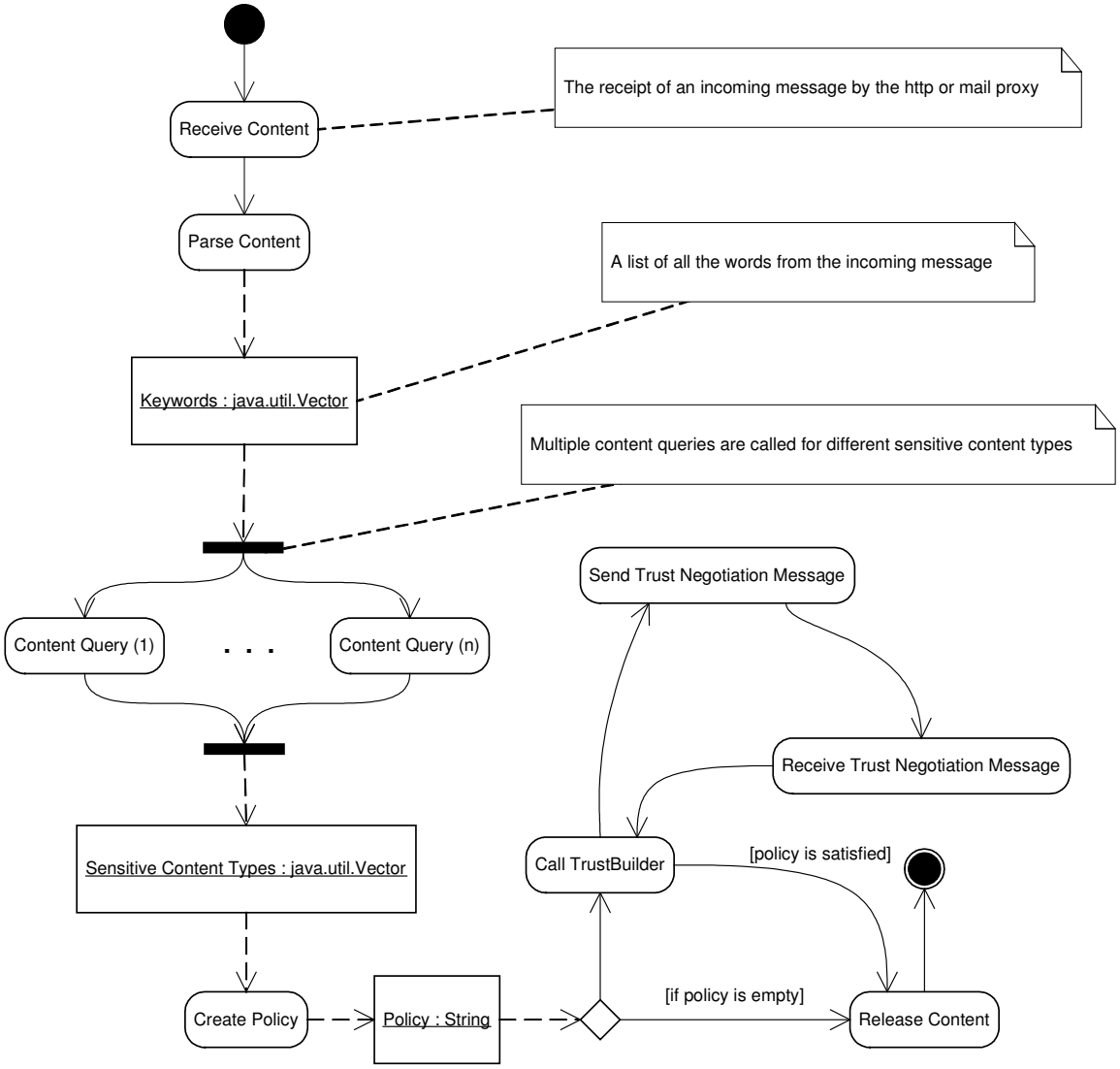
1. Use Case for HTTP System



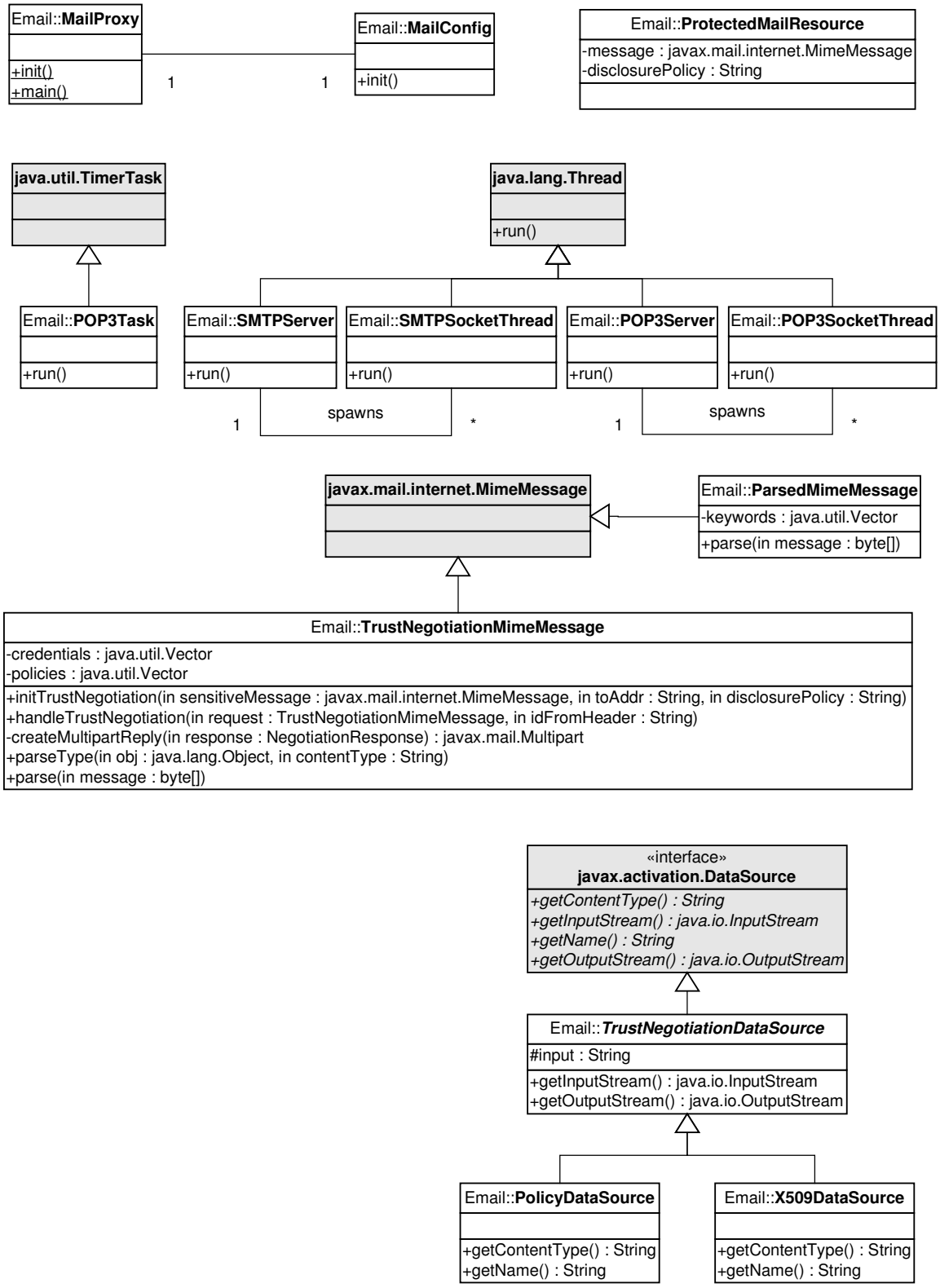
2. Use Case Diagram for Email System



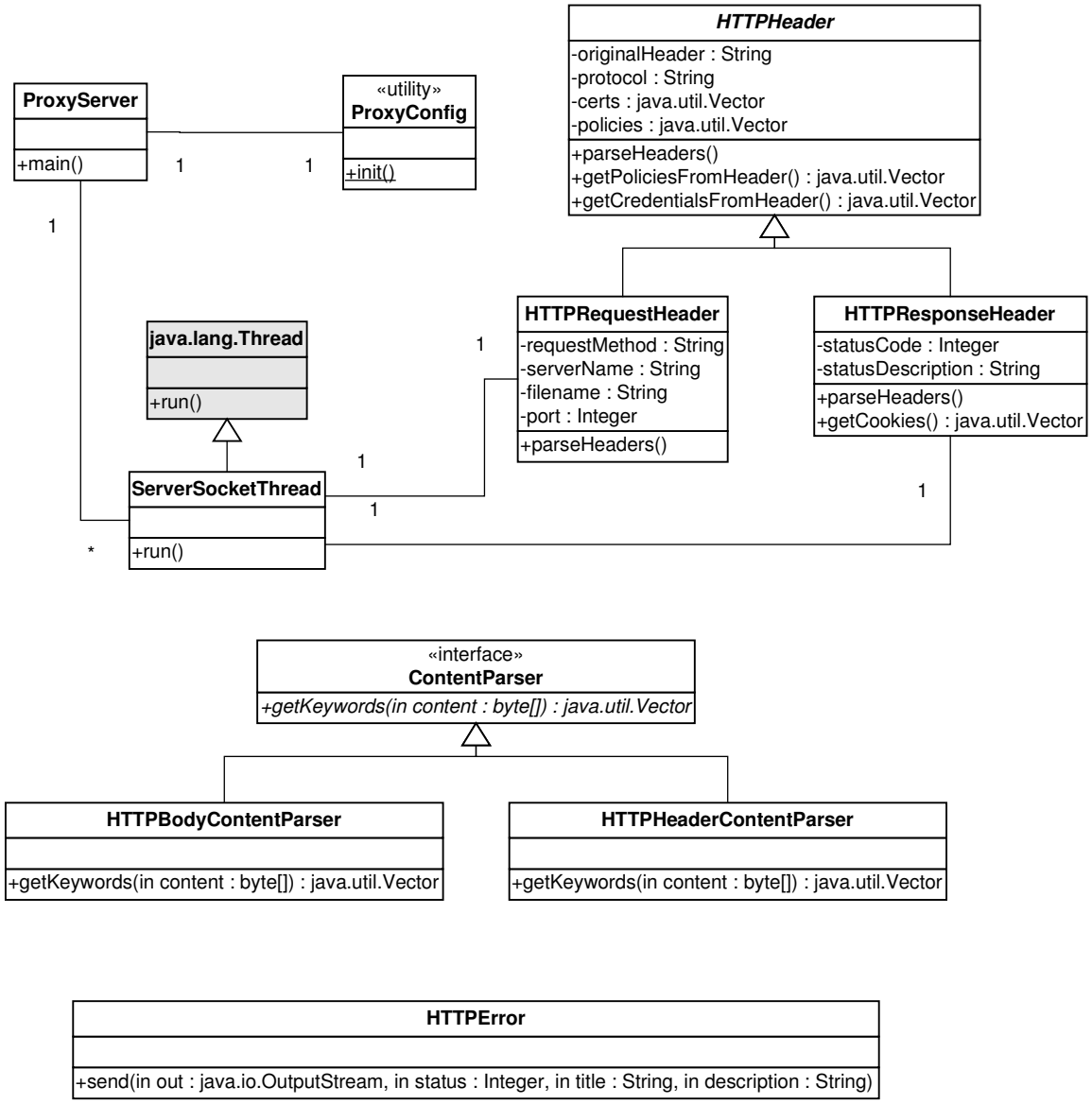
3. General Overview Activity Diagram



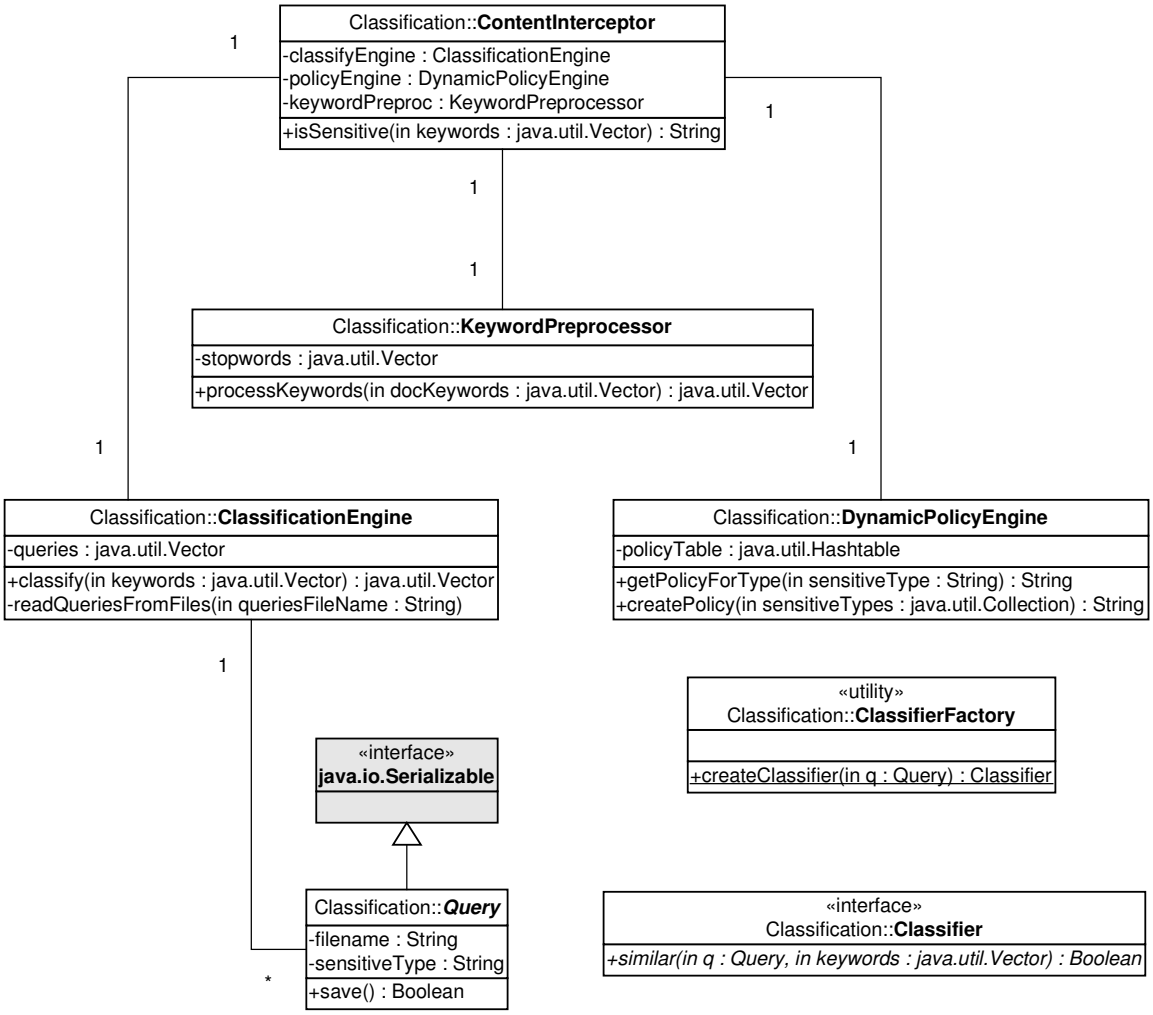
4. Email Proxy Static Structure



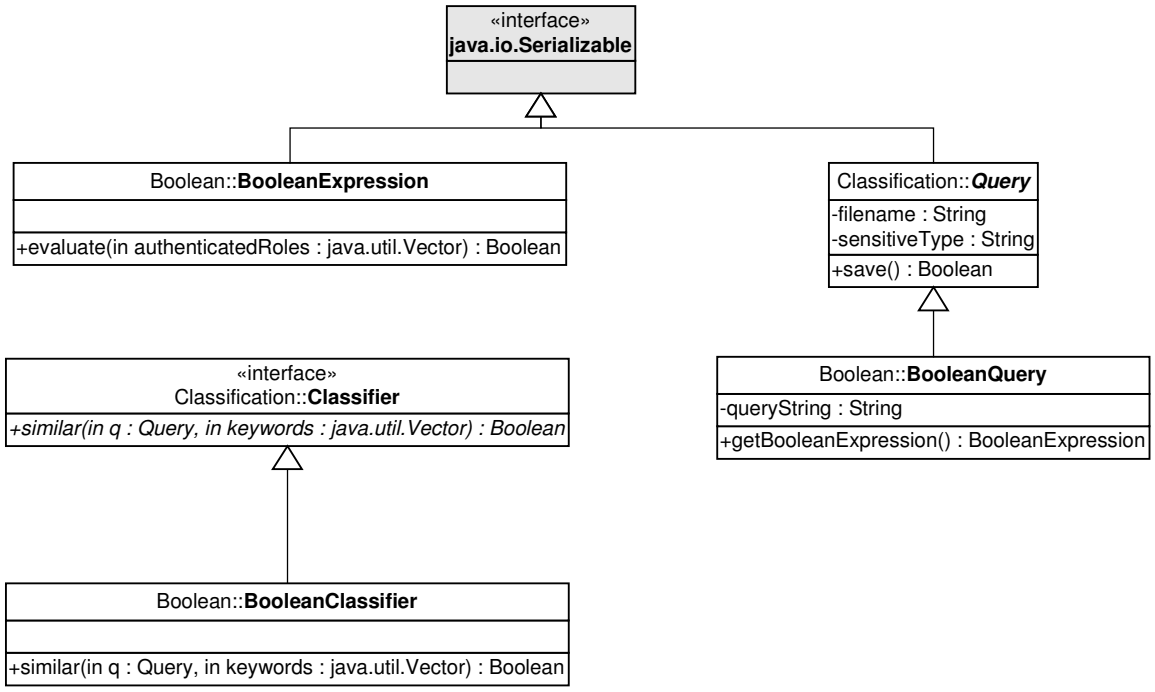
5. HTTP Proxy Static Structure



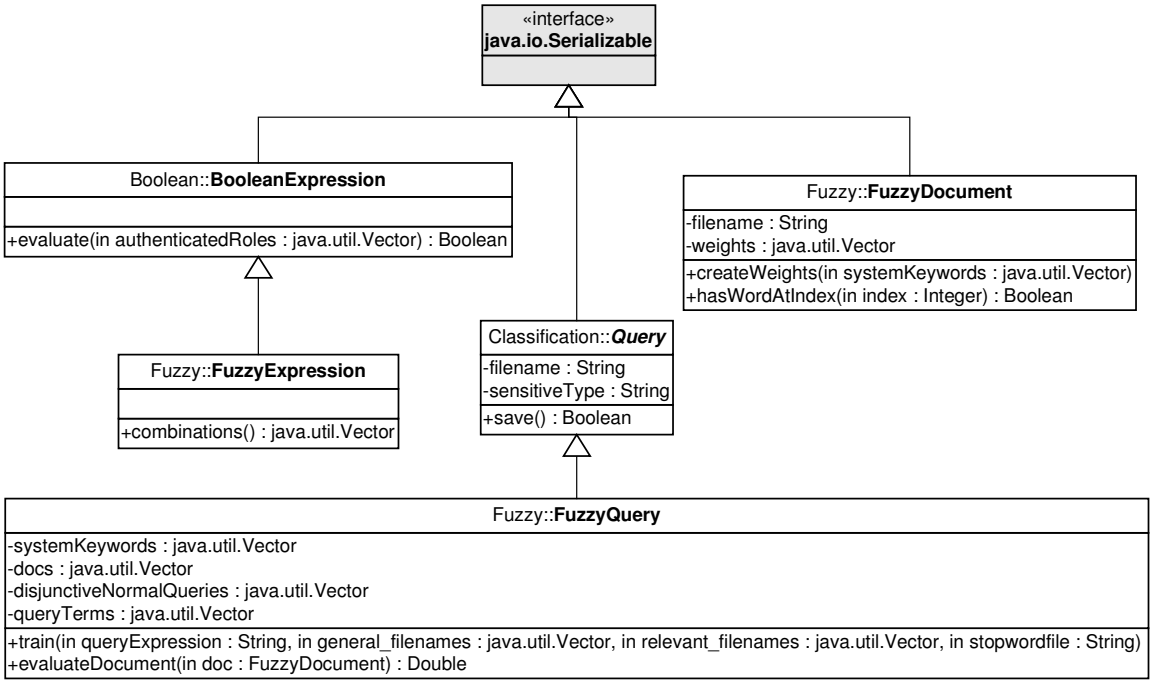
6. Classification Framework Static Structure

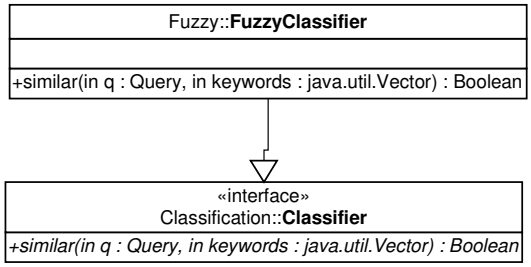


7. Boolean Classification Static Structure

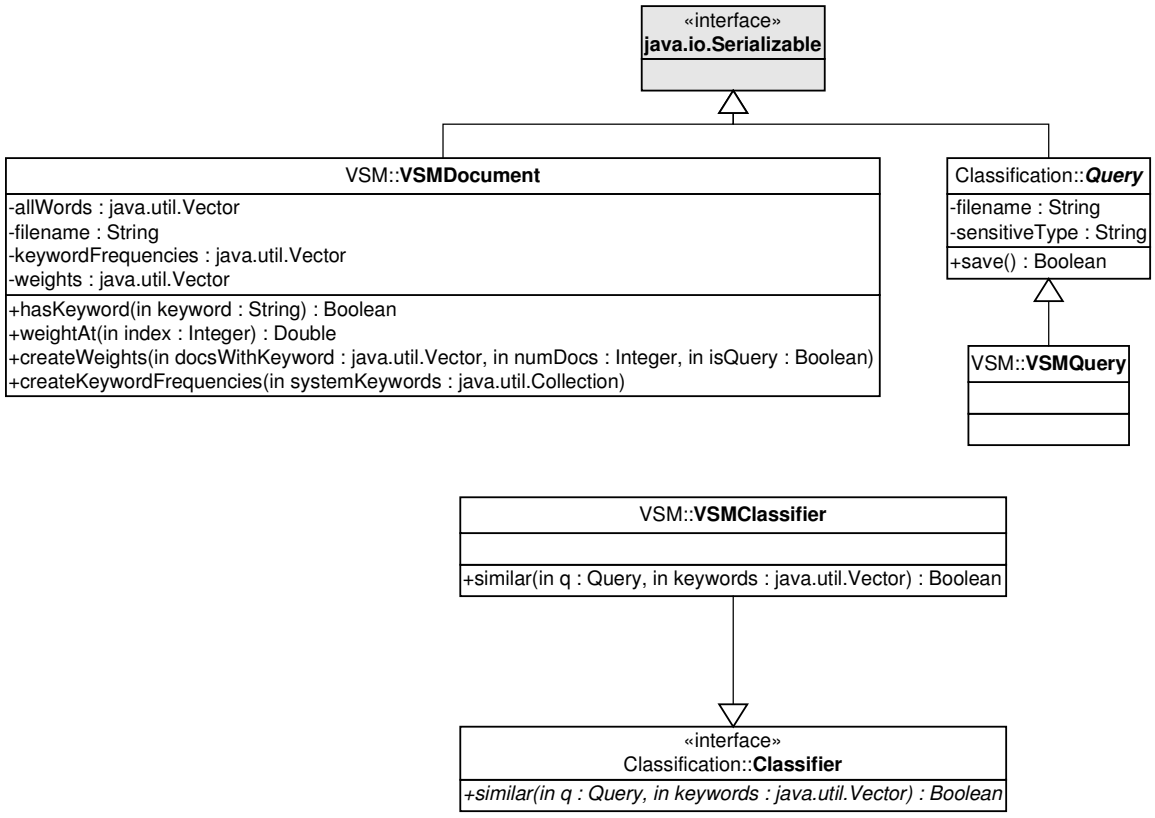


8. Fuzzy Classification Static Structure

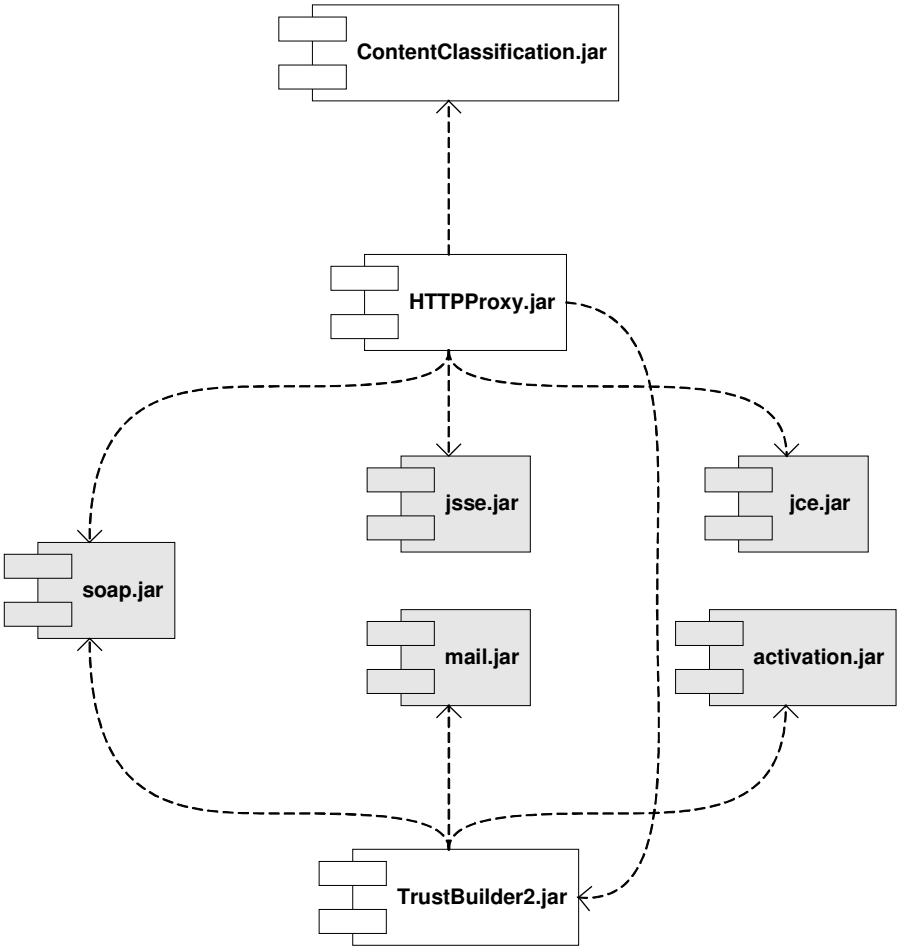




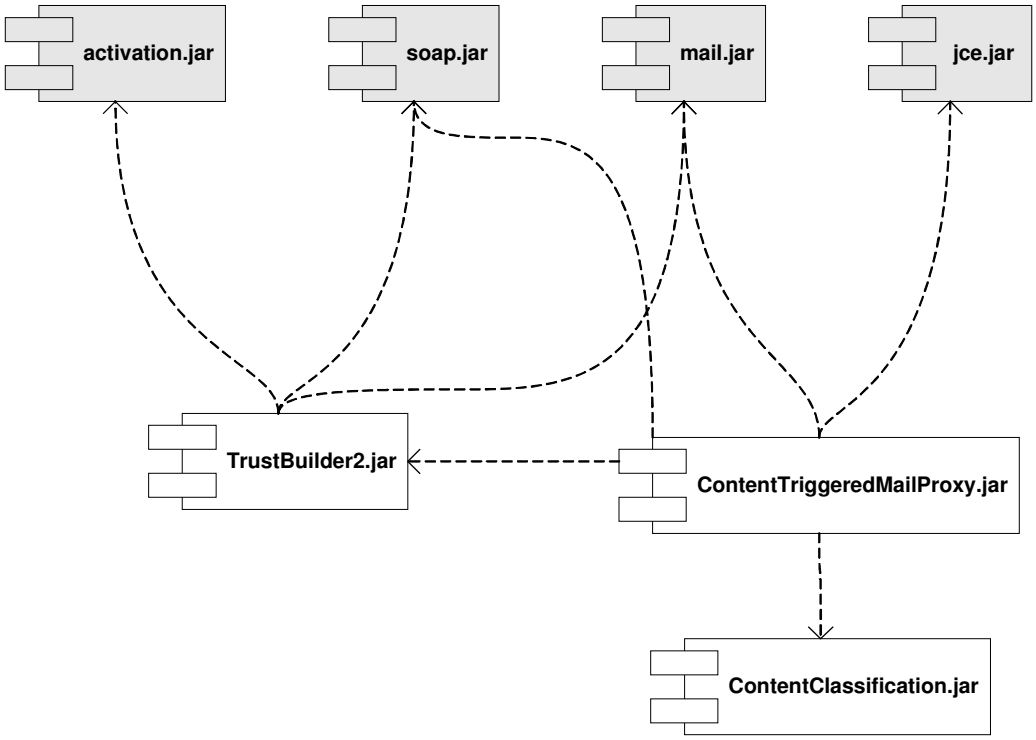
9. Vector Space Classification Static Structure



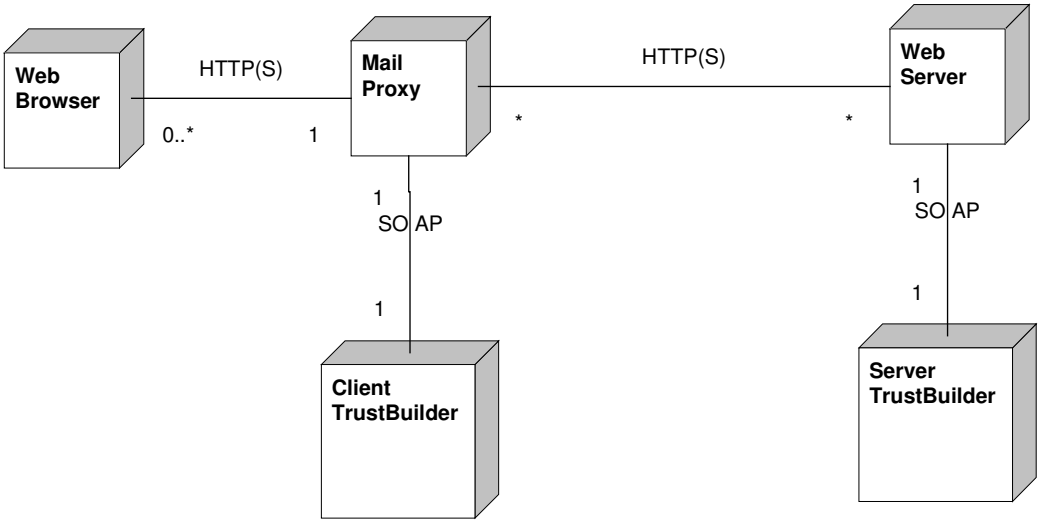
10. HTTP Proxy Component Diagram



11. Mail Proxy Component Diagram



12. HTTP Proxy Deployment Diagram



13. Mail Proxy Deployment Diagram