Switchboard: Secure, Monitored Connections for Client-Server Communication

Eric Freudenthal, Lawrence Port, Tracy Pesin, Edward Keenan, and Vijay Karamcheti
Department of Computer Science
Courant Institute of Mathematical Sciences
New York University
{freudent, lport, tracyp, woodiek, vijayk}@cs.nyu.edu

Abstract

Prolonged secure communication requires trust relationships that extend throughout a connection’s life cycle. Current tools to establish secure connections such as SSL/TLS and SSH authenticate PKI identities, validate credentials and authorize a trust relationship at the time a connection is established, but do not monitor the trust relationship thereafter.

To maintain security over the duration of a prolonged connection, we extend the semantics of SSL to support continuous monitoring of a credential’s liveness and the trust relationships that authorize it. Our implementation isolates trust management into a pluggable trust authorization module. We also present an initial design for a host-level secure communication resource that provides secure channels for multiple connections.

1 Introduction

Present tools for establishing secure communication channels such as SSH [16] and SSL/TLS [6] authenticate PKI identity; however, SSH provides no mechanisms for authorization beyond access control lists, and SSL/TLS is limited to validation of X.509 certificate chains at the time a connection is requested. In addition, these systems present serious time-of-check to time-of-use (TOCTTOU) [1] risks: No monitoring of authorization is performed after a secure connection is established.

These semantics may be reasonable when the connection is short-lived (for example, protecting the transfer of a single web page) and access control lists or X.509 certificate chains provide an appropriate security abstraction. However, these protocols can be used to provide long-term connectivity, but do not provide continuous monitoring of required trust relationships. In addition, these trust models are not well suited for applications that require enforcement of rapidly changing transitive trust relationships that span multiple administrative domains.

We believe that the tight integration of a particular trust management system into secure communication substrates (such as access control lists in SSH and X.509 in SSL/TLS) has discouraged exploitation of these techniques. For example, SPKI/SDSI [5, 12] provides a role-based abstraction and on-line certificate validation; dRBAC [8] includes mechanisms to monitor sustained trust relationships. However, the non-modular design of current SSL systems inhibits incorporation of such mechanisms.

The temporal nature of trust, which we believe is important, is not respected by present tools. SSH does not support revocation of authorization; the SSL specification describes a need for it, but no clear implementation is specified. An SSH-authorized login may extend for an indefinite period (potentially extending beyond an employee’s dismissal); an established SSL connection is not automatically closed when an authorizing certificate expires.

Finally, continuous connectivity can be utilized to authorize security relationships, and therefore connection liveness is an important security property not monitored by existing tools.

While these challenges may be manageable in small systems using administrative overrides, this approach is inappropriate for massively distributed systems. While an administrative user can easily kill all processes belonging to a dismissed employee on a handful of systems, this is impractical when the set of systems spans multiple administrative domains.

In the present paper we present switchboard, which extends the abstraction for authorized secure connections presented by SSL/TLS and SSH to include continuous monitoring of authority and liveness. Monitoring of authorization is enabled by a pluggable trust monitor component that both authorizes connections and continuously monitors the required trust relationships. Liveness is monitored by an
integral resilient heartbeat mechanism.

Two generations of switchboard implementations are described. The initial version, like TLS/SSL, is implemented as a library that establishes a distinct TCP connection for each secure conversation. The second generation, which is presently under construction, operates as a centralized host-level resource. We anticipate that this design, which multiplexes multiple secure conversations over a single TCP connection, will reduce the cost of establishing and monitoring connections.

The need for switchboard's abstractions arose directly from our efforts to develop secure middleware for massively distributed systems maintained by multiple trust authorities. The existence of the switchboard architecture has accelerated our larger research efforts (see [8], [7]) by providing a convenient abstraction and API.

This paper is organized as follows: Section 2 describes switchboard functionality. Section 3 presents the switchboard API, architecture, and implementations. Section 4 evaluates the benefits of switchboard, and Section 5 discusses related work.

2 The Switchboard Abstraction

Switchboards provide an serialized communication substrate with semantics that extend the abstraction provided by SSL. To participate in a switchboard conversation, both clients and servers must have access to the appropriate switchboard interfaces. Two generations of switchboard implementations are described below, each of which provides distinct communication abstractions that achieve similar goals.

Throughout this paper we will use the term service to refer to the exchange of information between remote machines via serialized channels. Like a conventional TCP connection, a switchboard conversation can be used for serialized object delivery, RPC, data distribution, and arbitrary programmed communication.

The term principal henceforth refers to users, programs, agents or other controlling entities which can make requests to the switchboard. Correspondent refers to the remote party to which a local principal is connected. A trust relationship is a condition that must be satisfied for corresponding entities to communicate.

We refer to the initiator of a conversation as a client and the receiver as a server. Once a conversation is established, switchboard considers them peers, and the distinction between client and server becomes irrelevant.

As in SSH, principals are identified by their PKI identities (public keys), and therefore authentication only includes validation that a subject has possession of a complementary private key. In switchboard, the process of determining whether a principal identified by its public key, and their credentials is entitled to perform some restricted action is called authorization.

2.1 Abstraction Extension beyond SSL/TLS

Extended Connection Request Specification As in SSL/TLS, potential correspondents must specify their PKI identities, credentials, acceptable ciphers, and TCP/IP addresses prior to establishing a connection. In addition, they specify an authorization module: a local software entity responsible for evaluating a potential correspondent's credentials. If a potential correspondent's credentials are accepted, the authorization module generates a trust monitor object. The trust monitor object will inform the switchboard (via a callback interface) should the correspondent's credentials become invalid.

Trust Monitoring Once a trust relationship is established, and a connection between client and server is made, trust monitors enable a switchboard to disconnect a conversation when necessary. Furthermore, the set of credentials that initially authorizes a connection may require renewal throughout a connection's lifetime to avoid disconnection caused by expiration or revocation. To support trust monitoring, interfaces are provided for both correspondents to transmit credential updates to their partner's trust managers.

Connection Liveness Monitoring A mechanism is employed to ensure connection liveness. The maximum acceptable period of quiescence is an exposed parameter.

A connection can fail as a result of either an unsatisfied trust requirement or a data communication failure (indicated by extended quiescence). In either case, all affected connections are immediately disabled and attempts to communicate through their interfaces raise exceptions.

2.2 An Example of Use of Switchboard

Switchboard’s continuous, secure trust monitoring is particularly useful in dynamic coalition environments. Such an environment is characterized by numerous hosts under multiple administrative domains with changing trust relationships.

Consider the following scenario in which Nation_A maintains sensitive reconnaissance information that is distributed from any of a number of replicated servers named S that securely distribute data using switchboard’s monitored connections. Nation_A now decides to temporarily make its data available to multi-national forces under the auspices of the U.N. Nation_A issues a certificate authorizing the U.N. to further delegate authorization for access to S’s data.

The U.N. then issues Credential_{U,N} authorizing members of its multi-national force to access S. In consideration of the sensitivity of the data, both the U.N. and Nation_A require the ability to quickly terminate data sharing should
the alliance be canceled. Therefore, \textit{Credential}_{\textit{UN}}’s validation tags require continuous on-line authorization using an additional liveness-monitored switchboard connection to a \textit{ValidationServer} operated by the U.N.

When \textit{beta}, an agent from the multi-national force, requests a connection to an instance of \textit{S}, \textit{beta}’s credentials (including \textit{Credential}_{\textit{UN}}) are presented to \textit{S}’s trust monitor \textit{TM}. To validate \textit{Credential}_{\textit{UN}}, \textit{TM} establishes a switchboard connection to the U.N.’s \textit{ValidationServer} with appropriate liveness monitoring.

When the multi-national mission is terminated, the U.N. informs its \textit{ValidationServer} that \textit{Credential}_{\textit{UN}} is revoked. This information is communicated to networked trust monitors via their switchboard connections to the U.N.’s \textit{ValidationServer}, causing \textit{beta}’s connection to be severed.

The only mechanism available for \textit{beta} to avoid disconnection is by interfering with communication between the \textit{ValidationServer} and \textit{S}. However, switchboard is resilient to these attacks: switchboard’s encrypted channels include message authentication codes that detect message tampering, and switchboard’s liveness monitor detects loss of connectivity.

3 Architecture and Implementation

Two architectures have been developed for switchboard. The first generation utilizes an open source SSL/TLS implementation, PureTLS [11], to implement its secure transport.

Authentication, authorization and cipher configuration are tightly coupled within the SSL handshake protocol. Switchboard decouples authorization from this process. Switchboard’s handshake phase is only responsible for authentication of the PKI identities of parties wishing to communicate and the establishment of a session cipher; we altered PureTLS’s handshake phase to perform only these tasks. Interfaces to this modified transport are modular and it can easily be replaced by another with similar properties.

In switchboard, credentials authorizing communication are transmitted after the handshake is complete and the encrypted channel has been established. Credential updates are transmitted by either party over the encrypted channel when needed to avoid disconnection due to revocation or expiration of previously established credentials.

Our second generation host-level implementation of switchboard consists of two major components. The transport module, which contains host managers, listeners and connectors is responsible for establishing and maintaining secure and liveness-monitored connections between switchboards on a pair of hosts. The services and conversations module, which is built upon the transport module, is responsible for establishing and maintaining connections between clients and services. As in Taos [9], switchboard serves as locally trusted resources: Clients and servers must provide credentials authorizing their switchboards to act in their stead.

The principal components of these modules are described below.

3.1 Switchboard Transport

Host Managers implement a single secure connection to a host manager on another system. As in SSH, one or more multiplexed channels of communication are routed through this encrypted connection. Encryption, message authentication, flow control and support for liveness monitoring are embedded into the host managers’ messaging protocol. All switchboard conversations between a particular pair of switchboards are multiplexed through the same host manager connection. Requests to establish new switchboard conversations are resolved by switchboard’s advertised services module, described in the next section.

Liveness Monitors and Flow Control We are investigating protocols that minimize message traffic required to monitor liveness. A technique resembling acknowledgment in TCP is being considered: in the place of sequence numbers, message headers are annotated with timestamps relative to both hosts. In this scheme, active two-way communication through any combination of multiplexed channels can generate sufficient traffic to guarantee liveness without the transmission of additional messages. When no data is transmitted via a particular host manager for an extended period, liveness is automatically probed via the transmission of a header for an empty message.

Flow control and message size limits reduce the risk that transmission latency will exceed a liveness monitor’s maximum refresh interval. Channel characteristics are conservatively estimated from message timestamps and message length is computed from liveness requirements. Bandwidth is partitioned among multiplexed channels; channel starvation due to queuing of long messages is minimized by limiting the aggregate size of transmitted but unacknowledged messages.

Connectors and Listeners When a new host manager is needed, a connector is utilized, which establishes a secure connection to a switchboard listener on the remote host. As in SSH, connectors and listeners implement a key-exchange protocol that authenticates each remote party’s claim to represent the identity of its public key and establishes a session key. Once a session cipher is established, responsibility for the secure connection is passed to newly generated host managers, which provide symmetric interfaces to both switchboards. A race condition is exposed if two hosts simultaneously attempt to establish new host monitors with each other. This race is resolved in favor of the switchboard whose public key has greater value.
3.2 Services and Conversations

Conversation Objects Open connections between switchboard clients and servers are represented as conversation objects. Conversation objects associate a particular conversation with its transport (host manager and multiplexer channel) and trust monitor.

Conversation Handles provide interfaces to conversation objects. This interface includes methods to update credentials, set and query liveness and authorization parameters, close the connection, transmit and receive data. In addition, a callback interface is exposed for asynchronous notification should the connection be lost.

Service Providers Every service is identified by its name and a specification of the property required for authorization (for example, a discriminator value specifying “SDSI/SPKI” and an authorizing SPKI path). A process P attempting to provide service type S supplies credentials authorizing its switchboard the privileges of the service and a trust monitor to authorize potential clients. These credentials can be used by switchboard, potential clients, and discovery mechanism to validate P’s claim to provide service S.

Service Clients A client process C requesting service S via its switchboard specifies a host name and port number, the service name and credentials authorizing its switchboard to connect to S. If the request is authorized, a conversation object is generated at both the client’s and server’s switchboard, and a conversation handle is presented to the newly connected parties.

Advertised Services Switchboard’s advertised services registry provides a mapping from service names to registered service providers. Host managers use the advertised services registry to resolve and authorize incoming client connections.

3.3 Authorization

Trust authorization plug-ins for switchboard can be easily constructed that implement a variety of trust-management and authorization schemes based on PKI identities. We note that a portion of the X.509 authorization protocol that maps a public key to endorsed identity is commonly referred to as a form of authentication. This mapping and its related validity checks can be performed as a component of switchboard authorization.

Two object types implement switchboard authorization. Authorizers are a factory class that build trust monitors which monitor credentials that enable connections. Trust monitors have interfaces to accept credential updates and to notify switchboard components if authorizing credentials become invalid.

In a related effort, our group has developed a distributed role-based access control system called dRBAC [8]. dRBAC implements mechanisms that monitor sustained trust relationships that rely on revocable certificates. The dRBAC trustMonitor plug-in utilizes switchboard conversations to authorize and reliably invalidate credentials.

3.4 Example Centralized Switchboard Connection Sequence

Figure 1 indicates the sequence of operations performed when client C on host Hc requests a switchboard conversation with service S on host Hs. Objects in this figure are labeled as follows:

- C is a client process on host Hc, S is a server process on host Hs.
- Each switchboard (SBc, SBs) contains a connector Co listener L, and service registry R. We shall assume that Hc and Hs have not previously communicated, and therefore have not previously established the secure connection supervised by host managers HMcs and HMsc.

1. Service S provides its service registry R with authorizer for clients As and credentials Crs that demonstrate SBs has permission to serve S.
2. Client C requests a conversation with S on Hs from its switchboard. It provides As, an authorizer for S and credentials Crs to validate SBs’s role as a client for S.
3. In order to establish host monitors for conversations between SBc and SBs, SBc requests that its connector Co negotiate with listener Ls to authenticate each other’s PKI identity and establish parameters for a cipher.
4. Cs and Ls create host managers HMcs and HMsc, which establish an encrypted channel with liveness monitoring.
5. SBc forwards C’s request to HMcs.
6. HMcs forwards C’s request to HMsc.
7. HMsc forwards C’s request to service registry Rs.
8. Rs authorizes C’s request by providing Crs and the PKI identity of SBc to As, which creates trust monitor TMcs. Rs then sends credentials Crs to SBc via host managers HMsc and HMcs, which is similarly authorized and results in the creation of trust monitor TMcs.
9. SBc and SBs establish conversation objects COc and COS, which are associated with trust monitors TMcs, TMsc, and multiplexer channel in HMcs HMsc respectively.
10. Connection handles are returned to C and S, allowing them to communicate.
3.5 Implementation Status

The initial implementation of switchboard is an extension of an SSL library. We are presently designing a second generation as a host-level resource.

4 Benefits of Switchboard

The development of switchboard was motivated by needs of the DisCo effort of the NYU Distributed Sanctuaries project. DisCo is developing application-neutral middleware to support secure decomposable service delivery. Such infrastructure is suitable for massively distributed systems that span multiple trust domains.

Applications developed for DisCo require infrastructure for establishing secure communication channels between agents executing on multiple hosts administered by disjoint trust authorities. DisCo utilizes switchboard’s facilities for secure object delivery, RPC, and code distribution.

To evaluate switchboard, we compare both the “code bloat” and increased execution and communication overhead involved in switchboard functionality to the overhead involved in performing the same functionality using SSL/TLS without switchboard.

First, we consider communication overhead. We observe that the extended SSL implementation of switchboard generates no additional communication traffic when extended switchboard features are not engaged. Credential updates and heartbeat messages generate no additional communication overhead beyond their encryption and transmission. Note that equivalent messages would have been necessary if similar functionality was layered upon a bare SSL system.

Of course, the primary benefit of switchboard is that it abstracts a broader security model than SSL/TLS. It provides a clear API for developers so that they can harness powerful functionality without delving into implementation details of trust management liveness monitoring.

Programs that currently employ SSL can be converted to switchboard with minimal code changes (normally, only a handful of lines must be modified). The developer can then easily specify liveness and trust monitoring properties for the modified program.

Without switchboard, coordination is required to associate services with TCP ports. By associating multiple services with a single listener, switchboard extends the model presented by portmap. The trust model is similarly extended: portmap requires that the system administrator authorize all services. Switchboard replaces this administrative role with trust management techniques based on public key cryptography.

The switchboard API extends the model presented by the BSD socket system calls listen(), accept(), and connect(). As a result, the number of lines of code required to establish a switchboard connection is equivalent. Switchboard does incur the extra overhead of establishing Host Managers, which is dominated by the cost of establishing a cipher. This cost is equivalent to that of existing secure communication protocols, and switchboard can amortize this cost among multiple conversations.

Multiplexed connections can result in a reduction of the number of signaling messages passed between hosts when compared to approaches that utilize multiple TCP ports. A single acknowledgment or heartbeat message applies to all connections sharing the same MUX. MTU requirements of many networks and ciphers reduce the efficiency of small messages; a multiplexed channel can aggregate small messages from multiple senders into a single larger block, resulting in higher efficiency.

5 Related Work

The switchboard secure communication abstraction unifies several mechanisms into a single modular API. SSL [6]/TLS [4] and SSH provide widely used tool-sets for
establishing connections where authorization decisions are made using access control lists or X.509 certificate chains. Considerable work recently has been done in the area of trust management systems [13, 14], which are more suitable for expressing the dynamic hierarchical nature of trust in large organizations (see KeyNote [2], PolicyMaker [3], SPKI/SDSI [5, 12], and dRBAC [8]). The SPKI/SDSI specification includes support for on-line credential validation; dRBAC includes a mechanism for validation of a trust relationship over a prolonged period. Switchboard’s monitored connections, which enable dynamic tracking of access rights of connected entities, are rooted in Publish/Subscribe systems, such as IBM’s Gryphon project [15].

Peer-to-peer infrastructures such as JXTA [10], or Groove provide a set of generalized protocols including dynamic discovery that allow any connected device in a network to communicate. JXTA pipes resemble switchboard connections, but the underlying trust model [10] is not extensible. Furthermore, like SSL/TLS and SSH, JXTA neither monitors connection liveness nor the status of required trust relationships.

6 Conclusion

Current secure communication protocols only guarantee trust and liveness at the time a connection is established. These guarantees are required to protect the integrity of prolonged interactions such as user logins and connections that monitor critical services.

Switchboard extends the abstraction presented by these protocols. As with SSL/TLS, the availability of a substrate with these properties can enable the construction of applications with enhanced security.

Trust management for distributed systems is presently an active area of research. Switchboard’s modular trust-management interface provides a platform to enable exploration of these techniques.

7 Acknowledgements

Oliver Kennedy, Vladimir Vanyukov, and Jordan Applebaum implemented several prototypes of switchboard components.

This research was sponsored by DARPA agreements F30602-99-1-0157, N66001-00-1-8920, and N66001-01-1-8929; by NSF grants CAREER:CCR-9876128 and CCR-9988176; and Microsoft. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright annotation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as representing the official policies or endorsements, either expressed or implied, of DARPA, Rome Labs, SPAWAR SYSCEN, or the U.S. Government.

References