Implementing the Intrusion Detection Exchange Protocol

Tim Buchheim,
Harvey Mudd College,
tcb@cs.hmc.edu,

Michael Erlinger,
Harvey Mudd College,
mike@cs.hmc.edu,

Ben Feinstein,
Guardent, Inc.
Ben.Feinstein@guardent.com

Greg Matthews,*
Harvey Mudd College,
gmatthew@cs.hmc.edu,

Roy Pollock,
Harvey Mudd College
rpollock@cs.hmc.edu

Joseph Betser,
The Aerospace Corporation,
betser@aero.org,

Andy Walther
The Aerospace Corporation
awalther@aero.org †

Abstract

We describe the goals of the IETF's Intrusion Detection Working Group (IDWG) and the requirements for a transport protocol to communicate among intrusion detection systems. We then describe the design and implementation of IAP, the first attempt at such a protocol. After a discussion of IAP's limitations, we discuss BEEP, a new IETF general framework for application protocols. We then describe the Intrusion Detection Exchange Protocol (IDXP), a transport protocol designed and implemented within the BEEP framework that fulfills the IDWG requirements for its transport protocol. We conclude by discussing probable future directions for this ongoing effort.

1 Introduction

Intrusion detection is an area of increasing concern in the Internet community. In response to this, many automated intrusion detection systems have been developed. However, there is no standardized way for them to communicate. To remedy this, the Intrusion Detection Working Group was chartered under the auspices of the Internet Engineering Task Force.

This paper gives an overview of the working group and the task it faces. The paper then describes attempts to define and implement a transport protocol for intrusion detection alerts. The paper focuses mainly on the Intrusion Detection Exchange Protocol and the implementation experience of the Aerospace Corporation/Harvey Mudd College 2000-2001 clinic team.

2 The IETF-IDWG

2.1 IETF

The IETF is the engineering, development and standardization arm of the Internet Architecture Board (IAB). The IETF is where the Internet protocol specifications are developed.

The IETF's mission includes:

- "Identifying and proposing solutions to operational and technical problems in the Internet.
- Specifying protocols and architecture to solve such technical problems.
- Making recommendations to the Internet Engineering Steering Group (IESG) regarding the

*Greg Matthews now at CSC/NASA Ames

†Supported by the Aerospace Institute, The Aerospace IRAD Corporate REsearch Initiative, and UC Davis Subcontract K971414-01 to US Air Force DARPA Cyber Panel GlobalGuard contract F30602-98-I-0201.

standardization of protocols and protocol usage in the Internet

- Providing a forum for the exchange of information within the Internet community." [4]

To accomplish its mission the IETF has divided itself into various areas of interest (e.g., Security Area), with an Area Director for each area. Each area has a number of Working Groups (WG), each dedicated to a particular issue, e.g., creation of a particular Internet protocol specification. A working group is composed of a chair and all interested parties. The WG charter sets the agenda and timetable for the achievement of some goal. Anyone may choose to join any WG. Participants, while making their affiliations known, act individually.

To accomplish its mission the IETF meets three times a year in various worldwide locations. At these meetings the working groups meet to accomplish specific tasks as established in their charter.

2.2 IDWG

The Intrusion Detection Working Group (IDWG) is one of the working groups in the IETF Security Area. Its charter is "to define data formats and exchange procedures for sharing information of interest to intrusion detection and response systems, and to management systems which may need to interact with them."

This working group is trying to develop a protocol that allows the many different intrusion detection systems to communicate in a standard way. At one level this work is a follow-on to the DARPA CIDF (Common Intrusion Detection Framework) that produced CISL. The major enhancements are a limitation of the type of information communicated and the integration of the protocol into the TCP/IP suite of protocols.

IDWG was chartered as a working group after the August 1998 IETF meeting and the working group met for the first time at the December 1998 IETF meeting. Since that time, IDWG has met at each IETF meeting and held interim meetings at various locations. In the process of producing the desired protocol the IDWG has committed to producing various specifications. The first is a requirements specification detailing the set of requirements for the protocol. The requirements document, while difficult to produce, was desired because the working group wanted to ensure that a well-defined checklist existed to assist in making protocol choices. The requirements specification is currently an Internet Draft and has been forwarded to the IESG for publication as an Informational RFC.

The other documents to be produced all relate directly to the protocol; a data definition of the data items desired in the messages to be exchanged; a definition of the message format (IDMEF) [7]; and two protocol specifications (IAP [5] [3] and IDXP [10]) fulfilling the IDWG transport protocol (IDP) requirements for communicating IDMEF messages.

Thus in a little over two years, IDWG has produced a final version of a requirements document as well as draft versions of a message format (IDMEF) and two transport protocols (IAP and IDXP). Participation in the IDWG activities is encouraged via idwg-public-request@zurich.ibm.com.

3 System Description of IDS and IDWG

The activities of the IDWG are integrated closely with Intrusion Detection Systems (IDS), so descriptions of the IDMEF and IDP use general IDS terminology. See [9] for additional IDWG terminology. The following definitions will clarify the description of the IDWG system.

3.1 Activity

IDSs monitor network usage and other types of activity by analyzing data sources such as network packets and system logs. In particular, any activity that is of concern from a security perspective is relevant for IDS monitoring.

Event: Any activity that is potentially worthy of some type of action or response, such as multiple failed logins or a ping flood. The degree to which an event is worthy of a response is dictated by the security policy in place, typically manifested in the configuration of the IDS.

3.2 Communication

Once an event has been detected and deemed worthy of further action it is necessary to communicate the existence of the event, and possibly its details, to a party that is capable of further action.

Alert: IDS alerts contain, at a minimum, information stating that an event occurred. Alerts may contain much richer information about the event, in order to facilitate an informed action or response by the receiving party.

Notification: Once an alert has been received it may be deemed necessary (as defined in the security policy) to notify an external actor of the alert. This can involve sending pager or e-mail messages, creating database entries, or publishing web-accessible HTML reports.

3.3 Actors

Intrusion Detection (ID) actors are those ID entities that create, consume, act upon, or respond to ID alerts. These actors are either programs or humans.

Administrator: Human responsible for establishing the overall security policy. This specifies the way in which other ID actors behave in response to various types of activities.

Operator: Human responsible for initiating responses to alerts and/or notifications. The operator may choose to manually respond or may configure the IDS for some automated response.

Sensor: Collects data about activity from data sources, to be used in detecting events.

Analyzer: An analyzer detects events based on activity data, and according to the security policy possibly generates alerts based on these events. Alerts are formatted and transferred to managers using the format and transfer protocols standardized by the IDWG.

Manager: ID entity that consumes alerts, and facilitates control over the IDS by the operator. Managers inform the operator through different types of notification that alerts have occurred, as per the security policy.

3.4 System Assumptions

The IDWG has made two assumptions about the deployment configuration of an IDS. First, it assumes that only analyzers create and communicate ID alerts. The same program or computer may act as both an analyzer and a manager, but these will be two separate ID entities. This assumption is tied to the second, that analyzers and managers can be separate components that communicate pairwise across a TCP/IP network. Together these assumptions differentiate between creation and consumption of ID alerts, and allow the acts of alert creation and consumption to occur at different IDS components on the network. The assumptions affect the transfer protocol, since alerts will be transferred from one component over TCP/IP to another component. This communication must be done in a standard and secure fashion.

4 Transport Requirements

A diagram outlining the core architectural elements of IDP is shown in Figure 1. The working group has specified requirements for how these elements should interact. See [9] for these requirements. The design of IDP must meet these requirements to successfully take its place in a system utilizing IDS to maintain and enhance security. The most important of these requirements are summarized here.

4.1 Allow Communication Between Different IDS Elements

IDS elements may be either passive or active. Passive elements wait to be contacted by active elements, while active elements are capable of initiating communication with passive elements. The active/passive distinction allows for flexible deployment of IDS elements, such that some elements may only actively connect to a known set of peers, while others may passively listen for one or more connections from a set of known peers.

4.2 Reliable Transmission

Messages sent between IDS elements must get through and be acknowledged, especially under difficult network conditions. The severity of an attack, and therefore the importance of a prompt response, may be revealed in the number and frequency of messages generated by IDS analyzers. Achieving reliable transmission in an efficient manner is desirable, so as to avoid unnecessary duplication of messages. For these reasons the IDWG has specified that the IDP be based on the Transmission Control Protocol (TCP). This leverages the massive effort that has already been expended by the IETF community to develop and to improve TCP.

4.3 Mutual Authentication / Assurance of Message Origin

The party at each end of IDP must be able to verify the identity of its peer. This is important because an attacker masquerading as an analyzer could inject false data to mask a real attack, or an attacker masquerading as a manager could intercept traffic and lull the analyzer(s) into thinking that a response was being mounted. Assurance of message origin involves having a way to prove post facto which messages came from which IDS element.

4.4 Integrity and Confidentiality

IDP must guarantee both the integrity and confidentiality of its data. Integrity of data requires that if a message has been tampered with by someone “on the wire” between the two elements then this tampering will be detected. Confidentiality means that a third-party intercepting the traffic between the two elements will be unable to make sense of it.

4.5 Resist DoS Attacks

IDP is potentially a burden if it allows attackers to more easily shut down the network by exhausting a resource. Therefore IDP should attempt to resist such attacks by taking actions such as rejecting illegitimate connections as early as possible, and potentially modifying its behavior if it believes it is being attacked.

4.6 Further Network Specific Issues

Since IDP is designed to increase network security it should be able to operate through firewalls without compromising their security. This is why managers and analyzers can both be active, allowing them to connect outward through the firewall to the appropriate passive element. Additionally, a proxying capability allows connections between IDS elements separated by a firewall. Proxying facilitates initiation of IDS message exchanges regardless of the location of the initiating IDS element. These features are desirable because any number of IDS deployment scenarios are possible, with managers and analyzers residing on either side of a firewall (should one exist). A proxy can be configured to only accept connections and transfer messages to known IDS elements, reducing the potential security risk that additional transfers across a firewall might pose.

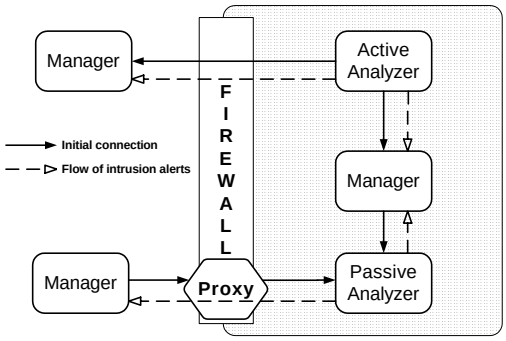

Figure 1: IDWG IDS Architecture

5 Intrusion Alert Protocol (IAP)

The working group's first attempt to meet IDP requirements was the development of the Intrusion Alert Protocol (IAP) [5]. The design of IAP was based on the Hypertext Transfer Protocol (HTTP) [11].

HTTP itself was deemed unsuitable for several reasons. The most important was that in HTTP, the HTTP client is always the party that initiates the TCP connection. For IDP, however, a passive analyzer should act as a client even though it receives the TCP connection. However, IAP still borrows many of its headers and response codes from HTTP.

IAP uses Transport Layer Security (TLS) [8] to insure confidentiality, integrity, and mutual authentication. To simplify the protocol, data only flows in one direction per connection. In other words, if a manager wanted to send configuration data to an analyzer it was already receiving alerts from, it would have to open a new connection. IAP defines a custom proxying method that examines the headers to make the initial connection, but then acts as an octet forwarder after that. Proxies, therefore, do not participate in security negotiation and are unable to see or undetectably tamper with the data.

The initial development of IAP was done by Dipankar Gupta of Hewlett-Packard. The authors joined him in an attempt to complete the specification and actually implement the protocol. We assisted Dipankar with several revisions of the specification and were given coauthor status on the internet draft. Additionally, in the course of our implementation, we found several flaws. One was in the specification of the proxy method. As originally written, a compliant implementation of the proxy could have allowed connections to arbitrary hosts and ports through the proxy. This is obviously undesirable from a security standpoint. This flaw was partly due to a related issue — the original specification contained no information about the source of the request. This meant that in a proxied connection, the originating entity would be unknown until it presented its key for security negotiation. This is undesirable since it requires security negotiation before a proxied connection can be refused, making DoS attacks easier. Additionally, it requires entity information to be encoded in the key to permit any identification of where the message came from. We also found and resolved several ambiguities that could have caused interoperability problems.

Our implementation of IAP is written as a module in Perl. Perl was chosen because its string handling capabilities and built-in HTTP header support make the implementation clean. Furthermore, it is very easy to interface a Perl module to various managers and analyzers. We also have developed sample implementations of both active and passive managers and analyzers, as well as a simple proxy. Our implementation is currently undergoing interoperability testing. The code can be found at: *http://www.cs.hmc.edu/clinic/projects/2000/aerospace*.

Our work was presented at the 49th IETF conference in December, 2000. At that meeting, however, the working group decided it should investigate the Blocks Extensible Exchange Protocol, or BEEP, as the basis for the IDWG transport protocol. The authors volunteered to spearhead this effort.

6 Blocks Extensible Exchange Protocol (BEEP)

The Blocks Extensible Exchange Protocol (BEEP) [22] is a generalized framework for the development of application-layer protocols (Figure 2). BEEP offers asynchronous, connection-oriented, and reliable transport. Thus an application level transport protocol such as IDP can be implemented using the BEEP framework. The key issue in such an implementation is determining whether BEEP provides all the features required. See [21] for a detailed discussion of application-layer protocol design and the development of the BEEP framework. The following subsections borrow heavily from [22] and [21].

Overall, BEEP supports higher level protocols by providing the following collection of protocol mechanisms:

- Framing: how the beginning and ending of each message is delimited.
- Encoding: how a message is represented when exchanged.
- Reporting: how errors are described.
- Asynchrony: how independent exchanges are handled.
- Authentication: how the peers at each end of the connection are identified and verified.
- Privacy: how the exchanges are protected against third-party interception or modification.

Note that other necessary protocol mechanisms such as naming and authorization are not provided by the BEEP framework. Other than the URI [2], there is no universal framework for naming. The naming issue represents a "domain-specific problem for each application protocol" [21], and thus naming is intentionally not handled by the BEEP framework. Likewise, authorization is excluded from the framework. Nearly every conceivable scheme for authorization must rely on naming to identify principals. It follows that without naming, the framework cannot provide authorization. Application protocols using the BEEP framework must explicitly deal with the issues of naming and authorization.

6.1 Message Framing

There are three traditional approaches to the problem of message framing, with each approach being taken by at least one of the major IETF protocols. The Simple Mail Transfer Protocol (SMTP) [13] uses *octet-stuffing* for its framing requirements. This method suffers from slower performance due to the comparison overhead of checking for the special sequence that indicates the end of a message. HTTP [11] uses *octet-counting* to determine the message delimiters. The drawback of this approach is that, unless messages can be segmented, the entire message must be available before transmission. The File Transfer Protocol (FTP) [18] uses *connection-blasting* as its framing mechanism. However, for small messages, the overhead of connection-blasting is high. In addition, the endpoint-addresses of the peers must be available at the application layer.

BEEP uses a variation of *octet-counting* to perform message framing. In addition to counting the bytes of the message to find the delimiter, BEEP adds the use of a trailer "END". This was presumably done to aid in the human comprehensibility of BEEP messages. The beginning of BEEP frames are delimited by a command line that provides the following information about the individual frame:

- "what kind of message is in this frame;
- whether there's more to the message than just what's in this frame (a continuation flag);
- how to distinguish the message contained in this frame from other messages (a message number);
- where the payload occurs in the sliding window (a sequence number) along with how many octets are in the payload of this frame; and,
- which part of the application should get the message (a channel number)" [21].

Since framing is at the core of the framework, BEEP specifies several consistency checks that catch most implementation errors in the framing mechanism. The BEEP framework's relatively straightforward framing mechanism, with its sequence number, octet count, and trailer, simplifies error detection.

The BEEP framework uses a flow control mechanism to allow for the multiplexing of channels over a single transport connection. The mechanism used is the same as the Transmission Control Protocol (TCP) [17], using sequence numbers and a sliding window. This mechanism is well understood and has been proven through extensive implementation experience with TCP.

6.2 Message Encoding

There are quite a few encoding mechanisms in use on the Internet today. SMTP [13] uses its RFC 0822 style of encoding. The Simple Network Management Protocol (SNMP) [6] uses the elaborate ASN.1/BER encoding mechanism. In the BEEP framework, messages may be arbitrary MIME [12] content. The framework's default content-type is application/octet-stream and its default content transfer encoding is binary.

6.3 Reporting

Application protocols must have a method to convey error information between peers. SMTP originally pioneered the use of 3-digit error codes. Many Internet protocols, including HTTP and FTP, use 3-digit error codes. In addition to conveying reply codes, most application protocols send textual diagnostics to aid in human interpretation. In any protocol being developed for the Internet, textual diagnostics should include the option for language localization. The BEEP framework provides for 3-digit error codes with a localized textual diagnostic. During connection establishment, BEEP peers exchange lists of language tokens (defined in [1]), ordered by peer preference. The textual diagnostics accompanying subsequent message replies will be localized with respect to the peers' language preferences. Additionally, replies to messages are flagged as either positive or negative. This makes the success or failure of a message more easily determined.

6.4 Exchange Asynchrony

In the BEEP framework, "frames are exchanged in the context of a *channel*. Each channel has an associated *profile* that defines the syntax and semantics of the messages exchanged over a channel" [21]. The concept of channels provides the basis for the BEEP framework's asynchrony. Within a single BEEP session, multiple channels can be open simultaneously. Messages are processed synchronously within each channel, but no limitations are placed on the order of processing for different channels. For example, in a multi-threaded implementation, each channel would map to its own thread. Implementors are not required to support this, however. For instance, an implementor may choose to use only a single process for a BEEP session and its associated channels, thereby forcing synchrony onto the intra-channel communications within the session. Figure 3 shows a BEEP session between peers with three *channels* each specified by a *profile*, i.e., IDXP, syslog, and Other.

6.5 Authentication

Traditionally, application-layer protocols have ignored the problem of authentication. That is, the protocols do not authenticate the identities of the peers or the authenticity of the messages. Other protocols that do provide for authentication often use a domain-specific mechanism. For instance, FTP uses its own scheme of user/password exchange to authenticate the client connection.

With the goal of creating a standard framework for authenticating protocol peers, the Simple Authentication and Security Layer (SASL) [14] was developed by the IETF. With SASL, one can describe how a particular authentication mechanism functions. It is the responsibility of the protocol using SASL to describe the actual exchange of messages. SASL can also use information from the underlying layer in the authentication process. For instance, if the connection is using IPsec, SASL can use the underlying credentials of each peer to perform authentication. In addition to authentication, SASL mechanisms can provide for guarantees of message integrity and privacy.

The BEEP framework supports the use of SASL. When a channel is authenticated, a single user identity is established for each peer on the session. In this way, peer authentication is done on a per-session basis. Each session must correspond to a single user, with all open channels being associated with that session's peer identities.

6.6 Privacy

Transport Layer Security (TLS) [8] is the IETF standard for securing transport connections. (See Section 3 of [22] for a more in depth discussion of transport security in BEEP and its use of TLS.)

The BEEP framework supports the use of SASL and TLS to provide for privacy. When "a channel associated with transport security begins the underlying negotiation process, all channels (including channel zero) are closed on the BEEP session. Accordingly, upon completion of the negotiation process, regardless of its outcome, a new greeting is issued by both BEEP peers" [22]. When transport security negotiation on a channel is successful, all subsequent traffic on that session is secured by the underlying transport security. Similar to BEEP authentication, privacy is provided on a per-session basis.

6.7 BEEP Profiles

In the BEEP framework, each channel is associated with a BEEP profile. A BEEP profile specifies the syntax and semantics of messages being exchanged over a

channel using the profile. An application-layer protocol (such as IDP) using BEEP would be designed as one or more related BEEP profiles. Each BEEP profile is identified by its profile URI [2]. These profile URIs are exchanged between the peers during the BEEP session greeting, advertising the profiles that each peer is willing to use.

There are two basic types of profiles, and these determine the operations a profile may make on its channel and the underlying session.

- *Tuning profiles*: these are used to perform initialization, or "tuning", once the BEEP session is established. For instance, profiles providing transport security, authentication, or application-layer tunneling would be tuning profiles. Thus *tuning profiles* establish the characteristics of a channel, such as authentication, privacy, etc..

- *Regular profiles*: these are used to perform data exchange. Usually, these profiles are used after the initial tuning has been completed. Thus, *regular profiles* establish the actual exchange of data between peers over the channel.

6.8 Transport Mapping

BEEP is a generalized protocol framework, and as such it does not rely directly on any particular transport protocol. Instead, BEEP specifies a collection of requirements on how an underlying transport protocol must support a BEEP session. Then, separate "transport mapping" documents are to be developed, defining how to use specific transport protocols to meet the BEEP requirements. BEEP will map easily to any connection-oriented transport protocol offering ordered, reliable delivery. This clearly suggests that TCP be used as an underlying transport. Attempts to map BEEP onto a connection-less, best-effort delivery protocol (i.e., UDP) are strongly discouraged, as it's anticipated that such an effort would be forced to duplicate much of the effort that went into the development of TCP.

A transport mapping for TCP has been developed, defining how a BEEP session is mapped onto a single TCP connection. In March 2001, this document was designated RFC 3081 [23] and is on the IETF standards-track. There has been some discussion of mapping BEEP onto the Stream Control Transmission Protocol (SCTP) [19]. SCTP is a reliable, connection-oriented protocol that provides multi-streaming functionality, allowing data to be broken into multiple independently sequenced streams.

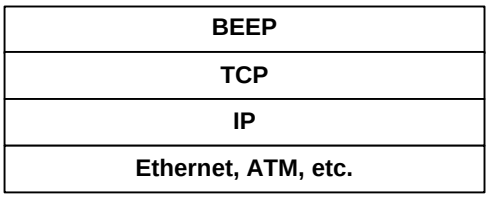

Figure 2: BEEP's Position in TCP/IP Protocol Stack.

7 Intrusion Detection Exchange Protocol (IDXP)

IDXP is an implementation of the IDWG application level transport protocol described in the IDWG requirements document as IDP [9]. IDXP is implemented as a BEEP *profile*. Thus BEEP provides the actual protocol while the IDXP profile specifies the BEEP channel characteristics necessary for implementation of the IDP requirements. IDXP can be broken down into four main phases: connection provisioning, security setup, BEEP channel creation, and data transfer. Basically, IDXP is implemented as a BEEP profile that uses other profiles to create a channel.

7.1 Connection Provisioning

Connections between ID entities may be established using zero or more intermediate hops. A simple BEEP connection from one ID entity to another is used for the case of no intermediate hops. When one or more intermediate hops are required, the protocol needs to set up an application-layer tunnel across those hops.

The IDXP BEEP profile requires that users of IDXP choose a BEEP *tuning profile* (6.7) designed to create such a tunnel, and requires that implementors include support for use of the TUNNEL profile [15]. Use of a *tuning profile* for this purpose preserves the orthogonality between this phase of IDXP and later phases, because later phases need only know that a connection exists from one ID entity to another. Requiring support for TUNNEL insures interoperability between implementations for this phase of the protocol.

It is important to note that no security-related requirements are placed on the creation of application-layer tunnels. This stems from IDXP's trust model, which states simply that endpoints alone are trusted once mutual authentication between endpoints has taken place. Intermediate hops are not trusted, and do not participate in end-to-end security measures except in their basic function as links in the application-layer tunnel.

The IDXP profile does require that some authentication mechanism be supported by the tunneling profile chosen, and in particular requires support for the use of

SASL [14] by users utilizing the TUNNEL profile for tunnel creation. Use of security measures in tunnel creation is encouraged if the cost of security during connection provisioning is acceptable, and support for an authentication mechanism is required to, at the least, provide an option for safe buildup of the tunnel across security perimeters. In particular, IDS deployment scenarios may require that ID entities outside a firewall connect to other ID entities located within the firewall, and in such scenarios it is highly desirable to authenticate any host attempting to connect through the firewall and into the protected network (Figure 1).

7.2 Security Setup

Once a connection has been provisioned the use of end-to-end security measures must be initiated. The IDXP profile requires that a BEEP *tuning profile*(6.7) be used to establish security properties of the connection that meet the IDWG requirements on transport security. IDXP requires that implementors support use of the TLS profile [8], using the TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA ciphersuite, to insure interoperability for this phase of the protocol.

Utilizing a *tuning profile* to establish end-to-end security preserves the orthogonality of this phase of the protocol with other phases, because other phases assume that sufficient security measures are in place (or will be put in place). The TLS profile is the only profile known to meet all the transport requirements of the IDWG [9] for IDP.

7.3 BEEP Channel Provisioning

Once a BEEP session has been established and secured, the next phase is to open one or more BEEP channels over which ID data can be exchanged. The client (source of ID data) and server (sink for ID data) must be established for each channel that is opened for use with IDXP, and so the first message exchanged across a channel running IDXP contains a URI [2] for the peer as well as a requested role, i.e., server or client. A fully qualified domain name and/or IP address may also be supplied. Once a peer receives this message it must decide whether the role requested by the ID entity with the given identification is acceptable, and reply with an affirmative or negative answer.

Messages identifying a peer and requesting a role on a channel running IDXP may be sent at any point in the lifetime of the channel, and the current role choices at any given time are determined by the last affirmatively acknowledged greeting message. The choosing of roles on a per-channel basis allows BEEP sessions to be created between ID entities in a fully peer-to-peer fashion, and permits different concurrent exchanges within a BEEP session. Additionally, the use of multiple channels running IDXP under a single BEEP session provides a reduction in per-exchange overhead because the costs of connection provisioning and security are only paid once at session creation.

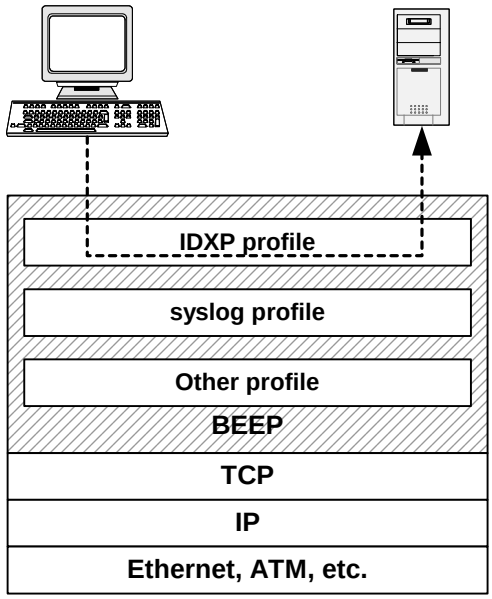

Figure 3: BEEP in Action

7.4 Data Transfer

ID data is sent from clients to servers in one of three MIME types [12]: text/xml, text/plain, or application/octet-stream. XML data must conform to the DTD for IDMEF messages, and the remaining two types are provided so as not to restrict the present and future formatting of ID data being sent using IDXP.

Transfer categorization can be achieved by appropriating a different channel to different categories of ID data, and transfer prioritization can be achieved by establishing an order in which channels are serviced (messages sent or received).

8 Future Work

Future work on IDXP can be grouped into four areas: improving the IDXP specification and preparing it for the standards process, completing and testing our implementation of IDXP, developing applications that use IDXP, and extending IDXP to do more than deliver alerts.

8.1 The IDXP Specification

Although there are no obvious problems with the IDXP profile, it is important for it to be scrutinized before the standards process, in order to ensure its security and usability. Any ambiguities in the definition of IDXP must be resolved in order to avoid the interoperability problems that have faced other protocols when the specifications have not been clear.

Part of this process will involve testing our implementation of IDXP with implementations developed by others. In addition to finding any ambiguities or errors in the text of the specification, this step is important in order to find any design problems that may hinder operation under various conditions.

After appropriate testing and refinements to the document, the working group will submit IDXP for consideration as an RFC. At that point, it will be up to the vendors to implement support for IDXP in their intrusion detection products.

At this time, the IDXP specification is undergoing 'final call' within the IDWG. After 'final call', the IDWG will request that IDXP become a Proposed RFC.

8.2 The Implementation

Currently, there are several major problems with the implementation. First, because the TLS profile is not done, the implementation can not check that it is being used correctly. It can check that the underlying session is using TLS but can not verify that the correct algorithm is being used. Second, the implementation has a problem servicing multiple channels. The fix for this should include the ability to prioritize channels. Third, the configuration interface needs to be improved. But as it currently exists our IDXP implementation demonstrates the ability to fulfill the IDWG requirements for a transport protocol using the BEEP framework.

8.3 IDXP Applications

In order to show the usefulness of IDXP, applications which make use of it must be created. The most useful demonstration of IDXP would consist of a complete system of analyzers, managers, and proxies for distributing alert data across the network.

The easiest way to implement analyzers would be to create IDXP interfaces for existing intrusion detection products. A plug-in for the popular Snort tool would be one useful example. For those applications which do not have easily extensible architectures, a small program could be written to convert and transmit proprietary output.

A simple manager application could demonstrate the utility of IDXP by collecting information from the analyzers and storing it in a database. A user interface for collating, summarizing, and searching the alert database would allow for powerful administrative abilities.

Currently, we have created a small environment of analyzers and managers with IDXP as the transport protocol. Our hope is to expand on this environment during the next year with commercial and public domain tools.

8.4 Extending IDXP

Although the IDWG is currently only working on exchanging alert data, it may examine other issues at a later time. One popular idea for future work would be to standardize methods for remote configuration of IDS. IDS that use IDXP to communicate with managers across the network might be physically located far away from those responsible for their configuration. Even if they are physically accessible, an operator might prefer to configure them remotely, especially if a large number of systems must be reconfigured. It might also be desirable for a manager application to reconfigure an analyzer automatically in response to a network attack.

In order to allow managers to configure analyzers, standard messages need to be defined, possibly as an extension of IDMEF[7], and need to be sent to the analyzers over the network. The natural mechanism for sending such messages is IDXP. The managers and analyzers already connect to each other through IDXP, so configuration messages could easily be carried on the same connection. One of the nice features of BEEP is the ability to open multiple channels over the same connection, so while one channel is used to exchange alerts, a separate channel could carry configuration information. BEEP even allows for assignment of priorities to channels, so a configuration channel could be given higher priority than the alert channel, in order to avoid the situation in which a flood of alerts might prevent timely delivery of configuration changes.

9 Summary

In this paper, we describe the goals of the IDWG and the requirements for a transport protocol to communicate among intrusion detection systems. Our focus has been on specifying and implementing the Intrusion Detection Exchange Protocol (IDXP) a transport protocol designed and implemented within the BEEP framework. There are still open issues in IDXP implementation, but IDXP appears to fulfill all the IDWG requirements for a transport protocol.

References

[1] ALVESTRAND, H. RFC 3066: Tags for the identification of languages, Jan. 2001.

[2] BERNERS-LEE, T., FIELDING, R., AND MASINTER, L. RFC 2396: Uniform Resource Identifiers (URI): Generic syntax, Aug. 1998.

[3] BETSER, J., WALTHER, A., ERLINGER, M., BUCHHEIM, T., FEINSTEIN, B., MATTHEWS, G., POLLOCK, R., AND LEVITT, K. Globalguard: Creating the ietf-idwg intrusion alert protocol (iap). IEEE DARPA Information Survivability Conference and Exposition (DISCEX), Anaheim, CA, June 2001.

[4] BRADNER, S. RFC 2026: The Internet standards process — revision 3, Oct. 1996.

[5] BUCHHEIM, T., FEINSTEIN, B., GUPTA, D., MATTHEWS, G., AND POLLOCK, R. IAP: Intrusion Alert Protocol, Mar. 2001. draft-ietf-idwg-iap-05, work in progress.

[6] CASE, J., FEDOR, M., SCHOFFSTALL, M., AND DAVIN, C. RFC 1157: Simple Network Management Protocol (SNMP), May 1990.

[7] CURRY, D., AND DEBAR, H. Intrusion Detection Message Exchange Format data model and Extensible Markup Language (XML) Document Type Definition, Feb. 2001. draft-ietf-idwg-idmef-xml-03, work in progress.

[8] DIERKS, T., AND ALLEN, C. RFC 2246: The TLS protocol version 1, Jan. 1999.

[9] ERLINGER, M., AND WOOD, M. Intrusion detection message exchange requirements, Feb. 2001. draft-ietf-idwg-requirements-05, work in progress.

[10] FEINSTEIN, B., MATTHEWS, G., AND WHITE, J. The Instrusion Detection Exchange Protocol (IDXP), Feb. 2001. draft-ietf-idwg-beep-idxp-01, work in progress.

[11] FIELDING, R., GETTYS, J., MOGUL, J., NIELSON, H., MASINTER, L., LEACH, P., AND BERNERS-LEE, T. RFC 2616: Hypertext Transfer Protocol — HTTP/1.1, June 1999.

[12] FREED, N., AND BORENSTEIN, N. RFC 2046: Multipurpose Internet Mail Extensions (MIME) part two: Media types, Nov. 1996.

[13] KLENSIN, J. RFC 2821: Simple Mail Transfer Protocol, Apr. 2001.

[14] MYERS, J. RFC 2222: Simple Authentication and Security Layer (SASL), Oct. 1997.

[15] NEW, D. The TUNNEL profile, Feb. 2001. draft-ietf-idwg-beep-tunnel-01, work in progress.

[16] NEW, D., AND ROSE, M. Reliable delivery for syslog, May 2001. draft-ietf-syslog-reliable-07, work in progress.

[17] POSTEL, J. STD 7, RFC 793: Transmission Control Protocol, Sept. 1981.

[18] POSTEL, J., AND REYNOLDS, J. RFC 959: File Transfer Protocol, Oct. 1985.

[19] R. STEWART, E. A. RFC 2960: Stream Control Transmission Protocol, Oct. 2000.

[20] ROSE, M. The facts on BEEP, Mar. 2001.

[21] ROSE, M. On the design of application protocols, Mar. 2001.

[22] ROSE, M. RFC 3080: The Blocks Extensible Exchange Protocol core, Mar. 2001.

[23] ROSE, M. RFC 3081: Mapping the BEEP Core onto TCP, Mar. 2001.