# X-Bone: Automated System for Deployment and Management of Network Overlays

## Surety Assessment Report

## Information Design Assurance Red Team (IDART™)

Sandia National Laboratories
P.O. Box 5800
Albuquerque, NM 87185-0784

John Clem, Organization 06512
Bret Badgett, Organization 09621
Timothy MacAlpine, Organization 09322

April 21, 2003

## ABSTRACT

This report details the results of an IDART™ surety assessment of version 2.0 of the X-Bone system application developed by the University of Southern California's Information Sciences Institute (ISI). The principle objective of the assessment is to identify design-related security vulnerabilities with respect to the software and the intended implementations of the system. Identified vulnerabilities can be mitigated through system design changes and or through implementation choices. This report includes recommendations to improve the security posture of X-Bone system for future versions and implementations.

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

This page left blank, intentionally.

# Table of Contents

ABSTRACT ........ 1
Table of Contents ........ 3
List of Figures ........ 4
1. Introduction ........ 5
2. X-Bone Objectives ........ 5
3. X-Bone Description ........ 6
3.1 Overview of Operations and Components ........ 6
3.2 The Application Code ........ 6
3.3 System Security Mechanisms and Capabilities ........ 7
3.3.1 Secure Overlay Creation ........ 7
3.3.2 Data Security ........ 8
3.4 System Views ........ 8
3.4.1 Secure Overlay Creation ........ 9
3.4.2 Example X-Bone Topology ........ 11
3.4.3 X-Bone Component Communications ........ 12
3.4.4 Overlay Configuration ........ 13
3.4.5 User Monitoring ........ 14
3.5 Additional System Views ........ 14
3.6 Detailed Description ........ 14
4. Scope of IDART™ Assessment ........ 15
4.1 Developer Claims ........ 15
4.2 Known Vulnerabilities ........ 15
5. IDART™ Analysis of X-Bone ........ 16
5.1 Attack Brainstorm ........ 16
5.2 Attack Scenarios ........ 16
5.3 Critical Success Factors ........ 17
5.3.1 Implementation Factors ........ 17
5.3.2 Operational Factor ........ 18
5.3.3 Host Security ........ 18
5.3.4 Certificate Authority Security ........ 19
5.4 X-Bone System Security Issues of Interest ........ 19
5.4.1 Access Control Lists ........ 19
5.4.2 Overlay Manager ........ 20
5.4.3 General System ........ 20
5.5 Analysis of the X-Bone Code ........ 20
5.6 Recommendations ........ 21
5.6.1 Security-Related Recommendations ........ 21
5.6.2 Improving the Utility of the X-Bone System ........ 22
5.6.3 Improving Application Robustness ........ 22
6. Conclusions ........ 23
Information Sources ........ 24
Appendices ........ 25

# List of Figures

Figure 1 .......... 9
Figure 2 .......... 11
Figure 3 .......... 12
Figure 4 .......... 13
Figure 5 .......... 14

# 1. Introduction

X-Bone is a software application package developed by the University of Southern California's Information Sciences Institute (ISI) that is intended to automatically deploy, coordinate, and monitor IP overlay networks. X-Bone is intended to simplify the deployment of overlays by including provisions for network resource discovery, user-defined overlay configuration and management, and automated overlay construction.[1] The system developers have included several measures to provide for secure overlay deployment as well as optional security measures to encrypt the overlay traffic.

The development project is currently funded under the Fault Tolerant Networks program of the Defense Advanced Research Projects Agency (DARPA). DARPA Program Director Doug Maughan tasked the Sandia National Laboratories Information Design Assurance Red Team (IDART™) to perform a design-level, surety assessment of the X-Bone system. Based on availability of the software and design documents, it was agreed by both IDART™ and ISI that version 2.0 would be reviewed.

The value of the surety assessment is in the opportunity to have an independent, third party team of system security analysts evaluate the system for its overall security posture as well as to explore specific security features incorporated into the system. Based on the team's findings, issues of concern are identified so that the developers can explore mitigation strategies, resulting in a more secure product.

Note that the purpose of this design review is not to gauge the ability of the system to meet all performance objectives, but rather to assess its security features, and security posture, as well as the potential to operate securely. Additionally, some consideration is given to implementation issues, namely those that have the potential to impact the security stance and operational capability of the system.

# 2. X-Bone Objectives[2]

The core X-Bone objective identified is to offer a system capable of automated network overlay creation, deployment, and management. An underlying motivation includes the development of a system that supports the following:

- Network experimentation (in a laboratory environment)
- Network labs (supporting multiple classes and experiments)
- Virtual Private Networks (isolated, restricted-access infrastructure)

Other objectives identified include:

- Portability
- Extensibility

Security-specific objectives identified include:

- Secure overlay deployment
- Secure overlay traffic (optional)

Because the primary interest of the design review is focused on security-related aspects of the X-Bone system, it is important to note what the system is not designed to do: namely, increase the security posture of the network resources on which it is deployed.[3]

# 3. X-Bone Description

This section will provide a brief overview of the concept of operations and key system components, as well as an explanation of the system software code. Additionally, an explanation of the developers approach to securing the overlays as well as the construction and management of the overlays is offered. Finally, the system views developed by the assessment team are presented.

## 3.1 Overview of Operations and Components[4]

X-Bone is a system for the definition, construction, monitoring, and management of virtual network overlays. As such, it has components designed to facilitate the discovery of network resources, resource invitation through multicast, resource selection, resource configuration, overlay maintenance, and a graphical user interface. A central overlay management (OM) object processes user requests to build, monitor, and deconstruct the overlays. OMs build overlays by sending specific configuration information and specific, trusted commands to individual network resources. Resource Daemons (RDs) reside on each network resource (host) in the X-Bone infrastructure. RDs process and respond to invitations, configure the host resources, respond to status requests, and remove their individual hosts from overlays. The OM maintains overlays by sending keepalive messages (heartbeats) to the RDs of hosts in any given overlay. Note that all resource-related tasks performed by the OM are done out of band of any overlays.

## 3.2 The Application Code

X-Bone 2.0 is written in Perl, version 5, to run on FreeBSD or Red Hat Linux operating systems. Additional open-source Perl modules are used for common functions like setting up SSL sessions, evaluating IP addresses, generating icmp pings, etc. Version 2.0 of the application provides a web CGI as the interface for configuring overlays. This method was chosen as it allows the use of a standard web browser as a client. In a secure implementation of the current version, the developer recommends installing the CGI on the same host as the OM to mitigate a known vulnerability; the distribution installs this way by default. Please note that the system was examined by IDART™ according to this recommended implementation choice. The team did not investigate the system under a scenario where the OM and the CGI would reside on separate hosts.

The list of X-Bone modules and their relationships to each other can be seen in the diagram in Appendix A, Figure 6. The top area of the diagram depicts the OM and CGI-

GUI components. Both the CGI process and the OM process use the API modules. The CGI uses some of the modules for formatting the overlay requests for submission, and the OM uses some for parsing and verifying these requests. Modules that make up the RD process and modules that are common to both the OM and the RDs – namely utilities used for secure communications – are illustrated in the lower portion of the diagram.

In addition to the Perl modules that ship with the distribution, the list of software that is required for this version of the application is as follows:[5]

- Perl5 5.005_03 or 5.6.0
- Perl modules:
    - Net::Netmask (OM/GUI & RD)
    - Net::SSLeay (OM/GUI & RD)
    - CGI.pm (OM/GUI)
    - LWP::Simple (libwww) (OM/GUI)
    - File::CounterFile (OM/GUI)
    - Mail::Sendmail (OM/GUI)
    - Parse::RecDescent (OM/GUI)
- OpenSSL 0.9.5a or above (OM/GUI & RD)
- Apache-SSL: apache_1.3.9+ssl_1.37 or above (OM/GUI)
- DNS server (named): bind 8.2.3 or above (OM/GUI)
- iproute2 *Linux Only* (RD)

## 3.3 System Security Mechanisms and Capabilities

The strategy employed by the developers to achieve secure overlay deployment and management is multifaceted, while that used to secure the traffic within the constructed overlays is data authentication and encryption.

Note: the use of authentication and encryption within the overlays is optional, and must be specified by the user.

### 3.3.1 Secure Overlay Creation[6]

The developers' approach to secure overlay construction includes the use of X.509 certificates, SSL-encrypted communications, S/MIME-signed communications, and access control lists (ACLs). Specifically, a hierarchical trust model using X.509 certificates is employed and communications are SSL-encrypted between the user's host and the web server, as well as between the central, coordinating host (that which holds the OM object) and the network resources (during overlay configuration and to maintain overlays with the use of keepalive messages). S/MIME-signed UDP messaging is used during the resource discovery process. Resource hosts (those with the RD object) utilize access control lists (ACLs) that are intended to provide a level of control over resource availability. Please refer to Figure 1 in section 3.4.1 for further discussion.

### 3.3.2 Data Security

IPsec is used to secure overlay traffic within the overlay tunnels, either authenticating the traffic or authenticating and encrypting it. Traffic is encrypted and decrypted at the tunnel endpoints in the overlay. Separate key pairs exist for all tunnel endpoints in any given overlay. Additionally, use of IPsec within the overlays does not hinder the use of IPsec on the base network, should such an arrangement exist.[7]

## 3.4 System Views

In performance of the assessment, the Red Team generated multiple views of the X-Bone system. These views represent the understanding of the assessment team, and were generated based on developer documentation and interviews conducted with the development team. The development of views is intended to enhance and confirm the assessment team's understanding, aiding in the characterization process.

### 3.4.1 Secure Overlay Creation

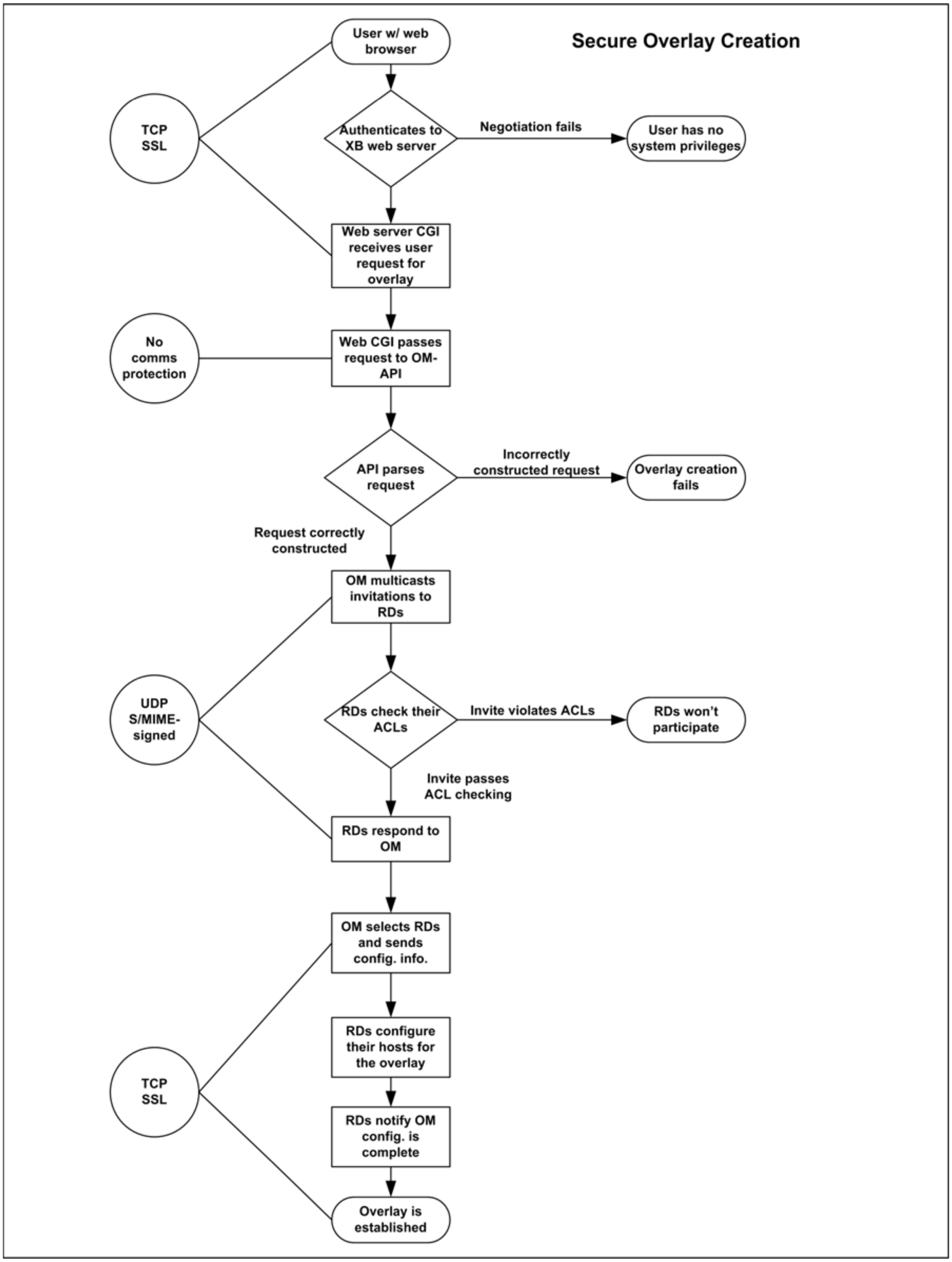

Secure Overlay Creation
User w/ web browser
TCP SSL
Authenticates to XB web server
Negotiation fails
User has no system privileges
Web server CGI receives user request for overlay
No comms protection
Web CGI passes request to OM-API
API parses request
Incorrectly constructed request
Overlay creation fails
Request correctly constructed
OM multicasts invitations to RDs
UDP S/MIME-signed
RDs check their ACLs
Invite violates ACLs
RDs won't participate
Invite passes ACL checking
RDs respond to OM
OM selects RDs and sends config. info.
RDs configure their hosts for the overlay
TCP SSL
RDs notify OM config. is complete
Overlay is established

**Figure 1**

Figure 1, above, reveals some of the security-related aspects of overlay creation. Users initiate SSL-encrypted sessions to the X-Bone web server, using X.509 certificates to authenticate. Users then can request overlay construction. The web server passes the request to the OM-API, which performs syntax checking of the request. The API passes correctly constructed requests to the OM.[8] The OM multicasts UDP invitations (S/MIME-signed for message replay protection, unencrypted) to all host RDs. Access control lists (ACLs) are used by the RDs and are intended to preclude positive replies to the OM where the user credentials in the invitations are unauthorized or where acceptance could result in a violation of a maximum overlay count that the host is allowed to participate in. RDs respond to invitations that are not filtered by the RD ACLs. The OM then selects hosts from the pool of responding RDs that will participate in the overlay, and initiates parallel TCP-SSL connections to those hosts. An overlay configuration (set of commands with parameters) is issued by the OM to the host-RDs over this secured channel. Host-RDs then issue commands to the host kernel for configuration at the host level. Once the overlay is active, the SSL connection between the OM and the host RD is released. OMs maintain overlays by sending regular keepalive messages over a TCP-SSL channel to each RD (not shown).

### 3.4.2 Example X-Bone Topology

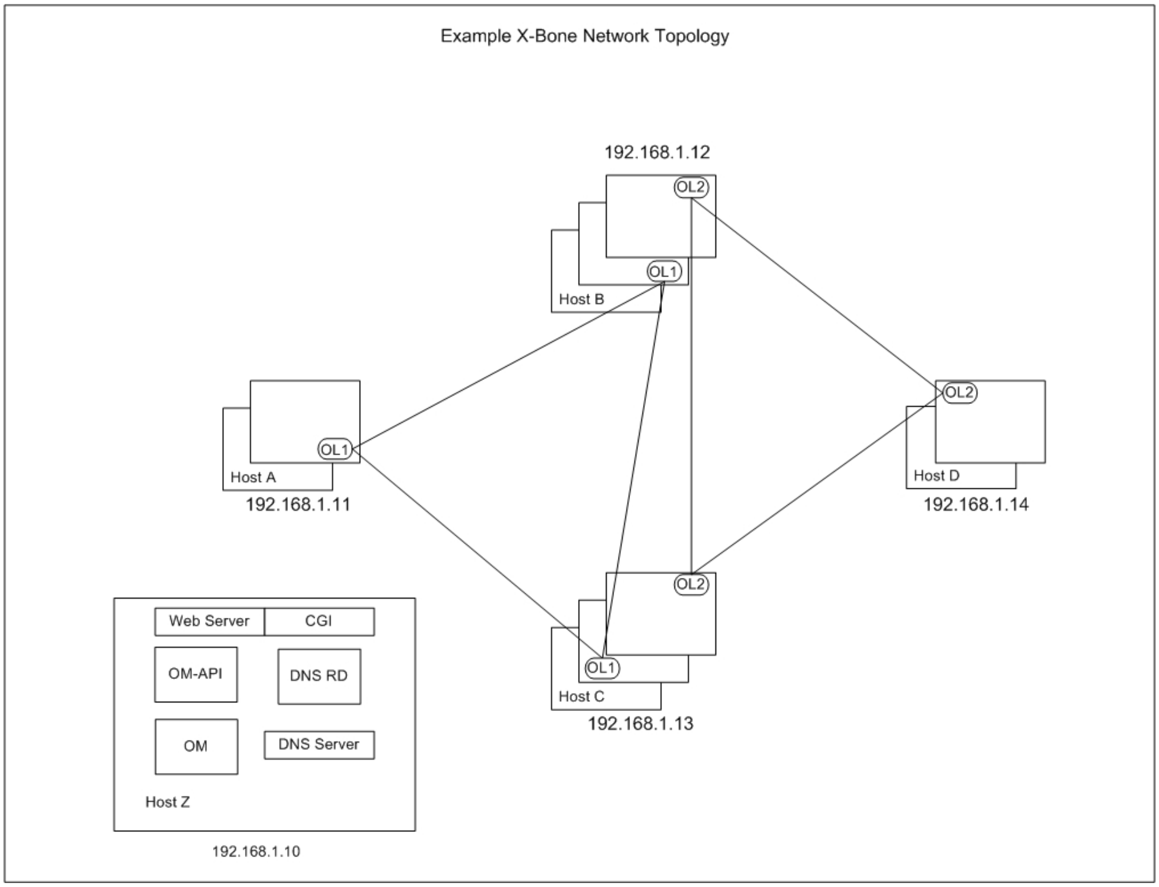


**Figure 2**

This example shows five hosts on the real, base IP network 192.168.1.0/24, all participating in the X-Bone infrastructure. Host Z maintains the Overlay Manager (OM) object and its API, the Web Server components, and the DNS-RD and Server components. This view demonstrates that an OM can manage multiple overlays. Note that only one OM object exists per OM-host. Each network resource (host) has only one Resource Daemon (RD) (not shown) regardless of the number of overlays that the host belongs to. This view also illustrates that the OM does not participate in overlays. Both the OM and DNS play supporting roles. DNS is not a required component of the X-Bone system. Note also that the base IP addresses are shown, but not the virtual host IPs for the overlays.

### 3.4.3 X-Bone Component Communications

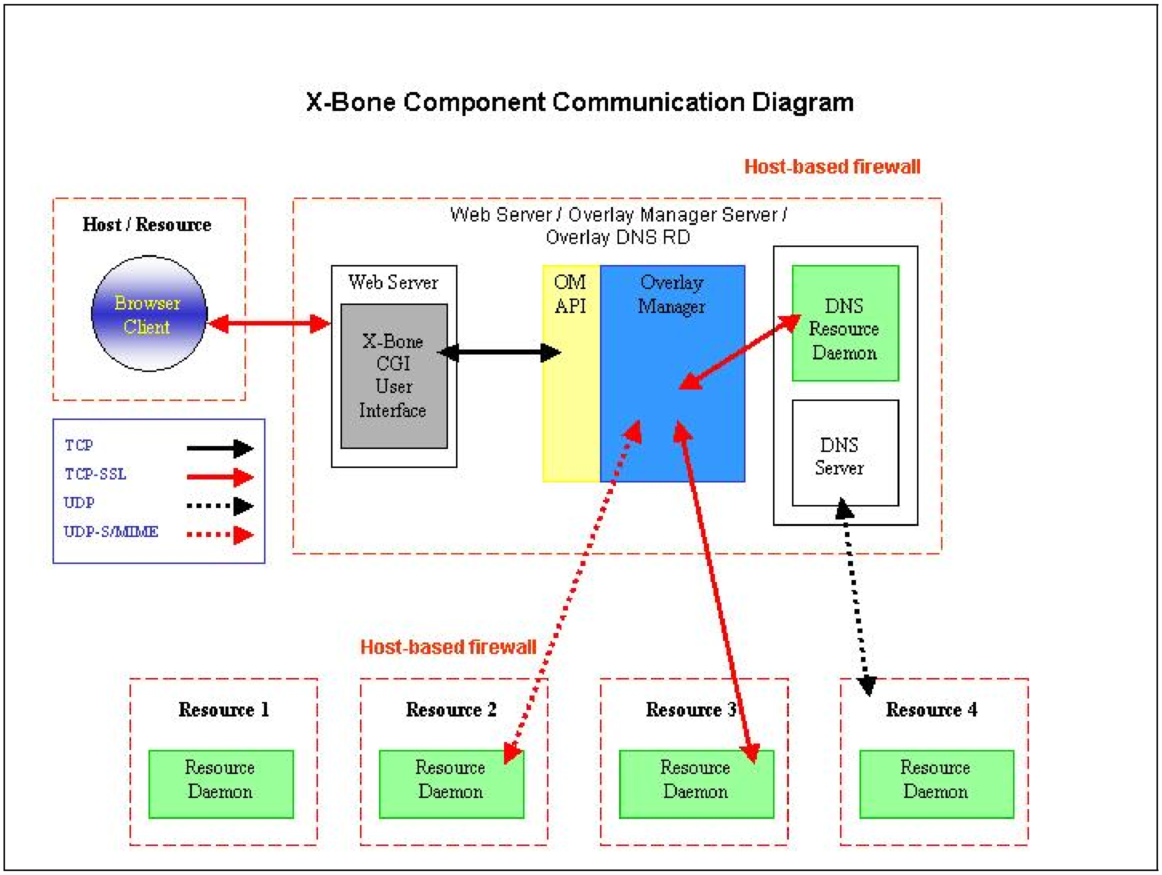
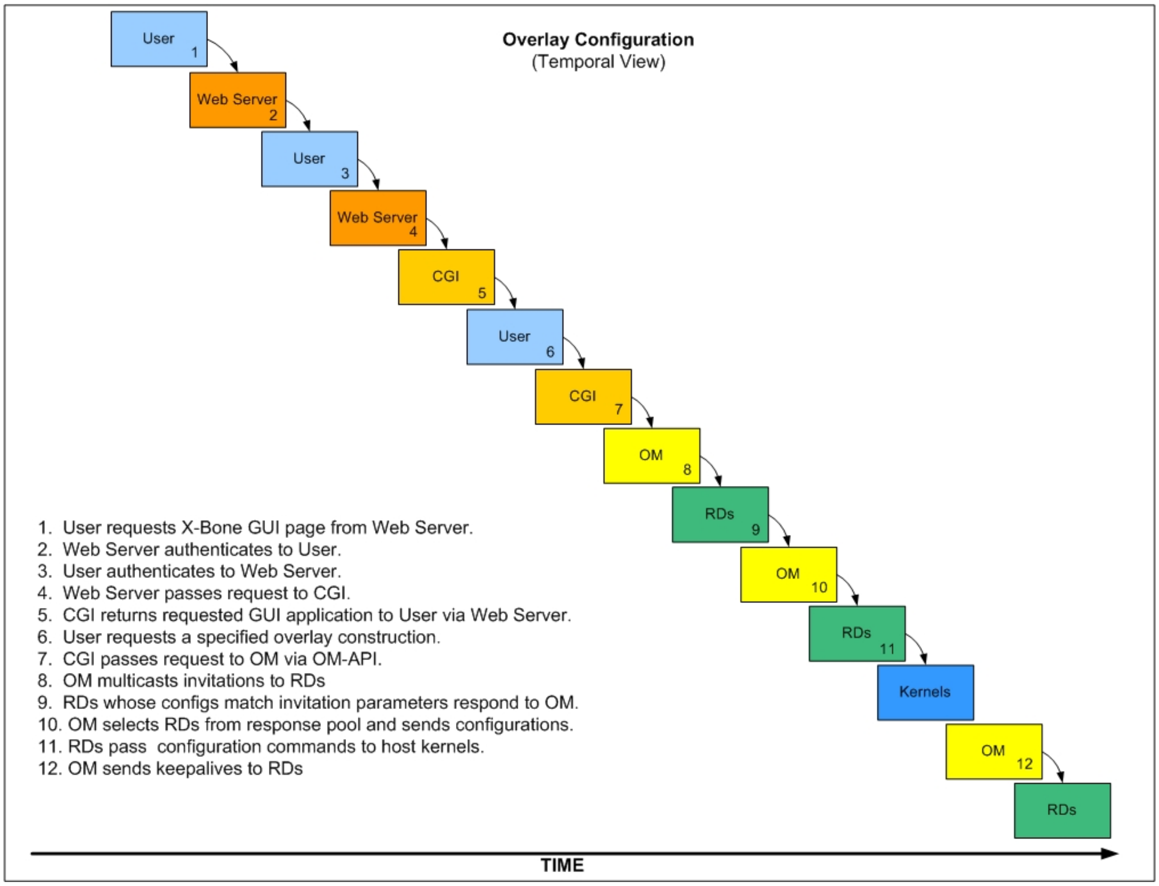


**Figure 3**

Figure 3 illustrates key X-Bone components and where they reside in a typical implementation of X-Bone v2.0. In this example, the Web Server and CGI, OM and OM-API, and DNS RD and Server reside on a single host. X-Bone network resources (e.g., Resource 1, 2, etc.) are indicated with their key X-Bone object, the RD. Note that the DNS RD and Server functions are not required to reside on the same host as the OM; the optional DNS service may be provided by another host. The DNS service operates over the real network-addressing scheme, as the DNS Server does not participate in the overlay addressing schemes (though in theory it may be possible). The DNS Server maintains tables mapping overlay hostnames to their respective virtual IP addresses.

Please refer to Appendix B, Figures 12-19 to view a typical sequence of system communications and the nature of the communications (e.g., TCP-SSL, UDP-S/MIME, etc.).

### 3.4.4 Overlay Configuration

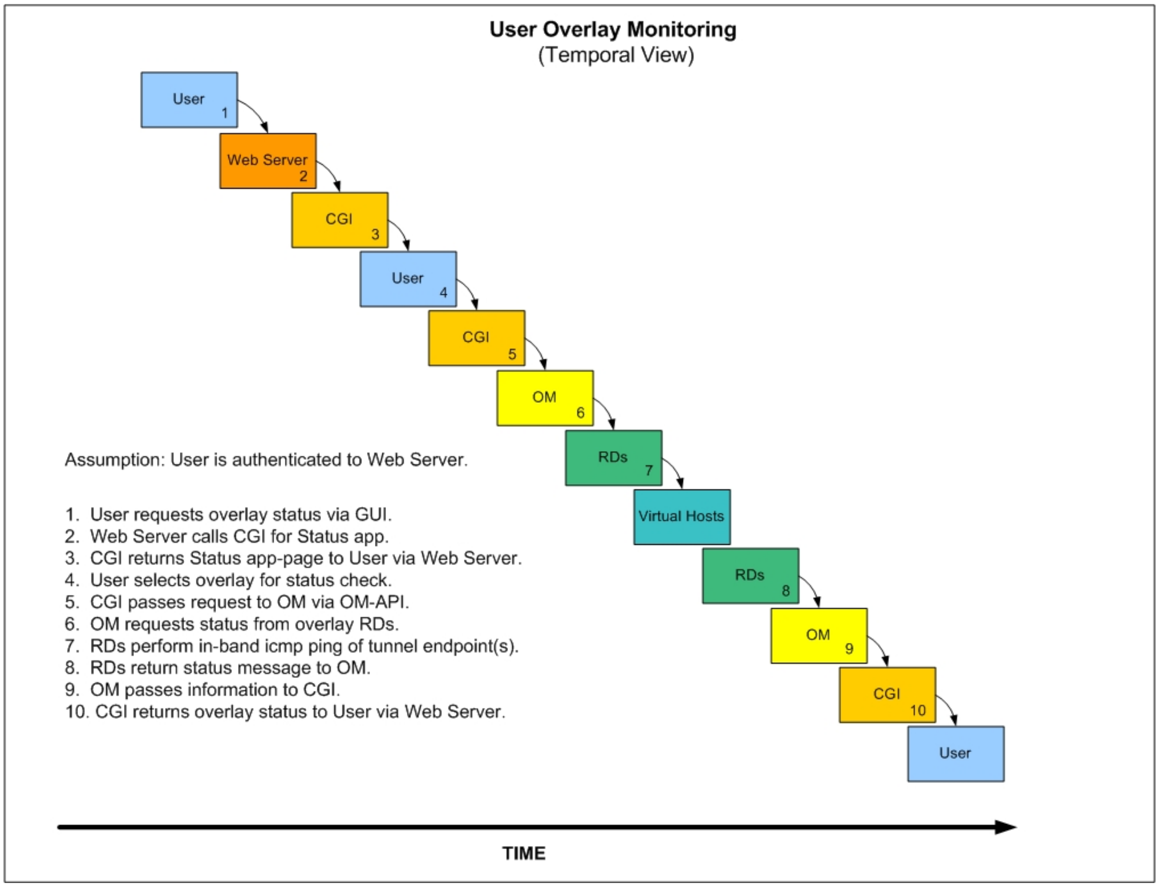


1. User requests X-Bone GUI page from Web Server.
2. Web Server authenticates to User.
3. User authenticates to Web Server.
4. Web Server passes request to CGI.
5. CGI returns requested GUI application to User via Web Server.
6. User requests a specified overlay construction.
7. CGI passes request to OM via OM-API.
8. OM multicasts invitations to RDs
9. RDs whose configs match invitation parameters respond to OM.
10. OM selects RDs from response pool and sends configurations.
11. RDs pass configuration commands to host kernels.
12. OM sends keepalives to RDs

**Figure 4**

Figure 4 illustrates the temporal sequence of events necessary for the OM to process a user’s request to build, deploy, and maintain an overlay.

### 3.4.5 User Monitoring



**User Overlay Monitoring**
(Temporal View)

Assumption: User is authenticated to Web Server.

1. User requests overlay status via GUI.
2. Web Server calls CGI for Status app.
3. CGI returns Status app-page to User via Web Server.
4. User selects overlay for status check.
5. CGI passes request to OM via OM-API.
6. OM requests status from overlay RDs.
7. RDs perform in-band icmp ping of tunnel endpoint(s).
8. RDs return status message to OM.
9. OM passes information to CGI.
10. CGI returns overlay status to User via Web Server.

**Figure 5**

A representation of the temporal sequence of events necessary for the user to perform an overlay status check is shown in Figure 5.

## 3.5 Additional System Views

Additional views procured from ISI and used by the assessment team are included in Appendix A.

## 3.6 Detailed Description

For a detailed explanation of the system architecture and operations of the X-Bone system, the reader is referred to the developer site:

http://www.isi.edu/xbone/

# 4. Scope of IDART™ Assessment

IDART™ was tasked to perform a design-level surety assessment of X-Bone. The assessment is limited in scope and was not part of a larger, active experiment with an associated engagement of red, white, and blue teams. Instead, the developers were asked to supply detailed design documentation, the application code, and make themselves available to the Red Team to answer specific questions and provide clarification on system purpose, function, and design features. An attack brainstorm was held at Sandia National Labs in which various attacks against X-Bone were theorized. Several, existing vulnerabilities were acknowledged by the developers.

Note: The X-Bone system was not installed on Sandia laboratory infrastructure for detailed testing.

## 4.1 Developer Claims

Regarding the security aspects of the X-Bone, the developers make two critical claims[9]:

- Data security within the overlays (optional)
- Secure overlay deployment and control

The focus of the assessment is on these two critical claims.

## 4.2 Known Vulnerabilities[10]

The system developers have acknowledged the following vulnerabilities in advance of this assessment:

- Storage of X-Bone specific information (including keys used for encryption) on participating hosts is unencrypted, with root-level access.
- A root-privileged user on a given host has complete control over all system parameters, including those used by X-Bone.
- Root compromise of a host participating in multiple overlays implies compromise of all overlays that host participates in.
- Users can deploy applications running at root level on X-Bone resources (disabled by default in 2.0).
- Source code is easily modified (and could be done so for malicious purposes).

These vulnerabilities were not explored in any significant detail by the IDART™ team; only so far as to understand the nature of the stated concern and where appropriate, to validate that concern. The developers have not indicated that a solution is in the works to address these vulnerabilities.

Additional, known vulnerabilities that the developers may address in future versions of X-Bone include:

- Communication between the web server and OM components is unencrypted, unauthenticated.
- All UDP system control communication is unencrypted (but authenticated).
- Shared keys distributed by the OM (for secure tunnels) are static for the duration of the overlay.
- OM maintains all keys (IPsec) for all links in all overlays it manages.
- Perl modules lack employment of taint mode.
- Calls to external programs could represent a security risk
- Antiquated code (e.g., that which authenticated users to http sessions by IP address).
- DNS lookups are unsecured.

As the developers are aware of these vulnerabilities and have discussed their intention to resolve them in future versions of X-Bone, they were not further analyzed in this assessment.

# 5. IDART™ Analysis of X-Bone

The assessment tasks included a review of all available documentation, analysis of the application code, and interviews with the system developers. Additionally, multiple views of the X-Bone system were built in order to further the Red Team's system understanding, and a brainstorm was held in which attacks were theorized against X-Bone.

## 5.1 Attack Brainstorm

An attack brainstorm was held in order to postulate potential attacks on X-Bone overlays and on the X-Bone system itself, including the component pieces that make up X-Bone. The IDART™ X-Bone assessment team leveraged the expertise of additional information system security analysts at Sandia National Labs, in order to allow an even larger set of analysts to participate in the review. Brainstorm personnel included the X-Bone assessment team, additional members of the Information Operations Red Team and Assessments (IORTA™) group, Networked Systems Survivability and Assurance Department, and Cryptography and Information Systems Surety group.

## 5.2 Attack Scenarios

Possible adversary starting points (ranging from a remote, non-RD host to a RD host) were defined and specific adversary objectives were developed for the brainstorm, including:

- Form a leak in an overlay (threatens confidentiality, authenticity, potentially availability, etc.)

- Breach an overlay (without joining: threatens confidentiality, potentially availability)

- Denial of Service (DoS)

  - Block user(s)/system from building overlays
  - Block user(s)/system from eliminating overlays

The majority of attack vectors produced in the brainstorm describe an effort to DoS the X-Bone system by one means or another. Misuse of X-Bone network resources (e.g., malicious consumption of resources) represented one of the more popular ideas discussed. “DoS-ing” specific target hosts on the base network represented a more obvious means; identification and targeting of OM-hosts would be a clear objective in a broad DoS attack. An additional, potential attack that generated interest included one whose objective would be to “leak” information out of the overlay via a “double agent” RD. Actual RD hosts and non-RD hosts on the real network segment able to become or spoof an RD host were also considered to be potentially valuable targets. A recurring node of interest was the DNS server.

## 5.3 Critical Success Factors

As part of the IDART™analysis process, it is customary for the Red Team to identify key factors that are critical to the ability of the system to fulfill its mission. Review of developer documentation, inspection of the application code, and discussions with the development team indicate the following critical success factors for X-Bone v.2.0:

Security-specific critical success factors identified:

- Implementation Factors
- Operational Factor
- Host Security
- Certificate Authority Security

These are discussed in the following sub sections.

### 5.3.1 Implementation Factors

Implementation issues affect both key critical developer claims, with respect to system security.

- Web Server Security

Until the developers modify the code to support a secured link between the CGI and the OM, they recommend that these components be installed on the same host, using the localhost channel to communicate for security reasons. The system architecture supports relocation of the Web Server to a separate host. As this component plays a critical role in the system, it necessitates use of security best practices to protect it.

- Use of the DNS components

Because the DNS Server maintains tables with overlay specific information (virtual hostnames and their virtual IP addresses) and the lookups are done insecurely, use of these components could have a degrading effect on the overall system security posture. Additionally, should DNS be utilized in an X-Bone configuration, the operation of the overlays may depend on many security-related configuration parameters of the DNS server(s) itself. The concern is that an attacker might be able to:

  - Use the DNS as a stepping stone
  - Change the zone files
  - Otherwise DoS the DNS service

- System is deployed correctly

Installation of the system is not trivial. There are opportunities for mistakes to be made in the configuration of the hosts with the X-Bone software. Proper deployment also depends on the correct installation of the underlying utilities.

- ACLs are configured to properly restrict access to resources

Failure to configure ACL parameters with thoughtfully chosen limits or manipulation of the ACLs could result in a system configured with controls insufficient to protect it from certain types of attacks (e.g., resource depletion).

### 5.3.2 Operational Factor

The following issue represents an operational consideration that defines whether or not a deployed overlay will secure the traffic.

- Users are responsible for selecting authentication and IPsec for traffic encryption

In order for the system to provide secure tunnels, it relies on the user to properly configure his/her overlay request to include both authentication and IPsec for traffic encryption. User choice or error can result in unsecured overlays.

### 5.3.3 Host Security

As it has been discussed that host compromise implies compromise of all overlays that the host is a member of, individual host security becomes a critical success factor. The following are some host-level considerations:

- Effective firewall rules
- Patched OS and security assets (e.g., SSL)
- Key/message ID generation technique
- Non-trivial passwords
- Account control
- File integrity

### 5.3.4 Certificate Authority Security

The use of trusted X.509 certificates is essential to the security of an X-Bone version 2.0 system. The application depends on a "chain of trust" that is based on host security and a trusted PKI. This chain begins at the point of user authentication during the SSL mutual authentication that occurs when a user accesses the web cgi GUI. The next link is the closed communication loop between the cgi and the X-Bone OM that it interfaces with – through a protected communications path (this link is not PKI secured, but may be in future). The final link is established between the OM and all RDs that are configured to trust that OM. This last trust takes two forms:

1) Signed S/MIME invitations that are sent out via multi-cast UDP by the OM. RDs authenticate these and check the verified distinguished name against their own ACL before they will respond.
2) Mutual authentication of OM and RD during the SSL session that is established to transfer the overlay configuration information (for each RD that is chosen to participate in the overlay).

A user's identity is based on the distinguished name that is extracted from the user's trusted certificate. This is passed to the OM with the overlay request and consequently on to each RD. Each RD will check the user identity string against its user ACL to determine if the requester has authorization to utilize the RD host in an overlay. This assumes that the user string has been passed unchanged from a trusted certificate presented to the web cgi during the overlay request.

If the X-Bone implementation is to have a limited and known user/host base, the certificate authority (CA) can be local - trusted by only those participating in X-Bone overlays. If X-Bone is to be used for larger, distributed overlays it would be necessary to utilize a more robust and recognized CA  (e.g., an enterprise-wide PKI or perhaps a commercial third-party vendor).

With the "chain of trust" used by the X-Bone system, it is essential that the certificate authority used to issue and verify these certificates be appropriately secured. Attackers able to successfully spoof a recognized certificate authority (CA) could compromise the entire system.

## 5.4 X-Bone System Security Issues of Interest

The following security-related issues of interest are grouped by category:

- Access Control Lists
- Overlay Manager
- General System

### 5.4.1 Access Control Lists

- ACLs fail to limit administrative resets by user ID. Administrative resets are validated by the OM permit list on the RDs. All overlays that an RD belongs to will be deleted by the RD when a reset message is received.[11]

- ACLs on RDs fail to limit replies to discovery messages.[12]
- OMs lack ACLs. Could the use of them at the OM limit exposure to potential resource consumption attacks?

### 5.4.2 Overlay Manager

- Version 2.0 does not incorporate back-up OMs. If the OM or OM-host fails, all overlays it manages are destroyed.
- OM state is not recoverable. A reboot of the OM host means that the users must rebuild all overlays.
- OMs can be easily identified by use of unencrypted UDP packets. Thus the host that deploys overlays is easily determined.

### 5.4.3 General System

- The user defines the security of the overlay.
- Version 2.0 lacks an alarm capability. RDs can disappear without any action taken by the OM. The OM will only discover a failed node upon the issuance of a status request by the user.
- X-Bone lacks a facility to self-check the security of the overlays that it constructs. There is a presumption that an overlay defined by the user to be secure is secure based on the status message that the overlay is up.

## 5.5 Analysis of the X-Bone Code

In general the code is well formed, and shows systematic top down development and revision. A consistent coding style was used throughout the project, making it easier to follow and verify assertions made in the documentation. The code uses variable names that aid in following flow, and is well commented. Some of the comments are obviously from older revisions, and this should be cleaned up. There are also debug segments of code that should be integrated into the overall design, rather than left as adjunct code.

The development scheme uses a layered approach, where communications packets are built and then wrapped in SSL, or IPsec. This allows for the logical separation of layers for debugging and testing. It is also a design that could easily lend itself to module level testing/debugging, and indeed there is some of this in the code.

The general scheme of the implementation is to build (at an administrative level) the configuration for all potential overlay members, and then allow a user to select specific options. This decreases the impact of user level error. In cases, such as the building of virtual IP addresses the code exhaustively checks for correct format and address collisions. Throughout the application code care is taken on the constructive and shared segments of code in both the RD and OM to ensure data integrity.

One issue of concern is the lack of “taint” mode in the Perl, although it is acknowledged that use of this would conflict with some of the external modules. A general design issue is that there seems to be little effort to guard against insider threat; this is where the inclusion of taint mode or conversion to compiled code could help.

Administrators of X-Bone hosts should also be warned about the generic components the system employs in order to avoid conflicts with other applications or intended host usage.

## 5.6 Recommendations

The following is a list of recommendations, some of which are offered to address a discussed vulnerability. Others are more general in nature. Both vulnerabilities discussed by the IDART™ team and some (but not all) vulnerabilities previously acknowledged by the developers are considered. Note: Many (but not all) of the security-related recommendations are targeted at the implementation rather than the application.

### 5.6.1 Security-Related Recommendations

As the security posture of the system is dependent on the underlying infrastructure, strong, resource-based defenses should be employed. Our recommendations include the following:

- Evaluate the possibility of building firewall rules for each resource which:
  - Deny traffic from non-overlay addresses into the overlay interfaces.
  - Deny traffic from overlay interfaces to non-overlay interfaces.
- Build rules that block traffic from entering a network interface from localhost (especially on the OM host, as it is currently using this channel for communications between the API and OM).
- Consider the need for host-based integrity checking to assist in the prevention / detection of unauthorized system software changes.
- Study the use of homogenous resources (e.g., common platforms and packages) to improve manageability and reduce implementation mistakes.

As DNSSEC is not yet a viable protocol for integration into the X-Bone system:

- Do not use DNS in security-critical implementations of X-Bone.

Where DNS is a required element of the X-Bone deployment, consider using a DNS server independent of the non-overlay DNS server. Use security best practices when configuring all DNS servers, including:

- Disable recursive queries and glue fetching to prevent spoofing attacks.
- Restrict zone transfers.
- Run *named* with a non-root user with sufficient permissions to accommodate X-bone requirements.
- Where possible eliminate the use of X-Bone and non-X-Bone DNS services on the same host.
- Evaluate the potential to use ACLs to limit lookups to the configured OL hosts if using a separate OL DNS Server.

If application or script deployment (disabled by default in 2.0[13]) is to be a capability of a particular X-Bone overlay:

- Configure ACLs to filter a specific subset of users able to deploy scripts.

As the developers have suggested, a security-minded package could be developed that eliminates security holes. Such a package could include the following recommendations:

- Remove the Administrative Reset feature from the GUI.
- Force all overlays to use both authentication and encryption (eliminate the user's ability to choose an unauthenticated, unencrypted overlay definition).
- Include a routine to verify the user's intent / confirm their selections in order to reduce user error.
- Inform users and administrators when debug mode is active.
- Reissue X-Bone in a compiled language, moving away from application scripts to an executable.

### 5.6.2 Improving the Utility of the X-Bone System

Suggestions to enhance the functionality of the X-Bone system include:

- Develop a system that can build multiple network overlays (version 2.0 handles only single network overlays).
- Provide the user with the capability to select specific resources in their overlay request. Currently the OM makes an arbitrary selection of OL resources.

### 5.6.3 Improving Application Robustness

Significant opportunities exist to eliminate single points of failure in the control system and provide greater system recovery capability. Incorporation of the following suggestions could strengthen the overall resiliency of the system, especially important in a high-utilization or operationally critical environment:

- Incorporate the use of backup OMs on separate hosts that are capable of hot standby.
- Incorporate the use of alarms to notify both the various X-Bone system components and the end user of system problems.
- Develop a dynamic, automated OL management capability. An OL reconfiguration-on-the-fly capability would preclude the need for user intervention to rebuild failed overlays. Currently, loss of a single RD-host can or will cause the entire overlay to fail.
- Make the OM state recoverable. Presently, only RDs are recoverable.
- Consider the need for redundant web servers to ensure availability (the use of a single web server is, potentially, a single point of failure).

# 6. Conclusions

X-Bone is not an application that enhances the base network fault-tolerance. That said, it is a cleverly conceived and designed application that offers the advantages of flexible overlay definition, configuration, and implementation. Although deployment and configuration of the system software are not foolproof, a correctly outfitted network provides to the end user an easily configurable infrastructure from a straightforward GUI. The developers have taken an approach that balances the incorporation of existing security-related mechanisms against system functionality. Much of the system security posture is left in the hands of the system owners and users. Besides the ease of use of the application, X-Bone's strengths are in its modular, object-oriented design, and extensibility.

Principle claims of secure overlay construction and secure overlay traffic are largely met. Of course no system can be guaranteed to be completely secure and X-Bone is not an exception. X-Bone's ability to provide secure network overlays is dependent on the security of the underlying infrastructure, as well as a variety of operational and implementation factors.

Aside from the known vulnerabilities, it is our opinion that no major flaws have been discovered within the system design. However, there appear to be opportunities to improve the security posture of the application code itself and to build additional utility and robustness into the system. A more secure package would eliminate certain operational and implementation dangers, and reduce the exposure of the code to malicious compromise. A more robust system could improve fault tolerance of the application significantly. Lastly, a system that gives the user greater control of overlay resources could improve the application's utility.

# Information Sources

[1] http://www.isi.edu/xbone, "What is the X-Bone?"

[2] Objectives are inferred from the following sources:
http://www.isi.edu/xbone
http://www.isi.edu/xbone/information/uses.html
http://www.isi.edu/xbone/OLD/faq.html
http://www.isi.edu/xbone/x-tend.html

[3] Yu-Shun Wang, X-Bone development team, ISI, e-mail 18 Ap. 2003, "Participating in an overlay does not make the host more or less secure."

[4] http://www.isi.edu/xbone

[5] http://www.isi.edu/xbone/software/release_2_0.html

[6] Multiple sources, including:
http://www.isi.edu/xbone/om.html
Lars Eggert and Yu-Shun Wang, ISI, X-Bone development team, Teleconference, 01 Nov. 2002
Eggert, Wang, and Joe Touch, ISI, X-Bone development team, Teleconference, 04 Nov. 2002

[7] Joe Touch and Lars Eggert, "Use of IPsec Transport Mode for Virtual Networks," Internet-Draft, 01 Mar. 2002: 4.2

[8] "Xbone Application Programmers Interface: Abstract and Overview."

[9] http://www.isi.edu/xbone

[10] Multiple sources, including:
http://www.isi.edu/xbone/information/issues/security.html
http://www.isi.edu/x-bone/security-issues.html

[11] Ibid

[12] Joe Touch, X-Bone development team, ISI, e-mail 04 Nov. 2002

[13] http://www.isi.edu/x-bone/security-issues.html