ISI Research Report
ISI/SR-93-358
October 1993

The ISI "Tunnel"

Annette DeSchon and Danny Cohen
ISI/SR-93-358
October 1993

University of Southern California
Information Sciences Institute
4676 Admiralty Way, Marina del Rey, CA 90292-6695
310-822-1511

This research was supported by the Defense Advanced Research Projects Agency under Ft. Huachuca contract number DABT63-91-C-0001, entitled "Gigabit Network Communications Research". The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed

REPORT DOCUMENTATION PAGE

FORM APPROVED
OMB NO. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	October 1993	Final Report June 1992 - October 1993

4. TITLE AND SUBTITLE
The ISI “Tunnel”

5. FUNDING NUMBERS
C-DABT 63-91-C-0001

6. AUTHOR(S)
DeSchon, Annette and Cohen, Danny

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
USC INFORMATION SCIENCES INSTITUTE
4676 ADMIRALTY WAY
MARINA DEL REY, CA 90292-6695

8. PERFORMING ORGANIZATON REPORT NUMBER
ISI/SR-93-358

9. SPONSORING/MONITORING AGENCY NAMES(S) AND ADDRESS(ES)
DARPA
3701 N. Fairfax Drive
Arlington, VA 22203-1714

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12A. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited

12B. DISTRIBUTION CODE

13. ABSTRACT ***(Maximum 200 words)***

The ISI Tunnel allows sites that are hidden behind “firewalls”, also known as “gatekeepers” or “mail-bridges”, to have IP-based internet access without being open to attacks over the Internet.

The Tunnel is a special router that provides smooth seamless internet access from a closed environment, the “inside”, to the “outside” while restricting the access from the outside to the inside. The advantage of the Tunnel, in comparison with traditional firewalls, is that it supports any IP-based communication; not just terminal access, file transfer, and electronic mail.

14. SUBJECT TERMS
firewall, gatekeeper, packet filtering, internet security

15. NUMBER OF PAGES
39

16. PRICE CODE

17. SECURITY CLASSIFICTION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UNLIMITED

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. Z39-18
298-102

The ISI “Tunnel”

Annette DeSchon and Danny Cohen
USC / Information Sciences Institute[1]

Abstract

The ISI Tunnel allows sites that are hidden behind “firewalls”, also known as “gatekeepers” or “mail-bridges”, to have IP-based internet access without being open to attacks over the internet.

The Tunnel is a special router that provides smooth seamless internet access from a closed environment, the “inside”, to the “outside”, while restricting the access from the outside to the inside. The advantage of the Tunnel, in comparison with traditional firewalls, is that it supports any IP-based communication; not just terminal access, file transfer, and electronic mail.

1.0 Overview

The Tunnel automatically allows any IP-based two-way communication between an internal and an external host when the communication is initiated from the inside. However, the Tunnel prevents communication initiated on the outside, unless this communication was explicitly authorized previously, either by a user identified by a password, or by the system administrator.

The Tunnel forwards a packet only if the Tunnel has a *visa* that matches (1) the source host, (2) the destination host, and (3) the protocol that is specified in the IP-header of that packet. Each *visa* is created implicitly (based on a packet from the inside) or explicitly (by an authorized user on the outside). A *visa* can be created only if the Tunnel has an entry in its *access-table* authorizing the creation of such a *visa*. Whereas a *visa* is typically defined for a host pair, entries in the *access-table* may have any granularity, such as a single host, a subnet, an entire network, or all the networks.

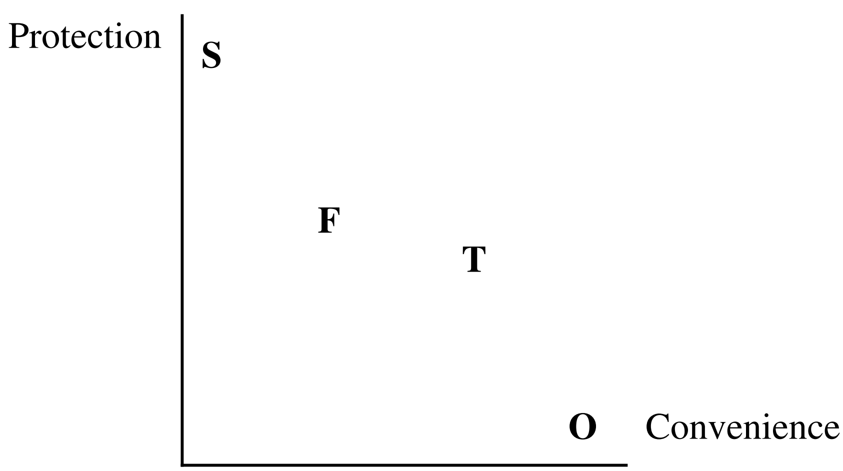
As mentioned above, a *visa* specifies, in addition to the host pair, a transport level protocol (such as TCP, UDP, and ICMP). The Tunnel can also be configured to disallow the use of certain ports for certain protocols. It is therefore possible to disallow the use of “*telnet*” and/or “*finger*” through the Tunnel.

1. ARPA supports the Tunnel Project through Ft. Huachuca contract No. DABT63-91-C-0001 with USC/ISI. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Advanced Research Projects Agency, the U.S. Army, or the U.S. Government.

The Tunnel provides a trade-off between protection and convenience. In general, the more protection the less convenience, and vice versa. There are several parameters that control this trade-off.

The Tunnel can be used either, (a) to “open” environments that are currently closed (behind traditional firewalls) or, conversely, (b) to add protection for environments that are currently completely open. If this is done, users in (a) would most likely enjoy the more open environment, whereas users in (b) would suffer from the inconvenience imposed, without necessarily appreciating the added protection. In this case, education and training may be required to make (b) acceptable to the users.

Figure 1 illustrates the relative position of the Tunnel in comparison with other common protection approaches. An organization in charge of national security might not want to connect their private network with the outside at world at all. An open environment, such as a university, might find the Tunnel to be too restrictive. There are many organizations, both in the government and in industry that have elected to hide behind a firewall that provides some protection, at the cost of great inconvenience. The purpose of the Tunnel is to provide a level of protection similar to the level provided by a traditional firewall, with greater convenience and flexibility.



S = Organization in charge of national security

O = Open environment, such as a university

F = Organization using a firewall

T = Organization using the Tunnel

FIGURE 1. Trade-off between level of protection and convenience.

2.0 Operation

The Tunnel supports any IP-based communication (such as TCP/IP or UDP/IP), and is not limited to *Telnet* connections. Therefore it supports applications that cannot run over *Telnet* or through traditional firewalls, gatekeepers and mail-bridges, such as remote X-windows, FTP, packet-audio, teleconferencing, and NeXT/NeXT communication.

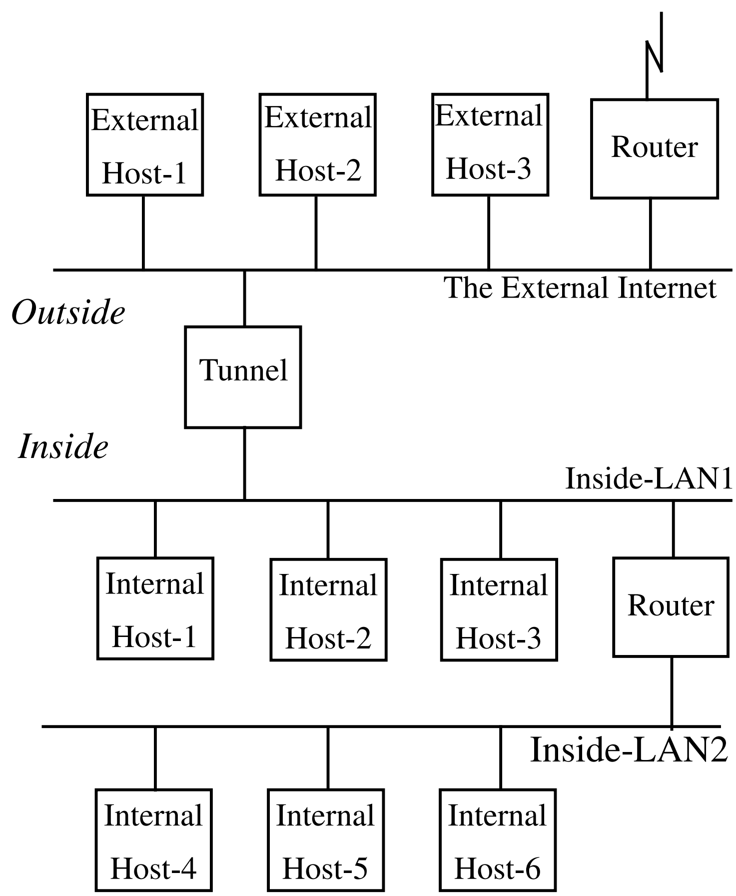


FIGURE 2. Basic Network Configuration

OUTWARDS: The tunnels allows inside users (i.e., on the inside-networks) to implicitly initiate communication with external hosts (i.e., on the external Internet), without any additional effort in most cases.

INWARDS: However, external users (such as local personnel on travel, using external hosts) must use explicit access control to initiate communication with internal hosts. This is implemented as follows. The authorized user first uses *Telnet* to establish a session on the Tunnel, logs in, and creates a *visa*, to allow direct communication (consisting of any IP traffic) between the external host and a specific internal host. Hence, to have direct communication with the inside, the external user must have access privileges (e.g., a password), on the Tunnel.

2.1 Access Privileges

A *visa* allowing two hosts to communicate with each other may be created only if there is an entry in the *access-table* allowing it. *Access-table* entries are typically set up when the Tunnel is booted, however they may also be added and deleted by the System Administrator, using the "*tunnel_access*" utility.

The *access-table* is a set of {from, to} pairs where each of the "from" and the "to" specifies a network, a subnet, or a single host, by specifying an IP-address and a mask, for a total of 2x(32+32) bits for each such pair of (address + mask).

Host-A is allowed to communicate with Host-B through implicit *visa* creation only if the *access-table* has an entry {from, to} such that Host-A matches the "from", and Host-B matches the "to".

However, for explicit *visa* creation, the requirements are less stringent. For example, between Host-A and Host-B, explicit *visa* creation is allowed when the *access-table* has an entry for the host-pair, e.g., {A,B} or {B,A}.

2.2 Visa Creation

The Tunnel allows communication between an internal and an external host if, and only if, there is a valid *visa* for that host-pair, and for that transport-level protocol type.

There are two ways to create *visas*, implicitly or explicitly.

Visas are created implicitly, automatically, for a host-pair when an internal host starts communicating with any external host, provided that there are access privileges for that pair, as described above in 2.1. This allows for smooth, transparent, IP-based communication provided that it is initiated by an internal host.

Visas are created explicitly by authorized users. Examples are as follows:

(1) Authorized users who happen to be outside (e.g., local personnel on travel, using an external host) log into the Tunnel and use the "*create_visa*" utility to create a *visa* for the external host and a specified inside host.

(2) The system administrator (logged in as root) may use the "*tunnel_visa*" utility to create a *visa* for a pair of hosts. *Visas* may also be created as a part of the Tunnel boot sequence.

No communication through the Tunnel may be initiated from the outside without a *visa* that was explicitly created for it.

The protocol type of a *visa* may be defined when the *visa* is explicitly created. The protocols recognized by name are as follows: *ICMP, IGMP, GGP, EIP, ST, TCP, UCL, EGP, IGP, PUP, UDP, IDP, HELLO, ND.* It is also possible to specify a protocol number for a protocol that is not listed. If the default protocol, *ANY*, is specified, the protocol field in the *visa* will be filled in from the protocol field in the IP header of the first packet that is matched with the *visa*. *ALL*, the wildcard-protocol value, matches all values of the protocol field in an IP header and may be specified, provided that the user creating the *visa* is logged in on the Tunnel as “root”.

3.0 Implementation Issues

An original goal of the Tunnel was to have communication that was initiated on the inside be transparent. However, we discovered that keepalives coming from the inside may keep a *visa* going indefinitely in the event that a user does not properly terminate an application. Therefore, a maximum lifetime *DT_life* was implemented to ensure that there was a limit on how long a *visa*’s lifetime could be extended by keepalives.

An additional problem surfaced; if the Tunnel expunged *visas* that had outlived *DT_life*, keepalive packets from the inside would cause a new *visa* to be created. To prevent the creation of a new *visa* caused by persistent keepalives from the inside, we decided to invert the role of the *visa* in this case. Namely, when a *visa* outlives *DT_life*, or the *visa* is explicitly deleted, its role will be to prevent access, rather than to grant access. To accomplish this we mark the *visa* “out of service” and employ a waiting period (*DT_wait*) before the *visa* is expunged from the system. Between the time that a *visa* exceeds *DT_life* or is deleted (*T_exp*), and the time that it can be expunged (*T_exp* + *DT_wait*), the *visa* remains “out of service”. During this period, each time a packet arrives from the inside, the packet is discarded and *T_exp*, the expiration time, is updated, effectively delaying the expunge time.

Unfortunately, however, the Tunnel software is unable to tell the difference between legitimate traffic and keepalives. Therefore the keepalive avoidance logic has two side-effects that can potentially cause inconvenience for the user.

(1) Many users are in the habit of leaving an application, e.g., a mail tool, running for extended periods of time. The keepalive avoidance logic blocks communication after the application has been running for a period of *DT_life*.

(2) Some transaction based applications run for a short time, but are run very often. If such an application is run more often than every *DT_wait*, communication will be blocked after the application has been repeated over a period of *DT_life*.

The solution for both of these problems is that the user must explicitly create a *visa* every *DT_life* time period, whether the user is on the inside or the outside. Practice (1) is incompatible with “life in a closed environment.” Both practices (1) and (2) result in “holes in the fence” that are convenient to use, but their duration should be minimized to reduce the risk.

Another goal of the Tunnel is to guard against the situation in which a person using an outside host leaves his workstation for an extended period of time, and the workstation is used by an unauthorized person. To counteract this problem, we implemented an activity timeout *DT_act* that blocks communication after a period of inactivity. The expiration time (*T_exp*) is set to the current time plus the activity timeout (*T_now* + *DT_act*).

Therefore, if a packet is received from the outside after *T_exp*, the *visa* is considered to be expired. Packets received from the outside when the *visa* is expired are discarded. Packets from the inside, which would result in the creation of a new *visa* if no *visa* existed, are forwarded and renew the *visa* by resetting *T_exp* to (*T_now* + *DT_act*). Hence, *DT_act* effects only traffic from the outside.

The protection against unattended workstations may cause a problem for users on the outside who need more time than *DT_act* to exit their session on the Tunnel and to get their application started, after creating a *visa*. To solve this problem. we allow an initial timeout (*DT_init*), which can be set to a value larger than *DT_act* if so desired. Before a *visa* is used for the first time *T_exp* is set to (*T_create* + *DT_init*), giving the user extra time to exit his *Telnet* session on the Tunnel and start his application.

The *DT_act* still has the potential for causing inconvenience to users on the outside. Therefore, it is important that this parameter be adjusted with the particular application and/or the communication patterns of the particular user in mind. Again, longer is more convenient; shorter is more secure.

3.1 Visa Renewal and Expiration

A *visa* is marked "in service" when it is created. A *visa* is marked "out of service" when it is deleted explicitly, or when it expires because its maximum lifetime of the *visa* has been exceeded. A *visa* is not marked "out of service" when it expires because of inactivity.

A *visa* expires when the current time (*T_now*) is after the expiration time (*T_exp*). *T_exp* is set as follows:

- If the *visa* is created at the time *T_creat*e, the initial value of *T_exp* is (*T_create* + *DT_init*). It is assumed that *DT_init* is less than *DT_life*.
- While a *visa* is not expired (*T_now* < *T_exp*), each time a packet arrives *T_exp* is set to the minimum of (*T_now* + *DT_act*) and (*T_create* + *DT_life*).
- When a *visa* is expired because of inactivity, i.e., is still "in service", a packet from the outside network will be rejected and will have no effect. However, if a packet from the inside network arrives, *T_exp* is set to the minimum of (*T_now* + *DT_act*) and (*T_create* + *DT_life*), as before. See the examples in Figure 3 and Figure 4.
- When a *visa* is expired and "out of service", a packet from the outside will be rejected and will have no effect. However, if a packet from the inside arrives, *T_exp* is set to *T_now*. This effectively delays the expunge time each time a keepalive arrives. See the example in Figure 5.

Once a *visa* has expired or been deleted, it will not be expunged from the system until a period of *DT_wait* has elapsed (when *T_now* = *T_exp* + *DT_wait*). During this waiting period, for *visas* that are marked "out of service", packets originating from the inside network will not cause automatic *visa* renewal to take place, but will instead delay the expunge time. Once the "out of service" *visa* has been expunged, the *visa* will be automatically re-created when a packet from the inside is received. Alternatively, an authorized user can put the *visa* “back in service” by logging on to the Tunnel system and explicitly re-creating it.

DT_wait, the period of time that must elapse before an expired *visa* is expunged from the system, is a system-wide constant. To change it, the Tunnel system must be re-built. *DT_wait* should be longer than the interval associated with any keepalive mechanism that is in use on the inside network.

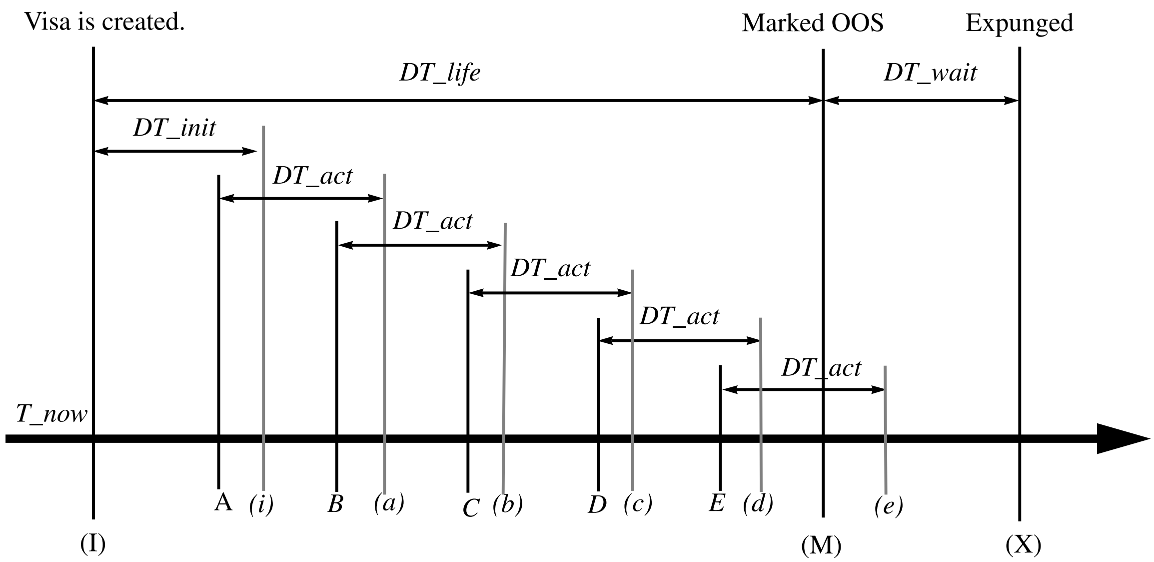
Stated another way, *visa*s expire under the following conditions:

- Implicitly, by not being used for *DT_act* time (or *DT_init*, initially)
- Implicitly, at *DT_life* after the *visa*’s creation.
- Explicitly, when a utility program (*delete_visa* or *tunnel_visa*) is used to delete a specific *visa* or a set of *visas*.

Visas are expunged from the system in any of the following conditions:

- *DT_wait* after expiration.
- *Visa* has expired or has been explicitly deleted and it contains a wildcard-host and/or a wildcard-protocol (*ALL*) specification.
- *Visa* has expired or has been explicitly deleted and it contains a default protocol specification (*ANY*), and thus has never been used.

Figure 3 illustrates the expiration of a *visa* that has exceeded its maximum lifetime, *DT_life*. In this example the *visa* was created at (I), or *T_create*, and the *visa* was no longer valid at (*M*) = (*T_create* + *DT_life*).



The initial expiration time is (i) = (*T_create* + *DT_init*).

Before (i), the *visa* is used at time (A), and extended to (a);

before (a), the *visa* is used at the time (B), and extended to (b);

before (b), the *visa* is used at the time (C), and extended to (c);

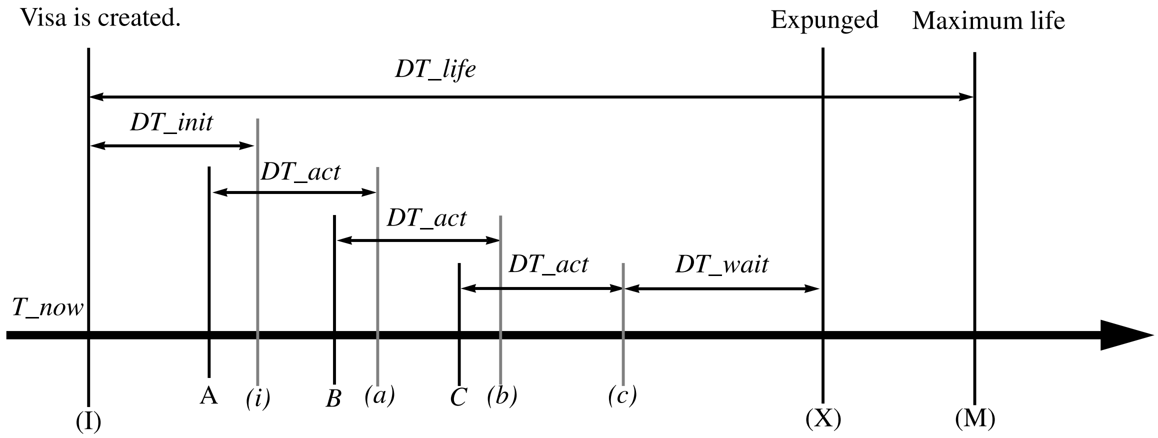
before (c), the *visa* is used at the time (D), and extended to (d);

before (d), the *visa* is used at the time (E), and extended to (e).

Since (e) > (M), i.e., beyond the maximum allowed life, the *visa* expires at (M), and marked as being out-of-service, until it is expunged at (X) = (M) + *DT_wait*.

FIGURE 3. *Visa* expires due to exceeding its max-life

Figure 4 illustrates how a *visa* expires due to inactivity. In this example a *visa* is created at (I) or *T_create*, and is no longer valid at (*M*) = (*T_create* + *DT_life*).



The initial expiration time is (i) = (*T_create* + *DT_init*).

Before (i), the *visa* is used at time (A), and extended to (a);

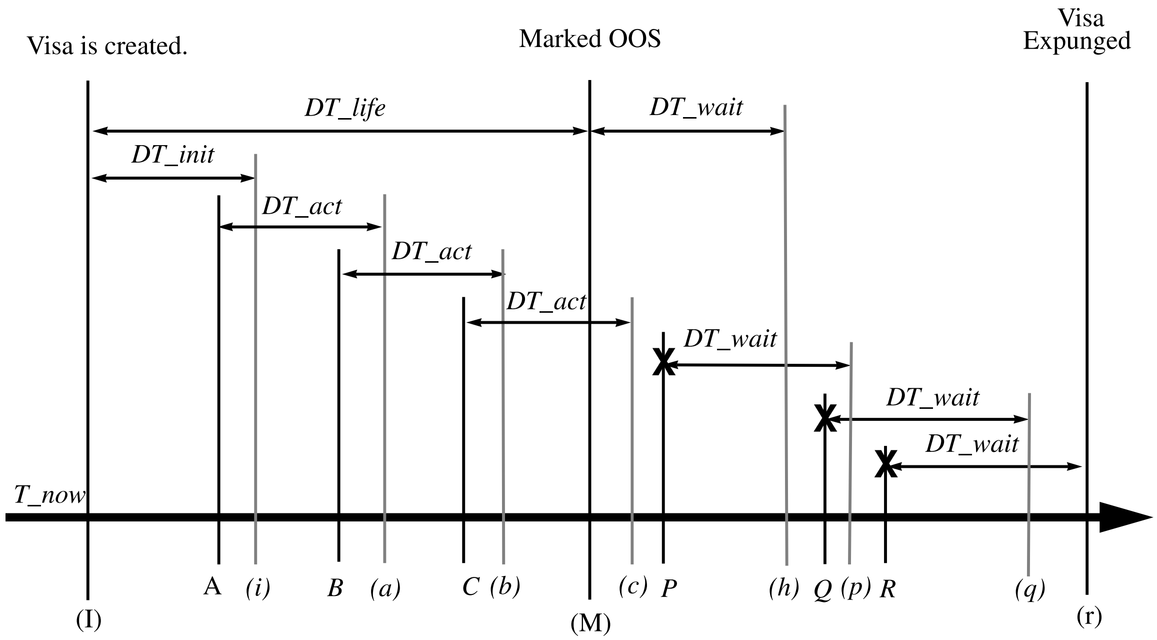
before (a), the *visa* is used at the time (B), and extended to (b);

before (b), the *visa* is used at the time (C), and extended to (c).

However,the *visa* is not used by the time (c) and therefore it expires then, without being marked as being out-of-service. It is expunged at the time (X) = (c) + *DT_wait*.

FIGURE 4. *Visa* expires due to lack of activity

Figure 5 illustrates how keepalives from the inside are handled. In this example, a *visa* is created at (I), or *T_create*, and is no longer valid at (M) = (*T_create* + *DT_life*). After the *visa* expires, it is marked “out of service”, because it was in use beyond time (M). After that, as keepalives are received from the inside, the expiration time, or *T_exp* is updated, delaying the time when the *visa* can be expunged.



The initial expiration time is (i) = (*T_create* + *DT_init*).

Before (i), the *visa* is used at time (A), and extended to (a);

before (a), the *visa* is used at the time (B), and extended to (b);

before (b), the *visa* is used at the time (C), and extended to (c).

Since (c) > (M), i.e., that time is beyond the maximum allowed life, the *visa* expires at (M), and is marked as being out-of-service.

At this point the *visa* is scheduled to be expunged at (h) = (M) + *DT_wait*, however after a packet from the inside is received at time P, the packet is discarded and the time for the *visa* to be expunged is extended to (p);

after a packet from the inside is received at time Q, the packet is discarded, and the time for the *visa* to be expunged is extended to (q);

after a packet from the inside is received at time R, the packet is discarded, and the time for the *visa* to be expunged is extended to (r).

Since no more packets were received between (q) and (r), the *visa* is expunged at time (r).

The packets P, Q, and R were discarded by the Tunnel.

FIGURE 5. *Visa* expires due to exceeding its max-life, fighting inside-keepalives.

3.2 Defaults

The following are the defaults for the parameters used to configure the Tunnel system:

Parameter	Default Value	Minimum	Maximum
DT_init	10 min.	30 sec.	1 hr.
DT_act	10 min.	1 sec.	10 min.
DT_life	4 hr.	30 sec.	4 hr.
DT_wait	1 hr.	N/A	N/A
Log Interval	15 min.	N/A	N/A
Generate ICMP Error Msgs	TRUE	(FALSE)	(TRUE)

TABLE 1. Tunnel Defaults

4.0 Additional Features

4.1 ICMP Messages

The system administrator can set the Tunnel to not generate ICMP messages in order to reduce the Tunnel's visibility. Alternatively, the Tunnel can be set to generate "ICMP Host Unreachable" messages when an external host attempts to communicate without a *visa*, with an internal host.

All such occurrences are recorded by the Tunnel, and the total number of packets rejected for each host pair is reported periodically.

4.2 Source Routing

Source routing of IP packets could potentially be used by someone on the outside to open *visas* from the inside. This might occur in two possible ways, depending on which addresses in the packet are deemed to be the source and the destination addresses to be matched with the addresses in a *visa*.

(1) If the Tunnel compares the IP-source-address and the IP-destination-address against the addresses in the *visa*, there is the risk that an unauthorized person might insert the addresses used in a legitimate *visa* into the middle of a communication between two hosts for which no *visa* exists. For example if a *visa* exists for Host-I and Host-X, the source route "Host-A -> Host-X -> Host-I" would allow communication between outside Host-A and inside Host-I, using that *visa*.

(2) If the Tunnel compares the first and last hosts listed in the source route against the addresses in the *visa*, an unauthorized person might fake the first address at the beginning of the source route. For example, if a *visa* exists for Host-I and Host-X, the source route “Host-X -> Host-A -> Host-I” would allow communication between outside Host-A and inside Host-I, using that *visa*.

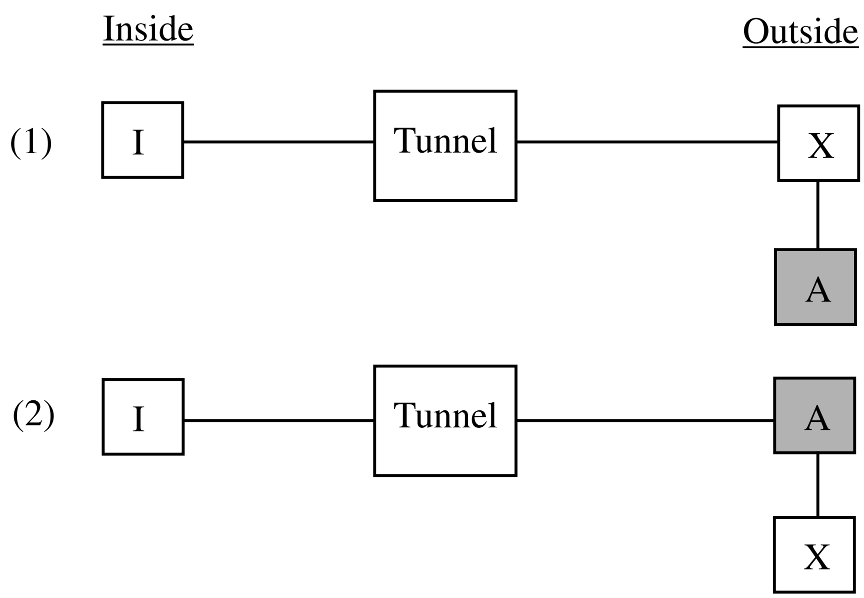


FIGURE 6. Problems potentially caused by source routing.

Therefore, because it is impossible for the Tunnel to determine which two hosts in a source route are actually communicating, the Tunnel drops all source-routed packets.

4.3 Port Filtering

A *visa* specifies a host pair and the transport level protocol for which the *visa* may be used (such as TCP, UDP, and ICMP). It is possible to statically configure the Tunnel to disallow the use of certain ports of certain protocols, for example *Telnet* (TCP #23).

We could have used a “good-ports” table or a “bad-ports” table. If only a few specific ports are to be allowed, it is more efficient to list the “good-ports”. However, since the Tunnel was designed with the assumption that very few types of communication were to be disallowed, we chose to list “bad-ports” explicitly. Listing “good-ports” is particularly difficult when applications that use multiple connections, such as FTP, are to be supported, since the ports that are to be used for the various connections may not be known in advance.

The *port-table* in the tunnel program, “tunnel.c”, is compiled into the system. The tunnel program is currently able to filter ports defined for TCP. This limited port filtering capability may be expanded so that the list of ports that are prohibited can be dynamically configured by the System Administrator in a future version of the Tunnel.

4.4 Tunnel Logging Facility

The Tunnel uses the standard Sun "*syslogd*" mechanism to log messages. See the section "TUNNEL CONTROL FILE" in Appendix B for details of how this is set up. The Tunnel logs the following events:

- Program errors.
- Table overflow (*visa* and *access-table* entry not created).
- *access-table* entry creation/deletion.
- Each time that the Tunnel defaults are set.
- *Visa* creation/deletion/expiration/expunge.
- *Visa* creation attempts rejected because no entry in the *access-table* exists.
- Count of packets rejected since previous report per [reason, src-addr, dst-addr, protocol].
- Count of packets forwarded and/or "out of service" per *visa*.

To change the set of events that is logged, the Tunnel must be rebuilt. In a future version of the Tunnel, logging may be dynamically configured by the System Administrator.

4.5 Permanent Visas

Warning: A permanent visa is a hole in the fence and is therefore a serious access control risk. We highly recommend that permanent visas not be allowed when access control is important.

To facilitate long term, machine-initiated interactions, a permanent visa may be created by the system administrator, logged in as "root". This approach may be used to set up a permanent communication path between a specific host on the inside and a specific host on the outside, for example two servers. Permanent visas should be used with extreme caution, since they create an opening that could potentially be taken advantage of by an attacker.

The following is the policy with regard to permanent *visas* that is implemented in the Tunnel:

- Only root may create a permanent *visa*.
- A permanent *visa* contains no expiration date and can be marked "out of service" only as the result of a system call.
- Only root may delete or modify a permanent *visa*, and must do an explicit deletion, specifying both hosts.

When a fully specified permanent *visa* is deleted, it is marked "NOT permanent" and "out of service", and is given an expiration time. In effect, it becomes a normal *visa* waiting for the *DT_wait* period to be expunged. This is to avoid keepalives in the same manner that they are avoided when any other *visa* is deleted.

4.6 Wildcard Visas

Warning: A wildcard visa is a hole in the fence and is therefore a serious access control risk. We highly recommend that wildcard visas not be allowed when access control is important.

To facilitate network monitoring applications, it is possible to set up a permanent *visa* in which either host (or both hosts) is a “wildcard” value (0.0.0.0). Wildcard *visas* should be used with extreme caution, since they could potentially create an opening that could be taken advantage of by an attacker.

The *visa* [*, *, *] totally bypasses the Tunnel for all packets.

A *visa* that contains a wildcard-host specification or a wildcard-port specification is expunged as soon as it expires or is explicitly deleted. This is necessary because it might otherwise block the creation of *visas* that should legitimately be “implicitly” created, if the system attempted to do “keepalive avoidance” processing on a *visa* that contained wildcard parameters.

5.0 Risk Analysis

The addition of the Tunnel to sites currently behind traditional firewalls will make the interactions with the outside smoother, with little added risk.

Replacing a conventional router with a Tunnel (e.g., at ISI) would significantly reduce the risks of the interactions with the outside. However, this added protection may cause inconvenience.

Some of the risks of using the Tunnel are identical to the risks of using any shared system. It is possible to break into a firewall and attack internal systems by:

(a) Stealing the password to the firewall. Since passwords are typically sent in the clear, often over an Ethernet, this is relatively easy.

(b) The use of other attack techniques such as address spoofing, nameserver spoofing, or the exploitation of bugs in software running on the firewall.

A risk that is specific to the Tunnel is the use of a valid visa that was created by an authorized user, but used by an unauthorized user. This can take place under the following conditions:

(c) The unauthorized user uses an external computer at the same time that an authorized user (located either inside or outside) is using it. On multi-user systems (like Sun workstations and servers), at the same time that a legitimate session takes place, another job can be started in parallel.

(d) The unauthorized user uses an external system after the authorized user (located either inside or outside) has concluded his use. For example, an inside host may send "keepalives" on a connection that has not been closed properly, thus improperly causing *visas* to be regenerated in the Tunnel.

The most serious threat is (a), the communication of a password in the clear, especially over LANs, like those in most of the sites visited by authorized users, especially those users who are anxious to read their e-mail. This risk applies equally to communication with the current firewalls and the Tunnel. Password theft may be accomplished easily using any of the many popular "protocol analyzers". This may be done practically anywhere on the Internet, and especially in LAN-based sites. Some of the methods to reduce this risk are: (1) One-time passwords (requiring the traveling authorized users to carry a list of one-time passwords), (2) using a challenge/response or time-based scheme, requiring the traveling authorized user to carry an hand-held credit card size "calculator", or (3) providing end-to-end authentication (e.g., by Kerberos). There are several variations on the above such as the use of a personal notebook computer or a password controlled software to generate the response for the challenge in (2).

We recommend the implementation of some password protection means, for both the "firewall" (independently of the Tunnel) and the Tunnel.

The second risk, (b), of breaking into the Tunnel is practically the same as breaking into the current firewalls. Hence, the risks of (a) and (b) are the same as those that already exist. The Tunnel does not provide any added protection against these threats.

Another attack possibility, (c), is for an external attacker to log into the same system as the one being by used an authorized user, and to use an existing *visa* for the attack. However, the attacker is limited to the use of the same protocol used by the authorized user. If the protocol in use is TCP, this limitation does not provide much protection. Configuring the Tunnel to disallow packets destined for certain ports can reduce this risk. To protect against this kind of attack, authorized users should be advised to minimize the use of external multi-user systems, and/or to watch (via "who" on UNIX, for example) other users of the same external system. In addition, users should explicitly delete *visas* that are no longer in use, by using the *delete_visa* utility.

To counter the risk (d), an attack through a *visa* left by improper termination of a session, users should always be sure to exit an FTP session, for example, properly. Users should run *delete_visa* on the Tunnel after access to an internal system is completed. If the workstation being used by the authorized user crashes, the user may use another workstation to log into the Tunnel and to delete all the *visas* that exist for the inside and/or outside workstation(s) he was using.

A short *DT_act* protects against (d), but is painful for authorized users on the outside. A waiting period, *DT_wait*, is used to counter the problem of *visa* renewal and implicit re-creation caused by keepalives originating from hosts on the inside network (d). However, since the Tunnel cannot differentiate between keepalives and legitimate traffic originating on the inside, *DT_wait* also prevents deleted and expired *visas* from being automatically revived in the manner that they are automatically created. A short *DT_life* provides protection against (d), however it may be inconvenient for users on the inside, as well as on the outside, since *visas* must be periodically renewed.

Packets containing IP-level source routing do not constitute a risk, as all source-routed packets are dropped by the Tunnel.

The above risks should be understood by all users, as well as the proper procedures to counter them.

6.0 Software Architecture

The Tunnel software is implemented as a part of the IP protocol layer in the Sun kernel. The defaults in the Sun OS are set up in such a way that the Tunnel will serve as a router, performing IP-level forwarding between networks or subnetworks whenever there is more than one interface attached to it. At the point in the IP processing where it is determined that a particular IP packet is not destined for the local host and that the packet can be sent out on one of the host's interfaces, the Tunnel software is invoked. The Tunnel processing determines whether the packet should be dropped or forwarded, depending on the results of a comparison with the various access control structures and parameters that are part of the Tunnel software.

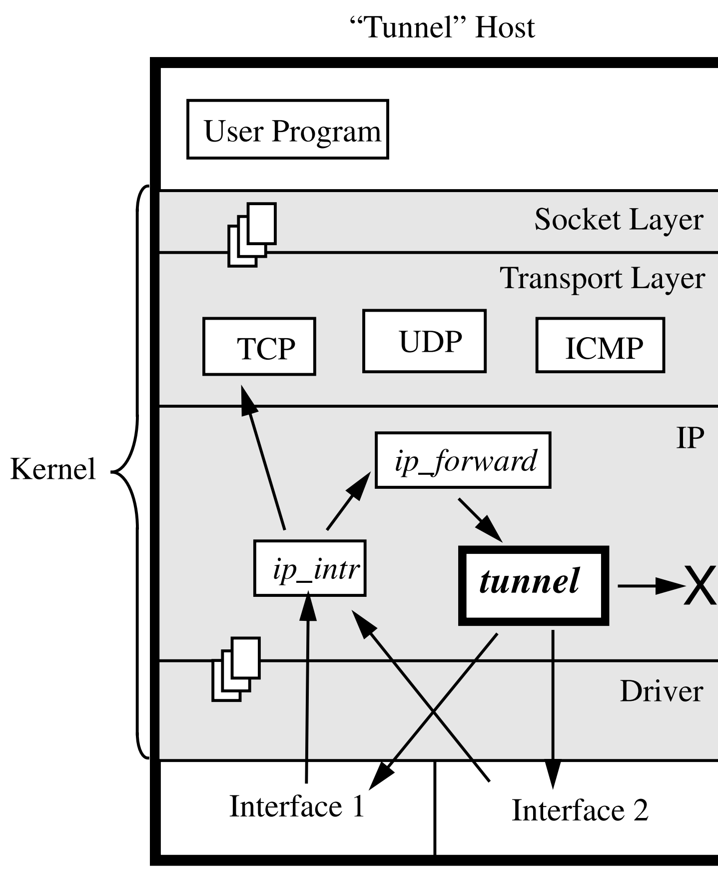


FIGURE 7. Software Architecture

The following diagram describes the logic and the processing performed when a packet not addressed to the Tunnel itself is received.

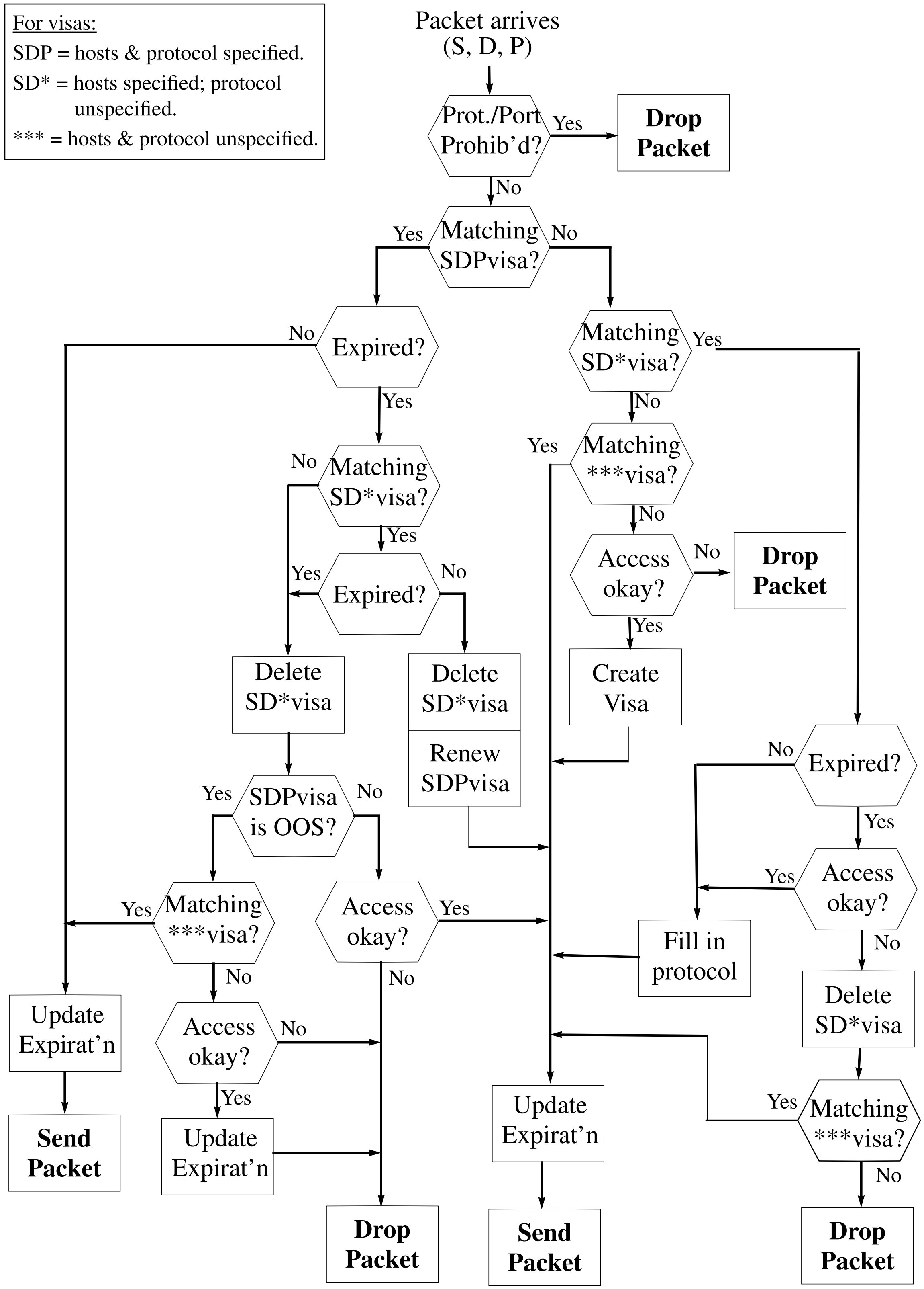


FIGURE 8. The Tunnel determines whether a packet should be forwarded.

7.0 Experience

We tested the Tunnel System at ISI. The Tunnel was logically positioned in place of the Cisco router that normally connects the Ethernet subnet used by our division with the rest of ISI and the world.

The Tunnel was designed to open up a closed environment, however in doing the testing at ISI, we were closing a very open environment. Table-2 summarizes the differences between the ISI environment and the environment that the Tunnel was designed for.

Functionality	ISI	Closed Environment
Mail servers	Inside & Outside	On the Firewall
Network Monitoring	100s of hosts, often	None
Host-Server	Inside host, outside server	None
Server-Host	Outside host, inside server	None
Server-Server	NFS	None
Transit	Encapsulated IP Multicast	None

TABLE 2. The ISI environment versus a closed environment.

Figure 9 illustrates the network configuration that existed in the ISI environment.

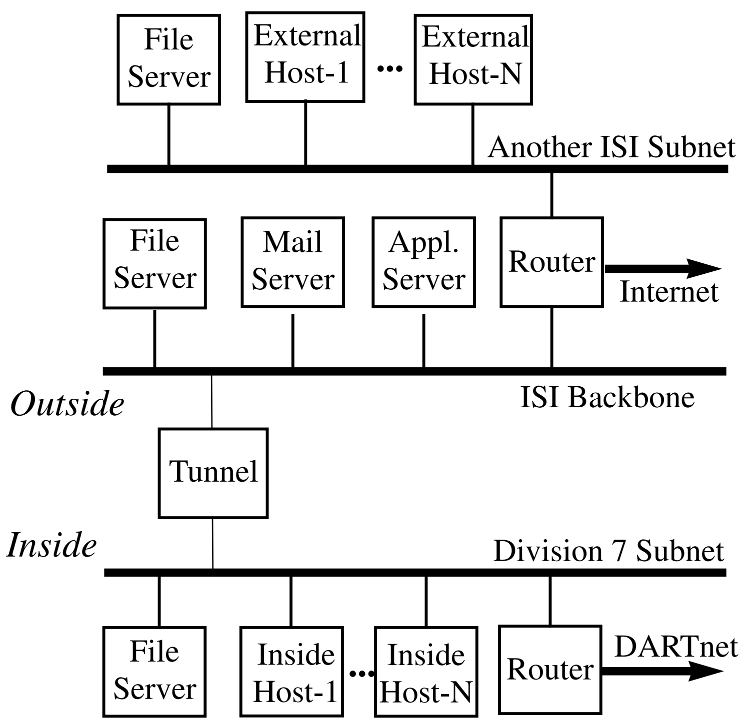


FIGURE 9. Configuration Used to Test the Tunnel at ISI

In the ISI testing, a number of adjustments had to be made to allow some network functions to continue (a) without *visas* being explicitly created, and (b) over a longer time period than a normal *visa* is permitted to exist, *DT_life*. The addition of permanent *visas* and wildcard *visas* was, in effect, the same as deliberately making "holes" in the Tunnel, to allow certain classes of packets through the Tunnel. Some adjustments were made as follows:

- Permanent *visas* were added to support applications on inside workstations that make use of outside servers. A broader problem is that users are often not aware of interactions that take place between their workstations and outside servers. Some examples of this at ISI are network file servers, the "*phoneserver*", and various printers. When files are printed, the user may type "lpr myfile.txt" and thus start up an interaction between his local workstation and an outside host that he's never heard of, much less thought to create a *visa* for.
- Permanent visas were added to support outside workstations that make use of a fileserver that was located on the inside.
- Wildcard *visas* were required to support network monitoring applications.
- Wildcard *visas* were added to support encapsulated-IP multicast tunneling between the Internet and DARTnet.

In addition, several static routes had to be installed on the Tunnel. At the start of the testing we experienced a number of routing problems, partially because of difficulty in adapting the manual configuration, and partially because of errors in the configuration, such as hosts that were set up with an incorrect subnet mask. There were a number of outside hosts that erroneously directed packets addressed to other outside hosts, to the Cisco router. The Cisco router had previously forwarded the packets. The Tunnel, which was deliberately set up to forward only between the inside subnet and outside, did not forward them. Routing tables on the offending hosts had to be individually corrected. A "domino effect" resulted in some routing instability that lasted for the next few hours.

None of the problems at ISI that required these fixes exist at sites that are now behind firewalls!

It is our assessment that if an environment is supposed to be closed for protection against attacks, the use of permanent *visas* and *visas* that contain wildcard-host specifications should be minimized, or ideally, eliminated. Every permanent *visa* is a hole in the fence around the protected "inside".

Similarly mailers and domain name servers should not be inside the fence because they create traffic which creates *visas* that may be exploited by attackers.

8.0 Comparison

The Tunnel provides an alternative to a traditional firewall that contains a fixed access list. When a fixed access list is used, all interactions between specific hosts, or between classes of hosts defined by an address and a mask, must be anticipated. On the Tunnel, an interaction initiated by a host on the inside sets up access for packets to be sent in the opposite direction. Therefore, it is not necessary that a system administrator be specifically concerned with the identification of participants in network communication.

Since the Tunnel is implemented on a Sun workstation, it is potentially vulnerable to any break in techniques that can be applied to a Sun. Users log in directly on the Tunnel, in order to set up access from the outside. Firewalls that are based on a fixed access list avoid some of the risks associated with workstation-based firewalls, such as the Tunnel. In addition, firewalls that are based on a more secure system architecture may prove to be more resistant to various attacks.

The Tunnel allows greater flexibility than an application level gateway, which has to be reprogrammed each time an application is added. Whether the support of a new application is seen as an advantage or a disadvantage, depends upon one's point of view. Since new applications can contain bugs and can introduce security vulnerabilities, an application level gateway can also be viewed as protection from new, untested technology. However, in an environment where convenience and the use of the latest software tools is considered to be of high importance, use of the Tunnel provides access to all IP-based communications.

9.0 Conclusion

The Tunnel was implemented at ISI as a small scale project. We believe that it has good potential for providing increased convenience with very little increased risk, in a variety of environments. The Tunnel can be configured to serve as a router for an open environment, such as a university, or for an organization concerned with reducing its vulnerability to attacks.

Proposed extensions include:

- One-time passwords,
- Additional dynamic configuration capability,
- On-line alerts on certain conditions, and
- Negative access (black lists).

10.0 References

R. Ganesan, "BAfirewall: A Modern Firewall Design", to appear in the *Proceedings of the Internet Society Symposium on Network and Distributed System Security*, February 1994.

J. Kohl, (DEC) and C. Neuman (ISI), "The Kerberos Network Authentication Service (V5)", *RFC-1510*, September 1993.

J. Mogul, "Simple and Flexible Datagram Access Controls for UNIX-based Gateways," *USENIX Conference Proceedings*, Summer 1989.

M. Ranum, "Thinking About Firewalls," *Proceedings of Second International Conference on Systems and Network Security and Management (SANS-II)*, April 1993.

C. Stoll, *The Cuckoo's Egg*, Doubleday, 1989.

Appendix A -- What Users Should Know

All users should understand the risks associated with the use of the Tunnel and the procedures to counter them.

For usage from the inside:

- In most cases, no additional action is required for communication with external hosts, for IP-based communication initiated from the inside.
- If an inside user initiates a communication with an external host that would result in the external hosts replying a long time later (e.g., "factor a 7,000 bit number and report"), the internal user must explicitly create a *visa* that will last as long as necessary. This can be done by running the *create_visa* utility on the Tunnel and setting the activity time limit (*DT_act*) to a large value.
- If a session with an outside host needs to last longer than *DT_life* (4 hours), the user must use the *create_visa* utility to extend the lifetime of the *visa*, preferably before *DT_life* has elapsed. Use of *create_visa* must be repeated at an interval of *DT_life* in order to prevent the *visa* from expiring.
- At the end of a session, the user should explicitly use the *delete_visa* utility.
- Once a *visa* has been explicitly deleted, or has expired because its usage exceeded *DT_life*, *create_visa* must be used before communication with the outside host can be reinitiated.
- Inside users who use external services should be aware of their hostnames and/or addresses to allow the creation/renewal of *visas*.

For usage from the outside:

- The user must first *telnet* to the Tunnel, log into it, and use the *create_visa* utility. Then the user should start the communication within *DT_init* (defaulted to 10 minutes). If needed, the user may enter other values for *DT_init* and *DT_act*.
- If a *visa* needs to last longer than *DT_life* (4 hours), the user must use the *create_visa* utility to renew the *visa*, preferably before *DT_life* has elapsed.
- At the end of a session, the user should use the *delete_visa* utility to explicitly delete any *visas* that were in use.
- Once a *visa* has been explicitly deleted, or has expired because its usage exceeded *DT_life*, *create_visa* must be used before communication with the inside host can be reinitiated.

Appendix B -- System Administration

WHAT THE SYSTEM ADMINISTRATOR SHOULD KNOW (AND DO)

- Control access to the Tunnel (defining authorized networks in the *access-table*)
- Review existing *access-table* entries
- Enable/Disable ICMP messages
- Create/modify the file that lists commands/programs invoked during booting, including Tunnel control ("/etc/rc.tunnel_hostname")
- Create *visas*
- Delete *visas*
- Review the existing *visas*
- Review the system log

THE TUNNEL CONTROL FILE

Note that the utility programs *tunnel_access, tunnel_visa,* and *tunnel_defaults* must be in the "/etc" directory.

The Tunnel control file, on the Tunnel machine, is "/etc/rc.tunnel_hostname", where "tunnel_hostname" is the host name of the Tunnel machine.

To configure an Ethernet interface, the following line is added to to the control file, "/etc/rc.tunnel_hostname":

ifconfig le1 162.111.0.1 netmask 255.255.255.0 broadcast 162.111.0.255 -trailers up

("le1" is "el-ee-one", the name of the interface. "162.111.0.1" and "162.111.0.255" are examples of an interface address and a network mask, respectively.)

Access-Table ("*tunnel_access*"): To add access for the "inside network" "162.111.0.0", add the following line to "/etc/rc.tunnel_hostname":

/etc/tunnel_access a 162.111.0.0 255.255.255.0 ANY

Permanent *visas* ("*tunnel_visa*"): To add a permanent *visa* for TCP access between two hosts add the following line to "/etc/rc.tunnel_hostname":

/etc/tunnel_visa p 162.111.0.2 128.9.0.32 TCP

Defaults *DT_init* and *DT_act* and limits on *DT_init* and *DT_act* ("*tunnel_defaults*"): To set the defaults for the Tunnel, add the following line to "/etc/rc.tunnel_hostname". These are the defaults that the Tunnel is already initialized with, however other values may be specified:

/etc/tunnel_defaults i 600 I 600 a 600 A 3600 l 14400 u TRUE

The *syslog* file (path, and recording parameters): On the Tunnel machine, add the following line to the end of the file "/etc/syslog.conf":

local7.info@log_machine

where "log_machine" is the host name of the machine on which the log messages will be collected.

On the log machine, add the following line to the end of the file "/etc/syslog.conf":

local7.info /var/log/syslog.tunnel

Log messages will be written into the file "/var/log/syslog.tunnel" on "log_machine".

The parameter that controls which events are logged is a constant that is compiled into the Tunnel kernel. "LOG_DEFAULT" is currently defined such that everything will be logged. Individual flags are defined as follows:

- LOG_ERROR: Program errors
- LOG_NO_ACCESS : *Visa* creation attempts rejected because no *access-table* entry exists.
- LOG_NO_SPACE: No space (*visa* and *access-table* entry creation failed).
- LOG_VISA: *Visa* creation/deletion/expiration/expunge.
- LOG_ACCESS: *access-table* entry creation/deletion.
- LOG_DEFAULTS : Each time that the Tunnel defaults are set.
- LOG_BAD_P_CNT : Count of packets rejected since previous report per [reason, src_addr, dst_addr, protocol].
- LOG_FWD_P_CNT: Count of packets forwarded and/or "out of service" per *visa*.

SAMPLE TUNNEL CONTROL FILE -- /etc/rc.tunnel.isi.edu

route add drax-net-yp 128.9.32.2 1
route add quark-net-yp 128.9.32.3 1
route add vlsi-net-yp 128.9.32.4 1
route add darkstar-net-yp 128.9.32.3 1
route add rocky-net-yp 128.9.32.2 1
route add isi-net32-yp 128.9.32.130 0 route add trader-net-yp 128.9.32.130 0
route add net 128.9.112.0 128.9.160.132 1
route add net 140.173.0.0 128.9.160.153 1
route add default 128.9.32.1 1

Start proxy ARP daemon
/local/etc/in.parpd &

allow for inside network -> anywhere
/etc/tunnel_access a 128.9.160.0 255.255.255.0 ANY

allow for anywhere -> “ant” for DARTnet
/etc/tunnel_access a ANY 128.9.160.49 255.255.255.255

for LosNettos
/etc/tunnel_visa p ALL ant ALL
/etc/tunnel_visa p ALL los ALL /etc/tunnel_visa p ALL fji ALL

establish permanent visas for the ISI servers
/etc/tunnel_visa p zephyr venera UDP
/etc/tunnel_visa p zephyr venera TCP
/etc/tunnel_visa p zephyr venera ICMP
/etc/tunnel_visa p zephyr darkstar UDP
/etc/tunnel_visa p zephyr darkstar TCP
/etc/tunnel_visa p zephyr darkstar ICMP
/etc/tunnel_visa p zephyr quark UDP
/etc/tunnel_visa p zephyr quark TCP
/etc/tunnel_visa p zephyr quark ICMP
/etc/tunnel_visa p zephyr drax UDP
/etc/tunnel_visa p zephyr drax TCP
/etc/tunnel_visa p zephyr drax ICMP
/etc/tunnel_visa p zephyr lepton UDP
/etc/tunnel_visa p zephyr lepton TCP
/etc/tunnel_visa p zephyr lepton ICMP
/etc/tunnel_visa p zephyr rocky UDP
/etc/tunnel_visa p zephyr rocky TCP
/etc/tunnel_visa p zephyr rocky ICMP
/etc/tunnel_visa p zephyr alibi UDP
/etc/tunnel_visa p zephyr alibi TCP
/etc/tunnel_visa p zephyr alibi ICMP
/etc/tunnel_visa p zephyr trader UDP
/etc/tunnel_visa p zephyr trader TCP
/etc/tunnel_visa p zephyr trader ICMP

set defaults (These are the defaults built into the Tunnel.)
/etc/tunnel_default i 600 I 600 a 600 A 3600 l 14400 u TRUE

Appendix C -- Manual Pages

The following are manual pages for the various Tunnel utility programs.

NAME

create_visa - Create a *visa* on the Tunnel System.

DESCRIPTION

create_visa is used to create a *visa* for communication between the host from which the user has *telnet'd* (or *rlogin'd*) to the Tunnel, and another host specified by the user, on the other side of the Tunnel.

The following is an example of how to create a *visa* for outside host "enclyclopaedia.britanica.edu" and inside host "myhost.isi.edu" , while *telnet'd* to the Tunnel, from "enclyclopaedia.britanica.edu":

% create_visa<CR>

Complete host name: enclyclopaedia.b*ritanica.edu<CR>*

(If the name of the host from which you *telnet'd* or *rlogin'd* is longer than 15 characters, you will be asked to type in the characters required to complete it.)

Enter password: *secret<CR>*

(Enter the password for your account on the Tunnel machine.)

Host you wish to connect to: *myhost.usu.edu<CR>*

(Enter a host name or number, for example "venera.isi.edu" or "128.9.0.32".)

Protocol [ANY]: *<CR>*

(Enter a protocol such as "UDP", or if *create_visa* does not recognize the protocol name, the decimal number of the protocol, e.g., "77", or type <CR>. The protocols recognized by name are as follows: *ICMP, IGMP, GGP, EIP, ST, TCP, UCL, EGP, IGP, PUP, UDP, IDP, HELLO, ND*. Note that the default protocol, *ANY*, signifies that the protocol field in the *visa* will be filled in from the protocol field in the IP header of the first packet that is matched with this *visa*.)

Initial time limit [600 sec]: *<CR>*

(Enter the time required to exit *telnet / rlogin* and start your application, or type *<CR>*.)

Activity time limit [600 sec]: *<CR>*

(Enter the maximum time between transactions, or type *<CR>*.)

OPTIONS

NONE

SEE ALSO *tunnel_defaults, tunnel_access, tunnel_visa, delete_visa*

BUGS

Please report any bugs to Annette DeSchon <deschon@isi.edu>.

NAME

delete_visa - Delete a Tunnel System *visa*.

SYNOPSIS

delete_visa [Host1] [Host2 **|** *ALL*]

DESCRIPTION

delete_visa works as follows:

- "*delete_visa*" deletes all *visas* in which local host is involved.
- "*delete_visa* Host1" deletes all *visas* in which Host1 is involved, and deletes all *visas* in which local host is involved.
- "*delete_visa* Host1 *ALL*" deletes all *visas* in which Host1 is involved.
- "*delete_visa* Host1 Host2 " deletes all *visas* in which both Host1 and Host2 are involved.

This will allow the administrator to delete a specific *visa* or a class of *visas*. It will also allow a user to delete his *visas* from another host following a crash of a system that he was using.

One drawback is that it will be possible for anyone who has access on the Tunnel machine to delete other people's *visas*, however we believe that it is important for users to be able to remove *visas* in the event of a crash, rather than having them depend on the system administrator.

The following is an example of its use deleting any *visas* in existence for hosts "enclyclopaedia.britanica.edu" and "myhost.isi.edu", while *telnet'd* to the Tunnel, from "enclycolpaedia.britanica.edu":

% delete_visa myhost.isi.edu<CR>

Complete host name: enclyclopaedia.b*ritanica.edu<CR>*

(If the name of the host from which you *telnet'd* or *rlogin'd* is longer than 15 characters, you will be asked to type in the characters required to complete it.)

Enter password: *secret<CR>*

(Enter the password for your account on the Tunnel machine.)

OPTIONS

NONE

SEE ALSO *tunnel_defaults, tunnel_access, tunnel_visa, create_visa*

BUGS

Please report any bugs to Annette DeSchon <deschon@isi.edu>.

NAME

tunnel_access - Create/delete/list a Tunnel *access-table* entry.

DESCRIPTION

tunnel_access is used to create/delete/list an *access-table* entry in the Tunnel tables. The system administrator must be logged in as "root" to run this utility program.

An *access-table* entry consists of a [Source, Destination], e.g., an [Inside, Outside], pair of networks, which allows hosts on the source network to send packets to hosts on the destination network without the explicit creation of a *visa*. A network consist of a four-byte dotted decimal format address and a four-byte dotted decimal format mask. *ANY* is equivalent to "0.0.0.0 0.0.0.0".

OPTIONS

-l List all *access-table* entries:

tunnel_access -l

-a Add an entry:

tunnel_access -a src_net src_mask dst_net dst_mask

-d Delete an entry:

tunnel_access -d src_net src_mask dst_net dst_mask

SEE ALSO *tunnel_defaults, tunnel_visa, create_visa, delete_visa*

BUGS

Please report any bugs to Annette DeSchon <deschon@isi.edu>.

NAME

tunnel_defaults - Set system defaults for the Tunnel.

SYNOPSIS

tunnel_defaults [-*i* t] [-*I* t] [-*a* t] [-*A* t] [-*l* t] [-*u* *TRUE* | *FALSE*]

(All times “t” are in seconds.)

DESCRIPTION

tunnel_defaults may be used by the system administrator, logged in as “root”, to change the Tunnel system defaults. The Tunnel is configured with minimum and maximum values for each of these parameters, so if a value that is out of the acceptable range is entered, *tunnel_defaults* will use the closest possible legal value instead.

If no options are supplied on the command line, *tunnel_defaults* lists the current values.

OPTIONS

-*i* DT_init

Default time limit for first use. Initially 600, maximum 3600, minimum 30.

-*I* DT_init_max

Maximum time limit for first use. Initially 600, maximum 3600, minimum 30.

-*a* DT_act

Default time limit after last use. Initially 600, maximum 3600, minimum 1.

-*A* DT_act_max

Maximum time limit after last use. Initially 600, maximum 3600, minimum 1.

-*l* DT_life

Maximum lifetime of a *visa*. Initially 14400, maximum 14400, minimum 1.

-*u* Send_ICMP_Host_Unreach

Whether to send ICMP error messages. Possible values are *TRUE* and *FALSE*. Initially *TRUE*.

Note that *DT_wait* is a constant equal to 3600 seconds (1 hour).

SEE ALSO *tunnel_visa, tunnel_access, create_visa, delete_visa*

BUGS

Please report any bugs to Annette DeSchon <deschon@isi.edu>.

NAME

tunnel_visa - Add/delete/list Tunnel System *visa(s).*

DESCRIPTION

tunnel_visa is used to add, delete, or to list *visas* in the Tunnel System.

The protocols recognized by name are as follows: *ICMP, IGMP, GGP, EIP, ST, TCP, UCL, EGP, IGP, PUP, UDP, IDP, HELLO, ND.*

The default protocol, *ANY*, signifies that the protocol field in the *visa* will be filled in from the protocol field in the IP header of the first packet that is matched with this *visa*. A protocol number may also be supplied.

The wildcard-protocol value, *ALL*, will match any value in the protocol field in the IP header of a packet. The user must be logged in as "root" for *ALL* to be accepted as a legal port value.

OPTIONS

-a To add *visa*(s):

tunnel_visa -a host1 host2 [protocol [setup_time_limit [usage_time_limit]]]

-p To add permanent *visa(s):*

tunnel_visa -p host1 host2 protocol

For both a permanent *visa* and a *visa* in which the wildcard-host, *ALL*, appears, the user must be logged in as "root", and a specific protocol or *ALL* (not *ANY*) must be provided.

-d To delete *visa(s):*

tunnel_visa -d host1 [host2]

When "host2" is omitted, all *visas* that contain "host1" are deleted. For a *visa* containing the wildcard host *ALL* to be deleted, both hosts must be specified; when "host2" is omitted, "host1" cannot be *ALL.*

-l To list all *visas* in the Tunnel System:

tunnel_visa -l

SEE ALSO *tunnel_defaults, tunnel_access, create_visa, delete_visa*

BUGS

Please report any bugs to Annette DeSchon <deschon@isi.edu>.