Improving Unix Network Security Through Host Rings

Prepared For:	**General Release**
Author:	**Geoff Halprin**
Reference:	**WP-1995-01**
Version:	**V1.01**
Date Created:	**26 September 1995 16:17**
Date Modified:	**20 August 1997 11:59**

A SysAdmin Group Technical Report

Abstract

This paper presents a simple mechanism, "***host rings***," for implementing Multics style execution domains and protection rings to improve access control of Unix hosts within a LAN environment.

The SysAdmin Group Pty Ltd (ACN 069 951 677)
P.O. Box 441, North Balwyn, Vic 3104, Australia
Email: Geoff.Halprin@sysadmin.com.au

Phone: (03) 9686-3233
Fax: (03) 9686-3399

1.0 Preface

1.1 Scope

This document presents the background, theory and implementation details of *host rings*, a technique developed to improve access control of Unix hosts within a LAN environment.

1.2 Intended Audience

It is assumed that the reader is familiar with Unix and TCP/IP and in particular, with Unix security mechanisms.

1.3 References

[1] "Building a Secure Computer System," Morrie Gasser, 1988.
Van Nostrand Reinhold Company. ISBN 0-442-23022-2.

[2] "Firewalls and Internet Security," William R. Cheswick and Steven M. Bellovin.
Addison-Wesley Publishing Company. ISBN 0-201-63357-4

[3] "The TCP/IP Daemon Wrapper Package (tcpd)," Wietse Venema.
ftp://ftp.win.tue.nl/pub/security/tcp_wrapper

[4] "The SSH Secure Shell Package (ssh)," Tatu Ylonen.
ftp://ftp.cs.hut.fi/pub/ssh

[5] "The RDIST Software Distribution Package (rdist)," Michael Cooper.
ftp://ftp.usc.edu/pub/rdist

[6] "The SATAN Network Security Analysis Package," Dan Farmer.
ftp://ftp.auscert.org.au/pub/coast/mirrors/csrc.ncsl.nist.gov/tools/satan

1.4 Change Control

Version	Release Date	Author	Comments
V01.00	05-Jul-96	Geoff Halprin	Release for Comments

2.0 Introduction

Much work has been done in recent years on Internet Firewall technology. This technology creates a "choke point" through which all traffic must pass between (typically) the Internet and the internal network. This technology is equally useful wherever a clear separation can be made between any two networks where one does not trust the other.

This is extremely valuable technology in the fight to control the security of a network. The basic assumption that this technology makes, however, is that everybody on the outside is bad and everybody on the inside is good.

In order to bring better control and security to an internal network, another security model is required. This model needs to recognise that physical and logical security attributes, and the separation of functions, as is commonplace with good client-server network design, can contribute to identifying where and what type of trust should exist between hosts.

Experience has shown that the current client-server computing environment closely matches the environment of traditional time-sharing hosts, and we should be able to draw on work performed for host security, and adapt this to network security.

The model for network security developed in this paper is called "Host Rings," or "Rings of Trust." It is similar in notion to the Multics concepts of *execution domains* and *protection rings*. See [1] for details on these.

Applying these techniques using the commonplace public domain tools of **tcpd**[3], **ssh**[4] and **rdist**[5] can provide us with a formal security model providing many distinct advantages over the intricate and inherently untrustworthy web-of-trust so common at present.

This paper describes the host rings model, how it can be implemented, the advantages of this model, and experiences with implementing it at various sites.

2.1 Scope, Assumptions and Limitations

The basic assumption made here is that a number of core hosts are providing services to a large number of workstations, desktops (PCs and Macs), Xterminals, etc. This seems a reasonable assumption in today's client-server environments. The system presented here has very low overhead, so can be deployed in very small networks with immediate benefit, although obviously larger benefits will be seen when deployed in a larger network.

This work has been developed for deployment within corporate, mission critical environments where policies, procedures and practices can, and are controlled in order to provide consistency of service. The principles can be applied equally in a number of environments although some, such as educational and research institutions, may require careful design.

The work presented here describes a set of rings of trust within a *single administrative domain*. In this context; *administrative domain* means a group of hosts which are under the administrative control of a single systems administration team. Each administrative domain can define its own set of rings, and can place external hosts (those outside the team's administrative control) in the appropriate ring to allow interaction between systems of different administrative domains, whilst still ensuring that a suitable level of trust restricts the services that will be provided across domains.

The basic mechanism for decisions regarding trust is the hostname or IP number. Discussions on the security of this system are dealt with later in this paper.

3.0 The Present Situation

The present system used at large for managing the security of a distributed network is the ad hoc implementation of an informal web-of-trust. As a service on one host requires a service from another host, a suitable "hole" is opened up to allow that request to be fulfilled. This is the equivalent of the security stance of "that which is not expressly prohibited is permitted."

For example, "I need to get backups from machine **A** onto the tape drive on machine **B**, so I'll allow **A** to **rsh** into **B**." Another example might be, "database backend; please tell me which network port to connect to you via." This would often also be implemented as an **rsh** call.

The major problem with this approach is that, almost invariably, it is not implemented with a mindset of "least privilege," and this leads directly to problems of transitive, hidden trust between unrelated hosts. The complexity of such trust makes this a non-simple management exercise.

The inadequacy of the web of trust model has been graphically illustrated by packages such as **SATAN**[6] which report on this web of trust as a maze of hypertext links in the output report.

3.1 A Formal Security Model

Clearly, there are many unknowns in such an informal methodology as the web-of-trust. Moving to a formal security model gives us a number of quantifiable advantages over the web-of-trust;

- **Benchmark**.

  Most importantly, we gain a benchmark against which we can analyse and rate the various services we provide. We can clearly identify breaches of security in the allocation of services.

  For example, a development database backend clearly has lower availability expectations that a production mission-critical database backend, so having these share the same host would violate such a benchmark, just as putting NFS user accounts on a firewall machine would violate the rules of good isolation.

- **Proactive Security Management**

  Just as Firewalls take on the all important explicit stance of "that which is not expressly permitted is prohibited," and force us to analyse the security implications of a service before we provide it, so too use of a formal security model forces us to examine the security implications and requirements of each new service before we provide that service, and hence, places us in a position to implement a least-privilege solution to the many potential breaches of security policy that a service may require.

- **Quality Assurance**

  By using a formal model which defines policy, technology and implementation, we are in a far better position with respect to predictability as to how the system will behave, and repeatability and consistency as to the configuration of the system. These attributes directly contribute to the quality of the computing environment.

3.2 A Word About Security

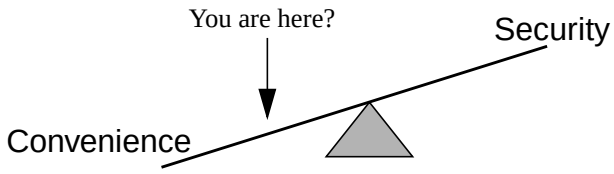

Figure 1 - The Security Trade-Off

Security is almost always a direct trade-off against convenience. Making something more secure usually involves making it more controlled, which implies more steps in the process. This equates to reduced convenience. (If it is still as convenient for the user, you can bet it isn't as convenient for the administrator!)

The migration to a formal security model is about converting implicit decisions into explicit decisions. How loose or tight a security policy is depends upon the organisation. This model does not dictate such things. It merely proposes a formal model within which to make and manage such decisions.

3.3 Types of Security

Security is primarily thought of in terms of access to information or services. This can be characterised as *Confidentiality*. In this paper we examine services under not only this security light, but also from the perspectives of *integrity* and *availability*.

4.0 Classification and Consolidation of Services

Each service that we provide to users can and should be assessed as to its importance to the user base. If we rate services as *Low*, *Medium* or *High* in terms of their *Confidentiality*, *Integrity* and *Availability* requirements, we can begin to compare the overall security requirements of one service against another.

No longer are we looking at security in terms of hosts or networks, but in terms of services provided to a user base. Specific security requirements might dictate specific measures. For example, policy might dictate that all services rated High Availability must reside on fully mirrored hosts. Services rated High Privacy might require secured communications channels to the service, such as encrypted terminal sessions. High Integrity might dictate full Tripwire signature databases are to be maintained.

A basic notion behind Host Rings is that we can classify and separate out the duties performed (services provided) by each host on the network. Further, we can identify certain key attributes, such as physical security, service availability, etc. as determining factors when allocating services to hosts. The intention is to group all services with similar security requirements onto the same host or group of hosts, and isolate these services from other less secure services by placing them on alternate hosts.

If you examine the services that "the system" provides to users in a distributed computing environment, from the perspective of their relative security requirements, you will typically see that a natural hierarchy unfolds;

1. **Administrative Services**. This includes naming services (DNS, NIS), network configuration services, host jumpstart images and profiles, host configuration baselines (e.g. rdist source tree), etc.
2. **Production Application Services**. Database backends, Fileservers. Database applications.
3. **Development Services**. Machines used to provide a programming development environment, be it C, an RDBMS, or other language environments. These hosts typically have lower availability and privacy requirements than do the production servers.
4. **General Purpose Hosts**. X-Terminal servers, Print servers, etc. Generally these do not have any core information on them, which can not be readily regenerated from an administrative machine.
5. **Internet Related Hosts**.Mail spooling, News spooling, Web servers.
6. **The Desktop**. PCs, Macs, diskless workstations. These machines use the facilities of the hosts described above to provide services to the end-user.

Each organisation is different. Although the above basic breakdown works for most organisations, there are always variations and complications; remote sites, X-terminals, PCs, and a myriad of other things to consider when allocating hosts to rings. The varying level of physical security also plays a part in the allocation process.

5.0 Host Security Rings

The security model proposed in this paper is called ***Host Rings***, or ***Rings of Trust***. It can best be characterised as implementing a **uni-directional, layered** model of trust. As mentioned previously, this is similar in concept to the *execution domains* developed by the Multics project.

This model basically consists of dividing the hosts into rings, based upon the security rating of the services they provide to the network, and then using these rings as the basis for trust between hosts.

There are many basic questions of trust which we can ask in order to help determine the hierarchy of the rings;

- Is this host in a physically secured computer room?
- Does this host have normal user accounts?
- Is this host at a remote site, and hence less trustworthy than the ones in the central computer room?
- Does this host run large service software which sources its data from the Internet?
- Does this host provide mission-critical services? How many people in the company would be affected by down-time on this host?

Part of implementing this scheme often requires moving certain services between hosts in order to consolidate the services being offered to the network at large. This, in itself, is a good thing, and results in better tuned hosts with less administrative overhead, better documentation and fewer interactions. Indeed, Parnas' goals for structured programming of High Cohesion and Low Coupling apply equally to this task as they do to programming.

5.1 The Rules of the Rings

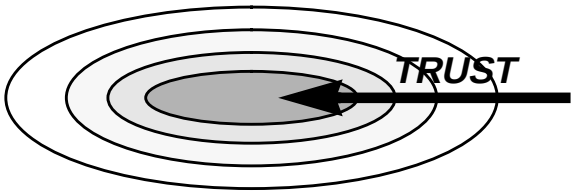

Figure 2 - The Unidirectional Trust of Host Rings

The following general rules apply to host rings;

1. **Each host trusts those hosts in a more inner ring than itself**.
2. **No host trusts any host in a more outer ring than itself**.
3. **Each host *may* trust those hosts in the same ring as itself**. This is best determined on a ring-by-ring basis. For example, it is very useful for all the major servers to trust each other. If one has been compromised, then the network has already fallen, so there is no real harm in such a choice. Conversely, not trusting hosts in the same ring is quite restrictive here, and would lead to increased administration overhead in crafting scripts to provide services between hosts. See the example section for more on this.
4. **Where a ring has been segmented, a host in one segment does not trust hosts in other segments**. More on this next.

5.2 Segmented Rings

One enhancement to this scheme is to recognise that an administrative domain may cover a number of geographic sites which do not need or wish to trust each other. They will all draw certain services from central machines, plus local services from local site servers. Hence several rings may be better segmented to reflect this more fine-grained trust. We will examine an example of such a site shortly.

5.3 Calling an Inner Ring

Often, a need is present for a workstation or desktop to either obtain information from a server (such as which IngresNet port to connect to), or to make a request for service from an inner ring. This must be done in a controlled, audited manner, and must be easily extensible and customisable on a site-by-site basis, or the mechanism will not be used by administrators.

In short, we wish to replace or supplement the standard **rsh** general purpose service mechanism with a more controlled, audited, least privilege service mechanism.

In order to be true to the concepts developed in Multics, we need to provide a secure mechanism for a host to make a request of a more inner ring. This should only be through a publicised “service gate,” and we can perform appropriate authentication and verification of the validity of the data and the request, before processing it.

6.0 An Example Scenario

In this section, I present an example scenario where a network has been broken down into several host rings. This example is then expanded upon to provide detailed insight into the advantages of this scheme from the security and administration viewpoint.

The example chosen is representative of many sites in which I have personally been involved with the systems administration practices consolidation effort.

The diagram below represents a multi-level breakdown of hosts, by their function, and their relative requirements for the security of their data;

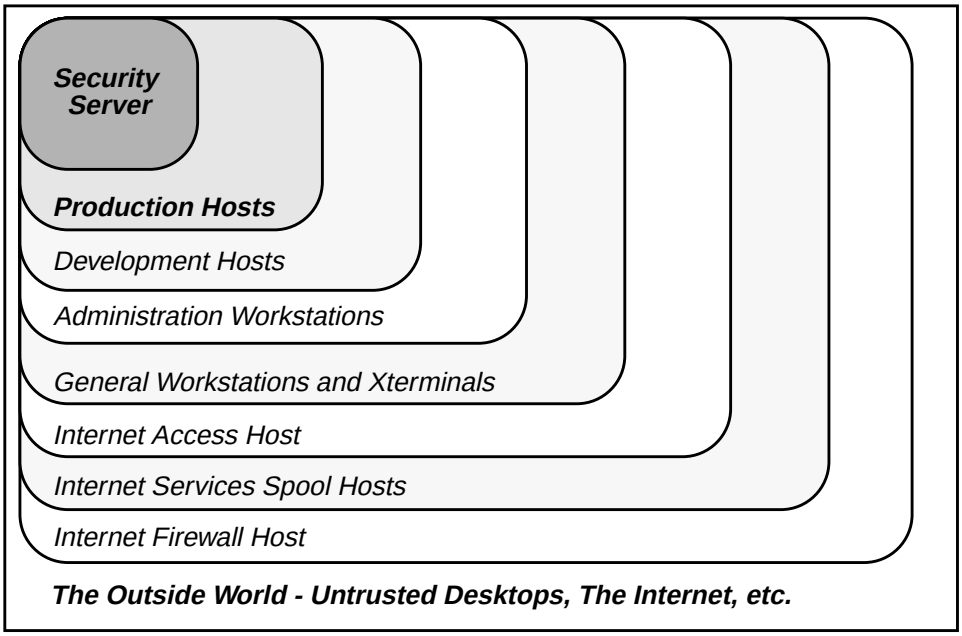

Figure 3 - The Natural Hierarchy of Distributed Services

As we move out through the rings we see that there is less physical and administrative control over the hosts, hence we extend less trust to them. Typically, the inner two rings are physically secured within a computer room, and share the backbone network. The next two rings are workstations present on the LAN, and physically secure within the building, but not as securely located as the main servers.

We might have subsidiary offices, where the lines between the offices mean that we wish to place remote production servers in a ring outside local production servers. So the rings system may become more complex. Indeed, this is an example of where we might best treat each site as a separate administrative domain, even though they are controlled by the same team of people. Alternately, we can segment these rings, as shown in the next example.

For the sake of simplicity in this example, we will assume that all hosts are located within the one building, and that the security server and production hosts are located within a computer room.

6.1 Ring 0 - The Network Security Host

Central to the successful implementation and management of Host Rings is the introduction of a new host to control and manage the security of the network. This host is known as the *security server,* or *core services host*.

This machine controls the configuration and security of the network. It does not provide actual production services to the network at large, but rather it provides the *production management* services (change management) for the network.

This host should be physically secured in a computer room, and on the same network segment as the production hosts (discussed next). The security server does not trust ***any*** other host. Conversely, ***all*** other hosts trust the security server. Typically access is either via physical access to the console, or by encrypted session or one-time password login from an administration workstation.

This is the single most important host to this scheme. This host performs several important functions;

1. **The Security Server**. It has the master copies of all files pertaining to the implementation of host rings. These are distributed to all other hosts via **rdist**. (See the section on implementation for more on this.)
2. **The Map Server**. Central to this scheme is the trust of host name and number information. Hence, this machine must manage the maps used by the various naming services such as DNS and NIS. It does not actually provide these services directly to the network, however. It, instead, propagates this information to one or more production servers which run the daemons to provide these services to the network at large.
3. **Other Centralised Administration**. Discussion of other possible uses is beyond the scope of this paper, although it is worth mentioning that all security critical images must be maintained on this machine, as this host must not trust other hosts. This includes source code for utilities used by the security server.

The nature of this workload is such that only a small server is required to fulfil this role. Often an obsoleted server or workstation can be rebuilt and used as the security server. Most important is that it is built from known trusted sources such as CD-ROM, and freshly obtained public domain software. Only a bare minimum of people should have access to this machine, and even fewer as *root*.

6.1.1 Notes About Configuring the Security Server

1. Almost all services into this machine are denied.This specifically includes the **rsh** family of commands, printing, **finger** and **smtp**. This will eliminate the ability to exploit **sendmail** bugs (which are numerous) and other potential holes to gain control of this host.
2. Remove all machines from the server's trusted host list (**/.rhosts** and **/etc/hosts.equiv**) (This is extra security, as **tcpd** should block any attempts to contact these services anyway.)
3. This machine does not export any filesystems read-write. All software is distributed via **rdist**. In practice, you may also need to export operating system images ***read-only*** for JumpStart.

4. There are no user accounts on this machine, and only the bare minimum of administration accounts. This machine must be set to ignore NIS if that is running in the network.
5. The EEPROM password if it is a Sun, or the local equivalent, should be set to allow automatic reboot. This machine should never be off the air for long, as this provides a window for someone to impersonate the security server.
6. Optionally, C2 level security is implemented (Sun). This will ensure that no files are modified without these being audited. This is the only host which requires C2, so the administrative overheads are minimised.

Maintaining the integrity of the security server is critical to this scheme. Below are a few guidelines which may help;

1. Maintain regular connectivity (ping) testing of the security server.
2. Monitor routers via SNMP to ensure the security server remains visible to the network.
3. Use encrypted sessions or one-time passwords when accessing the security server over the network. Restrict network access to the production servers. This will cause the inconvenience of having to log in through the rings to get to the security server, but this inconvenience may be worthwhile.

6.2 Ring 1 - Production Hosts

These hosts provide the production services to the network, such as database backends, NFS file serving of home directories and shared applications, printing, etc. These hosts also provide naming services for the network.

The production hosts trust the security server, and (usually) each other. They do not trust any more outer rings. This, again, specifically denies services such as the **rsh** family of commands.

They will often have full user databases, to perform their duties, but only system administration staff will be allowed any form of login access to these machines.

A good rule of thumb for these machines, is that they must be configured so that they do not rely on any other machine to boot to a full production status. This specifically implies that they all ring 1 servers should run slave naming services. This avoids the problems of cascading failures.

Too often I have found sites breaking the golden rule of client-server; “servers must not be clients.” When this rule is broken, machines must be powered up in a particular sequence, and one machine crashing can affect all other machines and effectively bring the entire network down. This is often a good indication of a need to migrate services between hosts.

Each production server should provide a self-contained set of services to the network. As in the case of naming services, this may require some duplication of services between these servers, but this overhead is a small price to pay.

6.3 Ring 2 - Development Hosts

These hosts provide a software development environment. It is worth noting that these are not trusted by the production hosts, and a special process should be implemented to effect an upgrade of an internally developed application from **development** to **test** to **production**. Such a process must be initiated from a production host.

6.4 Ring 3 - System Administration Workstations

These workstations are placed in a more inner ring than the general workstations, Xterminal servers, and other similar servers, to make the administrator's job as convenient as possible. By having these workstations in a more inner ring, **rlogin** access to workstations and general servers is available, so problems can be resolved quickly.

These workstations are not trusted by the production or development machines, however, so a more secure access channel such as **ssh** or one-time passwords must be used to gain access to a host which is in a physically secured environment.

These workstations probably do not reside in a secure area, so they should have EEPROM passwords (or equivalent) set to prevent access to the EEPROM (boot commands). It may also be possible to restrict login access to systems administration staff.

It has been suggested that system administration workstations are more trustworthy than development hosts, and so these two rings should be reversed. There is certainly validity to this argument, and such a decision can easily be taken by a site should they feel that appropriate.

6.5 Ring 4 - General Workstations and Xterminal Servers

Typically, hosts in this ring do not trust other hosts in this ring. There is often little need for this, so it makes good security to not allow it. Administration is performed from a more inner ring, so there is no loss of convenience to the administration staff.

As with all hosts, EEPROM passwords should be set, and network access to the *root* and other administration accounts denied. As described earlier, any attempt to access a service from outside the local network should set off immediate alarm bells (in the form of **syslog**).

6.6 Ring 5 - Internet Access Host

This is a special purpose host. It is used to provide restricted access to the internal network from the Internet.

So that it can access mail and news, it must be in a higher ring than the Spool Host (described next). But it is not trusted by any other workstations, so compromise of this host does not grant access to the rest of the network.

Access to this host is typically after appropriate authentication on the firewall (such as one-time passwords), which then performs an **rlogin** to this host (perhaps over a non-standard port).

If it is deemed necessary, then this machine could access NFS filesystems. Better still, this host might use a secure telnet session to gain access to a workstation on a more inner ring after further authentication.

6.7 Ring 6 - Internet Services Spool Host

This is a special purpose host. It supports all of the services of a store-and-forward nature which are provided by the Internet. These services are typically provided by large, complex public domain software packages, and are thus inherently insecure.

These packages (as has been shown with both **sendmail** and **inn**) can often be exploited *through* a firewall. For this reason, these services are isolated onto a single host (as much as possible) and that host is not trusted by the network at large.

The services provided by this host are, typically, mail (SMTP), news (NNTP), and the Web cache (HTTP). These services can be provided to the internal network as NFS exported areas, or by protocol transfer (internal NNTP). No hosts trust this host for anything other than those filesystem directories and service ports required for the allowed services.

This host is not trusted by the other hosts for access to maps such as the password or username maps. Where this information is required for correct mail operation, this database should be generated on the security server, and **rdist**'ed to the spool host.

6.8 Ring 7 - Internet Firewall

[For the sake of this discussion, a reference to the firewall host, refers to all hosts which make up the firewall as a single entity.]

This host is not trusted by any other hosts, expect the Internet Access Host described earlier (for remote login service), and the Internet Spool Host (for SMTP and NNTP transfers).

6.9 Ring 8 - PCs and Other Desktops (The Unwashed Masses)

All those machines which do not support **tcpd**, and restrictions over privileged port usage are inherently untrustworthy. Almost always, there is no form of centralised administrative control over these machines. Yet, we will often need to provide services such as file serving to these desktops. Hence, these machines are in the outer most ring, the *outside world*.

Given that, hopefully, we can still guarantee valid IP addresses, we can still perform some control, however. We can still use netgroups to restrict NFS exports, and only the absolute minimum area of a filespace should be writeable by hosts in this layer. Such areas as must be writeable should be separated into their own filesystem, and only that filesystem exposed to this ring.

6.10 A Multi-Site Example

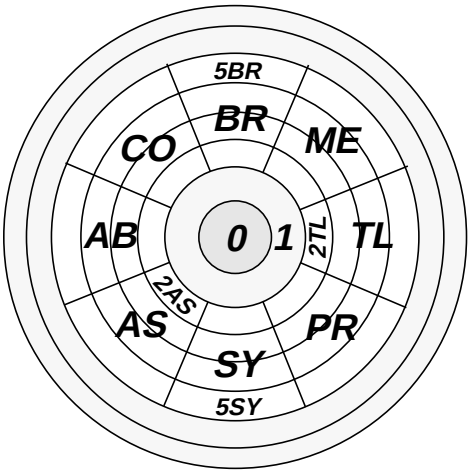

Figure 4 - Segmented Rings

In this example, we look at a company where one team manage computers at eight geographic sites. The central site manages core shared services including the Internet gateway, production management, backups, etc., and each site houses servers which provide services local to that site such as file serving, printing and site-specific applications.

An extra ring as been added to support Site Production servers, and several rings have been segmented to keep the sites out of each others way.

As you can see from the above diagram, the 8 sites are translated into ring segments. As per rule 4 of the rings, no host in one segment trusts those hosts in other segments, hence we have introduced a finer grain of trust into the system. The table below helps understand this more clearly;

Ring - Description	Segments / Sites							
Ring 0 - Security Servers	(Shared Resources)							
Ring 1 - Central Production Servers								
Ring 2 - Site Production Servers	AB	AS	BR	CO	ME	PR	SY	TL
Ring 3 - Development and Test Hosts	AB	AS	BR	CO	ME	PR	SY	TL
Ring 4 - Support Workstations				CO				
Ring 5 - General Workstations	AB	AS	BR	CO	ME	PR	SY	TL
Ring 6 - The Unwashed Masses; PCs, etc.	AB	AS	BR	CO	ME	PR	SY	TL
Ring 7 - Internet Services Spool Hosts	(Shared Resources)							
Ring 8 - Customer Accessible Test Platforms								
Ring 9 - Firewall / Security Perimeter								

The CO site is the support organisation, hence the only site with support workstations. These workstations are trusted by workstations at all sites, so that support is still relatively painless.

7.0 Implementation Details

7.1 TCPD Protected Services

The basic tool for implementing host rings is the TCP Wrapper program (usually called "**tcpd**"), developed by Wietse Venema, and described in [3]. This program allows us to control, via a single configuration file (**/etc/hosts.allow**)[1], which services will be provided to which hosts, based upon either IP address or host name.

On each host, we use **/etc/netgroups**[2] to define which hosts are in which rings, then use **tcpd** (for services invoked from **/etc/inetd.conf**) and **potmapper**/**rpcbind** to control access to all network services. All hosts share a common **/etc/inetd.conf** file which redirects all services to the **tcpd** daemon. This minimises the effort involved in administration of security policy. This also provides a mechanism for logging any attempt to access any service on any host.

The **security server** (the inner most ring) controls the master copy of these files, and regularly distributes new copies to every host (with **rdist**), ensuring that this file hasn't been overwritten by someone else (a sure sign of an unwanted visitor). We use the "notify" and "-b" options of **rdist** to ensure we are informed of any attempts to subvert these control files.

Also, all hosts within each ring share common trust files, such as **/.rhosts**, **/etc/hosts.equiv** and **/etc/hosts.lpd**. These files are also maintained on the security server and regularly distributed with **rdist**.

Whilst the **tcpd** software is not strictly necessary on the outer rings, it provides "defence in depth" so that we are not relying solely on the firewall to block attempts to breach system security from the Internet. This software will alert us to any attempt to breach the site, such as a request for a service from outside the local network.

In order to get around the problems of host authentication, **ssh** is used in place of **rsh**, and a modified **rdist** ("**sdist**") which works over **ssh** is used to distribute files between hosts. See the section on risk analysis for more on this.

The figure below shows a typical rdist Distfile;

1. Assuming it has been compiled with the appropriate options. The single configuration file option is recommended as it is easier to check rules, and one less file per host to maintain.
2. This assumes that the site is using NIS. If this is not the case, then we must generate explicit host lists in each **/.rhosts** file and the equivalent for each service.

```
#-------------------------------------------------------------
# Ring 0 - Security Server Only (me)
#
ring0:
( ring0/sshd_config ) -> ( ${RING0} )
        install /opt/local/etc/.;

ring0:
( ring0/ssh_config ) -> ( ${RING0} )
        install /opt/local/etc/.;

ring0:
( ring0/hosts.allow ) -> ( ${RING0} )
        install /opt/local/etc/.;

ring0:
( ring0/inetd.conf ) -> ( ${RING0} )
        install /etc/.;

#
#-------------------------------------------------------------
# Ring 1 - Central Production Servers
#
ring1:
( ring1/root.shosts ) -> ( ${RING1} )
        install /.shosts;

ring1:
( ring1/root.rhosts ) -> ( ${RING1} )
        install /.rhosts;

ring1:
( ring1/sshd_config ) -> ( ${RING1} )
        install /opt/local/etc/.;

#EOF
```

Figure 5 - Example Rings Distfile

7.2 Service Gates to Inner Ring Services

At the time of writing, this tool has been designed but not crafted. At the client side, **sudo** (or similar) invokes a control script which packages up the request and authenticates the user on the client side. This script then uses **ssh** to send the request to a special port on the server. At this point **ssh** has authenticated the host making the request, and hence trusts the credentials of the user making the request.

This server process can then verify the user's authority to make the request, verify the validity of the data stream, and then invoke a suitable service script to process the request. All of which does not require particularly much programming or execution overhead.

8.0 Risk Analysis

1. **Internal network Domain Name Server**. The security server controls the files which are used to provide all naming services; if it has been compromised, we are stuffed anyway. In addition, the security server should provide its own naming services. If there are too many hosts to do this via files, then the security server should run its own DNS server, and a tool such as **ipfilter** should block any external attempts to connect to it.
2. **Window of compromise between rdist updates**. Technically, we could be paranoid and **rdist** these files every five minutes. In reality, the host integrity should be checked regularly with programs such as **Tripwire**.
3. **Host compromised**. Should a host be compromised, the attacker still cannot proceed further into more inner rings, except via those services which are permitted by the inner hosts (mostly restricted to service-gate based requests described earlier). Progress must be into one of the two inner most rings before any real damage can be done, and hopefully a number of warning bells would be going off by this time. This is one of the most powerful advantages this scheme has over the normal **web of trust** that Unix networks tend to implement by default.
4. **Impersonation of the Security Server**. This is the most vulnerable point of this scheme. If someone successfully impersonates the security server, then they can compromise any host in the network. Of course, this is typically no worse than before the implementation of host rings, except that in such a case, the administration team would have a false sense of security (if no warning bells had gone off).

   Impersonation of the security server can be in several forms;

   1. *Disconnect/Down the Real Security Server.* If this is done, then any machine on the network can claim to have that IP address (routing tables aside).
   2. *Isolate the network segment that the security server is on*. If the network segment that the security server is on is disconnected, then another host can take over the IP address of the security server, and wreak havoc. In practice, the security server is generally located on the network backbone along with the major production servers, so it is not easy to take the security server off the air without someone noticing. Likewise, if any form of subnetting is used, it will be difficult to pretend to be on a secured subnet when you are not.
   3. *Inject IP packets which appear to be from the Security Server.* This is the real threat, and goes back to the window of compromise discussed earlier.

   The basic solution to all of these problems of impersonation is to use public key cryptography to sign distributions from the security server. This would stop all attacks dead. As it happens, using the aforementioned **sdist**, which is **rdist** compiled to use **ssh** as its file distribution transport, this is exactly what we do. Hence all these problems go away.
5. **Session on Security Server is Subverted**. If a terminal session over the network is subverted, then the network is effectively compromised. It is important, therefore, to do as much as possible through the service gates, and to only login to perform a specific function when necessary. The security server should have a very short inactivity timeout. Encrypted sessions to the security server (via **ssh**) also alleviate this problem.

The general guidelines outlined in the earlier discussion of the security server, help in minimising the risk of the security server being compromised.

8.1 Other Security Measures

The scheme described here is in no way a substitute for other security measures. Nor does it alleviate the need for integrity checking (such as with **Tripwire**) on all hosts, especially the administration stations and above.

8.1.1 IP Address Spoofing

Refer to [2] for a detailed discussion on IP address spoofing and related topics. The technology described here should be used in conjunction with a Firewall. One would hope that, within the confines of an internal network, we could trust the IP numbers presented with a reasonable degree of safety. Of course this may not be a valid assumption in very large corporations, or other large network environments.

Although liberal use of **ssh** and public key cryptography goes to the heart of the problem of authentication, it has a high cost in terms of CPU resources. Good defence-in-depth suggests that we should not place all our trust in the firewall to block all attempts at address spoofing, and it would be prudent to implement **ipfilter** or equivalent on all important hosts to further enhance security.

8.1.2 Use of SSH - A Warning

SSH provides end-to-end encryption for a session. This implies that we trust (a) the copy of the client **ssh** program invoked, and (b) the security of the environment within which we call the **ssh** client program.

Most importantly, using **telnet** or similar to log into a workstation so that you can run **ssh** to get from there to the destination machine defeats the entire purpose of **ssh**. You must invoke **ssh** as early in the sequence as possible, and be sure that no traffic from your keyboard to the **ssh** program is visible in clear text on any wire.

You should also be satisfied that a valid copy of **ssh** is being invoked.

I have seen people **rlogin** to their administration workstation so that they can then **ssh** into the security server. This immediately means that their secret key (actually their pass phrase which protects the secret key) travels in clear text across a hostile network. Oops.

9.0 Discussion and Conclusions

Although the concepts and implementation of Host Rings are not particularly difficult, the implementation within an operating production environment is not a simple task. Many undocumented examples of the web of trust are scattered throughout existing production environments, and to “flick the switch” would cause many services to break.

The initial advantages of this scheme come from the mindset it embodies, and from the new perspective on assessing the security requirements of services.

By defining policy as to Availability, Confidentiality and Integrity of services, we are able to make explicit decisions about how we manage services. Indeed, we can set explicit policy per ring, and ensure that services receive the attention that the users expect.

At one site, an administrator had a problem where it felt wrong, but he couldn’t quantify that feeling. One machine was providing the production backup services, but also the development database services. Once we defined the host rings it became clear that these were incompatible services for the one host to be providing. The development services were quickly migrated to another host. Similarly, it became clear that we needed to separate the web and mail services from the file-server services.

The implementation of a security server with a restricted set of administrators, helps refine and control many system administration practices. These benefits all begin accruing long before a true host rings environment is implemented.

As with all changes to system administration practices, it is easier to define the rings, and use them on new services, than it is to retrofit this model to existing services; usually by at least one magnitude. So the implementation of a system like this is a slow path that we journey down, never a switch that we just turn on.

9.1 Conclusions

The host rings scheme seems to form a useful addition to the security administrators toolkit. It is not intended to replace other tools, nor make them less important, but the simplicity of the implementation and administration of the scheme make it a valuable tool for the centralisation of computing policy.

A single directory of **host.allow** files and one **Distfile** can control and restrict unnecessary access within a multi-hundred node network.

The mindset and attitudes that this formal model embodies can help shape the way that an administrator approaches the provision of a new service and, like the firewall, it encourages investigation, analysis and explicit decisions to be made prior to the provision of a new service which must be supported within a production environment.