

Back to the Basics: Solaris and inetd.conf Part Two

Hal Flynn 2000-03-20

Solaris File Access Control Lists

Scope

This collection of material has been written to further clarify one of the more mystical configuration files on today's Solaris Operating System. For this example, we make use of the latest distribution of Solaris (at this time, 7). The topics covered are quite familiar to most seasoned administrators, although this publication has been written with administrators from beginning to advanced skill levels in mind. This document will focus on rpc services started via inetd.

An Analysis of RPC

To make a short analysis of Remote Procedure Call (RPC), let's look at the internals, and how it works. First, a query is sent to rpcbind on port 111. rpcbind keeps a table of the rpc services registered on the system, and sends the request for the service to the appropriate port. RPC is capable of use both the socket and TLI API, making a number of different transport protocols available. Here, we'll only discuss the protocols used on the current Sun Solaris version 7 (11/97). You'll notice there are some rpc services contained in /etc/inetd.conf. There's a valid reason for this. Much like the other services that run under control of inetd, these services don't run until they're needed. At the time they're called for, rpcbind makes the call for them, then inetd starts the service. Let's get close and personal with some of the internals.

The Protocols

Before getting into the protocols, let's explain a few of the standards to make the terms a little more meaningful. rpc has a few different types of protocols, with more protocols called semantics. Semantics are the basics of how the connection functions.

tpi_clts is the first and most commonly used semantic. It functions as a transport provider interface and connection-less protocol. When descending the stack, this semantic makes use of UDP to traverse a network.

tpi_cots is the next semantic. It is also a transport provider interface. This semantic is connection oriented in nature. As you might have guessed, this means it uses TCP when

descending the stack to the network.

`tpi_cots_ord` is the last we'll discuss. This semantic is basically the same as `tpi_cots`. However, it has an added feature. This semantic supports orderly release. "Orderly release?" you say. Here's the difference. The first type uses what's called "abortive release." This means that at any time, the connection between the two ends may be broken, and all unsent data discarded. Orderly release does the opposite, preventing the loss of data by sending all data before closing the connection. At any time during transit between the two ends, if the connection is aborted, the data transport continues until all data is sent from the server to the client.

The first commonly used protocol is `circuit_v`. The official reference is the man pages for `rpc` and `netconfig`, but we'll give a brief summary of it. The `circuit_v` protocol is a connection oriented transport medium which uses only visible connection oriented protocols from `/etc/netconfig` (semantics `tpi_cots` and `tpi_cots_ord`).

The second commonly used protocol is `datagram_v`. It's similar to `circuit_v` in that it queries `/etc/netconfig` to use only visible protocols, except for the fact that, as the name may indicate, it uses connection-less datagram transports (`tpi_clts`).

In reference to the DoD model, `rpc` is a high level protocol, functioning on the application level. It makes use of External Data Representation (XDR) in packets. XDR is another unique part of `rpc`, which serves a few good purposes, such as resolving differences in hardware architecture and byte ordering (Big Endian versus Little Endian).

RPC used with Solaris is what's known as Transport Independent RPC (TI-RPC). TI-RPC is capable of running over any protocol. This means that although the default is to run over TCP and UDP, it can be fitted to function in any environment. A RPC request can additionally be specified to run over a specific protocol exclusively by setting the Environment Variable `NETPATH`. When an RPC procedure starts, it first searches out the `NETPATH` variable, then defaults to `/etc/netconfig`. With this said, let's look at the services.

RPC Services

`admind` is the distributed systems administration daemon. This daemon interfaces with the Solstice Admin Suite package, making remote administration (such as adding users, setting passwords, and the like) possible. It functions using TLI over standard UDP. This daemon has been the topic of numerous discussions in security circles, as it has been used to exploit a number of Sun systems due to a buffer overflow (See Sun Security Bulletin #00191). This

daemon supports security features worth investigating if using it is a must.

rquotad is the remote quota daemon. It manages quotas for local file systems mounted on remote systems, and returns the info to the remote system. This service functions over the datagram_v protocol.

rusersd is the remote network username server. It allows a remote server to view a list of the users on a local machine. This server users datagram_v as well as circuit_v.

sprayd is the spray server. It's used mainly for testing, and often to simulate a network load. It's also capable of reporting the dropping of packets under special circumstances. This service uses datagram_v.

rwalld is the remote wall daemon. It provides all the usual functions of wall, making it possible to send wallops to remote systems. It makes uses of datagram_v.

rstatd is the remote statistics server. It provides performance information to from the kernel, and also makes the neat little graph on the CDE Desktop work. rstatd uses datagram_v.

rexed is the remote execution server. It's similar to rlogin and rexec, as it allows a user to remotely execute commands, and for interactive commands spawning a pseudo-terminal to handle the request. This service uses standard TCP and TLI. It by default has relaxed security. If its use is necessary, security specifications can be made in pam.conf. It's often a greater risk than rexec, rlogin, and rsh.

ttdbserverd is the Sun Tooltalk Database Server. To talk briefly about Tooltalk, it is an integrated application environment, acting as a mediator of communication between applications, and allowing them to exchange data. Tooltalk additionally allows the launching of applications with different respective data types. For example, a .txt file, when double clicked upon, goes to Tooltalk, which in turn launches a text editor. If a hyperlink is contained in this document, clicking upon the hyperlink could additionally launch a web browser, and so forth. This daemon process is a database daemon responsible for keeping the TT_DB file at the mount point of each partition, containing files that will be used by Tooltalk, or Tooltalk objects. This will allow a user with a remote ttsession to launch applications using the protocol. There have been recent problems with ttdbserverd, namely a buffer overflow which has allowed the remote penetration of numerous Solaris Systems. See Sun security advisory #193.

ufsd is currently part of an incomplete add-on package, which in the future will be capable of fixing ufs file systems.

kcms_server is part of the Kodak Color Management System. This daemon uses rpc over tcp. The purpose of this daemon is serving profiles to remote workstations. inetd starts the daemon when a request is made from a remote KCMS library. Here's a little detail as to what this all means. KCMS is essentially a C++ framework of objects for showing color on the display. The *kcms_server*'s sole client is the KCMS library. When a request for profiles is made by a remote KCMS library, the *kcms_server* searches `/usr/openwin/etc/devdata/profiles` and `/etc/openwin/devdata/profiles` for information in a read-only capacity. When the query is made, the server is started by inetd, and only runs as needed. See the KCMS documents on <http://docs.sun.com> for object and structure information.

cachefsd is the cache filesystem daemon. It functions using tcp. This daemon is responsible for the cache filesystem, which takes frequently accessed files and stores in a central directory on the disk. It can be administrated with the *cfsadmin* utility.

kerbd is the Kerberos daemon. It's responsible for validating rpc requests using the Kerberos encryption scheme for authentication. It additionally generates Kerberos tickets (authentication keys) and maps remote users into their respective uid and gid. This service is used for Secure RPC, and more notably Secure NFS. *kerbd* uses a protocol not previously mentioned, called *ticlts*. It functions over loopback (local to host).

gssd is the Generic Security Service Daemon. This daemon is an additional level of security, coupled with *kerbd* for Secure RPC. It is responsible for identification of both sides of the transaction, data integrity, privacy, and additionally encrypts data between the points of the transaction. This process uses *tpi_cots_ord*. It uses loopback as well.

Finally, there's *rpc.cmsd*, the Calendar Manager Service Daemon. It is primarily used by *dtcm*, the GUI Utility for managing calendars. It functions over udp. The primary function of this daemon is database management of local calendars. It additionally permits users on remote hosts to create calendars locally by configuration of *cmsd.conf*. This daemon has also had some recent security issues. Refer to Sun Security bulletin #188 for more information. Recent Security Developments

Unless you've been out on Safari, the likelihood of having heard about the recent DDoS of some rather large name Internet Businesses is pretty good. I'll only make mention of a few of the recent things, as there's still developments being made in detection and removal of DDoS servers.

Trin00, the Tribe Flood Network (tfn), and stacheldraht are Distributed Denial of Service tools

used to consume networks resources, and create a general annoyance. Rather than going into explanations of these tools and talking at great about them, I'll simply refer to Dave Dittrich's excellent analysis of them found at:

<http://www.washington.edu/People/dad/>

Now while these are not directly related to inetd, bear with me while I make a connection. When these clients were initially discovered, it propagated itself on a large number of Solaris Systems. This is due to some of the recent problems with rpc code and the servers deployed with a Solaris System, most notably, the Calendar Manager (rpc.cmsd) and the Distributed System Administration Daemon (sadmind). These two daemons, along with a barrage of other rpc problems, make anything rpc related undesirable for the security conscious.

What Can You Do?

There are a number of approaches one can take in securing inetd. The seemingly most successful and rational way is the "Minimal" approach, cutting out everything that's not needed. Trimming services reduces the risk of being compromised by some exploitable, intermittent problem in code, and lightens the load on a system. It's a win-win situation.

Additionally, there are some programs out there to make inetd a little more secure. Tcp Wrappers is a widely used wrapper program for all daemons running in inetd.conf. It's capable of providing access control lists based on services, and clients wishing to use them. It's highly configurable, and can also be customized to generate alert mails and verbose logs. The shortcoming with this program is it's inability to protect rpc services.

Tripwire is another program used to protect your system. It's not directly related to inetd, but warrants further investigation for the curious. Tripwire makes and stores md5 hashes of all system binaries to external media (such as a floppy disk), which may be write protected and compared against the system on a regular basis. There is currently a free distribution, although there is much recommendation against its use. The commercial distribution is the latest and greatest, and there's also currently gnu versions being developed.

Although not directly inetd related, rpcbind by Wietse Venema can make help as well. Rpcbind is a cross between Sun's rpcbind and tcp wrappers, placing access control on rpcbind. This will not necessarily defend your rpc services, as a determined person will just scan for rpc services directly, but it will make it a little harder for the less determined and less knowledgeable.

If your organization has either not taken firewalling seriously, or doesn't have the funding to purchase firewalls, there are a number of good open source and free software solutions available out there. Ipf is one that I'm particularly fond of, and is coming along quite nicely. There are also many low cost commercial solutions available.

Conclusion

Inetd is a simple program, making system performance better and more scalable. It supports a number of complex services, and is a critical point of review when securing a Solaris machine. Rpc services should be approached with caution, and disabled when unnecessary. Awareness, Operating System patching, and routine security audits are all part of a good administrative schedule, making a system less vulnerable to downtime, and other indirect results from negligence. DDoS is real and present problem, and can be stopped by responsibility of administrators and security conscious staff. Added layers of security are something that all responsible administrators, managers, and security personnel should investigate. Not to is to flirt with inevitable disaster.

Acknowledgments and special thanks to: Alan Cox at Red Hat Inc., Josh Uziel and Casper Dik at Sun Microsystems, Steve Goldsby at Integrated Computer Solutions, Andy Kroll, and anybody I might have missed.

[Privacy Statement](#)

Copyright 2006, SecurityFocus