Who's Trusting Whom?
How To Audit and Manage Users' .rhosts Files

Michele D. Crabb (crabb@nas.nasa.gov)
- Sterling Software / NASA Ames Research Center

RNS-94-002 April 1994

ABSTRACT

The use of *.rhosts* files to provide trust between systems has generated much controversy in the UNIX world. On one hand, people argue that by using the *.rhosts* file, you are reducing or even alleviating the number of times a user's clear text password is transmitted across the network, which is considered a plus. On the other hand, if you use and allow *.rhosts* files, you may increase the possibility of your system being compromised by Internet intruders. At the Numerical Aerodynamic Simulation (NAS) Facility at NASA Ames Research Center, users depend on *.rhosts* files to run remote interactive programs and to remotely login without the use of passwords. At the same time, security is a paramount concern at NAS. This paper describes the methods to monitor and manage users' *.rhosts* files at the NAS facility.

Introduction

In today's large, multi-system environments it is not uncommon for users to have accounts on multiple systems at a single site or accounts on multiple systems at multiple sites. The proliferation of accounts causes a number of problems for users and system administrators. From the users' standpoint, having a large number of accounts can make it difficult to remember the password for each account. Some users get around this problem by using the same password for all of their accounts, which is a very poor security practice. Another problem from the users' standpoint is how to transfer files, when needed, between the different hosts. One solution to both of these problems is to have the different systems "trust" each other. This would alleviate the need to use or even remember the password for each account. On most standard UNIX systems today, there are two methods to accomplish this trust. One method is to use a configuration file called */etc/hosts.equiv*, which lists all of the hosts which are to be implicitly trusted by the local host. The other method is to use a user configurable file call *.rhosts*, which is located in the user's home directory. The *.rhosts* file acts as a personal */etc/hosts.equiv* file. Both of these files are consulted when using any one of the r-commands (*rlogin*, *rsh*, *rcp* and *rcmd*).

Issues of Providing Trust

The issue of "trust" between various systems presents a whole set of problems for the system administrator or security analyst. The use of these two files, especially the */etc/hosts.equiv* file, is considered a poor security practice by many people in the UNIX world. One vendor's man page for *.rhosts* even says "Use of the *.rhosts* file presents a security risk; use the file very cautiously or not at all in situations where security is a concern." David Curry, in his paper "Improving the Security of Your UNIX System", states "The only secure way to manage rhosts files is to completely disallow them on the system."

While the use of the */etc/hosts.equiv* file is almost universally seen as an evil by system administrators and security analysts alike, there are two schools of thought on the use of *.rhosts* files. On one hand, people argue that by using the *.rhost*s file, you are reducing or even alleviating the number of times a user's clear text password is transmitted across the network, which is considered a plus. Of course, if you are using Kerberos or something similar, the transmission of a clear text password over the network is not an issue. On the other hand, if you use and allow *.rhosts* files, you may increase the possibility of your system being compromised by Internet intruders. If an intruder is able to break into a host that a user on your system has an *.rhosts* entry for, the intruder has an open door into your system if the path of vulnerability, via the *.rhosts* file, is discovered. This paper will present an in-depth analysis on the pros and cons of using *.rhosts* files, some typical types of *.rhosts* entries, and some methods to monitor and manage users' *.rhosts* files.

To start off the discussion, I ask the question, "Should you allow *.rhosts* files at your site?" The answer to this question depends largely on the specifics of the configuration at the site. Is the internal network gate-wayed to the Internet via a firewall system? Is the internal network connected to the Internet? Are the internal users trusted? Is security a major factor at your site? Is Kerberos available or already installed at the site? Does the type of work performed by the users at your site require "trust" among the internal hosts or remote hosts on the Internet? These are some of the questions you need to ask yourself before you can determine if the use of *.rhosts* files is appropriate for your site. From the users' standpoint, the use of *.rhosts* files is very beneficial. The trust mechanism provided by the file allows them to remote copy (*rcp*) files from one host to another. It allows them to remote login without needing a password or having to remember a large number of passwords. The *.rhosts* file also allows users to run remote interactive programs. For the latter function, the use of *.rhosts* files is imperative.

At the NAS facility, it has been necessary to allow users the freedom to use *.rhosts* files, should they want or need the functionality. The NAS facility is comprised of over 300 computer systems running some flavor of the UNIX operating system. These systems are interconnected into a heterogeneous network using various types of network hardware and are supported by over 100 personnel who provide ongoing support and software development. The NAS facility, which primarily provides supercomputing services to other NASA sites, large aerospace corporations and a large number of universities, has over 1500 users nationwide. Most of the NAS users are at remote locations, and many of them run remote, interactive graphics or plotting packages from their remote sites. Hence, the *.rhosts* file is a crucial part of their daily work.

If you have made the decision to disallow *.rhosts* files at your site, then there are several precautions that can be taken to ensure users do not attempt to create one in their home directory. The simplest action to take would be to run a cron job (daily or weekly) which searches for *.rhosts* files and removes them. A more permanent solution would be to modify the sources for *rshd* and *rlogin*d so they ignore the *.rhosts* file and only consult the */etc/hosts.equiv* file. .

Types of *.rhosts* File Entries

Once you have made the determination that *.rhosts* files will be allowed at your site, you need to decide what types of *.rhosts* entries will be allowed. Before discussing the different types of good vs. bad entries, let’s examine the format of the *.rhosts* file and the different types of entries which can be used.

The format of the *.rhosts* file is almost the same as that of the *hosts.equiv* file. Each line contains a hostname [username] pair or a special case entry. The system name can be the hostname of the remote system, the IP address of the remote system, or a netgroup name on the remote system. The hostname included in the entry must be the official hostname of the system and not some alias or secondary name. The username can be the name of any user on the remote host, the name of a netgroup, or the field can be left blank, in which case, the username defaults to the user of the account on the local host. The ability to classify a group of systems or users into a “netgroup” is a function of NIS (Sun’s Network Information Services). The *.rhosts* file also allows the use of a spec~~ial designator (a “+”) in the hostname or username field. The “+” provides trust for all hos~~ts or all users from the host specified. See *Figure 1* for an example of typical *.rhosts* file entries.

```
foobar.nas.nasa.gov crabb
badboy.nas.nasa.gov crabb
mustang.arc.nasa.gov crabb
```

Figure 1: Typical *.rhosts* file entries

The above entries would allow the user “crabb” to remotely access the local hosts from foobar, badboy and mustang.arc.nasa.gov without needing to supply a password.

Now that you have some familiarity with the types of entries found in a *.rhosts* file, how do you decide what should be allowed or disallowed? Again, this will depend largely on the configuration of your site and the desired level of security. Prior to the Morris Worm experiment, it was quite common for sites to provide trust between all hosts on the local network via the */etc/hosts.equiv* file. This was the case at the NAS facility. However, many sites no longer allow the use of the file to provide unilateral trusts on the local network. Generally, it is safe to allow users to add *.rhosts* entries for other systems on the local network; however, the entries should only allow logins for the local user. Account sharing via the use of *.rhosts* entries is very popular at UNIX sites. At NAS I frequently find users who are sharing their accounts via *.rhosts* entries.

The next level of *.rhosts* entries would include those for non-local systems (e.g., systems outside of the local domain). In a site that uses a firewall to connect to the outside world, it would not make sense to have such entries in the *.rhosts* file. The same would apply to sites using host-based filtering which filters out all non-local domain connections. However, at a site which allows non-restricted access to and from the Internet, the use of non-local *.rhosts* entries may be appropriate or even needed. At the NAS facility, many of our users are at remote sites, and as a part of their daily work, they may run remote, interactive programs which require trust between the local and remote system. As such, we allow *.rhosts* entries for non-local domain systems on our main systems. Access to NAS workstations is restricted to the local domain, hence non-local *.rhosts* entries on NAS workstations are non-functional. The next, and final level of *.rhosts* entries would include the use of the special designator “+”. The “+” designator can be used in the hostname or username fields to add trust for all hosts or all users. This type of entry is the most insecure, and should never be allowed. The “+” designator is not allowed at the NAS facility.

When deciding if you want to allow *.rhosts* files, you should also consider whether *.rhosts* files will be allowed for the *root* account or any other system account. Many system administrators and security analysts may consider to use of *root .rhosts* files an extreme evil to be avoided at all costs.

However, in a large heterogeneous environment, many system administration tasks would be very difficult or impossible without the use of *root .rhosts* files. At the NAS facility, *root .rhosts* files are used on the workstations, but are not allowed on mainframe systems such as the Convex and Crays. If you do allow the use of *root* account (or any uid 0 account) *.rhost*s files, you should minimize their use and only allow *root* entries for a limited number of systems in the local domain. For example, if you have a small number of file servers and a large number of workstations, you might want to use a *.rhosts* file that would provide trust for all file servers among the workstations and file servers.

Once you have determined what kinds of *.rhosts* entries will be allowed, you need to outline these in a formal policy, which is distributed to all users. The policy should also state the required permissions on the *.rhosts* file, what type of entries are explicitly not allowed, and what actions will be taken against users who violate the policy. At the NAS facility, we have a policy which covers the use of *.rhost*s files, */etc/hosts.equiv* files and batch configuration files such as the *.netrc* file. The files are monitored on a regular basis and appropriate action is taken against users who violate the policy.

Methods to Manage and Audit *.rhosts* Files

One of the major drawbacks of using or allowing *.rhost*s files is that they are often abused by users and targeted by intruders. Users frequently add entries for other users to allow them access to their account or files (e.g., when doing an *rcp*). Sometimes, out of ignorance, users will even add a "+" entry to their *.rhosts* file. Intruders who break into systems often use the *.rhosts* file as a means to provide themselves a back door into the systems for future "visits". The intruders will add a *.rhosts* file for a system account such a *bin* or *daemon* or they may add an entry into a regular user's account. Once a user creates a *.rhosts* file, the file is usually forgotten, and the contents are rarely looked at. Hence, should an intruder gain access to the system and add a *.rhosts* entry, it may go unnoticed for weeks or months. This actually happened at NAS on one occasion involving a break-in from Australia. The intruders added a *.rhosts* file for the *bin* account with a single entry of the form~~ "+", and they also added a "+" entry to several users' accounts. Fortunately, the incident~~ and *the .rhosts* entries were discovered within several weeks.

```
Rhosts File Audit Report for Chaos
========================================
Summary Report For User: crabb
# of .rhosts entries: 2
# of non-operational entries: 0
# of non-NAS entries: 0
# of possible illegal entries: 0
========================================
Summary Report For User: kensiski
# of .rhosts entries: 69
# of non-operational entries: 12
# of non-NAS entries: 8
# of possible illegal entries: 0
========================================
Summary Report for Host: chaos
----------------------------------
The total number of rhosts files is: 2
The total number of .rhosts entries is: 71
The total number of non-operational entries is: 12
The total number of remote entries is: 8
The total number of possible illegal entries is: 0
```

Figure 2: Example of a long form *raudit* report

Due to the increased risk that the use of *.rhosts* files bring, it is apparent that some method of monitoring and managing all users' *.rhosts* files is needed. The remainder of this paper will discuss an *.rhosts* file auditing program written and used at the NAS facility, and actions taken against users who abuse the policy. But first, I would like to provide a little background information and motivation for writing the audit program. Back in early 1990, I received a phone call from Dan Farmer (author of cops), who was working at CERT at the time. He was considering a joint project with a friend which involved doing an analysis of users' *.rhosts* files. He was interested in getting a copy of as many NAS users' *.rhosts* files as he could. Dan and his partner were interested in obtaining a large sample of files to see just how many users were using the *.rhosts* facility and what type of entries they were adding. One of the items Dan was interested in was determining how many *.rhosts* files were located on a single system -- sort of a "the security of this system is reliant on the security of N systems".

Since a user's *.rhosts* file was considered sensitive information, I told Dan that I would not provide him copies, but instead I would be glad to provide him the statistics. Out of curiosity, I wrote a quick shell program that would tell me how many users had *.rhosts* files and how many entries each user had and whether the entries were for systems that were included in the */etc/hosts.equiv* file or not. I ran my primitive script on few of the systems at NAS and was shocked at the numbers. Some users had 20-30 *.rhosts* entries for systems all over the Internet.

After this eye-opening experiment, I decided that a more extensive auditing program was needed. The initial *raudit* program was written in the Bourne shell. The first version of the program collected and reported on the following items for each user and for each host: the total number of *.rhosts* entries, the number of non-operational entries (i.e., hosts which were included in the */etc/hosts.equiv* file), the number of remote entries (i.e., hosts that were not in the local domain), the number of entries that were either malformed or possibly illegal (e.g. the username in the entry did not match the local user name). This program was run once a week on all NAS systems via a cron job. Due to the popularity of *.rhosts* files at NAS, the weekly output from the *raudit* run was close to 300 reports a week. Some reports were 5-10 pages long. See *figure 2* for an example of a full *raudit* run on a local workstation.

The first run of the report generated a large number of entries where the username in the *.rhosts* file did not match the user's name who owned the account. The process of determining if each of these entries was really a case of account sharing or just a case of different login ids for the same user, was a very arduous and time-consuming task. For each questionable entry, I would look up the user's full name in our user database to determine if the *.rhosts* username was an alternate login id for the same user. If the information from our user database did not provide a sufficient answer, I would do a *finger* of the *.rhosts* username at the remote host. In many cases this would provide an answer. If the *finger* daemon was not enabled on the remote host, I would *telnet* to the smtp port and do a *vrfy* on the *.rhosts* username. If these measures failed, I would send email to the user requesting verification that the *.rhosts* entry was valid. If the entry was valid, I would make note of it for future reference.

After several months of running with the initial version, it was apparent that much of the information being reported was not necessary or even useful in a weekly report. Also, the method I was using to keep track of all the alternate login names for each user was inefficient. What I was really needed was a list of possible illegal entries for each user. Hence, version two was inspired. The second version of *raudit*, written in Perl, incorporated the use of an alternate login id database, and the use of command line options. Version two has the option to print a short report of just illegal ("bad") entries. Since this version incorporates the use of an alternate login id database, the weekly reports show truly illegal entries, and in some cases, new accounts where alternate login ids need to be added to the alternates database. See *figure*3 for an example of an *raudit* short report (bad entries only).

```
Rhosts File Audit Report For foobar
-------------------------------------
WARNING:  Possible  illegal  or  malformed  .rhosts  entries  for  user
johndoe:
 sunny.larc.nasa.gov majdi

WARNING:  Possible  illegal  or  malformed  .rhosts  entries  for  user
janedoe:
 uxh.cso.uiuc.edu aae391ac

WARNING:  Possible  illegal  or  malformed  .rhosts  entries  for  user
jsmith:
 grace.nas.nasa.gov root
 grace root

WARNING:  Possible  illegal  or  malformed  .rhosts  entries  for  user
jnsmith:
 rtccd.arc.nasa.gov *
```

Figure 3: Example of an short form *ruadit* report

Version two also incorporates an option to search *.rhosts* files for a key word, such as a specific hostname. Prior to version two there had been several incidents involving systems in the *arc.nasa.gov* domain. When auditing the NAS systems to see if there had been any attempts to break in, one of the items I was interested in was to know if any NAS users had *.rhosts* entries for hosts which had been compromised on the *arc.nasa.gov* sites or some other domain. At the time I didn't have an easy method to accomplish this task on over 300 systems. However, the task lent itself well to a feature of the *raudit* program. The command line to run *raudit* to find a match for a hostname is: *raudit -m -h host_keyword.* The host keyword can be the short hostname, the fully qualified name or any sub-string. Sometimes I use this feature to locate *.rhosts* entries for a specific host. This feature of the *raudit* program has been very useful in the follow-up of security incidents. With just a few commands, I can locate all host entries for a specific host or domain from all systems at the NAS facility.

Once I populated the alternate login id database, reading and processing of the weekly *raudit* reports became a much simpler task. The master copy of the alternates database and *raudit* program are kept in the master source tree on a file server. The alternates database is updated as the reports are read and processed each week. Each week the alternates database and the *raudit* program are distributed to all systems before the cron job to run *raudit* executes.

Auditing and managing of *root .rhosts* entries is another issue. At the NAS facility we allow *root .rhosts* entries on all workstations and the file servers. Instead of auditing the *root* account *.rhosts* files on a weekly or daily basis, we have a nightly cron job which reinstalls the *root .rhost* files with a master copy. All workstation *root .rhosts* files are the same. This method of auditing was chosen to reduce the need for human interaction. Also, replacing the *root .rhosts* file on a nightly basis is helpful because sometimes during the course of system work or testing, the *root .rhosts* file will be modified to add a temporary entry. Occasionally, the support people will forget to remove these "temporary" entries. Replacing the file on a nightly basis ensures temporary entries are really temporary. Non-root entries in *root .rhosts* files are not allowed at the NAS facility.

Methods To Handle Abusers Of The Policy

Invariably, users will abuse the *.rhosts* file policy, either intentionally or out of ignorance.

Appropriate action must be taken against the policy abusers to ensure that future infractions do not occur. A policy regarding the use of *.rhosts* files and what are the allowable entries should be clearly defined and provided to all users on the system. The policy should also state what action will be taken against users who abuse the policy. A written policy will also provide justification for the actions taken against people who abuse the policy.

At the NAS facility, the action taken against policy abusers depends on the severity of the violation. The NAS account policy states that account sharing is not allowed and that adding *.rhosts* entries for other users is considered account sharing. Per policy, we have the right to disable a user's account for suspected account sharing. However, when the account sharing is via a *.rhosts* file entry, the user will get one or two warnings prior to the account being disabled. If the entry in question provides trust for what appears to be a general use account (e.g., *guest* or *visitor*) at a remote site, I remove the entry immediately, and send the user an message such as the one shown in *figure* 4.

```
Mr. Doe -
The following illegal .rhosts entry was REMOVED from your .rhosts on
wk00:

Rhosts File Audit Report For wk00
------------------------------------
WARNING:  Possible  illegal  or  malformed  .rhosts  entries  for  user
johndoe:
36.65.0.120 guest # ctf unix

Adding entries for other users, especially guest accounts, is against
NAS policy. Please do not add such entries in the future or your NAS
accounts will be disabled.

Michele Crabb, crabb@nas.nasa.gov
Computer Security Analyst/Distributed System Support
Sterling Software Incorporated/ NASA-Ames Research Center
Mail Stop 258-6
Moffett Field, CA 94035
(415) 604-4337
```

Figure 4:Example email message sent to users with *.rhosts* entries for general use accounts

The user is given one email warning about adding such entries in the future. If the user adds the same entry again or another illegal entry, the user's account will be disabled. Per NAS policy, we can disable the user's account for up to three days before attempting to make contact. Depending on what the user has to say when we call, the account will be re-enabled or remain disabled.

If the entry in question provides trust for another user at the NAS facility, I will send email to both users. Usually, in these types of cases, the users add entries for each other in their *.rhosts* files. The users will be warned via email that future infractions will result in both accounts being disabled. The users are given one week to remove the entry. If the entry is present at the next *raudit* report, then I will remove the entry and attempt to contact the users by phone. In some cases, the users have not read their email. If the *.rhosts* entry appears to be for another user at a remote site, I will send an email warning message to the local NAS user requesting the entry be removed. The user is given one week to remove the entry. As in the case above, if the entry is reported on the next *raudit* report, I will manually remove the entry, and try to contact the user by phone. A typical email message sent to users shown in *figure* 5.

All email correspondences sent to users regarding *.rhosts* entry violations are archived. New reports of illegal entries are checked against the archived mail. If a user re-adds an entry that was previously declared illegal (or a similar entry) weeks or months after the first warning, the user's account will be disabled for a minimum period of three days. Users can permanently lose their NAS account privileges over *.rhosts* policy violations, although there have yet to be any cases

```
Hi -
You have the following illegal .rhosts entries on wilbur:

WARNING: Possible illegal or malformed .rhosts entries for user
janedoe:
catfish.cs.umd.edu robertz
hyena.cs.umd.edu bkmone

Adding .rhosts entries for users other than yourself is against NAS
policy.
Please remove these entries and any others like them on any of your NAS
accounts as soon as possible. Adding such entries in the future may
result
in having your NAS accounts disabled.
```

Figure 5: Email message sent to users with illegal *.rhosts* entries

.When corresponding to users who violate the *.rhosts* file policy, I suggest other methods the user can use to accomplish the needed tasks. In some cases, it may require more work on the user's part, but the user understands the policy must be followed. Many times I receive email replies back from the users apologizing for the entries which include various explanations. Frequently, the users will tell me the entries were added by mistake. Some users have told me that the other user never logs into the account, but only uses the *.rhosts* entry to remote copy files back and forth. There have been a number of a cases where users at universities add *.rhosts* entries for their "assistants". In these cases, I inform the users (usually a project Principal Investigator) they need to request accounts for their assistants as well. I even had one case where the owner of the account was rarely using his account. Instead, his two "assistants", from two different universities, were using the account. The owner of the account didn't think he was violating any policy. Some of the stories I have heard from users regarding their illegal *.rhosts* entries have been quite amusing. In some sense, I know how a police officer feels when listening to the different excuses people provide for speeding.

Future Directions

Ideally, it would be nice to have an alternate means of providing login authentication without the use *.rhosts* files or the sending clear text passwords over the network. One solution to this problem is the use of the Kerberos software or something similar. Another solution would be to use the challenge and response cards, such as the SecureID card. The NAS facility is currently considering these two possibilities. However, for very large sites it may be difficult to install Kerberos on the growing number of systems and platforms, and it might be difficult to provide challenge and response cards to a user base that exceeds a thousand. Until there are more cost effective solutions, the use of the *.rhosts* file will remain an issue system administrators and security analysts will have to address.

There are several modifications that could be made to the *raudit* program to improve functionality and performance. The current method used to store and search the alternates login id database is inefficient. When I first developed the idea of the alternates database, I was unaware that it would grow to its current size. This feature of the *raudit* program needs to be redesigned. As a part of the maintenance of the alternates database, it would be nice to have some function which could attempt to do a verification of *.rhosts* entries where the user has different login ids on different hosts. For example, a function which could do a *telnet* to the smtp port to do a *vrfy* command on the login id, would be a solution. Another improvement related to the use of the alternates database is the need for an automated method to know when a user in the database needs to be deleted. Currently, I have to manually check the alternates database against the official NAS user database to ensure archived users are removed. Although, this does not cause any problems with the *ruadit* program, reducing the size of the database would reduce the execution time of the program. A method to validate a hostname would also be a helpful feature. I have found cases where users have entries in their *.rhosts* files for hosts which no longer exist.

Another idea I have been considering for the management of *.rhosts* files and the *ruadit* program is to limit the number of non-local *.rhosts* entries a user would be allowed to have. There would be no limit on local *.rhosts* entries; however, some small finite number of remote entries, such as five or maybe ten would make the process of auditing and managing *.rhosts* files much easier. If such a scheme were to be implemented, users could specify which remote hosts they wish to use in their *.rhosts* files on their account request forms. Users could request additions, modifications, and deletions to their list of allowed hosts. This information could be stored in a database that would be accessed by the *raudit* program. If a user were to add an entry that was not previously requested, it would be reported as a illegal entry by the *raudit* program. Minimizing the number of *.rhosts* entries for remote hosts would also reduce risk level to the local network.

Availability

The *raudit* program is freely available from the NAS facility. To receive a copy via email, send an email request to *doc-center@nas.nasa.gov.* If you have any questions regarding the use of the *raudit* program or how security is handled at the NAS facility, send email to *crabb@nas.nasa.gov.*

Author Information

Michele Crabb has been the primary computer security analyst for the NAS Facility at NASA Ames Research Center for over four years. During her nine years at Ames, Michele has worked in several divisions, in a variety of positions ranging from applications programming to UNIX system support. Prior to becoming the NAS security analyst, she was actively involved in providing system administration support for the large number of workstations at the NAS facility. Michele can be reached via electronic mail at *crabb@nas.nasa.gov*, or via US Mail at NASA Ames Research Center, Mail Stop 258-6, Moffett Field, CA. 94035-1000.

RNS TECHNICAL REPORT

Title: Who’s Trusting Whom?

How to Audit and Manage Users’ .rhosts Files.

Author(s):
Michele D. Crabb

Clearance:
Form 427 has been filed with the division secretary. This report is unclassified. _______ Author’s initials.

Reviewers:
“I have carefully and thoroughly reviewed this technical report. I have worked with the author(s) to ensure clarity of presentation and technical accuracy. I take personal responsibility for the quality of this document.”

Signed: ________________________________

Name: ________________________________

Signed: ________________________________

Name: ________________________________

Branch Chief:

Approved: ______________________________

Date & TR Number: