A Unix Network Protocol Security Study: Network Information Service

David K. Hess, David R. Safford and Udo W. Pooch
Texas A&M University
dhess@cs.tamu.edu

Introduction

This note is a study of the security weaknesses present in a widely used Unix network protocol, Network Information Service (NIS). NIS (formerly known as Yellow Pages or YP) was developed by Sun Microsystems primarily to reduce the effort required to setup and maintain a network of Unix workstations. This is accomplished through the centralization on a NIS *server* of the major configuration files required to setup a Unix machine for a particular site. Thus only the set of configuration files on the NIS server need to be updated to effect all machines at a site. This has proven to be a very powerful network administration tool. However, it also becomes an enormous security problem. Spoofing of the NIS server can be used to completely penetrate security on an NIS client.

NIS Structure

NIS is based upon the Remote Procedure Call (RPC) protocol which uses the External Data Representation (XDR) standard. Below this level are the raw communications services of Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) provided by the Internet Protocol (IP). The relationships between these protocols are shown in Figure 1.

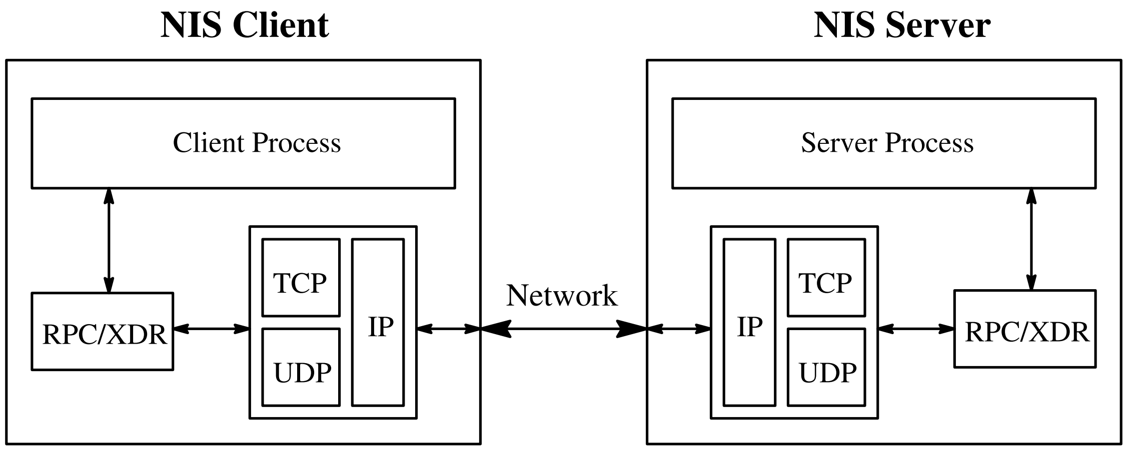

Figure 1. NIS Protocol Structure

RPC implements a method by which a *client* process on one machine can perform a virtual procedure call to a *server* process on a remote machine. The client is considered to be accessing a feature of a service provided by the server. The client calls an RPC procedure with the arguments for the remote procedure and does not return from the call until the request has been sent to the server, processed, and a reply received. The message is encoded using XDR so that RPC can be used between heterogeneous machines using different internal data representations. The actual transmission of data is performed using either TCP or UDP depending on the desires of the client and the design of the server.

Security for both the RPC request and reply is supported in three modes. The first is to perform no authentication checking at all. This is the default mode of a RPC service and is used for public databases and services which are not considered security risks. The next mode uses traditional Unix authentication based on machine identification and user id. While better than no authentication at all, this type of authentication is not very strong. Sun Microsystem's Network File System (NFS) by default uses this style of authentication. The last mode is Data Encryption Standard (DES) authentication which is considered to provide relatively strong security. This method is based on a user's password in conjunction with a public key/private key system and the ability to encrypt a timestamp. NFS allows the RPC authentication mode to be switched from Unix to DES if tighter security is desired.

The purpose of NIS is to provide a distributed network database. Since the configuration files on a Unix machine served by NIS are world readable by default and the NIS server is built only to query, not update the database, NIS is considered a safe, public service and uses RPC in the no authentication mode.

When the NIS server is configured, the Unix configuration files are processed to produce maps consisting of key/value pairs. For example the */etc/hosts* file contains the IP addresses and corresponding names for other machines. In this case the NIS configuration program generates two maps for the file. The first, *hosts.byname,* allows queries by the name of a machine while *hosts.byaddr* allows queries by address. Both queries respond with the original line out of the hosts file.

Access to NIS services is usually hidden from the programmer by adding NIS support to the library routines that access information in the Unix configuration files. The C library routine *gethostbyname()* which gets the entry for a host out of the */etc/hosts* file will generate a NIS/RPC request to the server if NIS is enabled and access the local file otherwise.

NIS Security Weaknesses

As was mentioned above, NIS is considered secure since the server does not modify the maps directly and the files are world readable to begin with. However, when the focus is placed on the NIS client, the security of NIS is actually quite poor.

Since NIS performs no authentication at the RPC level, any machine on a network could easily create a fake RPC reply simply by pretending to be the NIS server. This is shown in Figure 2. The difficulty of this depends on the underlying communications protocol used for the RPC call. For TCP, a virtual socket connection would have to be faked on the offending machine which is a difficult task. For UDP, it is much easier due to the simplistic structure of the protocol. NIS/RPC communication primarily uses the UDP protocol. A weakness of this attack is the fact that packet smashing in most local area networks is not possible and that therefore any fake NIS/RPC reply generated must be delivered to the NIS client before the NIS server responds with the valid reply. Also, the intruding machine must be able to both see the request and generate the fake reply. This requires that the intruding machine be on the same communication path used by the NIS server and the NIS client.

The security problems that result are due to the semantic meaning of the data provided by the NIS maps. Fake replies can in effect create any map desired to replace the one actually on the server. For instance, the *hosts.byname* and the *hosts.byaddr* maps provide information that is used in traditional Unix authentication. An intruder could access a NIS client machine through the Berkely r–commands (rsh, rlogin, etc.) by mapping in the *hosts.byaddr* map the address of the intruder's machine to a machine name contained in the NIS client machine's */etc/hosts.equiv* file or in a particular users *.rhosts* file.

Example Intrusion

This section describes an example of an actual exploitation of the weaknesses in NIS as described above. The Unix configuration file that is affected is the */etc/passwd* file which is mapped as both *passwd.byname*

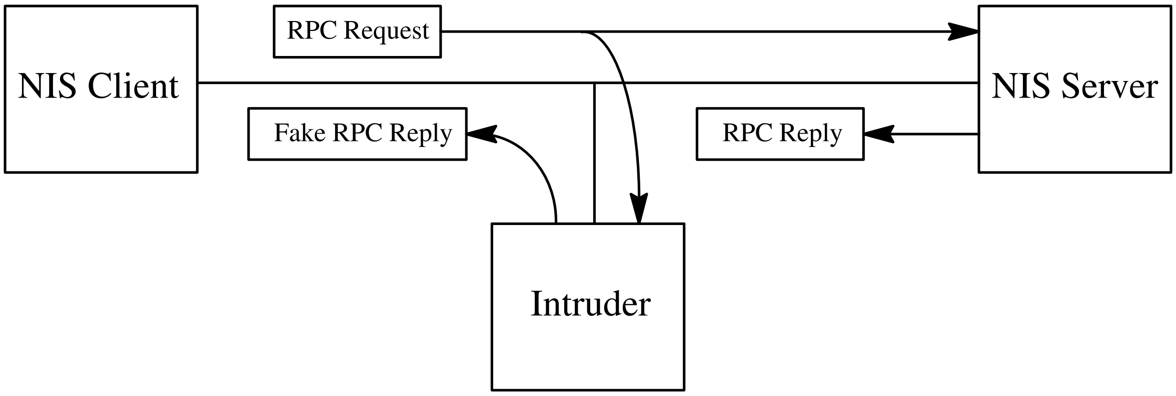

Figure 2. Intruder Spoofing NIS Server

and *passwd.byuid*. Conceptually many attacks are possible including denial of service (by supplying invalid encrypted passwords) and impostering valid users (by supplying predetermined encrypted passwords). This example describes a third type of attack which creates two new users to gain root access on a NIS client machine.

The intrusion is accomplished by a program written using the Network Interface Tap software module provided by Sun for direct access to the network on Sun workstations. This was used to change the IP source address of an outgoing packet so that the intruding machine could spoof the NIS server.

The program, ypfake, is based heavily on David Curry's public domain source for nfswatch. ypfake watches the network for NIS/RPC/UDP requests going to a specified NIS server for two particular usernames and user ids. When the requests are seen, the program generates a successful NIS/RPC reply with a fake */etc/passwd* file entry that looks like it came from the NIS server and sends it to the NIS client as quickly as possible. In this way, the two user accounts are virtually added to the true */etc/passwd* maps. The valid replies from the NIS server (which in this case indicate that the */etc/passwd* map entries for the two users do not exist) are dropped by the NIS client since the RPC code only looks for one UDP response and then disposes of the communication facilities used to make the call.

The first user account is used to login to the machine. The second user account is given a user id of 0 so that becoming that user is the same as becoming root. The transcript in Figure 3 shows the use of the two fake user entries to infiltrate a machine and gain root access. In both cases the passwords for the users were made the same as the user names.

Note that while the fake entries show up when the ypmatch command is used, the ypcat command does not see them. This is due to an implementation feature of ypfake and NIS. ypfake only intercepts NIS/RPC/UDP yp_match procedure calls used by the login and su commands. ypcat uses the NIS/RPC/TCP yp_all procedure call (for efficiency reasons) which ypfake never sees.

As was mentioned above, this attack is based on the ability of the intruding machine to deliver a fake packet to the NIS client before the NIS server. This program was tested using a Sparcstation 1 and two Sparcstation 330s. The Sparcstation 1 acted as the NIS client while one Sparcstation 330 acted as the NIS server and the other as the intruder's machine. Each of these machines was running SunOS 4.1. Under nominal loads the intruding machine was able to consistently beat the NIS server with fake RPC messages. Even if this were not the case there are many means available to slow a Unix machine down across a network. A flood of finger

```
node1[1] telnet node2
Trying 128.194.67.67 ...
Connected to node2.
Escape character is '^]'.

SunOS UNIX (node2)

login: bogus
Password:
Last login: Tue Dec  4 10:38:37 from node1
SunOS Release 4.1 (NODE) #1: Wed May 16 10:48:28 CDT 1990
node2% su intruder
Password:
node2# whoami
root
node2# ypcat passwd | egrep "intruder|bogus"
node2# ypmatch intruder passwd
intruder:dh5GzX0IUG7S6:0:1::/:/bin/csh
node2# ypmatch bogus passwd
bogus:dhKBMbnVnGu5A:67:35::/:/bin/csh
node2# exit
```

Figure 3. Intrusion Transcript

and rwho requests are two examples. To gain even more speed a powerful PC with a lan interface could be set up to do nothing but watch for NIS requests. Physically this is easy to do given the modular nature of lans and could even be done covertly with the ever decreasing size of PCs.

Implications

This weakness also effects the DES authentication mode used by other RPC services. This is due to the fact that both the password file entries and the public/private key pair are both maps served by NIS. By faking RPC replies to both an NFS server and NFS client, one could get on to a machine as any user and have full access to all file systems on the machine including any provided through NFS using the DES mode. The intruder would have a precalculated password and corresponding public/private key pair so that both the NFS server and the NFS client agree on the DES authentication.

In general, the problem is that when NIS is enabled on a system it implicitly becomes a part of almost all authentication schemes used by the system. This leaves the system wide open for intrusion and disruption by a wide variety of methods.

Possible Solutions

An obvious solution to this problem would be to convert NIS to using RPC with DES authentication. However this brings about an implementation problem. How can a service which provides the keys for an encryption procedure encrypt using that same procedure? At some point either the keys must be transmitted in an insecure fashion or they must be manually installed. The encryption procedure must be bootstrapped using some secure method before secure exchanges can begin. Under current systems we believe this can only be accomplished by foregoing the elegance of using NIS at all levels and maintaining by hand at least one password and private/public key pair for each machine. Most likely this password and private/public key pair would correspond to root. This would allow NIS to perform secure DES/RPC requests without transmitting any key information on the network beforehand.

Some other practical and simple solutions are possible. Modifications to the library routines that access the NIS features would help improve security. A passwd file entry obtained through NIS should be discarded if the user id corresponds to root. This would prevent intruders from gaining root access on an NIS client. This follows the rule of thumb that root is not trusted across the network.

Another modification would be to keep local copies of the passwd file. This could be applied in varying degrees. However, this by itself does not protect user accounts. A new account which does not exist in a local passwd file (and will thus generate a NIS request) can be faked which has the same user id as the account that access is desired to. A modification would have to be made to the NIS access library routines such that any passwd entry which comes back from an NIS request and has a user id which is already present in the local passwd file is discarded.

One could also increase security physically. The requirement that the intruding machine be on the same communication path could be exploited by the strategic placement of paranoid gateways and the physical control of the lan transmission media. Some form of line continuity alarms or physical locks could be used to prevent unauthorized access to the network.

Conclusion

NIS is not a secure facility. Yet many of the authentication procedures used on Unix machines implicitly use NIS when it is enabled. The problem comes about from the assumption that a NIS access across the network is as secure as a disk access to the local file. It is safe to assume that a read from a local file cannot be interfered with short of actually changing the file. However with NIS and more generally RPC with no authentication, a corresponding read can be made to return whatever data or lack of data is desired.

This is not an acceptable situation. Security is not an issue that should be slighted in the desire for ease in maintenance. In this case compromising NIS gives an intruder complete access to a NIS client rendering useless many security features of the system.

References

[1] Stevens, W. Richard, *UNIX Network Programming,* Prentice Hall, Englewood Cliffs, NewJersey, 1990.

[2] Sun Microsystems, Inc., *Network Programming Guide,* March 1990.

[3] Sun Microsystems, Inc., *RPC: Remote Procedure Call Protocol Specification*, **RFC 1050, ARPA Network Information Center**, April 1988.

[4] Sun Microsystems, Inc., *SunOS Reference Manual,* March 1990.

[5] Sun Microsystems, Inc., *XDR: External Data Representation Standard*, **RFC 1014, ARPA Network Information Center**, June 1987.