Global Information Assurance Certification Paper

Name: Arvid Soderberg
Version Number: 1.2f
Title: Do I need to be concerned about these Firewall log entries?

I'm working in a large health care institution that operates a hospital, a day surgery center, nursing homes, urgent care and occupational medicine facilities, and medical office buildings on four campuses, two community health clinics, as well as eight home health agencies in surrounding counties. We have installed a firewall to segregate the main campus network, a Frame Relay network connecting remote locations, wireless network implementations, a remote access concentrator used for ISDN and analog remote connections, and an Internet connection. The firewall implemented is a Nokia IP-440 running Checkpoint's VPN-1 version 4.1 software. Although we probably don't need to be concerned about industrial espionage from competitors as some businesses do, we do have patient information to keep confidential. This has always been a priority, but as the recording media change from paper to electronic, additional methods are needed to maintain confidentiality. With the coming HIPAA regulations, the cost of failing to maintain confidentiality will also increase. In this paper, I'll highlight certain entries from the firewall log file and attempt to determine the level of concern that should be associated with them.

The type of log entry to be focused on is for "service" "SunRPC". First, "What is 'SunRPC'?" The RPC stands for Remote Procedure Call. This is a specification originally from Sun Microsystems that defines how a program on one computer can cause action to be taken on another (remote) computer. The first system is considered the client, with the system running the remote procedure being considered the server. The specification allows for many RPC programs to use ports that aren't pre-assigned. In order to know how to communicate with them the RPC port mapper program is contacted on port 111, with the program number desired, and it provides the IP port to use. The RPC specification is built on the XDR external data specification. One application RPC is used to implement is the Network File System (NFS) which is used for sharing file resources among *NIX systems primarily, with some implementations on Windows systems. Some commercial applications are also implemented using RPC. The RPC specification includes fields that can be used by clients and servers for identification and validation purposes, but use of them is optional. Security and access control mechanisms are in the specification also. However, they vary from version to version with the earlier versions having rather limited security capabilities. The perceived danger with RPC is that a system running an implementation of RPC could be vulnerable to attack. The degree of vulnerability would depend on the version of RPC implemented, what RPC programs and procedures are implemented, and to what extent security features are required. Some believe that RPC has no place on the Internet. The result of running RPC insecurely, even on a private network, could be very dangerous.

In order to learn more about the RPC traffic causing the log entries, a network trace was captured using a Sniffer Pro LAN from Network Associates. All of the traffic seen here specifies version 2. The version 2 RPC specification provides for four authentication methods. The "NULL" method specifies no authentication. The "UNIX" method uses a combination of machine name, user id and group id for authentication. The "SHORT" method uses data sent

from the server to the client for future authentication after the "UNIX" method authentication has succeeded. The "DES" method, sometimes referred to as the "DH" method, uses a Diffie-Hellman public key scheme with 192-bit keys. This method uses a time stamp which is encrpted by the client and sent to the server. The server decrypts the time stamp, reduces it by one second, encrypts the new time stamp and sends it back to the client. The client then decrypts the new time stamp and compares it to the original one. The client concludes that the server successfully decrypted the original time stamp if they differ by the one second. Additional authentication methods such as Kerberos and IPSEC are included in later RPC versions.

Sources of these log entries are on the Internet, as expected, but are also on the internal network as well as the Frame Relay network. The destination address for Internet sources is the outside address of the firewall or the mail relay address using TCP. Since, the firewall is preventing entry of this RPC traffic from the Internet into the internal network and since its arrival can't be prevented, the present concern for this traffic is only that the firewall and the associated router don't present opportunity for an exploit from this RPC traffic. The destination address for internal sources is the broadcast address for the IP network which contains the firewall internal address. UDP is used in these frames. The destination address for the Frame Relay source (a desktop system) is the address used by an NT server on the internal network. The internal sources are NT and Win98 workstations as well as AIX servers.

The RPC traffic from the internal network is of greater concern both because it is already on the internal network, but also because of the potential threat of an internal intruder. This theoretical internal intruder could be someone who is curious and competent enough to be dangerous. Another type of potential internal intruder would be a disgruntled employee. Our type of business dictates that our building must be open to the public at all times. As a matter of security as well as trying to avoid tying up unutilized resources, we attempt to disconnect network ports at the hub or switch connection point at the time that end user equipment is removed from service. In spite of our efforts, we often are not notified. It would still be possible for an outsider to find an unattended area with a still connected network port, connect a laptop or other device to the network, and spend time trying to learn what vulnerabilities may be present and then take advantage of them.

The next question to be answered is "What 's in this RPC traffic being broadcast on the internal network?" The Sniffer was used to gather a sample of this traffic and decode it.

One source of this broadcast traffic is the NT workstation of an AIX administrator. These broadcasts take place regularly at four-hour intervals. Decoding of the frame revealed the following RPC contents:

SUN RPC Header:
Transaction id = 4294967295
Type = 0 (Call)
RPC version = 2
Program = 100000 (Portmapper), version 2
Procedure = 5 (Read from symbolic link)
Credentials: authorization flavor = 0 (Null)

[Credentials: 0 byte(s) of authorization data]
Verifier: authorization flavor = 0 (Null)
[16 bytes of data present]
SUN Port Map:
Proc = 5 (Indirect call routine)
Program = 100005, version 1, proc=0

Additional frames from the same workstation were identical, even the "Transaction id". Program = 100000 refers to the port mapper program mentioned earlier, the basis for RPC communications. The version numbers are important to note since the specification is written to allow for different versions of RPC and programs. Version 2 is an old RPC specification. Procedure 5 is defined as PMAPALLOC_CALLIT. This procedure is intended for a client to call another remote procedure on the same remote machine without knowing the procedure's port number. It is for supporting broadcasts to specific remote programs via the well-known port mapper's port (111). So in this case, the source of the broadcast is attempting to execute a remote procedure on all systems within its broadcast domain (subnet). The procedure to be executed is procedure 0 of program 100005. Program 100005 is for NFS related procedures, however procedure 0 is specified here which, by convention, in all programs, accepts no arguments, does nothing, and produces no results. The stated purpose of including a procedure that does nothing is to for server response testing and timing. There are no responses from any systems in this trace, but that's to be expected, given that only broadcast traffic from the station is likely to be seen by the trace tool at this location in the network. Another network trace was taken of traffic both to and from the sending station. The trace was stopped when the sunRPC broadcast showed up in the log again. Then the trace was searched for the broadcast frames and the RPC related frames following them in the trace and they were examined. There were two broadcast frames separated in time by four seconds. Between the two broadcast frames there were responses from twelve different stations. Decoding of the frames reveals:

SUN RPC Header:
Transaction id = 4294967295
Type = 1 (Reply)
Status = 0 (Accepted)
Verifier: authorization flavor = 0 (Null)
[Verifier: 0 byte(s) of authorization data]
Accept status = 0 (Success)
SUN Port Map:
Proc = 0 (Do nothing)

At the very least they have admitted their presence and the fact that RPC is active on them. After the second broadcast, the trace shows thirty more responses. Of those, eleven are from some of the twelve stations whose responses are between the broadcasts, and nineteen are from additional stations for a total of thirty-one stations responding. These systems include *NIX servers and workstations in IS, patient care areas, pharmacy, plant engineering, housekeeping, patient registration, and security departments. It seems that some additional investigation is warranted to determine the need for RPC on these systems. The results of that investigation are beyond the scope of this document.

Another source of this broadcast traffic is the Windows 95 workstation of an application

support person. Decoding of the frame revealed the same RPC contents as the first example with the exception of "Transaction id", of course.

Another source of this broadcast traffic is the IP address of x.y.z.1 with a destination address of x.y.z.225. The x.y.z.0 network is not part of our network although the source MAC address is that of a router on our internal network. Decoding of the frame revealed the following RPC contents:

SUN RPC Header:
Transaction id = 1007036772
Type = 0 (Call)
RPC version = 2
Program = 100000 (Portmapper), version 2
Procedure = 5 (Read from symbolic link)
Credentials: authorization flavor = 1 (Unix)
Len = 32, stamp = 1007383146
machine = "xxxxxxx" [value changed to hide identity of application]
uid = 0, gid = 0
1 other group id(s):
gid 0
Verifier: authorization flavor = 0 (Null)
[16 bytes of data present]
SUN Port Map:
Proc = 5 (Indirect call routine)
Program = 100005, version 1, proc=4

This looks much more serious because the procedure number to be executed on the remote system isn't the "do nothing" 0. What's program 100005's procedure 4? And where did this come from? The source address is not from a private IP network and in fact is assigned to the European RIPE organization, so it certainly doesn't belong here on our network. Further investigation has not yet revealed the system from which the frame was broadcast.

Another source of this broadcast traffic is an AIX application server. Decoding of the frame revealed the following RPC contents:

SUN RPC Header:
Transaction id = 1007511440
Type = 0 (Call)
RPC version = 2
Program = 100000 (Portmapper), version 2
Procedure = 5 (Read from symbolic link)
Credentials: authorization flavor = 1 (Unix)
Len = 52, stamp = 1007409076
machine = "xxxxxx" [value changed to hide identity of application]
uid = 0, gid = 0
6 other group id(s):
gid 0
gid 2

gid 3
gid 7
gid 8
gid 10
Verifier: authorization flavor = 0 (Null)
[16 bytes of data present]
SUN Port Map:
Proc = 5 (Indirect call routine)
Program = 100005, version 1, proc=4

Once again program 100005, version 1, procedure 4 is to be executed. This frame exhibits the use of the "UNIX" authentication method also. The application vendor was contacted to determine the purpose of the RPC traffic. The application support person said that to his knowledge, RPC was not used by the application. He thought that others in the organization might know more about it. He checked with them and they asked for the trace file to be sent to them for analysis. A trace file with a single sample frame was sent. They have yet to respond.

In interpreting what a particular procedure will accomplish, it's essential to consider all three parameters: program, version, and procedure. By example, with the combination of program=100005, version=1, procedure=4, the function performed is "remove all mount entries". This is the implementation of the NFS version 2 specification. By comparison the NFS version 3 specification uses procedure 4 for "check access permission".

The intended purpose of "remove all mount entries" seems to be for a server to tell a client to "unmount" any connections that had been established. RPC and NFS are intended to be implemented in such a way as to be as stateless as possible. In analyzing possibilties for this function to be needed three situations come to mind. First, it seems that a server would either be aware of the clients that had resources "mounted". This would violate the stateless objective. Second, the server would anticipate the potential for connections to be beyond its broadcast domain. If this was the case not all clients would receive the "unmount" message. Third, the server would just not care what resources might be "mounted". This would be perfect for a stateless implementation. Since the server is supposed to be as "dumb" as possible, with the client being responsible for having the intelligence to recover from any problems, the last option should be the implemented one. So even though a broadcast of "remove all mount entries" could be considered benign, it still is not at all clear why a broadcast would need to be used for this purpose.

Another source of this broadcast traffic is a test NT workstation used by our NT administrators. Decoding of the frame revealed the following RPC contents:

SUN RPC Header:
Transaction id = 1007469354
Type = 0 (Call)
RPC version = 2
Program = 100000 (Portmapper), version 2
Procedure = 5 (Read from symbolic link)
Credentials: authorization flavor = 1 (Unix)

Len = 28, stamp = 1007469342
machine = "yyyyyyyy" [value changed to hide identity of application]
uid = 0, gid = 4294967294
0 other group id(s):
Verifier: authorization flavor = 0 (Null)
[16 bytes of data present]
SUN Port Map:
Proc = 5 (Indirect call routine)
Program = 390109, version 2, proc=0

This is a new program the same procedure 0. Program 390109 is identified as "nsrstat", and identified with "Legato Networker (Solstice Backup)". Legato is an application we use to manage many of our backups. It looks like it's time to do some interviews with the members of our staff who administer these backups.

The interview with the individual using the "yyyyyyyy" system reveals that it is trying to establish a session with another system for the Legato application. This second system provides some sort of authentication service. So far the two systems are not communicating properly so the client system continues to send RPC broadcasts looking for the authentication system. Fortunately, this is done only when the client application is started, so it seems that this can be discounted from being a serious problem.

The "unmount" procedures need further explanation. Perhaps there's a good reason for this traffic to be broadcast, or perhaps it's due to sloppy program design or implementation. In case you think these aren't likely to happen, let me tell you of one example we experienced. This situation came about when a particular application was converted from DOS to Windows. The developer designed the new client software to check for the presence of a certain file on the server. If the file was there, the client was to shutdown immediately. The purpose of this was so that support personnel could cause the clients to shutdown for maintenance of the server. The problem was that each client checked for this file every time it checked for a keystroke. Obviously, it generated unnecessary traffic on the network and load on the server. Although this example isn't related to RPC, it demonstrates the possibility of a poorly written program.

In summary, it seems that the procedure 0 traffic is not necessarily a direct threat, but it does generate traffic that could be used by an intruder to discover systems to attack. However, in order to minimize the possibility of any exploitation, I intend to attempt to identify which software is producing the broadcasts and verify that the broadcasts are really serving some useful purpose. I'll also check those systems that are responding to the procedure 0 broadcasts. I expect to find that most of them are running NFS unintentionally and without a need for even RPC. Part of the concern with these systems is the NFS version that's being used. This is due to the fact that it is an old version without robust security measures. In those cases where NFS is appropriate, it would be best if it was implemented in a secure manner, probably with a later version. In our situation, I seriously doubt that NFS is needed at all. Concerning RPC, it will be removed from all systems possible. Where it is necessary, we'll communicate with the vendors whose applications are using RPC. It will be pointed out that a more secure version should be used and that the security features should be implemented. Will this be enough? Absolutely not!

Continued vigilance is needed. The firewall log will be monitored to be certain that it is still keeping the RPC frames from the Internet from getting past it onto any of the protected networks. Network traces will need to be captured periodically to verify that no new situations have arisen. When new applications are to be installed they should be checked, before purchase, so that problems can be addressed before installation. Applications that rely on RPC to function should be required to implement RPC securely. The need for monitoring and auditing will never end.

"SunRPC" URL: http://www.networkice.com/Advice/Services/SunRPC/default.htm (11/9/2001)

"portmap" URL: http://www.networkice.com/Advice/Services/SunRPC/portmap/default.htm (11/9/2001)

"Port RPC" URL: http://www.networkice.com/Advice/Exploits/Ports/RPC/default.htm (11/10/2001)

"rpc.mountd" URL: http://www.networkice.com/Advice/Services/SunRPC/rpc.mountd/default.htm (11/10/2001)

"Tnm::sunrpc" URL: http://webmail.cotse.com/dlf/man/tnm_sunrpc/ (11/12/2001)

"sunrpc probe host program version protocol" URL: http://webmail.cotse.com/dlf/man/tnm_sunrpc/probe.htm (11/12/2001)

Schoenwaelder, Juergen. URL: http://wwwhome.cs.utwente.nl/~schoenw/scotty/man/sunrpc.html (11/15/2001)

Peterson & Davie. "COSC 4377, Spring 2001, RPC-Supplement" URL: http://www.cs.uh.edu/~jsteach/cosc4377/2001spring/notes/rpc.pdf (11/15/2001)

White, Stephen. "Remote Exploits" 1/16/2001. URL: http://www.ox.compsoc.net/~swhite/talks/sysadmin/node9.html (11/15/2001)

Yemini, Y. "Fun and Games with Sun RPC" The Network Book. 1995. URL: http://www.cs.columbia.edu/dcc/classes/CS4119/Chapter03/Section03/section.html (11/20/2001)

Quinton, Reg. "Solaris Network Hardening" 3/13/2001. URL: http://ist.uwaterloo.ca/security/howto/2000-09-19/baseline.html (11/20/2001)

"Port Mapper Protocol" Protocols for Internetworking: XNFS, Version 3W. 1998. URL: http://www.opengroup.org/onlinepubs/009629799/chap6.htm#tagcjh_07_02 (11/21/2001)

Quinton, Reg. "Solaris Network Hardening: First Steps" Solaris Network Hardening. URL: http://www.samag.com/documents/s=1152/sam0104i/0104i.htm (11/22/2001)

"RPC: Remote Procedure Call Protocol Specification Version 2" June 1988. URL: ftp://ftp.isi.edu/in-notes/rfc1057.txt (11/22/2001)

Nowicki, Bill. "NFS: Network File System Protocol Specification" March 1989. URL: http://www.ietf.org/rfc/rfc1094.txt?number=1094 (11/23/2001)

Callaghan, B., Pawlowski, B., Staubach, P. "NFS Version 3 Protocol Specification" June 1995. URL: http://www.ietf.org/rfc/rfc1813.txt?number=1813 (11/24/2001)

Eisler, M. "NFS Version 2 and Version 3 Security Issues and the NFS Protocol's Use of RPCSEC_GSS and Kerberos V5" June 1999. URL: http://www.ietf.org/rfc/rfc2623.txt?number=2623 (11/25/2001)

Sheller S., Callaghan, B., Robinson, D., Thurlow R. "NFS version 4 Protocol" December 2002. URL: http://www.ietf.org/rfc/rfc3010.txt?number=3010 (11/27/2001)