Global Information Assurance Certification Paper

When Business Need Justifies Leaving RPC Services Enabled

By
Bertha Marasky
GCUX Practical Assignment v1.9
Option 3 – Topics in Unix Security
26 Nov 2003

Table of Contents

Abstract

The purpose of this paper is to discuss Remote Procedure Call (RPC) concepts and issues and to explore two RPC services, NIS and NFS. It is proposed that despite well-known security concerns, NIS and NFS can be the most reasonable solution for managing large heterogeneous networks. Alternative solutions to NIS and NFS are scrutinized. Default RPC, NIS and NFS installations are discussed. Recent vulnerabilities are reviewed. Best practices and options for securing and reducing risk are provided.

The Business Need

Managing a large network of systems presents a number of challenges. How to identify users from one system to another to aid in recognition of account owners and to ensure consistent file ownership when files are transferred between systems? How to make programs and data (including that on cdrom) available on multiple systems without replication? How to prevent users from having to manage passwords on an inordinate number of systems thereby encouraging them to utilize best practices for password selection and storage? How to disable user accounts, for terminated or transferred employees, in a timely fashion? How to present data stored on multiple filesystem types to a single client? And, if that isn't enough to think about, these tasks become more daunting when users and data are managed by disparate and incompatible mechanisms.

To meet the challenges, two tools often used in the management of heterogeneous networks are Network Information System (NIS) and Network File System (NFS). Consider a development shop that utilizes multiple operating systems: Sun Solaris, Tru64 Unix and Red Hat Linux. This is a medium sized shop with 75 developer workstations and 25 principal servers all supported by one systems administrator. Because it is a development environment, the criticality and sensitivity of the data is diminished. The network has been subnetted so that these servers are segmented from production systems. NIS and NFS have both been deployed for years with every one of the current systems configured to enable the services. Application developer turnover rate is not too high however reassignment to another project (thus a different set of systems) is common. Grouping of systems supporting the development of a given application can also change frequently reflecting the changing business needs and software life cycles. Unfortunately, even though security awareness is on the rise, budget dollars for security implementation are hard to come by. NIS allows centralized management of system configuration files. The administrator can modify and distribute common information without having to edit files on every one our one hundred machines. Users need memorize only one password. Accounts can be disabled in minutes instead of possibly hours. NFS allows efficient utilization and management of storage assets. Rather than have 100 installations of a given software, development tools can be installed on

principal machines and served out via NFS to the workstations. As an added bonus, it is then ensured that all developers are using consistent versions of software (including patch levels!) while allowing compilations to utilize local workstation CPU and memory resources. Developers can have one working home directory regardless of where they are logged in – all their scripts and personal preferences literally the same everywhere. Underlying these two very useful tools, is a network programming mechanism known as Remote Procedure Call (RPC). RPC is the fundamental building block upon which NIS and NFS are based. RPC simplifies writing applications that involve communication across a network and between heterogeneous systems. The programmer only has to write the client program and the server procedures that the client calls. It is not necessary for the programmer to know how to deal with network timeouts and retransmissions because RPC handles it. Issues with differences in data type storage (e.g. integer and floating point) are also handled by RPC and do not need to be considered by the programmer (Stevens, p. 462).

There are security concerns introduced by the implementation of RPC, NIS and NFS because of the lack of strong authentication mechanisms, misconfiguration of the tools and of course, coding deficiencies in the associated programs. Unacceptable availability of information can happen in a variety of ways. If an NFS exported filesystem is improperly configured without specifying which host(s) are allowed, any system on the network has access to the data. Because NIS transfers configuration map data as clear text, packet sniffing can provide information needed to gain access to a system. Unless access lists and firewall filtering are used, undesired viewing of NIS map data can be achieved via standard NIS commands. Lack of RPC authentication may result in NIS server masquerading allowing an intruder to send an NIS client bogus replies and subsequently gain unauthorized access. If file protection and export guidelines are not followed, file handle theft could not only lead to unacceptable availability, but also file damage on NFS partitions. Lastly, programming errors will always exist. If relative system patches are not applied in a timely manner, denial of service and unauthorized root access may follow.

Alternative solutions have been discussed for quite some time now. A search on the internet will produce discussions from the mid 1990's where RPC, NIS and NFS security risks are acknowledged. Yet, there are still no alternatives that have been widely accepted and integrated into all three of the operating systems examined here. (http://lists.insecure.org/lists/bugtraq/1995/Apr/0087.html)

Lightweight Directory Access Protocol (LDAP) is probably the most promising alternative to NIS however, "Note that LDAP is only a directory access protocol; it is not by itself an authentication mechanism" (Mani, May 2003). The implication being, for secure communications, applications must be LDAP aware with SSL enabled or there must be a gateway or underlying secure authentication framework. Red Hat Linux 8.0 and Solaris 8.0 support Pluggable Authentication Mechanism (PAM) as an authentication framework. PAM provides an

authentication interface to LDAP and can be used in conjunction with PAM SSL support for secure transmission. Additionally, PAM can be used in conjunction with Nameservice Switch (NSS) technology to allow integration with NIS and standard Unix flat files. This may ease the migration to LDAP where NIS is already widely used. While also based on OpenLDAP, Tru64 uses Security Integration Architecture (SIA) for a proprietary secure authentication solution underneath LDAP. While LDAP may have a future, it still has not been tightly integrated into all operating systems and often support must be installed as layered packages on the client and the server. All of the pieces necessary to enable secure authentication make it non-trivial to implement and configure. While we are beginning to see LDAP integration into operating systems it will be a while before it could be considered inherent. Solaris 8 includes an LDAP client but not a server. Tru64 5.1A offers an LDAP client and server available on the Internet Express cd that comes with most new servers. A separate annual right to upgrade fee is required for software updates. Red Hat 8 offers the most integrated solution. The future direction of SSL vs. Transport Layer Security (TLS) utilizing the Simple Authentication and Security Layer (SASL), as specified by RFC2829 (ftp://ftp.rfc-editor.org/in-notes/rfc2829.txt), should be considered before a huge investment, in time or money, is made in migrating to an LDAP solution. It was noted in the original TLS RFC 2246 (ftp://ftp.rfc-editor.org/in-notes/rfc2246.txt) that, "The differences between this protocol and SSL 3.0 are not dramatic, but they are significant enough that TLS 1.0 and SSL 3.0 do not interoperate (although TLS 1.0 does incorporate a mechanism by which a TLS implementation can back down to SSL 3.0)." TLS support was introduced in Solaris 8 via patch kit 108993. Communication with the Tru64 LDAP server is via SSL. Red Hat supports both SSL and TLS. While these solutions provide secure transmission, they also require certificate management by an internal or external trusted certificate authority. This, of course, adds to the complexity and potential maintenance costs. If migrating from an existing solution, for example NIS, implementation may be even more obfuscated. Even though os integration is underway; don't forget other applications, that make their own authentication calls, may need to be made LDAP aware. One last consideration should be overall corporate direction for directory services and single sign on mechanisms. Secure LDAP implementation today is a huge effort and its future is still being realized. Jumping on the "bandwagon" too soon may result in major rework down the road.

NIS+ is another alternative to NIS. In 1992, Sun Microsystems released NIS+ which was developed to address several NIS deficiencies including security (Stern, pg. 24). NIS+ introduced mutual encrypted client and server authentication via Secure RPC. According to Sun, http://wwws.sun.com/software/solaris/faqs/nisplus.html, NIS+ will be removed from the operating system after Solaris version 9. Customers are directed to consider migration to iPlanet Directory Server (based on LDAP standards). Linux development for NIS+ has stopped (Kukuk, July 2003). Obviously, this is not the future!

A financially viable option to replace NIS may be maintaining static configuration files on a master host and pushing them out using rsync with the SSH option enabled. This is however, "putting all your eggs in one basket". If your central machine is compromised, your whole network is compromised. This would also require users to log in to the central server to change their password or provide some other mechanism such as a web based form (Pomeranz, 2003, p 159). This also implies that all those users will have to have accounts on the central server which would provide much more fertile grounds for compromise.

One possible alternative to NFS is the Server Message Block (SMB) protocol also now known as Common Internet File System (CIFS). CIFS is primarily a Microsoft Windows-centric solution however, there is a Unix implementation using SMB called Samba. While Samba exists, the original intent of SMB was to allow sharing among Windows clients. To enable this cross-platform functionality in our Unix-variants-only environment does not seem as suitable as a solution that was designed with Unix in mind. The following message from the Samba Team, at http://us1.samba.org/samba/whatsnew/samba-2.2.8.html from the 2.2.8 security bug fix release, suggests the gain over NFS may not be significant.

> The SMB/CIFS protocol implemented by Samba is vulnerable to many attacks, even without specific security holes. The TCP ports 139 and the new port 445 (used by Win2k and the Samba 3.0 alpha code in particular) should never be exposed to untrusted networks.

NFS may also be replaced by the Distributed Computing Environment Distributed File Service (DCE/DFS). DCE/DFS sits on top of DCE and uses DCE RPC.

> DCE is called "middleware" or "enabling technology." It is not intended to exist alone, but instead should be bundled into a vendor's operating system offering, or integrated in by a third-party vendor. DCE's security and distributed filesystem, for example, can completely replace their current, non-network, analogs. DCE is not an application in itself, but is used to build custom applications or to support purchased applications (Mauney, Q1.01, July 2001).

Digital Equipment Corp., later absorbed by Compaq, most recently acquired by Hewlett Packard, was an original proponent of DCE. Now, when visiting the DCE homepage at Hewlett-Packard, http://h30097.www3.hp.com/dce/, you will find

> In June of 1999, Hewlett-Packard (formerly Compaq Computer Corporation) contracted with Entegrity Solutions® Corporation to take over responsibility for development, maintenance and support of DCE (including DFS) for Tru64 and Windows. As part of this agreement, HP also transferred the ownership of the DCE product set on these platforms to Entegrity®.

While the Entegrity® DFS offering supports Red Hat and Tru64 clients, Solaris is not included (http://www.entegrity.com/products/dce/features.shtml#platforms). DCE/DFS for Solaris can be obtained from IBM. In addition to requiring a multi-vendor solution, it should also be noted that "ONC RPC [used by NFS] and DCE RPC are not compatible" (Mauney, Q2c-04, July 2001). Client and server applications must be written with matching protocols, using a gateway or to support both flavors. This would render migration from NFS to DCE/DFS more drastic. It would take quite the justification to get approval to not only implement a complete middleware replacement, but have to pay for it too!

One last alternative to NFS that will be mentioned is AFS. AFS provides security through mutual client/server authentication and Kerberos support. AFS provides more granular access using seven different access permissions instead of the three Unix mode bits. A FAQ is available at http://www.faqs.org/faqs/afs-faq/. AFS was originally developed at Carnegie Mellon's Information Technology Center and was known as Andrew File System. AFS spun off from Carnegie Mellon as Transarc Corporation. IBM later acquired Transarc. While maintaining a commercial version for legal reasons, IBM made much of AFS available as open source and called it Open AFS. Commercial AFS will not work as a solution in the heterogeneous environment discussed here as Tru64 is not supported. The Transarc Lab no longer officially works on the Open AFS version, but there is another limited open source AFS effort called arla. While there has been support for arla on Tru64, recent versions have not been specifically ported and may need extra fixes to work. Arla currently offers only a stable client, no server. Visit http://www.stacken.kth.se/project/arla/ for more information.

Reducing the risk of using RPC and implementing NIS and NFS involves ensuring access to the environment is restricted via network design and a solid firewall. Access to the host is restricted using filtering software and following best practices for most securely configuring the operating system (os) and tools.

Note: it is assumed that the reader can consult and follow posted GIAC GCUX practicals and other sources for securing the os therefore, step-by-step instructions for accomplishing minimal os security are not included here. Please keep in mind that dependent upon the role of the server you are building, you may need to modify actions listed in a step-by-step guide. For Tru64 v5.1A see http://www.giac.org/practical/GCUX/Marc_Despres_GCUX.pdf. A guide for Red Hat Linux v8.0 is http://www.giac.org/practical/GCUX/William_Souza_GCUX.pdf. See http://www.giac.org/practical/Robert_Donaldson_GCUX.doc for a securing Solaris 8 document. There are references in this paper to steps in the SANS Institute Solaris Security Step by Step version 2 guide, 2001.

Understanding Remote Procedure Call (RPC)

A simple diagram of network programming with RPC is depicted in Figure 1.

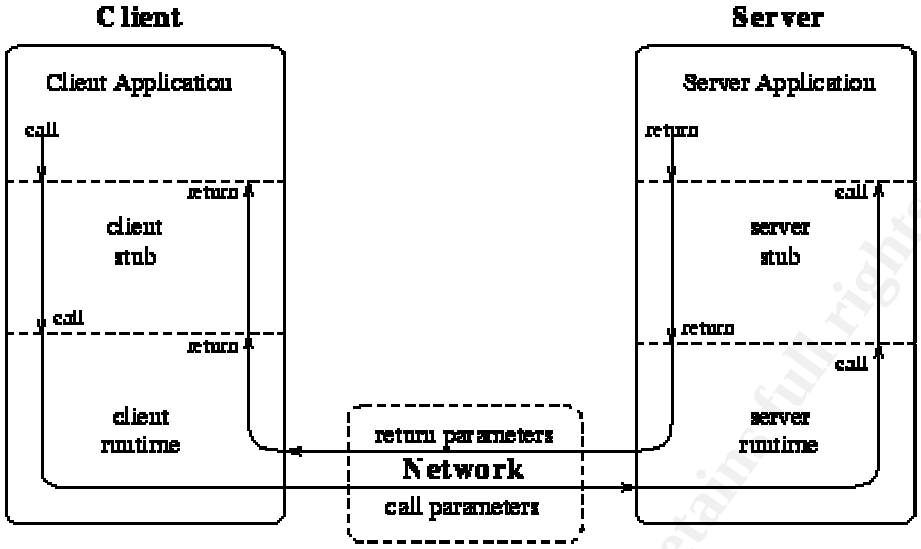


Figure 1: RPC Model (Barkley, October 1994)

As mentioned earlier, RPC allows the programmer to develop network based applications without having to really understand the underlying network interfaces. The client application makes a call to a procedure on the remote server via stub packages. The client side stub takes the call and associated arguments, translates it from the local data representation format to a common data format, bundles it up and hands it off to the server side stub via RPC. The server side stub translates it to local data format and hands it up to the server application. RPC uses the External Data Representation (XDR) protocol, RFC 1014, to describe the common format of its data. The translation to and from the common data format is what allows heterogeneous machines to communicate (and alleviate some of the network programming burden). It is assumed that differing common data representations is one of the reasons that some RPC flavors are incompatible.

There are a variety of RPC implementations including Open Software Foundation (OSF) DCE RPC, Secure RPC and Sun Microsystem Open Network Computing (ONC) RPC. As mentioned earlier, DCE/DFS (the NFS alterative explored earlier) uses DCE RPC. Secure RPC is used to refer to ONC RPC using the AUTH_DH (aka AUTH_DES) mode of security. AUTH_DH, using Diffie-Hellman public key exchange, is no longer considered secure by many security experts (Stern, pg 258-259). Secure NFS and NIS+ (the NIS alternative explored earlier) utilize Secure RPC. We will now focus on ONC RPC which is used by both NFS and NIS. (All references to RPC from this point forward are to ONC RPC.)

An RPC packet uses Internet Protocol (IP) addresses to identify hosts. Processes available on a given host are identified by a port number. Typically,

RPC applications use ephemeral port numbers and do not have the port number hard coded. They rely on a mechanism that accepts a registration request from the application and subsequently assigns it a port number. This mechanism is known as the portmapper. Solaris calls this daemon rpcbind. Tru64 Unix and Red Hat Linux call it portmap. The portmapper keeps track of registered RPC applications by program number, version and protocol. Upon startup, an application registers itself with the portmapper and obtains a free port upon which it will listen for client requests. A client application contacts the portmapper to find out which port it should use to communicate with a service. One big problem with RPC is that is provides weak authentication. This is summarized at http://hal.csd.auth.gr/thelug/unix-books/rpc.txt, "The standard Unix portmapper assumes that security will be handled by the servers, allowing any network client to communicate with any RPC server". Since the portmapper is an RPC service itself, it too is subject to this assumption. Most modern portmappers will only accept service registration from the localhost and requests to bind to a reserved port number (below 1024) from a privileged port. It would not be too difficult for a malicious user to craft an RPC packet that spoofs a request for service registration, appearing as if from the local host. Another issue with RPC is that even though the portmapper always listens on port 111 tcp and udp many RPC applications use ephemeral ports. It is difficult to configure packet filtering because the port number may change each time the program is started.

Of course, RPC is comprised of multiple applications and libraries. An RPC related vulnerability announced in early 2003 is known as follows: CERT VU#516825, CVE CAN-2003-0028, Sun Alert ID 1884, HP SRT2322 and Red Hat RHSA-2003:089-11. All os versions in the shop described in this paper were vulnerable. An integer overflow was found in the XDR libraries. This was a particularly bad one because the XDR libraries are used by multiple vendors in a variety of applications. Dependent on usage of the identified routine, potential damage is summarized as stated in CERT Vulnerability Note 516825 (http://www.kb.cert.org/vuls/id/516825):

> Exploiting this vulnerability will lead to denial of service, execution of arbitrary code, or the disclosure of sensitive information. Furthermore, because RPC services often run as *root* on affected systems, this vulnerability may be leveraged to gain remote root access on vulnerable systems.

Applying patches and effective router filtering would significantly reduce exposure to both this vulnerability and remote attempts to register services. Keep in mind that simply blocking access to the portmapper will not be sufficient because there are tools widely available that scan for any listening port. Ports or port ranges for all listening services should be blocked at the firewall. Be sure to filter spoofed packets coming from outside the firewall that appear to have originated within the intranet. Host based filtering, access controls and logging can reduce the possibility of internal address spoofing success and aid system administrators in becoming familiar with normal system access and activity (so

that abnormal activity will be more obvious). Use of the replacement portmapper by Wietse Venema disables proxy access. This replacement portmapper requires installation of tcp_wrappers which provides access control and logging. Implementing iptables or ipchains, or another host based firewall, enables host based packet filtering and optional logging. Do not overlook the very important step of disabling any RPC services that are not necessary. RPC services can be enabled in startup rc scripts or via the inetd daemon.

In the network described in this paper, Red Hat Linux version 8 is the most "security ready" operating system. It comes with tcp_wrappers, a portmapper that uses the tcp_wrappers library and iptables. Configuration is all that is needed. For Tru64 and Solaris, the iptables source can be obtained from http://www.netfilter.org. Alternatively, Tru64 offers a host based firewall called FireScreen. It is available on the Internet Express cd that comes with each new server. At http://wwws.sun.com/software/securenet/lite/download.html Sun offers SunScreen Lite for Solaris 8 which is a host based firewall that can be downloaded for free. There is a purchasable version of SunScreen however, Sun makes the following statement at http://wwws.sun.com/software/securenet/, "Sun has focused future efforts beyond SunScreen 3.2 towards more fully integrating stateful packet filtering into the Solaris OS rather than producing separate layered firewall products. This Solaris feature is still under development." Tcp_wrappers source can be obtained from ftp://ftp.porcupine.org/pub/security/index.html.

On Red Hat use "chkconfig --list |grep iptables" to make sure iptables is enabled. Use "iptables --list" to see the ruleset currently defined or view the /etc/sysconfig/iptables file. Use the "service iptables start/stop" command to start and stop filtering. The following iptables entries will allow incoming packets bound for port 111 via tcp or udp only if they are in the specified IP address range. All other packets are dropped.

```
iptables -A INPUT -p tcp -s! 192.168.0.0/24 --dport 111 -j DROP
iptables -A INPUT -p udp -s! 192.168.0.0/24 --dport 111 -j DROP
```

The IP address range can be specified in CIDR notation or netmask. Unfortunately, when dealing with ephemeral ports, the entries in iptables will either need to be a range of port numbers or configuration will need to occur after the RPC server has been started and the portmapper has assigned the port(s). See http://www.redhat.com/docs/manuals/linux/RHL-8.0-Manual/security-guide/ch-fw.html for more information on using iptables to filter traffic.

The following is provided to show that the NIS process ypbind will still function after applying our sample (somewhat loosely secured) iptables filters.

```
------------------------------------------------------------YPBIND DOWN, IPTABLES OFF
lnxtest# rpcinfo -p
   program vers proto   port
```

```
   100000    2   tcp    111  portmapper
   100000    2   udp    111  portmapper
   100024    1   udp  32768  status
   100024    1   tcp  32768  status
   391002    2   tcp  32769  sgi_fam
lnxtest#
lnxtest# ps -ef | grep ypbind
lnxtest#
lnxtest# iptables --list
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
lnxtest#
-----------------------------------------------------------------YPBIND UP, IPTABLES OFF
lnxtest# /etc/init.d/ypbind start
Binding to the NIS domain:                              [  OK  ]
Listening for an NIS domain server..
lnxtest#
lnxtest# ypwhich
t64tst
lnxtest#
lnxtest# ypcat -k netgroup
testusers bmar
testhosts nustest lnxtest
nustest (nustest,-,FrExmpl)
lnxtest (lnxtest,-,FrExmpl)
bmar (-,bmar,-)
#
lnxtest# rpcinfo -p
   program vers proto   port
    100000    2   tcp    111  portmapper
    100000    2   udp    111  portmapper
    100024    1   udp  32768  status
    100024    1   tcp  32768  status
    391002    2   tcp  32769  sgi_fam
    100007    2   udp    775  ypbind
    100007    1   udp    775  ypbind
    100007    2   tcp    778  ypbind
    100007    1   tcp    778  ypbind
lnxtest#
-----------------------------------------------------------------YPBIND UP, IPTABLES ON
lnxtest# /etc/init.d/ypbind stop
Shutting down NIS services:                             [  OK  ]
lnxtest#
lnxtest# service iptables start
Flushing all current rules and user defined chains:     [  OK  ]
Clearing all current rules and user defined chains:     [  OK  ]
Applying iptables firewall rules:                       [  OK  ]
lnxtest#
lnxtest# iptables --list
Chain INPUT (policy DROP)
```

```
target     prot opt source               destination
ACCEPT     all  --  anywhere             anywhere
LOG        all  --  anywhere             anywhere            LOG level warning prefix `iptables_input_stuff'
ACCEPT     tcp  --  anywhere             anywhere            tcp spts:0:1024
ACCEPT     udp  --  anywhere             anywhere            udp spts:0:1024
ACCEPT     tcp  --  anywhere             anywhere            tcp dpts:0:1024
ACCEPT     udp  --  anywhere             anywhere            udp dpts:0:1024
ACCEPT     udp  --  anywhere             anywhere            udp spt:nfs
ACCEPT     tcp  --  anywhere             anywhere            tcp spt:nfs
ACCEPT     udp  --  anywhere             anywhere            udp dpt:nfs
ACCEPT     tcp  --  anywhere             anywhere            tcp dpt:nfs
ACCEPT     udp  --  anywhere             anywhere            udp dpts:32700:33000
ACCEPT     tcp  --  anywhere             anywhere            tcp dpts:32700:33000
ACCEPT     udp  --  anywhere             anywhere            udp spts:32700:33000
ACCEPT     tcp  --  anywhere             anywhere            tcp spts:32700:33000
REJECT     all  --  anywhere             anywhere            reject-with icmp-port-unreachable

Chain FORWARD (policy DROP)
target     prot opt source               destination

Chain OUTPUT (policy DROP)
target     prot opt source               destination
ACCEPT     all  --  anywhere             anywhere
LOG        all  --  anywhere             anywhere            LOG level warning prefix
`iptables_output_stuff'
ACCEPT     tcp  --  anywhere             anywhere            tcp spts:0:1024
ACCEPT     udp  --  anywhere             anywhere            udp spts:0:1024
ACCEPT     tcp  --  anywhere             anywhere            tcp dpts:0:1024
ACCEPT     udp  --  anywhere             anywhere            udp dpts:0:1024
ACCEPT     tcp  --  anywhere             anywhere            tcp spt:nfs
ACCEPT     udp  --  anywhere             anywhere            udp spt:nfs
ACCEPT     tcp  --  anywhere             anywhere            tcp dpt:nfs
ACCEPT     udp  --  anywhere             anywhere            udp dpt:nfs
ACCEPT     tcp  --  anywhere             anywhere            tcp dpts:32700:33000
ACCEPT     udp  --  anywhere             anywhere            udp dpts:32700:33000
ACCEPT     tcp  --  anywhere             anywhere            tcp spts:32700:33000
ACCEPT     udp  --  anywhere             anywhere            udp spts:32700:33000
REJECT     all  --  anywhere             anywhere            reject-with icmp-port-unreachable
lnxtest#
lnxtest# /etc/init.d/ypbind start
Binding to the NIS domain:                                  [  OK  ]
Listening for an NIS domain server..
lnxtest#
lnxtest# tail -5 /var/log/messages
Nov 26 20:58:11 lnxtest kernel: iptables_input_stuffIN=eth0 OUT=
MAC=00:08:02:45:c3:aa:00:08:02:9e:6e:46:08:00 SRC=192.168.0.5 DST=192.168.0.14
LEN=120 TOS=0x00 PREC=0x00 TTL=64 ID=8914 DF PROTO=TCP SPT=1190 DPT=22
WINDOW=64120 RES=0x00 ACK PSH URGP=0
Nov 26 20:58:11 lnxtest kernel: iptables_output_stuffIN= OUT=eth0 SRC=192.168.0.14
DST=192.168.0.5 LEN=136 TOS=0x10 PREC=0x00 TTL=64 ID=18201 DF PROTO=TCP
SPT=22 DPT=1190 WINDOW=43800 RES=0x00 ACK PSH URGP=0
Nov 26 20:58:11 lnxtest kernel: iptables_input_stuffIN=eth0 OUT=
MAC=00:08:02:45:c3:aa:00:08:02:9e:6e:46:08:00 SRC=192.168.0.5 DST=192.168.0.14
LEN=120 TOS=0x00 PREC=0x00 TTL=64 ID=8961 DF PROTO=TCP SPT=1190 DPT=22
WINDOW=64096 RES=0x00 ACK PSH URGP=0
```

```
Nov 26 20:58:11 lnxtest kernel: iptables_output_stuffIN= OUT=eth0 SRC=192.168.0.14
DST=192.168.0.5 LEN=136 TOS=0x10 PREC=0x00 TTL=64 ID=18202 DF PROTO=TCP
SPT=22 DPT=1190 WINDOW=43800 RES=0x00 ACK PSH URGP=0
Nov 26 20:58:11 lnxtest kernel: iptables_input_stuffIN=eth0 OUT=
MAC=00:08:02:45:c3:aa:00:08:02:9e:6e:46:08:00 SRC=192.168.0.5 DST=192.168.0.14
LEN=120 TOS=0x00 PREC=0x00 TTL=64 ID=9014 DF PROTO=TCP SPT=1190 DPT=22
WINDOW=64072 RES=0x00 ACK PSH URGP=0
lnxtest#
lnxtest# rpcinfo -p
   program vers proto   port
    100000    2   tcp    111  portmapper
    100000    2   udp    111  portmapper
    100024    1   udp  32768  status
    100024    1   tcp  32768  status
    391002    2   tcp  32769  sgi_fam
    100007    2   udp    840  ypbind
    100007    1   udp    840  ypbind
    100007    2   tcp    843  ypbind
    100007    1   tcp    843  ypbind
lnxtest#
lnxtest# ypwhich
t64tst
lnxtest#
lnxtest# ypmatch testhosts netgroup
nustest lnxtest
lnxtest#
lnxtest# ps -ef|grep ypbind
root      6598     1  0 20:56 ?        00:00:00 ypbind
lnxtest#
```

Use of tcpwrappers requires entries in /etc/hosts.allow and /etc/hosts.deny. Keep in mind when configuring these files that the search stops as soon as a match is found. The allow file is checked first, then the deny file and if no match has been found, access is granted. Entries are in the following format: daemon_list : client_list [: shell_command]. For example:

ALL : @testhosts : spawn (echo `date` %c >> /var/log/access.log)

allows access from hosts in the netgroup testhosts, to execute any program controlled by tcp_wrappers and will log the access in the file /var/log/access.log. Combine this with a hosts.deny file containing ALL: ALL to disallow access to from any other host. One other advantage of using tcp_wrappers is the capability to display a banner before execution of the program.

One last note on RPC, is that security best practices are now suggesting it is a good idea to disable RPC altogether. Keep in mind that this is not possible in many clustered environments. Cluster implementations often use RPC services for intra-cluster communications, coordination and health monitoring. Be aware that network backup solutions, such as Legato Networker, may also use RPC.

Network Information System (NIS)

Network Information System (NIS) was originally developed to ease the burden of administration of host configuration files such as /etc/passwd, /etc/hosts and /etc/groups. Databases of information, built from flat text files and called maps, are maintained on a central server and the information is then made available to client systems via RPC calls. Redundancy can be achieved by utilizing slave servers. If the master server is unavailable, and the clients are properly configured, the requests will be answered by a slave server thus, eliminating the single point of failure.

One feature of NIS which actually introduces a security concern is that when enabled, NIS implicitly becomes part of system authentication. Support for the use of NIS is already built into standard Unix library routines. If NIS is enabled and an application makes a getpwnam( ) call, the local and NIS files will be searched for the username – no additional coding is required by the developer. It could lead to unauthorized access if key files are misconfigured or the implications misunderstood. One well known weakness of NIS is the lack of host authentication mechanisms while utilizing RPC. NIS uses the default RPC AUTH_NONE mode of security – which is just as it sounds, NONE. If inside the network, a machine controlled by a malicious user could be used to masquerade as an NIS server and send fake RPC replies to a client. If the NIS server and NIS domain name can be determined, ypcat commands can be used to freely view any of the NIS managed databases. Tools exist that are intended to guess the NIS domain name (search for ypx in the tools area of securityfocus.com). Additionally, information from all NIS maps is passed across the network unencrypted. Password hashes can be intercepted. Usernames and hostnames can be gleaned. If the usernames, passwords and hostnames can be gathered, using ypcat or by packet sniffing, it wouldn't be too hard to use a password cracker offline and come back later for a visit, appearing as a legitimate user. Password selection requirements cannot be enforced in our heterogeneous environment nor can shadow password files be used since we are forced to use the lowest common denominator. Unless enhanced security mode is enabled in Tru64, neither password criteria nor shadow files are supported. While policy enforcement tools that integrate with NIS exist, such as BMC Control-SA and TFS UnixControl, they usually rely on existing os mechanisms for feature availability.

Instructions for enabling NIS on our three operating systems can be found as follows:
Sparc Solaris 8
http://docs.sun.com/db/doc/806-1386/6jam5ahmj?q=nis+setup&a=view
Red Hat Linux 8
http://www.Red Hat.com/docs/manuals/linux/RHL-8.0-Manual/install-guide/s1-authconfig.html

Tru64 Unix 5.1A
http://h30097.www3.hp.com/docs/base_doc/DOCUMENTATION/V51A_HTML/ARPPCATE/TITLE.HTM (chapter 3) or
http://h30097.www3.hp.com/docs/base_doc/DOCUMENTATION/V51A_HTML/MAN/MAN7/0060____.HTM (man nis_manual_setup)

When installing NIS you will need to decide which server will be the NIS master. This system should have a non-descriptive name so it will be harder to deduce that it is an NIS server. Preferably, this server only has access to one network or minimally IP forwarding should be disabled. With IP forwarding disabled, the server will still be able to access more than one network, but will not act as a conduit between them. If the server is acting as a router, a rogue client may use ypset to gather information for multiple domains. Other considerations are availability and the expected resource load on the server. An NIS server that is slower or unavailable to respond allows more time for a masquerading server to respond to the client with bogus information. The master server should be locked down as much as possible and preferably not perform other duties such as acting as a web server. The number of slave servers depends on network performance, physical layout of the network and total number of clients. Minimally you would have at least one master and one slave so there would be redundancy in case of failure. If servers in an NIS domain are geographically dispersed and separated by network partitioning hardware, it would be wise to have a slave in each separate area. If server response time is degraded due to network performance or the load due to number of clients then adding another slave may be appropriate. The order in which clients choose a server can be controlled by configuration. In Tru64 use the NIS_ARGS parameter in the /etc/rc.config.common file. List the slave (or master) that is logically closest first in a comma separated list (following the –s and –S options).

Each flavor of Unix has a variety of NIS programs. Originally, NIS was called Yellow Pages but the name was changed to avoid trademark conflicts (Stern, 2001, pg ix). You will notice however, the commands retained the yp prefix. Common among them all are:
ypserv - NIS server process, must be running on an NIS master or slave server
ypbind - NIS binding process, must be running on an NIS client
ypwhich - determine current NIS server or master
ypcat - print values from an NIS database
ypmatch - print the value of one or more keys from an NIS database
ypset - point ypbind at a particular server
ypxfr - transfer an NIS map from the server to the local host
yppush - force propagation of a changed NIS map
yppasswd - change NIS password
yppasswdd - server daemon for changing NIS passwd file
domainname - show or set the current NIS domain

For this exercise the Tru64 server (t64tst) was chosen to be the NIS master and for simplicity there is no slave server. Also note, for readability, extensively randomized hostname and NIS domain name were not selected as is recommended for best practice. For Tru64 to function as the master, the Additional Network Services subset (OSFINET) must be installed. For NIS configuration, the sysman menu or manual setup can be used. See Appendix A for the nissetup session log. Note that both the –s and –S options were enabled. The –s option forces ypbind (the client program) to only listen to responses that originate from a privileged port - ports less than 1024. This means for an attacker to successfully spoof NIS by acting as ypserv responding with bogus replies to ypbind, the attacker must be root on a machine considered to be on the local network. This is similar to the modern portmapper functionality that requires service registration requests to come from a privileged port. Internal attacks may still be possible. Again, use host based filtering and access control to reduce the potential. The attacker would then not only have to be on the inside, he would have to know specific permitted hosts. The –S option locks ypbind to a particular domain and set of servers and provides some of that access control. Note that on the Tru64 server, the domainname and ypbind arguments are added to the rc.config.common startup configuration file as NIS_DOMAIN and NIS_ARGS respectively. Upon startup, if the /sbin/rc3.d/S##nis link exists, the nis script will check the rc.config variables and start the appropriate daemons. On t64tst we would see:

```
t64tst#  ps –eo cmd | grep ypbind
/usr/sbin/ypbind -s -S FrExmpl,t64tst
(Note: we specifically did a grep for ypbind, ypserv runs here also)

t64tst#  grep NIS /etc/rc.config.common
NIS_CONF="YES"
export NIS_CONF
NIS_TYPE="MASTER"
export NIS_TYPE
NIS_DOMAIN="FrExmpl"
export NIS_DOMAIN
NIS_SERVERARGS=""
export NIS_SERVERARGS
NIS_ARGS="-s -S FrExmpl t64tst "
export NIS_ARGS
```

For the Solaris client, if you have used the SANS Solaris Security Step by Step version 2.0 guide, you will need to reverse step 2.1.4 and renable RPC. Use ypinit –c to configure the client. If the NIS domain name has not been set it will exit and display “The local host’s domain name hasn’t been set. Please set it.” Type domainname FrExmpl. Also edit /etc/defaultdomain and add one line including FrExmpl so that the domainname will survive a reboot. Running ypinit –c again it will ask for server names to which it can bind. (An entry for the host server(s) must exist in /etc/hosts.) Enter t64tst and press return twice to exit. The /var/yp/binding/FrExmpl/ypservers file will now contain t64tst.

```
nustest# cat /var/yp/binding/FrExmpl/ypservers
t64tst
```

The /etc/init.d/rpc script, executed upon startup, calls /usr/lib/netsvc/yp/ypstart. Ypstart starts ypbind if the domain has been set and the ypbind executable exists. If the /var/yp/binding file exists for that domain then ypbind will start with no options. If the binding file does not exist it will be started with the –broadcast option. Using –broadcast will allow any responding server to act as our master which of course we do not want.

```
nustest# ps -ef|grep yp
   root   616     1  0 18:36:04 ?        0:00 /usr/lib/netsvc/yp/ypbind
```

To enable the NIS client on Red Hat Linux edit the /etc/yp.conf file. Enter domain NISdomainName server NISserverName.

```
lnxtest# cat /etc/yp.conf
# /etc/yp.conf - ypbind configuration file
# Valid entries are
#domain NISDOMAIN server HOSTNAME
#       Use server HOSTNAME for the domain NISDOMAIN.
#domain NISDOMAIN broadcast
#       Use  broadcast  on  the local net for domain NISDOMAIN
#ypserver HOSTNAME
#       Use server HOSTNAME for the  local  domain.  The
#       IP-address of server must be listed in /etc/hosts.
#
domain FrExmpl server t64tst
```

Multiple entries of this type can be added to accommodate the master and slaves. Again, do not use the broadcast option.

```
lnxtest# ps -ef|grep yp
root      6256     1  0 20:15 ?        00:00:00 ypbind
```

To survive a reboot, in the /etc/sysconfig/network file add the line NISDOMAIN=FrExmpl. The /etc/yp.conf file is read by /etc/init.d/ypbind at startup. Unfortunately, a specific port cannot be specified for ypbind (as it can for ypserv and ypxfr) so, you will not be able to use the automatic settings of medium or high for system security level – the security level will need to be set to none and will require manual configuration. The ability to specify port for the server daemons may be another consideration when choosing an NIS master.

None of the ypbind processes should be configured with –ypset or –ypsetme (you can verify via the ps command output). These options allow ypset commands to be used by remote and local root users, respectively, to change the binding for a domain. No need for the attacker to bother with spoofing, just needs a machine with root access and the NIS domain is no longer owned by you!

Be sure to create /etc/yp/securenets file on the NIS servers so that ypserv and ypxfrd will only respond to requests from hosts that are "trusted". While this does not completely eliminate spoofing possibilities, it does place limits on which networks NIS will service. All of the operating systems in use in this demo network support the securenets file. While Tru64 looks for the file in /etc/yp, Red Hat Linux and Solaris look for the file in /var/yp.

```
t64tst# cat /var/yp/securenets
255.255.255.0  192.168.0.0
```

This indicates that only machines on the 192.168.0 subnet are allowed to access the NIS data via the ypserv and ypxfrd programs.

We will need to modify the database service selection configuration file. On Tru64 this file is known as /etc/svc.conf. On Red Hat Linux and Solaris, it is called /etc/nsswitch.conf. For each map that will be served from NIS, the corresponding line must include yp (for Tru64) or nis (Red Hat Linux or Solaris). The following shows hosts, netgroups and the password file enabled:

```
t64tst# grep yp /etc/svc.conf | grep –v "#"
hosts=local,yp,bind
netgroup=local,yp
passwd=local,yp

lnxtest# grep nis /etc/nsswitch.conf | grep –v "#"
passwd:       files nis
hosts:        files nis dns
netgroup:     files nis
automount:  files nis

lnxtest# grep nis /etc/nsswitch.conf |grep –v "#"
passwd:       files nis
hosts:        files nis dns
netgroup:     files nis
automount:  files nis
```

Keeping map files updated on all servers (master and slave) is essential to the proper operation of NIS. Map information should only be modified on an NIS master. A code management system could be used for tracking and to control access when multiple system administrators are involved. After modifications are made to files in the source directory, /var/yp/src, a make command should be issued in the /var/yp directory. If the make command is issued without any parameters all maps that are out of date will be built and transferred, to all slave servers identified in the ypservers map, using the yppush command. If a map needs to be manually transferred, the ypxfr command can be used. The ypxfr command should be run out of cron on the slave servers to keep maps synchronized even if a previous push was missed (say the slave was down for some reason). Ypxfr only pulls a copy of the map if it has been updated.

Netgroup maps can be used to create sets of users and hosts. For example, say there are five developers working on the Sales project that need access to servers dev1, dev2 and dev3. These users and systems can be grouped together and referenced as SalesDev and SalesServ in configuration files (such as /etc/passwd and /etc/exports) as an entity instead of adding each individual user or host. This simplifies configuration files by reducing the number of entries required and eases maintenance by restricting modification to the entity (say one developer is no longer on the project but all other access needs remain) on the master server only. Netgroups reduce the opportunity for configuration errors since it reduces the total number of modifications required. Netgroups allow you to view the needs of a group as a whole and group membership is readily available. Note that netgroups are created here for each individual host and each user. This is for readability – testhosts is much easier to read without all the parentheses and extra field information. This provides a little more obscurity also. For example, if mmar was added to the testusers group in the same manner bmar is added, it would not become a valid account until added as a separate netgroup (or the full notation included).

----------Sample Netgroup File

```
t64tst# cat /var/yp/src/netgroup
# ----- hosts ------
testhosts nustest lnxtest
#
nustest (nustest,-,FrExmpl)
lnxtest (lnxtest,-,FrExmpl)
#
# ----- users ------
testusers bmar
#
bmar (-,bmar,-)
```

----------Adding The Netgroup to the Passwd File

```
lnxtest# grep + /etc/passwd
+@testusers
lnxtest# grep bmar /etc/passwd
lnxtest#
```

----------Accessing the Server via the Netgroup User

```
nustest# ssh bmar@lnxtest
This is a private system.  You are not authorized to be here.  All activity on this system is
monitored and recorded.  Using this system implies your consent to such monitoring.

bmar@lnxtest's password:
-sh-2.05b$
```

Now for a discussion of recent vulnerabilities and exploits.

Red Hat Linux 8 security alert RHSA-2003:173-07 describes a vulnerability which would result in a denial of service attack if exploited. If a client queries the ypserv daemon and then ignores the reply, ypserv will continue trying to reply to the client. In the meantime, other queries will not get serviced. The only way

around this one is to get the vendor supplied upgrade version installed. The srpm package provides a new ypserv daemon that forks a new child for every client request. This vulnerability was released around July of 2003 and is referenced in the CVE database as CAN-2003-0251. Red Hat Linux 8 by default comes with package ypserv2.5-1. The vulnerability affects all ypserv versions prior to version 2.7. The advisory has links to version 2.8. To see if you are vulnerable use rpm –q ypserv.

CERT Vulnerability Note VU #538033 describes an NIS vulnerability, made public in October 2002, that affected both Tru64 Unix 5.1A and Solaris 8. SSRT2339 and SSRT2368 from Hewlett Packard and Sun Alert ID 47903, respectively. This vulnerability affected both the ypxfrd and ypserv daemons. Basically, when a request is made to these daemons the passed variables, domain and map, are not checked for slashes or dots. Symbolic links could then be used to point to any file on the system. This attack could be done locally or remotely if the securenets file is not configured. The solution, in addition to always configuring securenets, is once again to apply the vendor patch. See also, CVE CAN-2002-1199. You are vulnerable if you are running Tru64 version 5.1A patch kit 3 to which the early release patch kit can be applied. The fix is included in mainstream patch kit 4. Solaris 8 is vulnerable if patch kit 109328-03 has not been applied.

In June of 2001, Sun released Sparc Solaris 8 security alert 27488 that describes a potential buffer overflow condition in which ypbind could be killed. In April 2002, HP released alert SSRT0781U affecting Tru64 version 5.1a, which indicates ypbind would core dump during an NMAP scan. Both of these vulnerabilities would result in a denial of service and both would be eliminated if the appropriate vendor patch was applied.

To reduce these risks it is imperative that the NIS domain be behind an efficiently configured firewall on a segmented network. The firewall should block packets bound for port 111 (rpc) if coming from outside the intranet. Apply a filter that prevents packets coming from outside your network that have a source address that should only be found inside the firewall. Host based filtering and access control adds to defense in depth. Logging allows systems administrators to become familiar with typical traffic so that potentially malicious activity is more recognizable. Periodically check the passwd map for null passwords. Restrict access to the NIS master both physically and logically. To reduce risks introduced by coding deficiencies, such as buffer overflows that lead to unauthorized host access, best practices for securing NIS include keeping patches up-to-date for all NIS related executables.

Network File System (NFS)

“The single biggest advantage of NIS is that it adds consistency to a network. Getting all hosts to agree on usernames, uid and gid values and hostnames and

host addresses is a prerequisite for adding other distributed services such as NFS" (Stern, 1991). Network File System (NFS) utilizes RPC in a client server relationship to allow a user to access remote filesystems transparently as if they were local.

Dependent on the implementation, in addition to the portmapper, the daemons involved in NFS communication are nfsd, mountd, lockd, statd and sometimes nfsiod. Nfsd accepts the NFS RPC requests and executes them on the server. Nfsd is an RPC server that usually has the port number hard coded in the application. Nfsd almost always listens on port 2049 tcp and udp. Mountd handles client mount requests. Lockd is the network lock daemon. Statd monitors the status of the client and server in response to lockd requests. Statd is used for lock recovery in the event of communication failure. Nfsiod, if present, is used for block I/O operations, performing some simple performance optimizations. Biod is not used in Solaris; performance is optimized via the tunable number of threads in the kernel. Red Hat does not use biod. The Tru64 implemenation does use nfsiod.

When a client requests the mount of an NFS filesystem the following occurs:

1. The client mount request issues an RPC call to the portmapper on the server to obtain the server's mount daemon port.
2. The server portmapper replies with the mount daemon port.
3. The client mount command issues an RPC call to the server mount daemon. The server validates the client, using the client's IP address and port number, to see if the server lets this client mount the specified filesystem.
4. Once validated, the server replies with the file handle for root of the given filesystem.
5. The server mount associates the filesystem filehandle with a local mount point on the client.
6. Further references by the client are via filesystem filehandle info.

It is important to note here that the validation occurs only at mount time. The filesystem filehandle is used for subsequent access. NFS LOOKUP requests for files and directories under the root of this mount use three pieces of information: a filesystem id (the mount returned filehandle), a file id (inode) and an inode generation count to create a unique filehandle. Filehandles mean nothing to the client; they are translated by the server only. An attacker can guess a filehandle fairly easily. Packets could then be crafted using the known filehandle to access or replace any file on the exported filesystem bypassing the server access controls (which are only validated at mount time).

One of the primary known weaknesses of NFS is that the default RPC implementation of NFS uses the AUTH_SYS security mode which is very weak. By spoofing a valid uid/gid pair in a crafted packet, information on exported filesystems could be viewed by someone that should not have access. To

enable NFS simply ensure RPC is available, start NFS related daemons, export the filesystems from the server and have the client issue the mount request.

Exporting the filesystems can be accomplished via a static file on each server or by utilizing NIS maps. The static file on Tru64 and Red Hat is /etc/exports. On Solaris is it called /etc/dfs/dfstab. On Red Hat the format of the exports file is as follows: export_point allowed_client(export_option[, export_option])
Multiple clients may be listed separated by white space. Export options must immediately follow the client, no white space is allowed. Allowed client can be a hostname, IP address or NIS netgroup. The \ backslash character can be used for line continuation and the # pound sign is recognized as a comment.

The automounter is a tool that mounts NFS file systems when access is requested and unmounts them after a period of inactivity. NIS can be used to maintain mount information hence be centrally managed as opposed to /etc/fstab (or vfstab) files on each server. This streamlines administration and reduces the opportunity for error. Unmounting file systems when not in use reduces visibility to the bad guy and reduces exposure to the dreaded hung NFS mount after an NFS server crash.

Of course, whether the filesystems are exported via export files on each host or via NIS maps, hard mounted or via the automounter, it is critical that best practices followed to limit unwanted availability of information through misconfiguration. The following guidelines are recommended:

- Exported files should be owned by root with no group write. By default attempts to write as root on an NFS filesystem get mapped to the nobody user. Ensuring root ownership will prevent inadvertent access to files owned by user nobody.
- To tighten access control a bit specify the hosts that are allowed to mount exported filesystems. For convenience, consistency and some obscurity, netgroups can be used. Always keep in mind that DNS can be poisoned and spoofed. List DNS last in the nsswitch or svc.conf files so that local host files and NIS are searched first.
- Export read only whenever possible.
- Disallow setuid execution to prevent unauthorized root access.
- Ignore requests that originate from ports higher than 1024 so at least root access is required to initiate a service request.
- NFS mounting home directories can be dangerous because it is possible for a malicious user with root on a local machine to place an .rhosts file on the NFS mount to gain access. Hopefully though you have the r-commands disabled (by removing from inetd.conf). Ssh authorized keys could be modified in the same manner. Sometimes the advantages of having NFS home directories are tough to give up so risk vs. gain will have to be assessed.
- Exporting binaries presents a couple issues: knowing versions of binaries makes searching for vulnerabilities easier and a configuration error could

lead to write access. Again, sometimes the advantages outweigh the risks. If you determine this is the case, triple check access bits and ownership and review them periodically.

- If there is a nodev option for the server to prevent character and block special devices from being interpreted it should be used. There is lots of interesting information stored in kernel structures that could be accessed via rogue device files.

Instructions for enabling NFS on our three operating systems can be found at:
Sparc Solaris 8
http://docs.sun.com/db/doc/806-0916/6ja8539fj?a=view
Red Hat Linux 8
http://www.Red Hat.com/docs/manuals/linux/RHL-8.0-Manual/custom-guide/ch-nfs.html or
http://www.tldp.org/HOWTO/NFS-HOWTO/index.html
Tru64 Unix 5.1B
http://h30097.www3.hp.com/docs/base_doc/DOCUMENTATION/V51B_HTML/ARPPCBTE/TITLE.HTM (Chapter 4)

The Red Hat server in our demo was chosen as the NFS server. From the Gnome menu select Server Settings and then NFS Server. After configuring the mountpoints to export, it will ask if you want to start NFS services. This process populates /etc/exports. To survive reboot, be sure to use chkconfig and look for NFS services. Configure the NFS server to only accept client requests that occur on privileged ports so that an attacker would have to at least have remote root access to attack. In our example on Red Hat Linux in Appendix B, this is the default. To configure improperly, under the General Options tab, you would have selected “allow connections from ports 1024 and higher”. Verify that you have not selected this option by looking for the insecure option in /etc/exports. Other Red Hat defaults when adding exported filesystems via the gui are read only, do not treat remote root users as root and do not treat all clients as anonymous.

```
lnxtest# showmount -e
Export list for lnxtest:
/home/bmar @testhosts
lnxtest#
lnxtest# ypmatch testhosts netgroup
t64tst nustest lnxtest
lnxtest#

t64tst# mount lnxtest:/home/bmar /test
t64tst#
t64tst# df
Filesystem          512-blocks      Used  Available Capacity  Mounted on
root_domain#root        550352    185102     353504    35%    /
/proc                        0         0          0   100%    /proc
usr_domain#usr         3009760    647142    2328736    22%    /usr
usr_domain#var         3009760     11220    2328736     1%    /var
lnxtest:/home/bmar    19860808     66768   18785168     1%    /test
```

```
t64tst#
```

The NFS server daemons, mountd and nfsd will start automatically at boot on a Solaris system if there are entries in the /etc/dfs/sharetab file that are labeled NFS. The client daemons, lockd and statd (must also run on an NFS server), automatically start at boot if found.

On Tru64, NFS can be configured manually or via sysman. When using sysman –menu you would select Networking, Additional Network Services, Network File System and in this case, Configure System as an NFS Client. The default number of client I/O threads is 7 and locking is enabled. The box must be checked to enable the automount daemon. Once checked, the default options are –m /net –hosts –nosuid. The –m option ignores directory-mapname pairs listed in the auto.master NIS map. The /net –hosts parameter is relative to NIS map handling which we have not described in our demo. The –nosuid disallows setuid execution. Possibly –v to log status messages to the console should be considered to understand default behavior. Once the configuration is saved, the following parameters are updated in /etc/rc.config.common:

```
t64tst# grep NFS rc.config.common
NFS_CONFIGURED="1"
export NFS_CONFIGURED
NUM_NFSIOD="7"
export NUM_NFSIOD
NFSLOCKING="1"
export NFSLOCKING
t64tst#
```

The client processes can be started when exiting the sysman menu or by executing /sbin/init.d/nfs start and /sbin/init.d/nfsmount start.

```
t64tst# rpcinfo -p|grep -v yp
   program vers proto   port
    100000    2   tcp    111  portmapper
    100000    2   udp    111  portmapper
    100011    1   udp   1030  rquotad
    100024    1   udp   1034  status
    100024    1   tcp   1176  status
    100021    1   tcp   1177  nlockmgr
    100021    2   tcp   1177  nlockmgr
    100021    3   tcp   1177  nlockmgr
    100021    4   tcp   1177  nlockmgr
    100020    3   tcp   1177  llockmgr
    100021    1   udp   1035  nlockmgr
    100021    2   udp   1035  nlockmgr
    100021    3   udp   1035  nlockmgr
    100021    4   udp   1035  nlockmgr
    100020    3   udp   1035  llockmgr
t64tst#
```

Discussion of recent vulnerabilities and exploits:

Sun released Sparc Solaris 8 security alert 47815 in October of 2002. This vulnerability may lead to an NFS denial of service attack by killing lockd. With lockd down, any NFS request that requires file locking will not be serviced. The specific method by which lockd is killed was not disclosed. A patch was released by the vendor to correct the problem.

In July of 2003, Red Hat announced an nfs-utils bug as security advisory RHSA-2003:206-08. The vulnerability is described by the discoverer at http://isec.pl/vulnerabilities/isec-0010-linux-nfs-utils.txt as follows:

> An off-by-one bug exist in xlog() function which handles logging of requests. An overflow occurs when function is trying to add missing trailing newline character to logged string.
>
> Due to miscalculation, if a string passed to the functions is equal or longer than 1023 bytes, the '\0' byte will be written beyond the buffer: [code excerpt not included here]
>
> Local or remote attacker which is capable to [sic] send RPC request to vulnerable mountd daemon could execute arbitrary code or cause denial of service.

Installing an updated version of the nfs-utils package corrects this problem.

Tru64 Unix 5.1B SSRT3533 from Hewlett Packard was released in April 2003. While specific details are a bit sketchy, Security Focus discussion, http://www.securityfocus.com/bid/7400/discussion/, gives the following discussion: A denial of service vulnerability has been discovered in the HP TruCluster server. The problem appears to lie in the way Cluster Alias/NFS services included with TruCluster systems handle malicious network traffic. Successful exploitation of this issue may allow a remote attacker to crash a vulnerable server. Again, to eliminate the vulnerability, apply a patch.

Best practices for securing NFS, of course, start with keeping patches up-to-date not only for the replacement rpc related daemons but the nfsd, mountd, statd, lockd, biod and automountd daemons also. To prevent attack from outside your networked world, block the ports on the firewall where the NFS server is listening (likely port 2049, both UDP and TCP, but use rpcinfo –p to verify). The /etc/exports and netgroups files should be audited frequently to validate which file systems are exported and which users have access. Ownership and access masks of exported directories and files should be reviewed periodically, possibly through a host based vulnerability assessment tool. Follow the best practices guidelines listed above for exporting files. The fsirand utility can be run on exported filesystems on a recurring basis to install random inode numbers. (Note filesystems must be unmounted to use fsirand.) This will reduce the likelihood that the attacker will correctly deduce the filehandle. Lastly, there is a tool

available at http://ftp.cerias.purdue.edu/pub/tools/unix/sysutils/nfsbug/ called nfsbug that will test for well known NFS bugs and problems. It is a tool worth checking out.

Conclusion

Although security issues are acknowledged, when faced with managing a heterogeneous environment NFS and NIS remain reasonable solutions. This is truer in shops where these tools are already pervasive. With security budget dollars limited, it would be difficult to justify the cost of retro-fitting alternate solutions, especially when the prevailing competitors have not been standardized, simplified and integrated into the majority of Unix operating systems. The preceding statements become even more applicable the larger the network. In shops where higher risk factors can be tolerated, because of lower levels of data confidentiality and criticality, NIS and NFS make even more sense. Lastly, once the precautions outlined here have been followed, confidence should be further increased. Consider that proactive patch application or package upgrade would have eliminated most of the exposure created by the recent vulnerabilities discussed. Once measures have been taken to prevent spoofed traffic from entering through the firewall and known ports that should not be accessed from the outside are blocked, the threat is reduced to primarily from within. After layering host based filtering and logging on top of diligent configuration of access control mechanisms, available via NIS and NFS, the risk is reduced to a malicious user with root access on the local network or a disgruntled insider. Reviewing log files and becoming familiar with normal activity and system performance will help a systems administrator recognize a compromise or maybe even an attack in progress. All that said, it is still recommended that progress of emerging and maturing technologies be monitored and migration plans be developed when appropriate for your shop. As the workforce becomes more mobile and more devices become network enabled, it will continue to become more difficult to define network perimeters. LDAP could very likely become the preferred replacement for NIS once standard implementations are agreed upon, the mechanisms tightly coupled with most operating systems and embraced in application development efforts. NFS version 4 is especially of interest given that in 1998, Sun Microsystems and the Internet Society completed an agreement giving the Internet Society control over future versions of NFS. The goals of the working group include mandatory common security and better heterogeneity (Stern, p. 144-145).

Appendix A – Tru64 Unix Sysman NIS Setup Session

```
Starting /usr/sbin/nissetup  ...

/usr/sbin/nissetup  run at Mon Oct 20 14:56:15 EDT 2003


**********************************************************************
*                                                                    *
*            Network Information Service (NIS) Setup          *
*                                                                    *
**********************************************************************

NOTE:  Please be sure the network is set up and running before you
       set up NIS.

       Also, if you are going to set up an NIS server, be sure you
       have installed the "Additional Networking Services" subset.

Enter "e" to EXIT or "c" to CONTINUE [no default]: c

Nissetup (formerly ypsetup) configures and runs the Network
Information Service (NIS, formerly YP) for your system.

NIS provides a distributed data lookup service for sharing data
among networked systems.  NIS data is stored in database files
called maps.  A domain is a group of systems which share a
common set of maps.

A NIS domain is NOT the same as a BIND domain.  If you are
unsure of your NIS domain, exit this script.  Setting an
incorrect domain name renders a system virtually unusable.

[ Press RETURN to continue ] :


For each domain, there are three types of systems:

        - a MASTER SERVER maintains the master copy of the
          domain's maps.
          There should be ONLY ONE master server per domain.

        - a SLAVE SERVER periodically receives updated versions
          of the domain's maps from the master.  A slave server
          can retrieve data from its private collection of maps,
          and can take over for the master server if the master
          server fails.

        - a CLIENT retrieves NIS data by requesting service from
          an NIS server.  A client does not have local copies of
          the domain's maps.

NOTE: To set up your system as a MASTER or SLAVE SERVER, you must have
      installed the "Additional Networking Services" subset.
```

[Press RETURN to continue] :

Default answers are shown in square brackets ([]).
Press RETURN to use a default answer.

What is t64tst's NIS domain name [] ? FrExmpl

Is FrExmpl the correct domain name (y/n) [no default]? y

Will t64tst be a

1. MASTER server,
2. SLAVE server, or
3. CLIENT ?

1 2 or 3 [3] ? 1

MASTER Server Setup

There can be ONLY ONE master server for each domain. If a master NIS server is already configured for domain FrExmpl please exit nissetup now.

Enter "e" to EXIT or "c" to CONTINUE [no default]: c

As a MASTER NIS server, you may run the yppasswdd daemon to allow remote password updates to the master copy of the passwd file.

Would you like to run the yppasswdd daemon [y]? y

Do you intend to use enhanced security with NIS [n]? n

If you serve very large maps, you may wish to have NIS maintain these as "btree" files instead of the default "dbm/ndbm"-style files. This will significantly reduce the time required to build and push very large maps. It may degrade performance slightly for relatively small maps.

Would you like maps to be maintained as "btree" files [n]? n

As a MASTER NIS server, this system maintains a list of slave NIS servers in the FrExmpl domain. This list is used to update the slave servers whenever a map is modified.

Enter the names of the SLAVE servers in the FrExmpl domain.
Press RETURN to terminate the list.

Hostname of slave server:

The list of NIS servers (including the master) for domain FrExmpl is:
t64tst

You may now redo this list, exit nissetup, or continue.

```
If you choose to continue, the default set of NIS maps for your host
will now be created.  Please be sure that the files that you wish
to distribute are in the /var/yp/src directory.

Enter "r" to REDO the servers list, "e" to EXIT nissetup,
or "c" to CONTINUE [no default]: c

Creating default NIS maps.  Please wait...
couldn't find /var/yp/src/passwd
couldn't find /var/yp/src/group
couldn't find /var/yp/src/hosts
couldn't find /var/yp/src/networks
couldn't find /var/yp/src/rpc
couldn't find /var/yp/src/services
couldn't find /var/yp/src/protocols
couldn't find /var/yp/src/netgroup
couldn't find /var/yp/src/mail.aliases
sh: test: unknown operator ENHANCED
*** Exit 1 (ignored)
Finished creating default NIS maps.
```

Ypbind option: -s

The -s security option will allow ypbind to run in secure mode. This requires the server be using a reserved port.

Would you like to use the -s security option (y/n) [y] ? y

NOTE: The default answers are RECOMMENDED.

Ypbind option: -S

The -S security option locks the domain name and an authorized list of NIS servers. With the -S option, your system will not change domains or bind to an unauthorized NIS server.

If your network uses SUBNETS, you must use the -S option in order to bind to any servers which are not on the local subnet.

Would you like to use the -S security option (y/n) [y] ? y

Enter the authorized servers. You may enter at most four servers; specifying three or four servers is recommended. Press RETURN to terminate the list.

Server 1 name: t64tst
(An NIS server must specify itself FIRST)
Server 2 name:

The list of authorized NIS servers is:
t64tst

```
Enter "r" to REDO the authorized servers list, "e" to EXIT nissetup,
or "c" to CONTINUE [no default]: c

                Ypbind options: -ypset and -ypsetme
                -----------------------------------

        The ypset command allows local or remote users to set your
        system's NIS server to a particular host.

        Would you like to:

                1. ALLOW LOCAL ypset requests only (-ypsetme),
                2. ALLOW ALL ypset requests (-ypset), or
                3. DISALLOW ALL ypset requests (RECOMMENDED) ?

        NOTE: If you choose 1 or 2, ypset requests will be limited to
             the authorized servers which you selected above.

        Enter your choice (1-3) [3]: 3

        t64tst will DISALLOW ALL ypset requests.

        Is this correct (y/n) [no default]? y

Nissetup will configure your system to use all of the NIS databases served
by your NIS master server.  If you choose not to have your system configured
this way, nissetup will help you to customize your configuration.

Would you like nissetup to configure your system to use all of the NIS
databases served by your NIS server [y]? n

To use NIS for passwd information, a "+:" must be appended to the
/etc/passwd file on t64tst.

Would you like to append a "+:" to the /etc/passwd file (y/n) [y]? n

To use NIS for group information, a "+:" must be appended to the
/etc/group file on t64tst.

Would you like to append a "+:" to the /etc/group file (y/n) [y]? n

To use NIS for any other databases, the /etc/svc.conf file must reference "yp".

Would you like to run the svcsetup script to edit /etc/svc.conf (y/n) [y]? y

*****************************************************************
*                                                               *
*         Svcsetup:  Modify the /etc/svc.conf file              *
*                                                               *
*****************************************************************


Svcsetup modifies the /etc/svc.conf file, allowing only valid
entries.

Your system uses the /etc/svc.conf file to locate system and
```

network information. For each type of information available, the file contains an entry which tells the system where to find that information.

For example, the /etc/services file may exist locally on your system and be served by the Network Information Service (NIS, formerly YP). If the /etc/svc.conf file contains the line "services=local,yp", your system will search the local /etc/services file for Internet service information and then query NIS if it could not find the information locally.

[Press the RETURN key to continue]:

For all databases EXCEPT the hosts database, the two choices for finding the information are "local" and "local,yp". Use "local" if you do not want your system to use NIS for a particular database, or if you are not using NIS at all. Use "local,yp" if you want your system to query NIS for information that is not found locally.

For the hosts database, eight additional settings are possible which allow the system to query the Berkeley Internet Name Domain (BIND) service for host information.

The passwd and group entries exist only for compatibility. To use NIS for passwd information or group information, a "+:" must be added to the end of the /etc/passwd or /etc/group file, respectively.

NOTE that "yp" corresponds to NIS which was formerly called "yellow pages".

Changes to /etc/svc.conf take effect immediately.

[Press the RETURN key to continue]:

Configuration Menu for the /etc/svc.conf file

Modify File => m
Print File => p
Exit => e

Enter your choice [m]: m

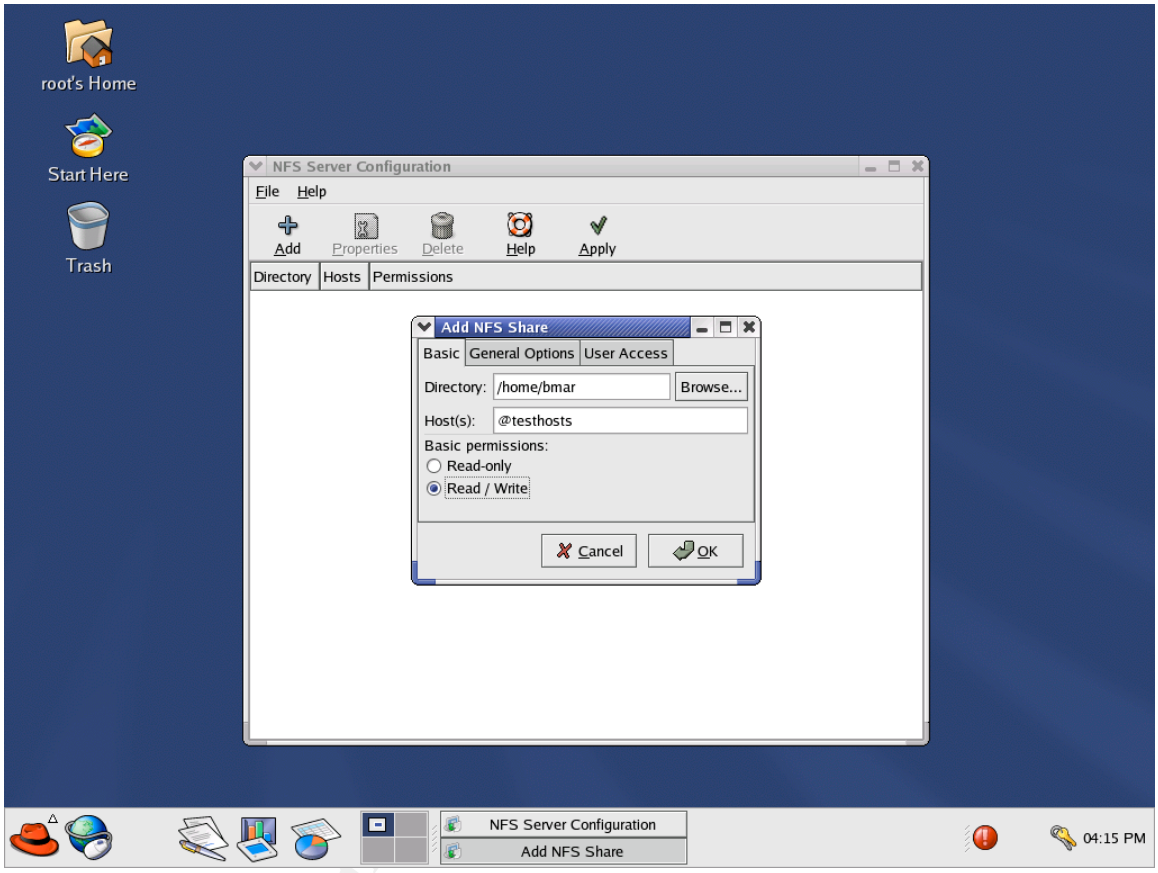
Change Menu for the /etc/svc.conf file

aliases => 0
group => 1
hosts => 2
netgroup => 3
networks => 4
passwd => 5
protocols => 6
rpc => 7
services => 8

```
        ALL of the above  => 9
        NONE of the above => 10

Enter your choice(s).  For example "0 3 5" [no default] : 2

*********************************
*  Name Service Order Selection *
*********************************

        local                => 1
        local,yp          => 2
        local,bind           => 3   (hosts database only)
        bind,local           => 4   (hosts database only)
        local,bind,yp      => 5   (hosts database only)
        bind,local,yp      => 6   (hosts database only)
        local,yp,bind      => 7   (hosts database only)
        bind,yp,local      => 8   (hosts database only)
        yp,local,bind      => 9   (hosts database only)
        yp,bind,local      => 10  (hosts database only)

Enter the name service order for each of the following databases:

"hosts" database [2]: 7

Updating file:
        /etc/svc.conf

***** SVCSETUP COMPLETE *****

Configuring your system to run NIS...
Updating files:
  /etc/rc.config.common

Would you like to start the NIS daemons now (y/n) [y]? n

You may start the NIS daemons by typing "/sbin/init.d/nis start",
or you can allow the new NIS configuration to take effect on the
next reboot.
 ***** NISSETUP COMPLETE *****

** /usr/sbin/nissetup exited, press 'OK' to close window **
```

Appendix B – Screen Shot Red Hat Linux NFS Setup

From the Gnome menu select Server Settings and then NFS Server. After configuring the mountpoints to export, it will ask if you want to start NFS services. This process populates /etc/exports. To survive reboot, be sure to use chkconfig and look for NFS services.



References

Barkley, John. “Comparing Remote Procedure Calls.” NISTIR 5277. October 1994.
URL: http://hissa.nist.gov/rbac/5277/titlerpc.html (Aug 2003)

Barkley, John, et al. “Security in Open Systems”. National Institute of Standards and Technology. Special Publication 800-7. July 1994.
URL: http://csrc.nist.gov/publications/nistpubs/800-7/main.html (Oct 1994)

CERT Advisory. CA-1994-15 NFS Vulnerabilities. Carnegie Mellon. 23 Sep 1997.
URL: http://www.cert.org/advisories/CA-1994-15.html (Aug 2003)

FacetCorp. “NFS and CIFS (SMB).” Technology Comparison.
URL: http://www.facetcorp.com/competition_nfs_cifs_comparison.html (Aug 2003)

Garfinkel, Simson, et al. Practical Unix & Internet Security, Third Edition. Sebastopol: O’Reilly & Associates, Inc., 2003.

Hess, David, et al. “A Unix Network Protocol Security Study: Network Information Service”.
URL: http://www.deter.com/unix/papers/nis_paper.pdf (Nov 2003)

Hewlett Packard Online Documentation. Administration Guide. Internet Express Version 6.1.
URL:
http://h30097.www3.hp.com/docs/iass/OSIS_61/documents/admin/TITLE.HTM (Aug 2003)

Hughes, Doug. Securing NIS (formerly YP). Auburn University, College of Engineering.
URL: http://www.eng.auburn.edu/users/doug/nis.html (Aug 2003)

IBM Corporation. AFS Administration Guide. 2000.
URL: http://www-3.ibm.com/software/stormgmt/afs/manuals/Library/unix/en_US/HTML/AdminGd/auagd007.htm#HDRWQ87 (Nov 2003)

Komar, Brian, et al. Firewalls for Dummies, Second Edition. Indianapolis: Wiley Publishing, 2003.

Kukuk, Thorsten. The Linux NIS(YP)/NYS/NIS HOWTO. The Linux Documentation Project. 1 Jul 2003

URL: http://www.tldp.org/HOWTO/NIS-HOWTO/which.html#AEN160 (Aug 2003)

Litt, Steve. IPTables. Linux Productivity Magazine. Volume 2 Issue 5, May 2003.
URL: http://www.troubleshooters.com/lpm/200305/200305.htm
Mauney, Jon. DCE Frequently Asked Questions. The Open Group, 12 Jul 2001
URL: http://www.opengroup.org/dce/info/faq-mauney.html (Aug 2003)

Mani, Greg. Summary: NIS+ and LDAP – Single sign on. Mailing list message thread. 16 May 2003.
URL: http://www.darklab.net/resources/sunman/10371.html (Aug 2003)

Open Group. Protocols for Internetworking: XNFS, Version 3W. Document number C702. The Open Group, 1998.
http://www.opengroup.org/onlinepubs/009629799/toc.htm

Pomeranz, Hal. Common Issues and Vulnerabilties in Unix Security. SANS Institute, 2003.

Red Hat Linux Online Documentation. Server Security (Chapter 5). Red Hat Linux 9: Red Hat Linux Security Guide
URL: http://www.Red Hat.com/docs/manuals/linux/RHL-9-Manual/security-guide/s1-server-nis.html (Aug 2003)

Red Hat Linux Online Documentation. Lightweight Directory Access Protocol (LDAP). Red Hat Linux 8.0: The Official Red Hat Linux Reference Guide.
URL: http://www.Red Hat.com/docs/manuals/linux/RHL-8.0-Manual/ref-guide/s1-ldap-daemonsutils.html (Aug 2003)

Reece, David P. "Is blocking port 111 sufficient to protect your systems from RPC attacks?" SANS Intrusion Detection FAQ. 26 Feb 2000.
URL: http://www.sans.org/resources/idfaq/blocking.php (Aug 2003)

Requests for Comments (RFC) URL: ftp://ftp.rfc-editor.org/in-notes/
rfc793.txt Transmission Control Protocol
rfc1831.txt RPC: Remote Procedure Call Protocol Specification Version 2
rfc2246.txt The TLS Protocol Version 1.0
rfc2829.txt Authentication Methods for LDAP
rfc2695.txt Authentication Mechanisms for ONC RPC
rfc2623.txt NFS Version 2 and Version 3 Security Issues and the NFS Protocol's Use of RPCSEC_GSS and Kerberos V5
rfc3377.txt Lightweight Directory Access Protocol (v3): Technical Specification

Scambray, Joel, et al. Hacking Exposed, Second Edition. Berkeley: Osborne/McGraw-Hill, 2001.

Stevens, W. Richard. TCP/IP Illustrated, Volume 1. Reading: Addison Wesley Longman, Inc., 1994.

Stern, Hal, et al. Managing NFS and NIS, Second Edition. Sebastopol: O'Reilly & Associates, Inc., 2001.

Stodte, Maya. "Opening up AFS". IBM developerWorks. 1 Sep 2000. URL: http://www-106.ibm.com/developerworks/opensource/library/os-afs.html (Nov 2003)

Sun Microsystems EOF. NIS+ End-of-Feature Announcement FAQ. Date unknown. URL: http://wwws.sun.com/software/solaris/faqs/nisplus.html (Aug 2003)

Sun Microsystems Online Documentation. System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP). Solaris 9 4/03 System Administrator Collection. URL: http://docs.sun.com/db/doc/816-7511/6mdgu0gu2?a=view (Aug 2003)

Van Meer, Roel, et al. LDAP Implementation HOWTO. The Linux Documentation Project. 30 Mar 2001 URL: http://www.tldp.org/HOWTO/LDAP-Implementation-HOWTO/pamnss.html (Aug 2003)

Westphal, Kristy. "NFS and NIS Security." INFOCUS. 22 Jan 2001. URL: http://www.securityfocus.com/infocus/1387 (Aug 2003)

A wide assortment of man pages on all Unix variants discussed in this paper.

Last Updated: April 26th, 2018

Upcoming Training

SANSFIRE 2018	**Washington, DC**	**Jul 14, 2018 - Jul 21, 2018**	**Live Event**
SANS Network Security 2018	**Las Vegas, NV**	**Sep 23, 2018 - Sep 30, 2018**	**Live Event**
SANS London October 2018	**London, United Kingdom**	**Oct 15, 2018 - Oct 20, 2018**	**Live Event**
SANS OnDemand	**Online**	**Anytime**	**Self Paced**
SANS SelfStudy	**Books & MP3s Only**	**Anytime**	**Self Paced**