

Securing NIS (formerly YP)



The following is a brief compendium of what we at Auburn University College of Engineering use to secure our NIS networks. We have a mix of about 65% NIS, 35% NIS+ network that is seeded from NIS -> NIS+ via periodic cron jobs. The following is our implementation of securing NIS using various vendor patches and free utilities from around the world.

NIS has a reputation of being extremely insecure. If you implement these steps it will lose most if not all of the reasons for this, and you will retain all the administrative advantages of NIS without any of the security risks. We only have experience implementing this with SunOS4.1.X, since we use NIS+ on Solaris 2.X machines and since we are a predominantly Sun shop. All other machines may have slightly different results and implementations. Hopefully others will find this useful, though. Here's a list of reasons why you should follow these steps.

1. People can grab your password map from any machine in the world and crack on it remotely.
2. It disables several of the holes found by [Satan](#).
3. Local people can use ypcat to grab all the encrypted passwords and crack on them.
4. Remote people can grab any map in your NIS domain. Some of these may have confidential information.

• Router Modifications

Most sites have a router connecting themselves with the outside world. If you have control of this router make sure you do the following to things, or everything else below could be completely useless. (Note, implementation details and configuration is router specific, and we can't help configure your particular brand of router.)

1. Turn off source routing
2. Apply a filter that makes sure that packets coming in from the outside network do not have source IP address that match the inside network. (IP spoofing) see the following [CIAC announcement on IP spoofing attacks](#)
3. If practical install a router filter that blocks ALL RPC packets. and everything on port 111 (and 2049 if you don't export NFS outside your LAN(s))

Replacing Daemons

- Wietse Venema has written 2 excellent utilities that are absolutely essential for securing NIS. They are replacements for the sun shipped portmap and rpcbind (SunOS4.1.X, Solaris2.X) respectively. These tools work much the same way as tcp_wrappers by defining access lists for

hosts that are and are not allowed to access your portmap. Without these tools in place a remote user can effectively use your own portmap as a way to circumvent security by accessing your NIS services through the portmap and appearing to be coming from a machine inside your domain. Click [here](#) to download portmap from <ftp.win.tue.nl>. Click [here](#) for rpcbnd 1.1 (Solaris2.X). Both of these require that you have libwrap.a compiled from tcp_wrappers. Click [here](#) to get tcp_wrappers7.2

Installing Vendor Patches

- For SunOS4.1.3_U1B and below get the NIS patch for securenets and install it on your system. On SunOS 4.1.3_U1 and U1_B this is patch 101435-03. On SunOS 4.1.[0-3] this is patch 100482-08. In your securenets file you should ONLY have those domains which require NIS maps. Use the smallest subset of domains possible. Try to not include subnets with PC's and macs on them if possible, since these machines have no concept of root; anybody could get your maps.

Transfer interrupted!

LI>

- It is a very good idea to make your NIS master machine restricted access for administrative staff only. This is where all the jewels are and if it gets compromised it does not good that NIS is protected when the raw text files are not. It is also a good idea to turn off routed on this machine and install static routes to all of your local networks that use NIS service. This way, even if a hacker can bypass your portmap/securenets security, the information drops on the floor on the way out of your machine because it doesn't have a way back to the hackers site.

Blocking TCP Attacks

- TCP sequence number guessing attacks are on the rise. These attacks rely on the fact the the kernel is not very good at picking random numbers for TCP sequence numbers. The sequence numbers control the order of the packets in the TCP/IP stream. If a hacker can guess what sequence numbers you are using (there are now automated tools to do this) he can intercept your session. In theory, removing the route to outside networks should prevent this attack, however, if it does not, or if you are not able to do this, you may wish to investigate some of the unsupported patches (for SunOS in particular) that block this kind of attack by altering the TCP sequence number generation. I've placed a README containing the rationale and instructions for one such patch [here](#)

Shadow Passwords

- Set your machine up to do Shadow Passwords. Contrary to popular belief, this does NOT require the C2 security patch. Shadow passwords can be enabled by following a few short easy steps. (If you use groups with passwords, you should do the following steps for the group.adjunct file as well, except for the AU* stuff)
 1. Make backup copies of the passwd maps on the NIS master and on all client and slave machines. Make a backup copy of the Makefile on your NIS master too.
 2. In the directory where you keep your NIS maps, make a directory called security (e.g. /var/yp/security).

```
cd /var/yp
mkdir security
chown root security
chmod 700 security
```
 3. Take all of the passwords out of your NIS master passwd map and stick them in a

passwd.adjunct map in the security directory. Replace all the password entries in the password map with entries that have the username preceded by two hash marks. Like this:

```
root:##root:0:0:::/bin/csh
```

The format for the shadow password map is then follows the form of
username:password:::::

Where password is what you removed from the passwd file. The 5 remaining colons will never have anything in them. They are used by C2 security for mandatory security accesses, but that is irrelevant here. The following awk script will generate a passwd.adjunct file.

```
nawk -F\: '{printf("%s:%s:::::\n", $1, $2)}' passwd >
security/passwd.adjunct
```

And the following script will fix your passwd file.

```
nawk -F\: '{printf("%s:##%s:%s:%s:%s:%s\n", $1, $1, $3, $4, $5, $6,
$7)}' passwd > passwd.new
```

Check the file passwd.new for any errors before replacing passwd with this new file. Now is also the perfect time to check for users that have no password and replace the empty password entry with a "*" in the passwd.adjunct file.

4. Create an /etc/security directory on every machine. And take the root password (and any other local passwords) out of your /etc/passwd file and into /etc/security/passwd.adjunct file (like above)

```
# mkdir /etc/security
# chmod 700 /etc/security
```

Then fill the file with something like this:

```
root:ZbAirHUqwr9w:::::
nobody:*:::::
daemon:*:::::
sys:*:::::
bin:*:::::
audit:*:::::
sync:*:::::
AUpwdauthd:*:::::
AUyppasswdd:*:::::
+:::::
```

Obviously, your root password will be taken out of your /etc/passwd file. The above password is nonsense anyway. If you want a different root password for each machine, make sure the root entry above has a valid password. If you want an identical root password for all your machines which comes out of NIS, delete the root line above all together.

5. Insert the following two lines in your NIS master password map AND in each machines local /etc/passwd file

```
AUpwdauthd:##AUpwdauthd:10:10::/lost+found:/bin/true
AUyppasswdd:##AUyppasswdd:11:10::/lost+found:/bin/true
```

Lost+found can be replaced with the name of any local directory.

6. Insert the following two lines into your NIS master passwd.adjunct, and make sure they are contained in every local /etc/security/passwd.adjunct map as well.

```
AUpwdauthd:*:::::
AUyppasswdd:*:::::
```

7. Make sure your rc.local file has the following set of lines uncommented. These provide a way for clients to authenticate a password as being valid or not without the password being passed over the net in clear text. The encrypted password is passed to the server for authentication.

```
#
# start up authentication daemon if present and if adjunct file exists
#
if [ -f /usr/etc/rpc.pwdauthd -a -f /etc/security/passwd.adjunct ]; then
    rpc.pwdauthd &                echo -n ' pwdauthd'
fi
```

8. Now you need to make sure that when you update passwords and users that your Makefile is setup properly to push out the passwd.adjunct map. After the all: rule in your Makefile add a new entry, we've called ours c2secure. Now you need to add a make rule for this entry.

```
c2secure:
    -@if [ -f $(DIR)/security/passwd.adjunct ]; then \
        if [ ! $(NOPUSH) ]; then $(MAKE) $(MFLAGS) -k \
            passwd.adjunct.time group.adjunct.time; \
        else $(MAKE) $(MFLAGS) -k NOPUSH=$(NOPUSH) \
            passwd.adjunct.time group.adjunct.time; \
        fi; \
    fi
```

9. You now need to make sure the rpc.yppasswdd on your NIS master is running with the right flags. Below is a sample. The -noshell and -nogecos flags specify that users are not allowed to change their shell or their full name using the passwd command. They are not strictly necessary for the proper functioning of rpc.yppasswdd. One further note: the rpc.yppasswdd supplied with 4.1.3_U1B seems to crash, so we use the one from 4.1.3 instead, and it works fine.

```
if [ -f /usr/etc/rpc.yppasswdd ]; then
    rpc.yppasswdd /var/yp/dbdir/passwd /var/yp/dbdir/security/passw
    echo -n ' yppasswdd'
fi
```

10. Finally, you need to push the maps out. First run make on your master. It will update the map and attempt to push it out. If you have slaves it may fail because the slaves don't know about this newly created map. Use ctrl-c to break the process and either ypxfr the map by hand or ypinit -s on each of the slaves to reinitialize and synchronize with the NIS master.
11. Test it out. If it doesn't work let me know, I may have inadvertently forgotten a step. In any case, you did follow step 1 and make backups right?

TCP Wrappers Addendum

- Here are some ways to setup your hosts.allow file when you have portmap installed. Let's assume you have a class B network with address 129.129, subnetted using mask 255.255.255.0. Now let's assume your sun network has 10 subnets. There are two ways you can setup your hosts.allow file. You can put all 10 subnets in one at a time like this:

```
portmap: 255.255.255.255 0.0.0.0
portmap: 129.129.1.0/255.255.255.0
portmap: 129.129.2.0/255.255.255.0
...
...
portmap: 129.129.10.0/255.255.255.0
```

Note: the first line is not always necessary, but it's safe to include it. It is necessary on NIS slaves and masters that have to answer ypbind broadcast requests on the local network. The second

approach is to just put a broad mask for your entire Class B network. Sometimes this is easier than doing every single subnet one at a time. (That would make for a HUGE /etc/hosts.allow file, 1 per machine. Obviously, this would be impractical. Here's an example of the latter approach.

```
portmap: 255.255.255.255 0.0.0.0
portmap: 129.129.0.0/255.255.0.0
```

- Read the documentation included with tcp_wrappers. It's very useful stuff. You can apply lots of options to the above. For example, you could put a statement up there such that any request from an unauthorized host would result in a reverse finger being mailed to the account of your choice plus a syslog entry that could trip an alarm.
 - If you get complaints about 'audit', you'll need to mkdir /etc/security/audit on the machines where you run shadow passwords.
 - Congratulations! You are now immune to most of the security holes and attacks that been publicized widely and have plagued NIS.
-

Related Documents

- [CIAC - Securing Internet Information Servers](#)
 - [Unishield](#) is a commercial product which has been designed to plug several of the above security holes.
 - [Cert Advisories](#)
 - [Bugtraq archive](#)
 - [Satan documentation](#) (compressed tar)
 - [SunOS satan advisory](#)
 - [Unix Security Topics at Dartmouth](#)
-

If you have any questions/comments, feel free to send me [email or comments](#).



[Back to home page of Doug Hughes](#)