NEXTSTEP In Focus, Spring 1994 (Volume 4, Issue 2).

Detecting Rogue Servers

Alan M. Marcum

Rogue servers are NetInfo servers that are set up improperly and wreak havoc on a network. Tracking them down can be tedious, but you can use a script to make the task easier.

Dastardly NetInfo Scoundrels

A rogue server. It sounds pretty nastyÐand it is! Sounds like it could foul up a NetInfo network pretty badlyÐand it can.

The question is, how do you go about finding a rogue server? And, come to think of it, what is one, anyway?

Case of the Hotshot Administrator

Consider the tale of Barb, the system administrator at Rhino Aviation. The network she runs had been working great. It was a pretty hot setup too, if she said so herself: a three-level NetInfo hierarchy, an Internet firewall, four mail servers, and hundreds of computers strewn to the four corners of the globe.

Or at least it was, until that hotshot whipper-snapper of a new system administrator, Sam, came along! About three days after that character showed up, things started acting, well, screwy. Not terriblyÐit's not like the whole network crashed or anything. But, occasionally a computer would go through an automatic host addition cycle when it rebooted. Or a computer would come up with odd mounts.

And, Barb just knew she had added that new user on Friday, in plenty of time for a Monday morning new-hire orientation. But, the account wasn't there last Monday morning when the new person started

NEXTSTEP training.

So Barb was convinced: This hotshot Sam fellow must be fouling things up. Undoing her perfectly good work. Screwing up the whole network. And probably doing it just for kicks or to make her look bad.

Time to tell the boss that this whippersnapper should be drawn and quartered. But wait, Barb figured, Maybe I should build up a rock-solid case first. Do a little investigating and collect evidence. All those detective novels she'd read over the years would come in handy. She put on her wide-brimmed fedora and got to it.

Looking for fingerprints

She started gathering evidence on a cold, dark FridayÐa storm was looming like an overbearing department head. Her, um, ªsuspectº was conveniently at a seminar.

Her first step was changing the root passwords in all the NetInfo domains on the network. She was an old hand at this. She had a shell script to do it, so she picked three new passwordsÐone for the networkwide domains, one for all the local domains, and one for automatic host additionÐand sent the shell script on its way. Now, she thought, I'll prove that it's Sam who's guilty. New passwords aren't readily guessed nor easily cracked by known tools, and only IÐas of nowÐknow the new password.

In the midst of all this, she got word that a new person was starting at the company next Monday. She broke off sleuthing to create the new account and set up a computer in the new person's office. Automatic host addition took care of getting the computer into some parts of NetInfo; she took care of everything else manually.

She finished rebooting the new computer after adding it and running the installation customization script. Suddenly, the lights flickered, there was a BOOM!, and the power failed completely. She started off for the server room, but then the power returned. Knowing the uninterruptable power source took care of the computers in the server room, she turned back to the recently installed computer, just to be sure it came up successfully. Hey, she thought, you never know with these new boxes, right?

And, there on the screen in front of her sat an automatic host addition prompt.

The plot thickens

Now wait just one cotton-pickin' minute! she thought. I know I just had that computer up on the network. I know it booted successfully, without automatic host addition! Sure, I just had a birthday, but I'm not getting senile yet.

And worse yet: There's absolutely no way Sam could have caused this.

Either there's a new villain or something else is going on that I haven't figured out. Yet.

Where's this stuff coming from?

Barb had to gather more data, and she had the perfect opportunity right in front of her. She grabbed her network analyzer, hooked it up, and checked which computer's **bootpd** was providing this host information. It should have been the one on the computer running the domain's master serverÐbut Barb was in the mood to be suspicious, so she checked anyway.

And the information wasn't coming from the master server's computer! Instead, it had come from a clone's computer! The clone Barb and her suspect had added that very first day.

Things were getting worse and worse, not better. Only the **bootpd** on the computer running the master server for the domain to which a new host is being added automatically should ever provide automatic host addition services. Was there some new bug? Or had the rules changed?

Barb decided to check out the database run by that clone. She tried the Connect by Tag command in the Open menu and began at the top with the root directory. Things looked fine: a **master** property, a **trusted_networks** property, a **security_options** property, and nothing else. But waitÐwhat was that **master** property doing saying **cadet/network**? It should say **exec/network**! How could a clone have a different value for the master property?

From enemy to ally

It was time to get all the help she could. Barb waited until Sam got back from his seminar and then described her findings to her new compatriot. Together they started looking aroundÐthey used NetInfo Manager to connect to the master's database and to the aberrant clone's. It didn't take them long to realize that there really were two entirely different databases. Almost none of the computers that should have been there were there, and only a few of the users and mounts were there.

Then Sam piped up, ªHey! Those look like the computers we had problems with. The ones we thought were added, but weren't? And same for the users, and the mounts!º

And it hit them both like a 2 gigabyte drive: All those added-yet-not-added additions really went into this ªotherº domain.

The Cause and the Cure

So, *that* was a rogue NetInfo server. The fix is straightforward: Remove that rogue database and reclone the domain to run on the desired computer. Of course, it would be nice to avoid this mistake happening again, especially immediately; to prevent that happening, you need to know where these rogues come from.

Here's one way to generate a rogue NetInfo serverÐa rogue master, in this case.

Assume you're adding a new computer to the network. You do some configuration and pre-installation testing first, so you have the new computer on its own network. Then, you create the networkwide NetInfo database on the computerÐbut you forget to switch it to the production network, the database that should be a clone of, in this case, **exec/network**. It's late in the day, you're tired, and you don't notice what you've done.

VoilÕ! You just created a new domain, with the same tag (**network**) as an existing domain. Now, this might not be all badÐyou can certainly have multiple domains on the same network with the same tag. But in this case it's not what you intended. To make matters worse, perhaps you had turned on automatic host addition; now you have two different domains on the same network with automatic host addition enabled. This can lead to computers being added to the different domains entirely at random.

Another kind of rogue is a rogue clone, a clone which, according to the master's database, shouldn't be running. These came into being if someone added a clone back in the Dark Ages of Release 2.2Ðwhen new NetInfo servers had to be administered by handÐand in creating the clone forgot one critical **serves** property. Not all rogue clones started in the Dark Ages: Maybe someone who added a clone learned how to do so in the Dark Ages and never re-learned to do it the new way.

Discovering rogue servers systematically

Unfortunately, discovering the presence of a rogue clone is a tedious operation. It requires checking every computer on your network to ensure that it's running the NetInfo servers it should and that those servers' databases are consistent with the master for the domain. This involves checking not only that each computer is running servers with the correct tags, but also that the contents of those servers' databases are all mutually consistent.

What does this entail? It's a bit cumbersome.

1. Look at the databases the computer serves and examine each to determine if the computer should be serving it or not, according to that database itself. Is there a **serves** property for the target computer

with a value of **./tag**? If not, you have a rogue clone that won't be updated.

2. Check that the computer that the target's database says should be running the master server for the domain is running the master server. Further, check that the value of the **master** property in the master server's database matches the value of the **master** property in the target computer's database for the domain. If it doesn't, there's an inconsistency, likely a rogue master.

3. Maintaining a ªbelt-and-suspendersº philosophy, verify that there's an appropriate **serves** property for the target computer in the master's database.

4. Verify that the **ip_address** property for the target computer in its database and in the master's database match. Otherwise, this would be another form of a rogue clone.

Tools to the rescue

This systematic searching sounds like the perfect thing for a program, doesn't it? A nice, big loop that iterates over a list of computersÐpossibly ªall the computers in the networkºÐchecking each case in turn. Fortunately, there's a shell script called **nicheck** that does exactly that; it's provided with this issue. It takes as arguments a list of hosts to check. You can use the options in Figure 1 with **nicheck**.

Figure 1: *nicheck* options

Option	Effect
-v	Produce verbose output. Otherwise, the output is terse and shows only problems.
-d *database*	Restrict checking to just the specified database.
-f *file*	Check the hosts listed in the specified file. Use - to specify input from **stdin**.
-V	Note hosts that seem to be down.
-P *packets*	Stop checking if a host is running after *packets* **ping**s. Default is 2.
-T *timeout*	Wait no more than *timeout* seconds for a NetInfo read. Default is 20.
-L	Check the domain tagged local, too. (This option is not recommended!)
-M *directory*	Get **/machines** information from the specified NetInfo directory. Default is **/machines**.

In a future release (likely 3.3), ***niutil*** *will provide for a timeout specification with* ***niutil -T****.*

Figure 2 shows some sample runs of **nicheck**.

Figure 2: *Output from **nicheck**, using the option for noting hosts that appear to be down and the terse and verbose options*

```
mustang [~]-133% nicheck pfm tls ranger sabre bonanza
***pfm is serving network inappropriately***
Host 'bonanza' unknown.

mustang [~]-134% nicheck -V pfm tls ranger sabre bonanza
***pfm is serving network inappropriately***
ranger appears down; can't check.
Host 'bonanza' unknown.

mustang [~]-135% nicheck -v pfm tls ranger sabre bonanza
pfm
   network..***shouldn't be served***
tls
sabre
   network...ok (master).
   Rhino.....ok.
Host 'bonanza' unknown.

mustang [~]-136% nicheck -v -V pfm tls ranger sabre bonanza
pfm
   network..***shouldn't be served***
tls
ranger appears down; can't check.
sabre
   network...ok (master).
   Rhino.....ok.
Host 'bonanza' unknown.
```

A shell script such as **nicheck** might run for quite a while in a large network, so it's the sort of thing a smart network administrator might start running just before leaving for the night.

One more note about **nicheck**: To check for rogue servers, one must examine specific NetInfo databases, not just any database in the domain that happens to be available. This is done using **niutil -t**. However, in NEXTSTEP releases through 3.2, **niutil -t** doesn't time out: If it doesn't get an answer, it just keeps on trying. If you did start **nicheck** when you left one evening, it sure would be disappointing to return the next day to find that the script hung because some computer was down. To work around this problem, a ªwatch dogº timer is included in **nicheck**. If you're interested in the gory details, take a look at the shell

script.

Epilogue

Rogue NetInfo servers, whether they be rogue masters or rogue clones, can play havoc with your network. They can lead to inconsistent results, intermittent failure, irritable users, and short-tempered bosses. If you suspect your network has one, put on your trenchcoat and track it down!

We expect that in a future release (again, likely to be Release 3.3), **netinfod** will perform some sanity checking at startup to ensure the consistency of a domain and a server's database. Errors will be logged to **syslog**. This checking will help you find problems that can lead to rogue servers.

Alan M. Marcum is a member of the Premium Support Team and specializes in large networks.

*What did you think of this article? Please send feedback to **journal_info@next.com** to let us know whether this article was useful and to tell us how we can make this journal a great resource for you!*