

## Conducting a Network Inventory Part 2

Michael J. Martin

In Part 1 of this series, I covered the process of collecting data for a network inventory. Next comes the part that I claimed was a "snap" -- generating a nice inventory report.

Before we go any further, however, I wish to clarify and revise my remarks. First, the good news is that the report will actually look better than it did in the last article. That's because, rather than go through a laborious file consolidation process, we now have a Perl script to do the work for us. Of course, to get a lot sometimes you need to give a little. In order to use the report script, you need to use an updated ouilookup.sh script, which formats the data a little differently than the older version. This change is well worth it, because the report script yields a nice tab-delimited file that can be imported into Excel to create this beautiful report:

IP	MAC	OID	DNS
172.30.71.1	0000.0c07.ac05	CISCO SYSTEMS INC.	gw.outlan.net
172.30.71.100	0004.ac5e.4810	IBM CORP.	obiwan,outlan.net
172.30.71.14	000b.cd83.53f6	Unknown OUI	dirserv.outlan.net
172.30.71.60	0008.02a1.4d80	Compaq Computer Corporation	foo.outlan.net
172.30.71.2	0001.812b.fe08	Nortel Networks	Unresolved
172.30.71.3	0005.5fe7.0800	Cisco Systems Inc.	Unresolved

This eliminates the need for running a bunch of separate text formatting processes and using the esoteric Unix diff command. To get started, you should have a collection of files similar to the ones listed here:

```
Trinity$ ls
172.30.100-IPdump-outlan-rs01-12-05-04.txt
Vlan-list-outlan-rs01-12-05-04.txt
172.30.100-dnsdump-12-05-04.txt
172.30.100-ARPDump-outlan-rs01-12-08-04.txt
172.30.100.1-acthost-12-05-04.txt
172.30.100-IPdump-fny-rs01-12-07-04.txt
oui-join.txt
Trinity$
```

Here's a quick review of how you got these files:

Vlan-list-\*.txt is the output of the vidump.sh script. It generates a list of a router's physical interfaces or a switch's VLAN interfaces, which you can use to determine which networks you should audit.

\*-IPdump-date.txt is the report output of arpextract.sh with the "-ir" flag that generates an IP address table from a router or Layer 3 switch.

\*-dnsdump-date.txt is the output of the oui-join.txt is the output of ouilookup.sh. This is a wrapper script for the Perl script ouiresolv.pl that resolves the OUI portion of the 48-bit Ethernet MAC address.

To generate the inventory report, only the acthost, arpdump, oui-join, and dnsdump data files are needed. Use the reporting script gen-net-inv.pl. The command syntax is

```
gen-net-inv.pl < acthost> < arpdump> < oui-join> < dnsdump>:
```

## Conducting a Network Inventory Part 2

Michael J. Martin

Here is a CLI example:

```
Trinity: # perl gen-net-inv.pl 172.30.71-acthost-01-24-05.txt 172.30.71-
ARPDump-outlan-rs01-01-24-05.txt oui-join.txt 172.30.71-dnsdump-01-24-
05.txt
Done
Trinity: #
```

When the script has completed its run, it reports "Done" to the CLI and exits. The report names are derived from the network designation associated with the acthost file. In the case that one of the report source files is missing from the CLI, the script errors with "Cannot open file" and exits.

```
Trinity: # perl gen-net-inv.pl 172.30.71-acthost-01-24-05.txt 172.30.71-
ARPDump-outlan-rs01-01-24-05.txt oui-join.txt
Cannot open file
Trinity: #
```

Before we go on to the reports the script generates (and why), there is one last thing you must keep in mind. The order in which the script reads the file is not flexible. The files must be processed in the following order if you want an accurate report:

```
acthost
arpdump
oui-join
dnsdump
```

If the files are not processed in the correct order, a report will generate but with incorrect data. Here is an example where the dnsdump and oui-join files have exchanges places:

```
Trinity: # perl gen-net-inv.pl 172.30.71-acthost-01-24-05.txt 172.30.71-
ARPDump-fny-rs01-01-24-05.txt 172.30.71-dnsdump-01-24-05.txt oui-join.txt
Done
Trinity: #
```

The script has generated a report, but take a look at the output:

```
Trinity: # more 157.191.5.1_consolidated_report.txt
IP      MAC      OID      DNS
172.30.71.1    0000.0c07.ac05  Empty    ->
172.30.71.100  0004.ac5e.4810  Empty    ->
172.30.71.14   000b.cd83.53f6  Empty    ->
172.30.71.15   0008.02a1.4d80  Empty    ->
172.30.71.16   0050.8b2c.f25e  Empty    ->
```

Compare the above to what the output should look like:

```
Trinity: # more 172.30.71_consolidated_report.txt
IP      MAC      OID      DNS
172.30.71.1    0000.0c07.ac05  CISCO SYSTEMS INC.  Unresloved
172.30.71.100  0004.ac5e.4810  IBM CORP.           overload.outlan.net
172.30.71.14   000b.cd83.53f6  Unknown             OUI foo.outlan.net
```

## Conducting a Network Inventory Part 2

Michael J. Martin

```
172.30.71.15    0008.02a1.4d80  Compaq Computer Corporation  home.outlan.net
157.191.5.16   0050.8b2c.f25e  COMPAQ COMPUTER CORPORATION  obiwan.outlan.net
```

This happens because the script is comparing the data in each of the files and parsing it into a set of reports, expecting certain values. If and when they are not there it reads this missing data as either valid or an error and reports on it. If data is there, but it's the wrong data, it also reports on it. It's a slippery slope because you want to see inconsistencies in the data, but you are not always sure what those will be. You also don't want to constrict the function of the script by requiring that the input files meet some explicit criteria. Just remember that the files must be processed in the correct order if you want the report to be correct.

What does all of this data provide us with? The reporting aspect of the network inventory should yield three things:

- A listing of all the active hosts on an IP subnet at the time the inventory was run
- Information on inconsistencies between ICMP collection and the router/switch's visibility of the subnet
- Inconsistencies between what is active on the subnet and what is in DNS.

There is also a fourth aspect that has value, depending on the environment: network hardware information, which can assist in the reading of the results of the network audit data to determine some of the validity of the findings and assist in the configuration of the network vulnerability scan tests. After all, the more you know, the more confident you can be in the findings. To help meet the above reporting criteria, the script generates three files:

\*\_consolidated\_report.txt is just what its name implies, a consolidation of all of the inventory data in a single report, including IP information, ARP information, OUI information and DNS information.

\*\_dns\_err\_report.txt is a list of IP addresses to hosts that do not have any DNS information. This report can be mailed off to the hostmaster so the problem can be fixed.

\*\_icmp\_err\_report.txt is the differential between the IP host data gathered by the ICMP scan and the ARP table report. Normally this report should be empty. If for some reason an inconsistency does come up, you may have to use Unix diff after all.

The IPdump report is used to resolve inconsistencies found in the \*\_icmp\_err\_report.txt file. Using the Unix diff command, the IPDump and acthost files need to be compared. Here is a syntax example:

```
Trinity$ diff *-IPdump*.txt *-acthost*.txt
```

This command compares the findings of the active IP hosts in the ARP table to those found using the ICMP ping scan. If a difference should come up, it will usually be hosts missing in the "acthost" file.

This is due to either ICMP filtering on a router gateway or some kind of local ICMP handling rules. Diff operates using file A -> file B comparison. It looks at the two files and makes suggestions on how file A can be changed to be the same as file B. So the "master" file should be file B. Diff reports its change recommendations using "c" to indicate a change is needed on a line, "a" where an addition is needed to a file at a specific line, or "d" where a line needs to be deleted. Needless to say, it takes some practice to easily work with diff, due to its esoteric output.

## Conducting a Network Inventory Part 2

Michael J. Martin

Let's take a look at the results of the "acthost" (which is the file we want the output to match) to "ipdump" (which is the file that "acthost" should match to):

```
Trinity$ diff 172.30.100-acthost-12-05-04.txt 172.30.100-IPdump-fny-rs01-12-05-04.txt
2a3
> 172.30.100.101
10c11
< 172.30.100.2
---
> 172.30.100.22
Trinity$
```

To make comparing a little easier, here is what the two files look like side by side:

172.30.100.1	172.30.100.1
172.30.100.100	172.30.100.100
172.30.100.14	172.30.100.101
172.30.100.15	172.30.100.14
172.30.100.16	172.30.100.15
172.30.100.17	172.30.100.16
172.30.100.172	172.30.100.17
172.30.100.18	172.30.100.172
172.30.100.19	172.30.100.18
172.30.100.2	172.30.100.19
172.30.100.3	172.30.100.22
172.30.100.4	172.30.100.3
172.30.100.5	172.30.100.4
172.30.100.83	172.30.100.5
172.30.100.85	172.30.100.83
	172.30.100.85

The 2a2 output from diff means that the line 3 of the "IPdump" file needs to be added after line 2 in "acthost" file. In terms of our inventory, we need to investigate why 172.30.100.101 did not respond to ICMP, aside from gateway or local ICMP filtering. The host could also no longer be active (with its ARP entry still stuck in the ARP table) or it could be that the host simply did not respond in time.

The 10c11 output from diff indicates that there is a mismatch between line 10 on the "acthost" and line 11 on the "IPdump" file (assuming that you have corrected the first difference). Diff recommends that line 10 be changed to reflect line 11. In terms of the audit, however, this represents another point to investigate. The ARP table sees a host 172.30.100.22, and the ICMP scan sees a host 172.30.100.2. This could indicate that host 172.30.100.22 is behaving similar to 172.30.100.101. As for host 172.30.100.2, it responded to ICMP, but is not in the ARP table. If the ICMP scan was run from a host connected to the target subnet, it is possible that the host could have just come online and had not yet forwarded a packet off the subnet yet. In that case, the gateway would have not seen any traffic from the host and would not have registered the MAC in its APR table.

## Conducting a Network Inventory Part 2

Michael J. Martin

Of these two examples, the former will be far more common. I use them to illustrate the point that there will be instances where the data from the two reports will not be in sync and you'll need to do some investigation to resolve the issue. Whenever discrepancies arise, the first action to take is to verify that the "odd" hosts are reachable. Then rerun the ICMP scan and arpdump and compare the results again. If the problem continues to exist, check to see if the "odd" host is running filtering software or has a local policy for responding to ICMP traffic. If all else fails, clear the ARP table on the router or switch using the exec command <clear ip arp> and rerun the tests again. Hopefully, you will not run into this problem often. But if you do, it should be investigated. This kind of scenario is also an indicator of a rouge access point or network monitoring point.

Now you've completed your network inventory are ready to run your network services audit. Stay tuned for the next steps in the process. In the meantime, if you have questions, comments, or article suggestions, please e-mail us.