An Elusive Diagnosis

reprinted from the Tejas Software Consulting Newsletter, v2 #2, April/May 2002
by Danny R. Faught, faught@tejasconsulting.com, http://tejasconsulting.com

Software quality folks seem to enjoy reading about bug hunts, so here is the tale of one particularly interesting bug that I hunted down lately. I'll go into some fairly low-level details, so my non-technical readers may want to skip down to the lessons listed at the bottom. I can't reveal which system I was working on, so I've changed some of the details.

One of my clients uses a script called "webupdate" to download new versions of the operating system software provided by an outsourced development team. The script copies the software from the outsourced vendor's web site and compiles it. One time when going through the process, something didn't seem quite right to me. It ran about as long as it usually does, but it seemed like the volume of output was quite a bit shorter than usual. I hadn't saved the output from a successful run before, so I couldn't know for sure. But I was pretty sure that this error near the beginning of the output hadn't been there before:

```
% webupdate
checking web site for new packages...
: web file not found
...
```

This was followed by the voluminous output that is the result of a successful build, and then a final message indicating that all was well. But none of the changes that were supposed to be in this release of the system were there. This was a showstopper, despite the misleading indication of success. I decided to investigate, to see whether this was a problem with our environment or whether I needed to report a bug to the vendor. Note that we ran on the previous version of the system to get the new one, and we hadn't had this kind of trouble with any previous version before.

Luckily, webupdate is a script rather than a compiled program, so I'm able to easily examine the implementation. By looking at the output right before and right after the mysterious ": web file not found" error, I determine that the error most likely came from a "getdir /" command. Okay, great. What does getdir do? I can't find any documentation for such a command, and I can't find a program by that name. Oh! Half a screen up in the webupdate script is a function named "getdir".

Okay, I'm getting closer to the source of the problem, but I don't know how close yet. There is nothing in the function directly that prints out the text "web file not found." But there are a few calls to external programs. The second one is preceded by an "echo downloading $name..." message, which I didn't get in the output, so I explore the first, which is a call to a program called "webls".

I find the webls program, which is also a script. Aha! There's the telltale code which produces the error - "echo $sub: web file not found". Hmmm, the $sub variable must be empty, since the error we got starts with simply a lone colon. Maybe that's the problem. I trace this variable through the program and find a pair of regular expressions that munge the parameter that is passed into the script. The parameter in this case is "/", indicating the root of the web server, and the regular expressions erase this character. Looking at the logic of the script, that seems to be okay. That was a dead end. So I turn my attention to a call to another external program: "webget www.vendor.com/download/$sub 2>/dev/null".

It turns out that this one is a compiled program. I do have access to the source code, but I really don't have any hints on where to start looking. I notice that errors from this program are hidden because of the "2>/dev/null" at the end of the line. So maybe there's some valuable information that's getting thrown out. To explore this, I run the webget command directly. Sure enough, I get an error that I wasn't seeing before:

```
can't reach www.vendor.com: The requested address is not valid.  (connect
```

That's odd - seems to be truncated. Maybe it's trying to say "connection failed"? I look at the webget source code that prints this message and it seems to be okay - looks like an operating system bug is causing the error to be cut short. Dang, why can't we hit just one bug at a time?

Well, now I'm stuck. I really don't know what's causing this error. I run a web browser and try to get to the web site. It works just fine. I go back to the webupdate script and scan the code, looking for inspiration. Way down at the bottom, I found some help:

```
} 2>&1 | tee /web/log/`cat /dev/time`
```

Matching the opening curly brace, I find that most of the script is enclosed in the braces, and all of the output from that code is copied to a file. I look under /web/log, and I find a stack of files, each with a long string of numbers for its name. Great! I do have the output from previous runs. I look at the file modification times to identify the log from the failed run, and the last successful run right before it. Sure enough, the failed run produced several kilobytes less output. That confirms my previous assumption, but I'm still no closer to knowing what's going on.

Also in my perusal of the webupdate script, I saw code that sets up a web proxy. The code appears after the getdir call, so it doesn't seem to be related. But I realize that the webget really does need to know about a proxy server in order to get outside the firewall. I check that the proxy file is set up properly, as it would be after any full run of the webupdate script. Ah, so the proxy may not work the first time webupdate is run on the system, but every run after that should be okay. I verify that the webget call still fails the same way.

Now is the point where inspiration strikes. Working with networks, I've seen that a common problem is for DNS to stop working - DNS is what converts symbolic Internet addresses like proxy.client.com to numeric addresses like 12.34.56.78. I've also seen cases where different applications used different DNS mechanisms, so one would work where others would fail. By using the less attractive numeric addresses, I can still do useful work if the DNS server is the only thing that's broken. So I look up the numeric address of the proxy server and edit the proxy setting from proxy.client.com:http to something like 12.34.56.78:80 instead. I also convert the "http" to the numeric port number "80" just in case. I run webget and shazam, it works! I modify the webupdate script to use the numeric address and port number, and the webupdate runs just fine - I check that the log file is about the same size as the last successful run. (Well, actually I had to track down two other unrelated problems, but I'll spare you that story for now. Bugs often seem to appear in clusters...)

To further isolate what's happening, I alternately set the proxy address and the port back to their symbolic versions. I verify that both must be numeric for the webget to work. I decide that I have enough information to report a bug to the vendor, which I do.

It takes a while to get an answer (I couldn't justify setting the bug at a high priority because I had a fairly easy workaround). The response is that I need to be running the DNS server, and it would have been run

automatically if I had logged in to a particular account on the system. I didn't know enough about the system to know that the DNS server wasn't started every time the system boots. So on the surface, it was a configuration problem after all, but there is also some room for improvement in the system.

So what did I learn?

- We need to document any requirements placed on the build environment, such as logging in to a particular account.
- To make the build environment more robust, important steps like running the DNS server should be automated from the webupdate script, not in the setup files for a user account.
- Throwing away program output, as is often done to clean up useless clutter, can also hide important error messages. Also, scripts should check for errors from the external programs they call so they don't erroneously report success. Use more than kind one check to verify that your software is working.
- I was lucky to have access to the scripts and source code. The problem would have been far more difficult to diagnose if I had simply reported the "web file not found" error to the vendor. Plus, I wouldn't have been able to find a quick workaround.
- Bootstrapping can be confusing. That worrisome code that seems to set up the proxy server too late actually sets it up for the *next* run, and it doesn't affect the current run, since we run on the previous version of the system to bootstrap the next version. This means that the very first install of the system had to be done manually.
- Keep copious logs. I was lucky that the webupdate programmer had the foresight to set this up.
- You're often fighting against more than one problem. I actually found four separate problems that all cropped up at the same time, not counting robustness issues (were you keeping count?).

There's a big overlap between bug reporting and debugging. How do you draw the line between the two? If I had been analyzing a test failure rather than a build process failure, should I have just filed a bug report based on the first symptom I saw? What if several other tests were blocked because of the problem? Food for thought.

#####