The network interoperability compatibility problem, first follow-up of many

devblogs.microsoft.com/oldnewthing/20060331-15

March 31, 2006

Raymond Chen

Okay, there were an awful lot of comments yesterday and it will take me a while to work through them all. But I'll start with some more background on the problem and clarifying some issues that people had misinterpreted.

As a few people surmised, the network file server software in question is Samba, a version of which comes with most Linux distributions. (I'll have to do a better job next time of disguising the identities of the parties involved.) Samba is also very popular as the network file server for embedded devices such as network-attached storage. The bug in question is fixed in the latest version of Samba, but none of the major distributions have picked up the fix yet. Not that that helps the network-attached storage scenario any.

It appears that a lot of people though the buggy driver was running on the Windows Vista machine, since they started talking about driver certification and blocking its installation. The problem is not on the Windows Vista machine; the problem is on the file server, which is running Linux. WHQL does not certify Linux drivers, it can't stop you from installing a driver on some other Linux machine, and it certainly can't download an updated driver and somehow upgrade your Linux machine for you. Remember, the bug is on the **server**, which is another computer running some other operating system. Asking Windows to update the driver on the remote server makes about as much sense as asking Internet Explorer to upgrade the version of Apache running on slashdot.org. You're the client; you have no power over the server.

Some people lost sight of the network-attached storage scenario, probably because they weren't familiar with the term. A network-attached storage device is a self-contained device consisting of a large hard drive, a tiny computer, and a place to plug in a network cable. The computer has an operating system burned into its ROMs (often a cut-down version of Linux with Samba), and when you turn it on, the device boots the computer, loads the operating system, and acts as a file server on your network. Since everything is burned into ROM, claiming that the driver will get upgraded and the problem will eventually be long forgotten is

wishful thinking. It's not like you can download a new Samba driver and install it into your network-attached storage device. You'll have to wait for the manufacturer to release a new ROM.

As for detecting a buggy driver, the CIFS protocol doesn't really give the client much information about what's running on the server, aside from a "family" field that identifies the general category of the server (OS/2, Samba, Windows NT, etc.) All that a client can tell, therefore, is "Well, the server is running some version of Samba." It can't tell whether it's a buggy version or a fixed version. The only way to tell that you are talking to a buggy server is to wait for the bug to happen.

(Which means that people who said, "Windows Vista should just default to the slow version," are saying that they want Windows Vista to run slow against Samba servers and fast against Windows NT servers. This plays right into the hands of the conspiracy theorists.)

My final remark for today is explaining how a web site can "bloat the cache" of known good/bad servers and create a denial of service if the cache did not have a size cap: First, set up a DNS server that directs all requests for *.hackersite.com to your Linux machine. On this Linux machine, install one of the buggy versions of Samba. Now serve up this web page:

```
<IFRAME SRC="\\a1.hackersite.com\b" HEIGHT=1 WIDTH=1></IFRAME>
<IFRAME SRC="\\a2.hackersite.com\b" HEIGHT=1 WIDTH=1></IFRAME>
<IFRAME SRC="\\a3.hackersite.com\b" HEIGHT=1 WIDTH=1></IFRAME>
<IFRAME SRC="\\a4.hackersite.com\b" HEIGHT=1 WIDTH=1></IFRAME>
...
<IFRAME SRC="\\a10000.hackersite.com" HEIGHT=1 WIDTH=1></IFRAME>
```

Each of those `IFRAME`s displays an Explorer window with the contents of the directory `\\a1.hackersite.com\b`. (Since all the names resolve to the same machine, all the `\\*.hackersite.com` machines are really the same.) In that directory, put 200 files, so as to trigger the "more than 100 files" bug and force Windows Vista to cache the server as a "bad" server. In this way, you forced Windows Vista to create ten thousand records for the ten thousand bad servers you asked to be displayed. Throw in a little more script and you can turn this into a loop that accesses millions of "different" servers (all really the same server). If the "bad server" cache did not have a cap, you just allowed a bad server to consume megabytes of memory that will never be freed until the computer is rebooted. Pretty neat trick.

Even worse, if you proposed preserving this cache across reboots, then you're going to have to come up with a place to save this information. Whether you decide that it goes in a file or in the registry, the point is that an attacker can use this "bloat attack" and cause the poor victim's disk space/registry usage to grow without bound until they run out of quota. And once they hit quota, be it disk quota or registry quota, not only do bad things start happening, but they don't even know what file or registry key they have to delete to get back under quota.

Next time, I'll start addressing some of the proposals that people came up with, pointing out disadvantages that they may have missed in their analysis.

Raymond Chen

Follow