the network police blotter

by Marcus J. Ranum

Marcus J. Ranum is CEO of Network Flight Recorder, Inc. He likes cats: they are complex yet manageable. When he's not working 10-hour days he plays console games and pursues too many hobbies for his own good.

<mjr@nfr.net>

As I write this, security is front-page news because Yahoo.com, Amazon.com, CNN.com, eBay.com, and others are under concerted denial-of-service attack from terrorists whose motivations and identities are, at present, unknown. As someone who runs a business, I can tell you how terrifying it is to contemplate being shut down by someone you can't identify or block. This kind of thing is about as elegant as a drive-by shooting and shows that, as a society, people couldn't resist bringing the worst aspects of "the real world" into cyberspace. I wonder if the perpetrators think it's funny. What motivates them? For that matter, what motivates the people who developed and released the tools the attackers are using? And what drives the virus writers? Come on, guys, it's not cool – you're hurting people.

Once again, security analysts and network managers are being presented with an extremely challenging problem that we really don't know how to solve conclusively. It's going to cost the Internet and its users millions of dollars in wasted time, lost revenue, and ulcers over the next few years. I've been happy, generally, with the media's response to this event, as compared to past problems. This time, at least, they are not saying it's being done by "brilliant" misunderstood "whiz kids" or anything like that. Perhaps the hackers made a big mistake going after CNN.com – media's "nudge-nudge-wink-wink" attitude toward hacking has done a lot to make it seem sexy and cutting edge. Biting the hand of the media is an error of judgment sure to get you slapped, as many politicians have learned. Perhaps their irritating the media will help adjust the press's coverage. Fundamentally, these are social problems, not technological problems, and are best solved by social means. The media has to take its part, since it is, arguably, the "voice" of society. This kind of nonsense isn't the work of "brilliant whiz kids"; it's the work of socially maladjusted losers.

Boundless Overruns

Lately, the problem of buffer overruns in code has become serious. It seems that virtually every application designed to work on a network has suffered some kind of buffer overrun recently. If an application hasn't, it's probably just because the hackers haven't looked at it – yet. Obviously, it's not a *new* problem, really – like many security problems, it's one we've known about for a very long time. I guess circumstances have pretty much conspired to let us ignore it for many years, but now the grace period is emphatically over. In the early 1990s I wrote a couple of firewall products; recently, in order to torture myself, I went back, pulled the source out of my archives, and did code reviews of some modules. When I wrote them, I know I was trying to be pretty careful, but in several places I was not careful enough. At least I tried, I suppose. What about the developers who don't know they need to be careful? What about the developers of applications that aren't believed to be security-critical? My guess is that the vast majority of networked applications are rife with security bugs.

What can we do about it, if anything? Recent flaws in "mission-critical" software seem to point out just how little progress we have made in securing our critical code, yet every day we field new applications, most of which were developed with only a cursory thought to security. I suspect that, as the software market continues to heat up, the environment will get worse, faster. We're seeing start-ups going from zero to product ship in months, and from product ship to installed bases of hundreds of thousands of users in weeks. About a year ago, I had a scary realization:

Today's toy is tomorrow's critical business tool.

We can't predict which ones will be "big," but a significant percentage of the buggy crud that is being written today will become part of our "mission-critical infrastructure." For example, lots of firewall administrators block instant-messenger–type programs because of various security concerns. I remember trying that when HTTP first came out – who'd need that stuff? My prediction is that within a few years, one of those online messenger programs will be an essential business tool, and people will be using it to buy and sell stocks or broker mergers and acquisitions. Because of the huge legacy installed base that will be in place by then, it'll be too late to add security into the application's protocols. I now believe it is impossible to accomplish security in such an environment – all we can do is struggle valiantly on the sharp end of the hook.

In order to make the situation improve we'll need to somehow completely revamp the way in which we develop software, publish it, and distribute it. We'll also need social change – the way we treat hackers and think about hacking is going to have to change. Another step in the right direction would be to scrap most of our applications base and replace our core software infrastructure with something simpler and better designed. None of that is going to happen, possibly ever. I don't want to contemplate the kind of disaster we'd need in order to trigger the degree of social change that is necessary – it'll take some kind of massive "software Chernobyl" and a couple of iterations of overreaction before we ever progress in the right direction.

That's a kind of depressing view of the situation, I realize. I guess that the news about the massive denial-of-service attacks has put me in a pretty negative frame of mind about the state of security. This kind of nonsense cannot continue.

Some Suggestions

Let me try to be a bit more positive with some suggestions for simple things you can do to help improve the state of computer security from the comfort of your workstation. One is simple and has to do with bug fixes; the other is more of a management issue and has to do with documentation and design.

One area I don't think we've adequately tapped is intrusion detection in applications. That's because the only people who can meaningfully add application intrusion detection are the developers themselves. It's such an obvious idea that I'm really kicking myself for not thinking of it until recently; this is something we should have been promoting for a long time. If a security flaw is identified in your code, don't just put a patch in place to fix the problem; put a patch in place that identifies attempts to trigger the problem. Log the attempt and block it. If we all did this in our code, we'd begin to collect extremely detailed information about who is trying to exploit specific vulnerabilities. This approach would also be pretty safe against false positives generated by legitimate scanning tools – such tools typically don't try to exploit the actual vulnerability, but simply test for its existence. It's free, cheap, utterly reliable intrusion detection with no performance cost. About the only drawback I see to this idea is that some bodies of code would be dramatically enlarged. Not coincidentally, a lot of them are already hugely bloated – so what'd be the harm of adding a few thousand more lines of code to a Web server, browser, or mail-transfer agent? There'd be a bit of a maintenance cost, but it's not too large.

Second, please stop making your applications identify themselves obviously over a network. While it's essential that protocol negotiations still work, it is not necessary to announce what version of a particular server you are running. Right now, huge numbers of applications effectively paint a target on their backs by announcing to the world at large, "Hello! I am whateverdV1.22!" Even the lowliest script kiddie can use such

The way we treat hackers and think about hacking is going to have to change.

An interesting thing happens when you hit a security-illiterate person with a tool that makes them realize how often they are probed and examined by hackers: they get furious.

obvious targeting information to search rootshell.com for vulnerabilities. Without identifying information, they'll have to use more sophisticated techniques such as stack-fingerprinting approaches applied to applications. That'll be okay because when someone starts fingerprinting your applications, that will be easier to detect (especially from within the application) and is more clearly hostile action. Remember, the more intrusive we can force probes to become, the easier they are to detect and the harder they are to laugh off as "innocent curiosity."

Imagine what would happen if a majority of applications were self-instrumented security alarms. Within an hour of installing your new Web server, you would begin receiving notices from the server of all the attempts being launched against it. My guess is that larger sites would get dozens of warnings a minute. The great part is that the hackers wouldn't really be able to tell if you were running a version of a server that was tattling on them, or a vulnerable version. Let's just make it a little harder for them, shall we?

Of course, I have a stealthy agenda in proposing this concept: it would stun people if they realized how often they come under attack. An interesting thing happens when you hit a security-illiterate person with a tool that makes them realize how often they are probed and examined by hackers: they get furious. If you're careful about how you do it, they don't get mad at you, they get mad at the hacker, complain to their ISPs, the police, the press, etc. That's the level of attention and frustration that leads to "there ought to be a law" thinking – a necessary part of the process of adding pain to cause a backlash.

Another thing developers can do is document whether their applications are safe for use in a privileged context. This is a minor point, but it might help. I've been amazed to see developers employ tools that had huge security flaws as components of critical applications. "But it comes with the system," they cry when someone points out the problem. That complaint is usually followed by "It's too late now, we have to push this into production and it already works." No, grasshopper, it merely appears to work. I'd like to see applications like ftp come with a warning label on them saying, in effect, "Not for use over public networks." We have to raise awareness of these constraints before the point at which system developers decide what components they will use to build their architectures. I am reminded of BSD's approach to getting rid of the obsolete and evil `gets()` function call: they modified it to print a message to the screen when the function was called: "WARNING: This program uses `gets()` which is insecure." Perhaps we need to have applications like ftp check and see if the destination is off the local subnet and print a warning like: "WARNING: This program uses insecure networking and should not be used off your local subnet." On the surface that seems like a decent idea, except that, of course, virtually every application would be printing terrifying warnings. Such warnings are more food for the eventual backlash.

One thing that's always frustrated me is the relative ease with which these broken applications could be upgraded. The only thing stopping us is the Evil Demon of Backward Compatibility, who grows stronger every day as tens of thousands of new users flock to the Internet. Eventually, I believe we will need to slay the demon by scrapping whole chunks of redundant protocols and layering them atop newer, better-designed ones. For example SSL, with whatever warts it may possess, is still a much better protocol, security-wise and in terms of port usage, than ftp. Someone could develop a new version of "ftp" that had the same user interface but that used SSL as an underlying protocol. Many users would never notice. The same could be done with telnet, rlogin, rsh, etc., etc. This would improve security not just in the obvious way (getting rid of passwords

crossing the Net in the clear) but by reducing the amount of redundant networking code that can be implemented incorrectly. Perhaps this might be something the open source movement could take on as part of general legacy code cleanup . . .

The Weather Sure Is Big Out

Well, I fear I may have rambled a bit. In my last column I promised that I'd try to keep things more technical in this column. I hope I haven't completely reneged on that promise. From where I sit, I'm becoming convinced that security isn't really a technical problem – or, more precisely, that the technical component of security is vanishingly small compared to the political, management, and financial components of the security landscape. Large weather out; looks like it's going to rain . . .

In the same vein as last column, I will propose another contest (prize will be a nifty keen "Network Police" jacket with a T-shirt for the runner-up). The topic of this contest is "Where do packets go when they die?" Be brief.