answer could, in fact, come from any machine, anywhere.

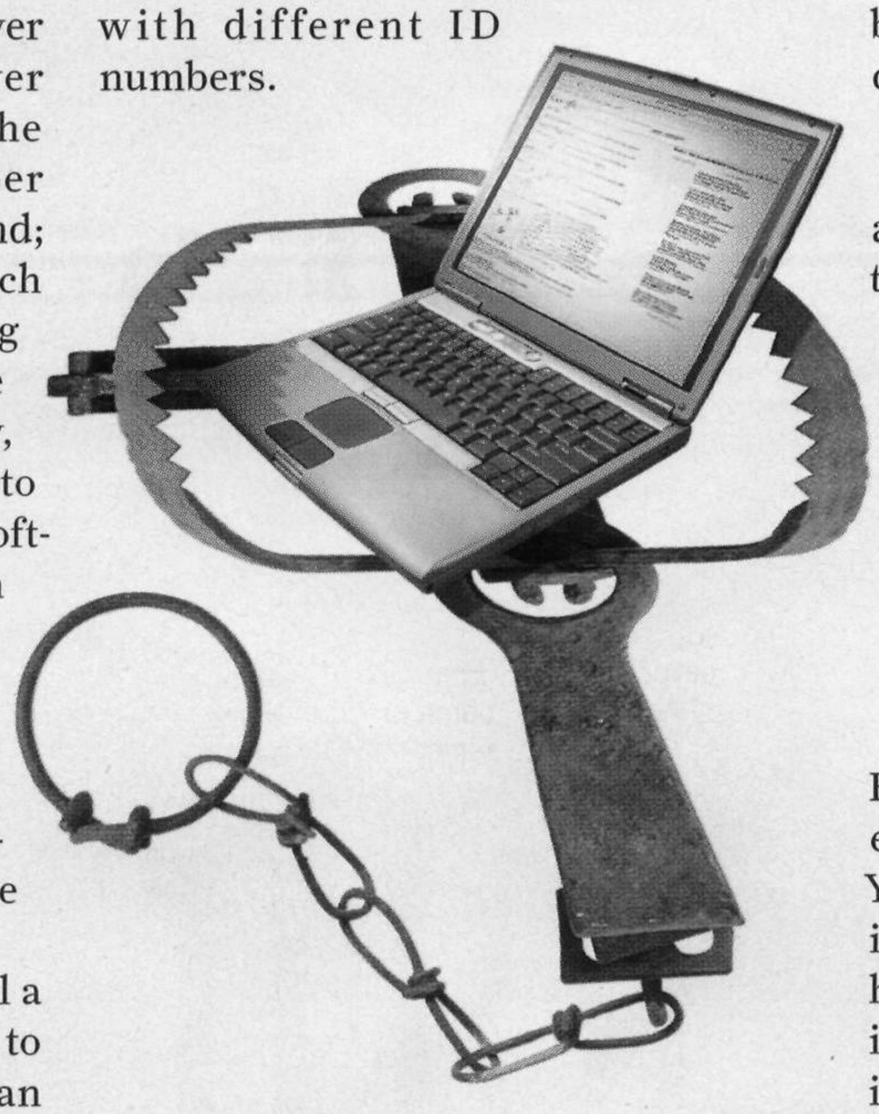
Well, almost—a few systems work differently. Anyone accessing sites in Sweden's .se domain, for example, can use a secure extension to DNS called DNSSEC (for Domain Name System Security Extensions) to carry out DNS lookups. But DNSSEC hasn't really caught on, in part because it is cumbersome for name-server administrators to manage. That attitude now seems due for an adjustment. "There certainly was a spike in interest in DNSSEC" after Kaminsky made his discovery known, says Cricket Liu, a DNS expert at Infoblox, a Santa Clara, Calif., provider of DNS hardware and software.

Still, most communication between name servers continues to take place over an insecure channel. The receiver does check the origin of an answer using a numeric tag attached to the request. But this query ID number wasn't designed with security in mind; it was intended originally just to match up outgoing requests with incoming answers. Early on, DNS software assigned those numbers sequentially, which made it easy for an attacker to guess them. Since about 1997, DNS software has been configured to assign those ID numbers randomly to make such attacks harder. But the ID number has only 16 bits, which translates to 65 536 possible values—few enough to give an attacker a reasonable chance of guessing right if he can try thousands of times.

And that, it turns out, is easy. All a hacker needs to do is send a request to the name server asking it to look up an IP address it doesn't have cached. Then he immediately bombards the server with answers, each containing a different query ID number. Many of the fake replies will beat the real answer back to the name server seeking the address. One of them, the attacker hopes, will contain the correct query ID. More sophisticated versions of this second type of attack don't have such long odds: one scheme, for example, takes advantage of the fact that nominally "random" query IDs can to some extent be predicted.

Up until now, guess-the-query-ID attacks hadn't been considered much of a menace, because the hacker would very likely fail on the first attempt, and the name server would then store the correct IP address in its cache, typically for a day or so. Thus, if the first such attack failed, the hacker would have to wait a day or more to try again. If he had to attempt this hundreds of times before achieving success, it could take years to poison the name server's cache.

Kaminsky's insight reveals how an attacker could sidestep that problem. Imagine that a hacker asked your ISP's name server to look up a nonexistent address within the paypal.com domain—for instance, aaa.paypal.com. This nonsensical name would, of course, not be cached in your ISP's name server, so it would pass the request up the line, eventually asking the real PayPal's name server for the corresponding IP address. While all that was going on, the attacker would send your ISP's name server a lot of spoofed answers with different ID numbers.



Because the hacker has a head start in the race, many of his simulated DNS responses will beat the real one back to your name server. If, for example, 100 fake answers arrive before the real one, the attacker's chances of having one of his bogus responses accepted improve, from 1 in 65 536 to a more worrisome 1 in 655. Even if all 100 spoofed answers fail to get the ID number right, there's nothing preventing him from repeating this attack as many times as he wants, asking about different nonexistent addresses each time: aab.paypal.com, aac.paypal.com, and so forth. Eventually—typically in about 10 seconds by Kaminsky's estimation—a false answer will be accepted.

What's so scary about someone giving your name server an IP address for, say, pdq.paypal.com? Nothing, of course. But remember that the attacker can add some devastating additional information to his spoofed DNS response—namely, a false IP address for www.paypal.com. Such fakery will pass the bailiwick checks because the extra information is for the same domain as the bogus name that was being looked up in the first place. So the bad address will go into the cache, taking the place of the previous entry for www.paypal.com if one had been stored there.

The 8 July patch makes such an attack much more difficult—though not impossible. It works by randomly varying not just the query ID but also the port number in use, which you can think of as being like the suite number on a postal address. To defeat this defense, a hacker must guess both the 16-bit query ID and the 16-bit port number—32 bits in all—requiring, on average, some 4 billion spoofed replies to be received. Such a colossal amount of traffic would be hard to conceal from network administrators.

SECURITY professionals are taking Kaminsky's attack scenario very seriously, and hackers appear to be testing the waters.

Computer-security expert Steven M. Bellovin, a professor of computer science at Columbia University, in New York City, says, "I've seen reports that it's started to happen in the wild." But he believes that DNS cache poisoning will ultimately affect few people—in contrast to what can happen with certain computer viruses and worms—given that system administrators are now taking appropriate steps. He nevertheless cautions, "It's one of the more serious problems we've seen, because it's going after the infrastructure. The hard-core bad guys are starting to wake up." ❑

TO PROBE FURTHER *Paul Vixie's pioneering report "DNS and BIND Security Issues," which warned that the Domain Name System was vulnerable to attack, appeared in* Proceedings of the Fifth USENIX UNIX Security Symposium, *Salt Lake City, June 1995.*

A full description of how the Domain Name System works is available in DNS and BIND, *Fifth Edition, by Cricket Liu and Paul Albitz, O'Reilly, 2006.*

VIKTOR KOEN