academic underground" (http://www.caughq.org)—precisely the kind of thing Kaminsky had hoped to forestall.

Whereas some network administrators may initially have been reluctant to patch their systems, fearing that the upgrade itself might cause problems, most of them now seem to have made the change. No definitive tally is available, but Kaminsky has created a tool on his personal Web site (http://www.doxpara.com) that allows visitors to check whether the server they are using has been patched. He reports that as of 9 July, about 85 percent of the name servers being tested were vulnerable. But by 6 August, the proportion had dropped to 30 percent.

But even those who have taken the appropriate steps are not exactly breathing easy. The patch is not a perfect countermeasure, as Kaminsky has emphasized on his blog: "This is just a stopgap—we're still in trouble with DNS, just less."

IF YOU FOLLOW computing at all, you know that security experts routinely uncover software glitches and vulnerabilities and then issue software patches and upgrades. What Kaminsky has found, however, is much bigger and much scarier.

To understand why, you need to know the basics of how the DNS works. The Domain Name System is essentially the Internet's phone book. It's a huge database containing the 32-bit numeric codes that identify every single site on the Internet. These are known as Internet Protocol addresses, or IP addresses for short. Amazingly, this database is distributed over some 12 million computers worldwide, known as DNS name servers.

When you type "www.google.com" into your browser, it must translate that human-readable text into an IP address before it can access the site. To do so, your computer sends a request to a name server upstream, probably one maintained by your Internet service provider.

Then, if your ISP's name server has the IP address for the requested site stored—or "cached"—it returns this information to your computer pronto. If not, it goes through what may be an elaborate process querying other name servers to find the address.

Kaminsky has discovered a way for a hacker to insert a false IP address into the cache of a name server. The hacker could, for example, change the name server's entry for "www.paypal.com," thus blocking access to PayPal, or worse, duping people into going to a site that mimics PayPal's. From there, it would be relatively simple to harvest user names, passwords, and other valuable data.

Such an attack would work much like the many "phishing" scams now plaguing the Internet, but in this case the victims wouldn't need to click on a link in a shady e-mail message. They could type the correct name, "www.paypal.com," directly into a browser and still get sent to the wrong place. (The real PayPal would upgrade the security of the connection from http to https, but the victim may easily fail to notice when this doesn't happen on the scammer's simulated site.)

The attacker could also use this tactic to redirect e-mail. By replacing the IP address of, say, a corporate mail server with the IP address of a mail server that he controlled, he could inspect incoming correspondence before passing it on to the targeted company's mail server. Even more troubling, he could add his own malicious code to e-mail attachments, which from the recipient's point of view might appear to come from known and trusted sources.

Security experts had long been aware of two general ways that a hacker could carry out such a "cache poisoning" attack on a name server. But both had been rendered ineffective years ago with changes to DNS software. Kaminsky, however, has found a way for a hacker to circumvent these fixes—and to combine the two exploits in a way that makes an attack especially potent.

THE FIRST KIND of attack causes the targeted name server to query a second name server, one that the bad guy controls. That turns out to be incredibly easy to do, even if the name server to be poisoned is behind a corporate firewall or otherwise protected from outside access.

Suppose this hypothetical villain creates a Web page that contains a description of Mother Teresa—perhaps an eBay ad for a copy of her definitive biography. Unbeknownst to you, the page includes an embedded image that is hosted on a machine in the hacker's evil domain, BadGuysAreUs.com. So when you access that eBay page seeking to purchase a book about Mother Teresa, your browser sends out a DNS query to look up the IP address of BadGuysAreUs.com.

Assuming the hacker doesn't try this too often, the address won't be in your name server's cache, so it will issue a series of queries to other name servers. Eventually, your name server will be referred to the bad guy's name server, which responds by supplying the requested IP address. Now here's where it gets ugly: the Domain Name System allows the answer to include additional information. The bad guy's name server could thus be programmed to send a false IP address for any other site—such as Citibank (http://www.citibank.com)—along with the requested IP address. The fake address would then take the place of the bank's real IP address in your name server's cache, where it would act to redirect traffic from anyone trying to use that server to reach www.citibank.com.

The ability to tack on additional information in a DNS response was considered a valuable feature when the Internet was first set up—it was designed to provide the IP addresses of name servers referenced in the main part of the response. At that time, which predates the Web by many years, nobody thought much about the possibility of scammers using this mechanism to take advantage of folks purchasing things through an Internet auction or doing their banking online.

To counter such mischief, DNS software was changed about a decade ago to do what is called bailiwick checking. With that, any extra information added to a DNS response is ignored if it pertains to a domain that is different from the one that was asked about in the first place. So your name server would disregard an IP address said to be for www.citibank.com if the original query was about BadGuysAreUs.com.

The second way an attacker can poison a name server's cache relies on the fact that the conversation your computer has with the name server upstream—or the conversation between two name servers involved in answering your query—is fundamentally insecure. One computer sends out a request to another and then waits for an answer from it. But the