

---

By DAN KAMINSKY

---

# EXPLORATIONS IN NAMESPACE: WHITE-HAT HACKING ACROSS THE DOMAIN NAME SYSTEM

*DNS cache scanning across a sample set of more than 500,000 name servers revealed the extent of last year's Sony rootkit infestation on client machines.*

**I**t's a fact that the larger the data set, the more difficult it is to update any individual entry within it and the more likely an individual record will become out of date. It's from this observation that the Domain Name System (DNS) was created to cope with the growing lists of domain names (such as `www.doxpara.com`) that need accurate mappings to Internet Protocol (IP) addresses (such as `209.81.42.254`). Originally designed in 1983 by Paul Mockapetris, then of the Information Sciences Institute of the University of Southern California, DNS offers a scalable, hierarchal, distributed approach to name resolution. DNS remains a core component of the Internet, but something has changed over the years. As the Internet has grown, attackers anywhere have become attackers everywhere, in part because of DNS.

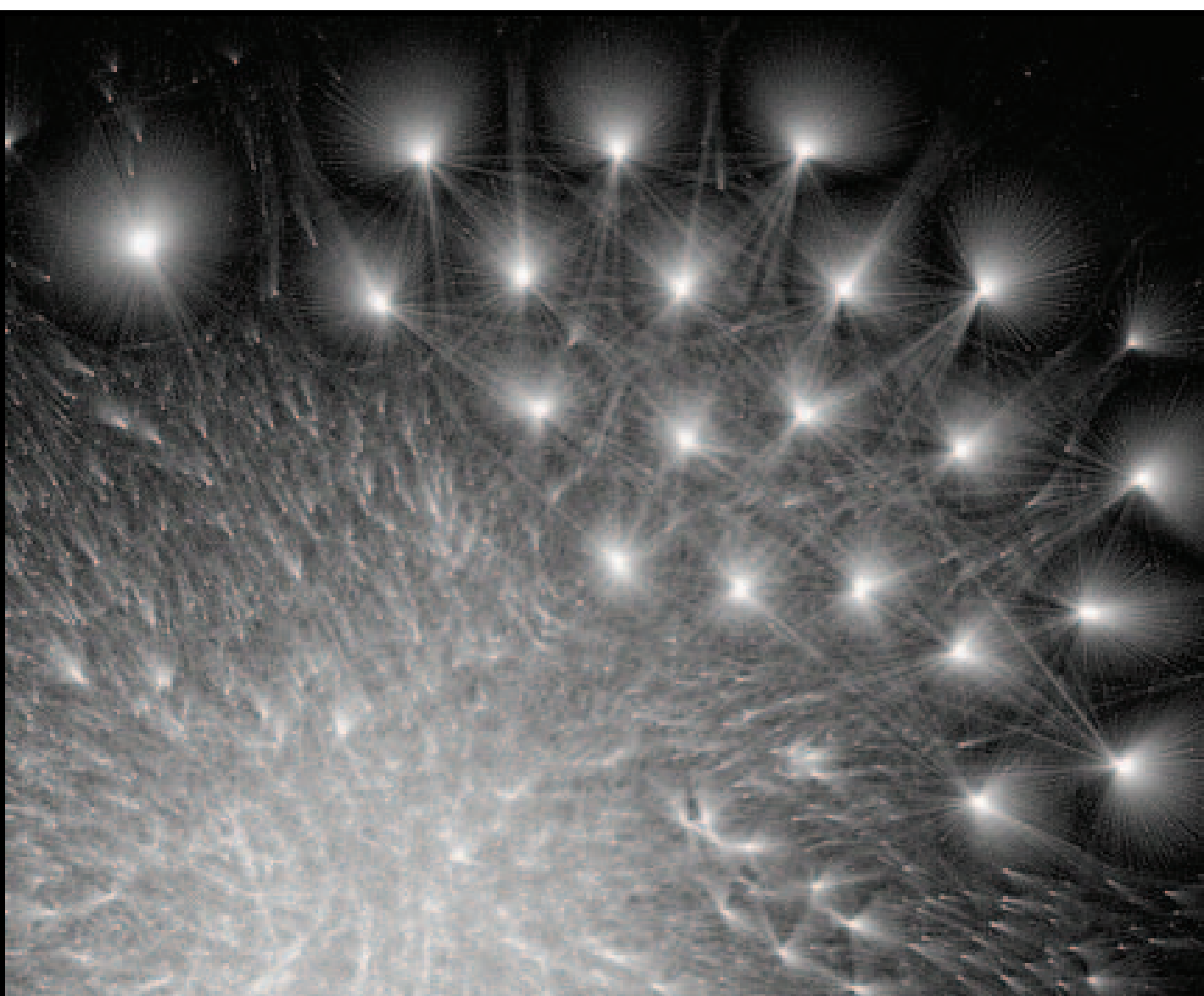


Figure 1. Unexpectedly complex inter-server forwarding relationships in the Domain Name System, Fall 2005 (data and visualization Dan Kaminsky).

My personal interest involves identifying and reporting these security faults on a global scale. The results have been surprising, even to me, considering that I look at every kind of DNS-related traffic I can access on the Internet; DNS is much more complicated than I thought, and name servers most certainly do not always tell the truth.

My scans for these faults have not been trivial. Code is easy, politics is not, particularly when dealing with organizations responsible for network infrastructure. But we cannot defend our networks if we are blind to what they contain. DNS is a massive, distributed, shared worldwide resource. We must not cede knowledge of it to those who wish to destroy it. DNS is one of the few remaining pre-firewall-era applications and may be the only one on the Internet today against which administrators of firewalls refuse

to interfere. From a firewall perspective, the configuration “problem” of properly securing DNS is not like the configuration problem with the File Transfer Protocol (FTP). FTP uses a port-selection strategy not used by other protocols; it is therefore only moderately at risk from devices in the middle of networks that seek to censor and alter traffic flows. More important, FTP fragility problems are limited to FTP transmissions.

On the other hand, DNS is a prerequisite service for almost every network-enabled codebase in existence. When something breaks the domain-name-to-

## THE INTERNET DOES NOT ROUTE NAMES, AND SONY'S ROOTKIT DID NOT HARD-CODE IP ADDRESSES. THE LACK OF HARD-CODED IP ADDRESSES MEANT (AHA!) DNS LOOKUPS WERE REQUIRED FOR THE ROOTKIT TO CONNECT BACK TO A MASTER SERVER.

---

IP-address service provided by DNS, the result is that random, highly visible services (such as email and Web browsing) begin to fail. Time and again, it takes so long for network administrators to trace the problem back to DNS—hindered by blocking libraries and poor runtime debugging—that by the time the problem is identified, the firewall vendor, rather than DNS, is often blamed instead. Firewall vendors fear inhibiting DNS in any way, while system administrators tend to feel that DNS is more likely to break if they touch it. So name servers chug on, providing their services as they always have.

Minimal oversight from system administrators and firewalls leads to other problems. For example, in March 2005, a number of name servers started returning bad data for Google servers. Attackers had exposed the servers to a variant of what's called a Kashpureff attack. Named after the erstwhile Eugene E. Kashpureff, whose zeal for alternative DNS top-level domains in 1997 exceeded his fear of deportation and eventual prosecution by the U.S. government. The Kashpureff attack on Google's servers involved sending more domain-name-to-IP-address mappings than had originally been requested; the name server would then cache these additional answers alongside what had actually been requested.

What made this variant interesting to security professionals was the fact that name servers are often not set up to operate independently but can be configured to run in fairly complicated trees and graphs that query other nodes before escaping out to the DNS root servers for resolution services. This preference for local nodes, called a "forwarding relationship," involves trusting that whoever you were speaking to will pass you only appropriate data (see Figure 1).

The new attack involved the Berkeley Internet Name Domain (BIND) (version 8) implementation of the DNS protocols, which didn't trust Kashpureff-polluted packets but would forward them unfiltered anyway. Other name servers (Microsoft DNS was targeted, though others were vulnerable) would trust anything forwarded to them. The attackers had revived an old technique by identifying and exploit-

ing it in new architectures.

Resurrecting old techniques is common in today's security environment; why find new bugs, when the old ones can be used to locate architectural mistakes programmers are likely to make. Regression testing for security flaws needs more work to ensure that already-identified security flaws do not make their way into new software applications. Luckily, the attackers didn't hit too many name servers. But how much damage could they have done had they succeeded on a global scale?

### CHALLENGING SCAN AND NOTIFY

Two things are not generally recognized by network administrators and governance bodies as being part of the status quo of operating on the Internet: servers will be scanned, and users who detect servers that appear to have been compromised can notify anyone else of their discovery.

Computer security professionals generally know that the average lifespan of an unpatched system on a network without a firewall can be measured in minutes. Less known is that at least one Class A network (16 million IP addresses) is constantly receiving 5MB/sec of traffic. But there's nothing there, because no machines reside at any of the 16 million addresses, and the addresses should not receive any traffic. From this we can surmise that unknown individuals and organizations are continually scanning the Internet's entire IP address space. Since "white hat" hackers do not break into systems as a matter of course, and systems are indeed being broken into en masse, we know that the "black hats" are largely doing the scanning. Whereas black hats ought not to have better intelligence than we do, white hats must be able to scan at least as effectively as their black-hat counterparts.

When it comes to notification, things get interesting, because the great advantage white hats have over black hats is they don't have to hide. Indeed, on my own scanning node ([deluvian.doxpara.com](http://deluvian.doxpara.com)), you'll even find my cell phone number. Because we white hats have no fear of being caught, we don't need stealth. White hats should push this advantage as

much as possible. Not only would you be notified by them that your scans have been noticed but that your servers and network hardware may have been broken into.

Black hats have long histories of breaking into systems, recruiting them for information gathering and using them as stepping-stones to exploit other third-party systems. A common line of communication involves an administrator who notices a scan from an external network, then informs the network's administrators that they have a compromised machine; remediation can then begin. Notification is part of the status quo.

My scan-and-notify proposition involves a caveat. While both scans and notifications are already taking place on the Internet, scanners generally do not notify the servers they are scanning, and notifiers are generally not also scanners. I intend to change the role of scanners to help improve large-scale Internet security. The security community's advisory model—in which recipients are obliged to evaluate their networks against the constraints documented in the advisory—doesn't work well. It also hasn't helped that it's become so difficult to acquire tools capable of verifying vulnerabilities.

Would it help to have site-customized advisories able to state that not only is there a vulnerability but that the scanners can see that your site is affected? Possibly, so I thought it would be worthwhile to find out. In 2005 after setting up a suitably uncloaked scanning host (with custom whois entries, reverse DNS, and my cell phone number being available via HTTP) I began trying to document the extent that other DNS servers across the Internet were vulnerable to a Kashpureff attack.

#### **SHEDDING LIGHT**

My first step in the scan-and-notify process was to sweep the IP space. I modified my high-speed TCP service scanner, Scanrand ([www.doxpara.com](http://www.doxpara.com), part of the Paketto Keiretsu suite of tools for manipulating TCP/IP networks), to emit User Datagram Protocol (UDP) packets on the standard DNS port—53. Although TCP scanning could have detected servers, many name servers fail to respond on TCP/53, suggesting a UDP approach was warranted. But which packets should be sent? Unlike TCP-based services, applications that expose their services over UDP must participate in announcing their presence. The message delivered to them must be understood by the DNS server, and the server must emit a response. So a DNS query is required, one that would be understood and responded to by any host receiving the request, preferably without

having to travel elsewhere in the DNS hierarchy. I then considered how to construct this query, ultimately using localhost (127.0.0.1), a special IP address machines use to refer to themselves.

I constructed a DNS PTR (pointer) lookup query for 1.0.0.127.in-addr.arpa. It asked the DNS server to return the domain name associated with a given IP address, a process also known as a reverse lookup. In theory, each server that receives the request should return "localhost" as a reply. In most cases, this was exactly what occurred in response to my query, but in a surprising number of cases the servers leaked additional information by announcing themselves as localhost.foo.com, where foo is the name of the domain associated with the DNS server. I detected other response patterns as well. One conclusion I made while doing these scans was that when scanning for a particular Internetworked application, domain-specific probes can yield more data than might otherwise be expected.

Another result of my scans was that I witnessed a second class of DNS traffic. I configured my DNS server to allow anyone attempting to lookup the IP address for my scanning computer (deluvian.doxpara.com) to receive a name directing this person to more information, including scanning.please-browse-to.http.deluvian.doxpara.com. Approximately 35,000 individual servers took me up on this request, many immediately after I probed their related networks. It is likely that automated systems—some designed for forensic logging—were announcing their presence to me, despite my being an unauthenticated party. I saw at least one packet from a Class A network from which I had never seen anything else. But more probable was that some of this traffic was coming from active forensics personnel trying to determine what I was up to.

I initially found nine million hosts, not all willing to keep talking to me. As my work progressed, I selected a 2.5-million-node subset for manageability purposes and extracted name-server version data from each host using Roy Arends' excellent DNS fingerprinting software ([www.rfc.se/fpdns/](http://www.rfc.se/fpdns/)). What remained was the difficult part of my search for servers still vulnerable to a Kashpureff attack—detecting and documenting forwarding interrelationships among DNS servers.

#### **PEERAGE**

All known mechanisms for reasonably detecting interrelationships among name servers rely on the fact that when one name server forwards a request to another name server, this other server ends up with the response in its cache. Luis Grangeia, a systems

and network auditor, wrote in [1] that an obscure mode of DNS, accessible via the Domain Internet Groper tool, allows cache contents to be probed nondestructively. Therefore, cache contents can be, at least to some degree, publicly monitored.

This mode of DNS offered three approaches for identifying interrelationships. First, I could issue a normal (but unique) query to one host, then nondestructively see if the results from that query showed up anywhere else. If, for example, I had asked Alice to retrieve data for me, and Bob ended up with that data, I could be pretty sure Alice and Bob were talking to each other. The problem was we weren't just talking about Alice and Bob; I wished to test millions of name servers on the Internet—an  $O(N^2)$  solution—meaning my approach would quickly become computationally infeasible.

An interesting variant of this approach is to recognize that old data can be differentiated from new data; it's possible to know to the second when a given record entered a particular host. We are able to determine such information because each record is associated with a Time To Live (TTL) value describing the number of seconds until the data is considered stale. Each node in the DNS is specified to decrement TTL, so it's trivial to subtract the original TTL from the retrieved value and determine the precise time a given value entered the cache.

This approach also had trouble dealing with complexity; for example, to scan more than a single node per second, the scanning machine would have to constantly shuffle its scan space, then identify relationships that show up at matching offsets, no matter what the order of the scan may be. This processing overhead is doable but can be prohibitively complicated.

I eventually implemented a more straightforward solution. In DNS, I can control parts of the namespace; if anyone in the world tries to look up something inside the doxpara.com domain, the packets come to me. So, I began probing DNS servers by asking them to look up a unique name, then monitored who came to me to service that request. If, for example, I asked Alice to look up something, and Bob came to my server hat in hand with a lookup in tow, that would mean Alice was talking to Bob. I used this simple solution to detect several forwarding relationships:

- 13,000 Microsoft DNS servers forwarding to BIND8;
- 18,000 BIND9 forwarding to BIND8; and
- 230,000 total name servers (almost 10% of the detected sample set) forwarding to BIND8, in direct contravention of the security disclaimer on the BIND home page. (Apparently, a very visible

security advisory on BIND's Web site wasn't enough for a number of people to properly configure their machines.)

This probing produced yet another inexplicable result. After generating two kinds of information—the type of nameserver software run by each node and edge information, or which nameservers were forwarding to which other nameservers—I tried to generate the full graph for the analyzed subset of the global DNS architecture. Building the graph, I mainly expected to find clusters of four or five nodes in some sort of cooperative arrangement for DNS lookups. This was indeed the case for the 40,000 networks I had probed, but I then found an anomaly consisting of 220,000 nodes, 330,000 edges, and a depth of 20. In one case, a DNS request sent to one particularly interesting server yielded traffic from one of a thousand other systems. I am still analyzing this result.

### LIVE IN YOUR WORLD; GET PWNED<sup>1</sup> IN OURS

During the late 1990s, the term “Halloween documents” referred to embarrassing leaks from Microsoft. More recently, during Halloween 2005, Mark Russinovich, author of a number of advanced Windows utilities, highly regarded technology blogger at [www.sysinternals.com](http://www.sysinternals.com), and probably one of the top five Win32 hackers in the world not working for Microsoft, documented how in the development of his RootkitRevealer tool, he'd found that his own system was infected by a rootkit. A rootkit is software that allows malicious applications to operate stealthily. He'd traced the matter back to a Sony CD he'd purchased; using “sterile CD” software from the U.K.'s First4Internet, a commercial provider of digital rights management (DRM) solutions ([www.first4internet.com/](http://www.first4internet.com/)), Sony had caused an indeterminate number of PCs to lose some of their ability to rip CDs into applications like Apple's iTunes or burn their own mix CDs.

This software cloaked itself using mechanisms so deep in the black-hat playbook it could be referred to as a rootkit. Indeed, black hats soon began borrowing access to its cloaking functionality to evade antivirus and antispy software. Worse, the initial uninstaller provided by Sony removed only the cloaking and not the DRM software—hardly the total removal users were looking for. It became clear to me that this code was designed to expect the loss of consent to execute. At some point, the rootkit developer must have asked himself whether users would want to get rid of it. The

<sup>1</sup>Pwned (Owned) is Internet slang for gaining root privileges on a computer by exploiting a security vulnerability (see [en.wikipedia.org/wiki/Pwned](http://en.wikipedia.org/wiki/Pwned) for more information).

answer, it seems, was “not if they don’t know it’s running.”

Such behavior from a company like Sony posed a problem for the security community and everyone else. It’s difficult to fight skilled hackers out for fun. It’s difficult to fight experienced, financially motivated criminal operations. But fighting a billion-dollar, corporate-funded hacking operation is impossible. Even the most experienced infantry will fall to aerial bombardment, and with four million CDs being sold directly to consumers, these CDs were a bombardment we (security experts throughout the Internet) hadn’t yet noticed; it took six months and Mark Russinovich to detect the presence of the rootkit. Where were the security vendors? Only Sony knew how many systems were infected. How could the security industry respond to something it could not measure?

I needed hard data that would allow me to measure the impact of the infection. Traffic dumps from rootkit-infected systems showed network connectivity to `connected.sonymusic.com`. While I did not expect traffic to this domain to be only rootkit-related, other researchers told me that `updates.xcp-aurora.com` had been identified as another suspicious domain.

The Internet does not route names, and Sony’s rootkit did not hard-code IP addresses. The lack of hard-coded IP addresses meant (aha!) DNS lookups were required for the rootkit to connect back to a master server. The lookups would execute, the responses would cache, and I could use the DNS cache snooping method [1] across my entire sample set to estimate penetration levels by the Sony rootkit. However, using this technique, I was unable to get per-host results, but the per-network results I could get would give me an idea (at least) of the scale of the rootkit infections. My analysis showed more than 500,000 name servers responding to nonrecursive queries with one of the two monitored names. This was rather more than I expected.

Over the next few days, I weeded out false positives through two approaches:

- Many servers look up data recursively even if they’ve been instructed not to. Originally, I found

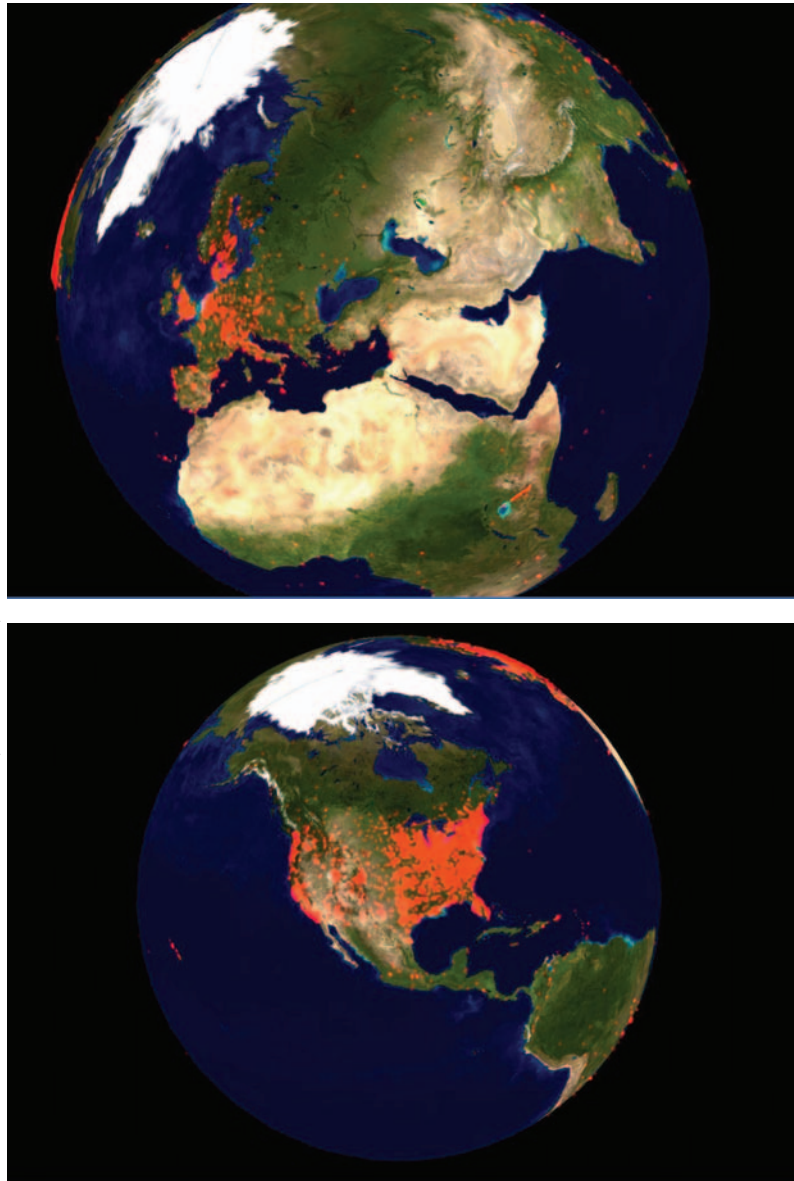


Figure 2. Geolocation of Sony rootkit infection for Europe (top) and the U.S. (bottom), November 2005.

almost a million hosts responding; due to the fault, I had to eliminate 350,000; and

- Some servers lie. Captive portals, commonly used in wireless portal applications, intercept all incoming DNS queries and return with links to themselves. Along with many recursive lookers, these portals were easy to detect, as their TTLs were always equal to some value divisible by 100.

I ultimately found that more than 556,000 name servers had returned names related to the Sony rootkit, with 76% of the servers returning both `connected.sonymusic.com` and one of the First4Internet addresses. Geolocation allowed me to identify infected nodes in 165 countries around the world (see Figure 2). It was clear that security researchers had a



new weapon—DNS—in the war against malware and that Sony had been its first target.

Even so, matters were still not clear. For example, `connected.sonymusic.com` was in fact an address used for almost eight years by Sony Enhanced CDs (but not rootkitted). Only gawkers and uninstallers went to the First4Internet sites; the rootkit itself would not natively go to that address. Security experts around the world were concerned (despite the CDs supposedly being sold only in the U.S. and Canada). Luckily, a breakthrough emerged that allowed new, much more accurate scans. In it, `connected.sonymusic.com` would issue an HTTP redirect to another site—`xcpimages.sonybmg.com`—but only in situations where an infected CD was inserted into a computer. Following these redirections yielded far more accurate data; for example, I found that more than 350,000 sites in 135 countries were linking to the `updates.xcp-aurora.com` site. While smaller than the original data set, I completed these new scans a month later (December 2005) after significant press and Sony's own genuine efforts to manage the repercussions of the rootkit infections—for which it deserves credit and respect.

I still wanted to know whether the rootkit was a small-scale event or one of, perhaps, global magnitude. For 350,000 or 550,000 name servers to be so affected, the best available data suggested that Sony had indeed caused an event of worldwide proportions.

## CONCLUSION

Hacking requires a mind-set that seeks to know not what a system is intended for but what it is capable of doing. Hacking is not limited to networks or to those who call themselves hackers; every programmer finds ways to repurpose old tools. White hats will never cede the creative ground to black hats. The greatest white-hat advantage—no need for stealth—exposes creative territory their adversaries cannot match. The results of using DNS as a global measurement platform can now be explored for the first time. ■

## REFERENCES

1. Grangeia, L. *DNS Cache Snooping for Fun and Profit*. White paper, Feb. 2004; [www.rootsecure.net/content/downloads/pdf/dns\\_cache\\_snooping.pdf](http://www.rootsecure.net/content/downloads/pdf/dns_cache_snooping.pdf).

---

**DAN KAMINSKY** ([dan@doxpara.com](mailto:dan@doxpara.com)) is a senior security consultant at DoxPara Research, Seattle, WA.

---