RIK FARROW

musings

Rik Farrow provides UNIX and Internet security consulting and training. He is the author of UNIX System Security and System Administrator's Guide to System V.

rik@spirit.com

IN MY PREVIOUS COLUMN, I POSITED the existence of individuals or groups that could break into computers and maintain access over a period of time without being noticed. While I have yet to actually learn of an individual being charged with crimes related to such trespass (the only people we see being prosecuted are the flamboyant Web site defacers, with a sprinkling of identity thieves), I continue to hear hints of the existence of these subtle invaders. Unlike those who sell credit card numbers for fifty cents each, these crafty groups would have access to IP (intellectual property) worth many millions of dollars or to bidding information, also worth millions.

What I have recently been reminded of are techniques that could be used to pass undetected through firewalls. These techniques permit the subtle invader persistent access to the founts of her wealth, all the while bypassing all the security products arrayed to detect any intrusion. I speak of tunnels that use two of the Internet's most common protocols: DNS and HTTP.

DNS and HTTP tunnels are nothing new. I have written of these in the past, and so have others. In the December 2003 *;login:*, Mudge wrote about techniques for detecting such tunnels (see http://www.usenix.org/publications/login/2003-12/pdfs/mudge.pdf), without actually mentioning any tools that could be used for tunneling.

You probably have heard of some forms of tunneling. Many botnets use connections to IRC channels as a method to communicate with the thousands of zombies attackers may control. IRC provides several advantages in that it can be used anonymously (through IRC relays) and that each zombie controlled through IRC connects outward through any firewall, an operation that is often permitted. If the firewall has been properly configured, outgoing connections that have not explicitly been permitted by an organization's policy will be prohibited, a restatement of an old saying by Ranum. But what of application protocols that will be permitted through a firewall by policy?

HTTP

HTTP provides the most obvious example. Not only has the Web become the favorite hole through the firewall for attackers, HTTP can also be used for remote communications. HTTPTunnel, which has been

around for many years (http://www.nocrew.org/software/httptunnel.html), provides a simple and easy method for tunneling protocols that might otherwise be blocked by policy and the properly configured firewall. HTTPTunnel converts requests from the client side of the tunnel into conforming HTTP PUT requests (that actually contain base64 encoded data). The server side of HTTPTunnel converts these requests back into IP packets, and converts responses back into legal-appearing HTTP replies. HTTPTunnel allows an internal user to communicate with an external IMAP or SSH server, even when the firewall would otherwise prevent this communication.

A similar technique, hinted at in Mudge's document, can be used to present a command prompt to a remote user. Instead of tunneling another service, this use of HTTP relies on a local server that makes routine requests of a remote server. Each request appears to be a legitimate HTTP request but results in a command prompt being issued at the remote server. The remote user then can enter a command that gets sent as an HTTP response to the local server. The local server executes that command and packs up the results as another HTTP request to the remote server. Simple enough, and although I cannot personally point you to a working version, I have no doubt this tool exists.

DNS has also been used as a tunnel. The NSTX tool (http://nstx.dereference.de/nstx/) provides a simple and slow tunnel using DNS that is not terribly exciting. Dan Kaminski, during his talk at BlackHat 2004, demonstrated a set of much more interesting tools, ones that you can download and try out yourself (http://www.doxpara.com/).

Kaminski stirred up a bit of a fuss while at CanSecWest (a security conference in Vancouver) by explaining how public DNS servers could act as a distributed network of file servers. Kaminski calculated that his scheme, called Domaincast, would require some 20,000 servers to hold the contents of a single ISO image. Although each server would only serve up 256 bytes, it would be possible to download 700 MBs within a reasonable time using this technique. If you are wondering about finding 20,000 DNS servers, Kaminski claims to have found over 140,000 DNS servers in a single Class A network.

While Kaminski did not demonstrate Domaincast (something the overloaded network at BlackHat would very likely have prevented from succeeding), he did use a tool named droute, written entirely in Perl for portability, to demonstrate tunneling SSH over DNS. Droute converts SSH packets into legal DNS requests, and those requests get converted back into SSH at the server end. The real advantage of this technique is the ubiquity of DNS, its ability to be relayed through firewalls, while its disadvantage revolves around the same data packet size and the use of UDP. But for tunneling, DNS works as well as HTTP.

Just for grins (and Kaminski does grin a lot), he also demonstrated passing an AM talk radio feed over DNS. Using the only codex in the public domain (speex), Kaminski received and converted the talk radio feed into a recognizable audio, all over DNS. For those who didn't want to believe in the efficacy of using DNS for tunneling, this silly example was really an eye opener.

By this point, I hope I have convinced you of the reality of tunneling, with ease, through firewalls. Because of the use of approved protocols, firewalls, IDSes, and IPSes have no chance of determining through the examination of individual packets or reconstructed streams of packets that a protocol has been tunneled through HTTP or DNS. The real hope lies in analyzing traffic, as these tunnels will generate traffic that is unlike normal HTTP or DNS traffic in many respects. The HTTP shell tunnel makes outgoing requests to the same server like clockwork and may keep the connection open for a minute while waiting for a command to be entered. An HTTP shell tunnel will also reverse the usual order for packet sizes, as client requests will be larger than server responses.

Kaminski's droute would reveal the tunneling of SSH through the many requests made to the same remote DNS server. If droute were being used for a remote shell, you could detect patterns in traffic similar to those in the HTTP shell tunnel.

These subtle effects would have to be teased out of the usual barrage of network traffic. While Argus (http://www.qosient.com/argus) would be very good at collecting traffic records for analysis, I know of no OS tool that would actually perform the analysis. Mudge, on the other hand, has been working for a company (Intrusic.com) that has a product that can allegedly do this analysis.

I hope that I have, by this time, at least made you a bit uncomfortable. The notion of outgoing tunnels has never made me feel the least bit happy. And while I am on the topic of uncomfortable things, I do want to encourage you again to vote early, as that guarantees a paper record in what may be the pivotal election for the future of the US.