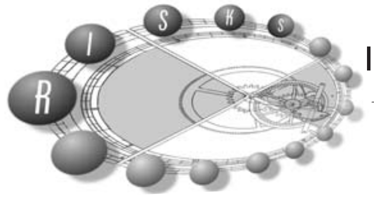

Inside Risks | Stephan Somogyi and Bruce Schneier

The Perils of Port 80

In the months that the Code Red worm and its relatives have traveled the Net, they've caused considerable consternation among users of Microsoft's Internet Information Server, and elicited abundant schadenfreude from unaffected onlookers. Despite the limited havoc it wrought, the Code Red family highlights a much more pernicious problem: the vulnerability of embedded devices with IP addresses, particularly those with built-in Web servers.

Thus far, the Code Red worms work their way through self-generated lists of IP addresses and contact each address's port 80, the standard HTTP port. If a server answers, the worm sends an HTTP request that forces a buffer overflow on unpatched IIS servers, compromising the entire computer.

Any effect these worms have on other devices listening on port 80 appears to be unintended. Cisco admitted that some of its DSL routers are susceptible to denial-of-service; when routers' embedded Web servers are contacted by Code Red, the router goes down. HP print servers and 3Com LANmodems seem to be similarly affected; other network-infrastructure hardware likely suffered, too.

HTTP has become Internet-connected computers' lingua franca. Since Web browsers are effectively ubiquitous, many technology companies can't resist making their product functions visible—and controllable—via a Web browser. Indeed, it seems as if all future devices on the Net will be listening on port 80. This increasing reliance on network-accessible gadgetry will return to haunt us; Code Red is only a harbinger.

Sony cryptically announced in April it would endow all future products with IP addresses; a technically implausible claim, but nonetheless a clear statement of intent. Car vendors are experimenting with wirelessly accessible cars interrogated and controlled from a Web browser. The possibilities for nearly untraceable shenanigans perpetrated by the script kiddie next door after working out your car's password are endless. This problem won't be solved by encrypting the Web traffic between car and browser, either.

The rise of HTTP as a communications common denominator comes from ease of use, for programmer and customer alike. All customers need is a Web browser and the device's IP address, and they're set. Creating a lightweight server is trivial for developers, especially since both in- and outbound HTTP data is text.

Even more attractive, HTTP traffic is usually allowed through firewalls and other network traffic barriers. Numerous non-HTTP protocols are tunneled via HTTP in order to ease their passage.

But HTTP isn't the miscreant. The problem is created by the companies embedding network servers into products without making them sufficiently robust. Bullet-proof design and implementation of software—especially network software—in embedded devices is no longer an engineering luxury. Customer expectation of reliability for turnkey gadgets is higher than that for PC-based systems. The successful infiltration of the Code Red worms well after the alarm was sounded is eloquent proof that getting it right the first time has become imperative.

Given the ease of implementation and small code size of a lightweight Web server, it's particularly disturbing such software isn't engineered with greater care. Common errors that cause vulnerabilities—buffer overflows, poor handling of unexpected types and amounts of data—are well understood. Unfortunately, features still are valued more than reliability. Until that changes, Code Red and its ilk will continue unabated.

One example of doing it right is the OpenBSD project, whose developers have audited its kernel source code since the mid-1990s, and have discovered numerous vulnerabilities before they were exploited. Such proactive manual scrutiny of code is labor intensive and requires great attention to detail, but its efficacy is irrefutable. OpenBSD's security track record—no remotely exploitable vulnerabilities found in the past four years—speaks for itself.

Like sheep, companies and customers have been led along the path of least resistance by the duplicitous guide called convenience. HTTP is easy: easy to implement, easy to use, and easy to co-opt. With some diligence and forethought, it is also easy to secure, as are other means of remote access. HTTP wasn't designed to be all things to all applications, but its simplicity has made it an understandable favorite. With this simplicity also comes the responsibility on the part of its implementers to make sure it's not abused. C

Stephan Somogyi (risks01@st.gyroscope.net); **Bruce Schneier** (Schneier@counterpane.com).

MARTIN MAYO