Global Information Assurance Certification Paper

Is Your Generic Port 80 Rule Safe Anymore?

Steve Riley
GIAC Security Essentials Research Assignment
5 February 2001

The problem

To be succinct: you can't trust your ports anymore. A typical firewall rule set will contain a rule allowing HTTP traffic on TCP port 80. Depending on your environment, such a rule is necessary for:

- Those behind the firewall to access the world-wide web
- Those outside the firewall to access your web server

Unfortunately, it's no longer advisable to trust that all traffic entering or exiting your network over 80/tcp is either a legitimate request or a response to some legitimate request. A search at http://www.google.com for *attacks or hacks "port 80"* resulted in over 7,100 documents.

Potential attacks

Many attacks today can be delivered over HTTP. Some of the attack types include:

- Exploiting buffer overflows in the web server's code
- Obtaining control of a server through installed sample code
- Triggering a fail-to-open-state vulnerability
- Tricking a user into installing a Trojan that runs over HTTP
- Use of port 80 for non-HTTP traffic

Typical packet filtering firewalls—even stateful-inspection types—are powerless to prevent these attacks.

Buffer overflows. Buffer overflows are some of the most common web server attacks today. Both Unix-based and Windows NT-based web servers are vulnerable. Briefly, a buffer overflow attack relies on the fact that in most code, memory allocations aren't bounds-checked. With careful analysis of the code, an attacker can determine how much data to stuff into a buffer to cause the server to execute arbitrary code. Usually, this "arbitrary" code is the attack code. Of

course, determining how to exploit a buffer can be difficult and requires skill. But once an exploitable buffer is found, a simple script can be written to perform the attack.

Patches and workarounds are common ways to address discovered buffer overflow vulnerabilities. Such an approach, however, won't stop the problem: programming disciplines must be changed. There is no excuse, given the performance of modern hardware, for not bounds-checking every memory allocation in any piece of code. A good commentary by Aleph One on buffer overflows is in the references.

A possible attack that relies on a web server buffer overflow and the utility Netcat might occur like this:

1. Via a buffer overflow exploit, run a command prompt with the parameters to tftp back to the attacker's machine to download Netcat to the victim machine.
2. On the attacker's machine, run Netcat listener on port 80.
3. Via the same buffer overflow, run another command prompt, this time Netcatting a command prompt back to the attacker's machine. Perform this over port 80.

The attacker now has an interactive command prompt running on the victim machine. If the victim is Windows NT-based, the user context is IUSR_*machinename;* by running one of many privilege exploits, the attacker can easily obtain administrator access. If the victim is Unix-based and the web server is running as root, the attacker will have obtained root access.

Sample code execution. Many web servers ship with sample code—the various servers for Unix often include CGI scripts, while IIS includes a number of ASP pages. These sample programs are not written with security in mind, and can often be exploited by supplying unexpected input. The best defense against such attacks is to remove the sample code from the server. An example of a sample code attack that reveals private information from Allaire Cold Fusion is in the references.

As with buffer overflows, programming disciplines must be changed to avoid attacks such as the one above. Simple checks for context will eliminate many of these vulnerabilities. For example, if an input field expects a string of digits in a particular format, ensure that the conventions are enforced by rejecting any input that fails to match the expectations.

Fail-open state trigger. One objective of securing a service is to ensure that if the service is subjected to attack, it reverts to a "fail-closed" state. While generally more applicable to firewalls than to web servers, new forms of denial of service attacks may be crafted specifically to trigger a web server into a "fail-open" state. Some administrative utilities commonly run on web servers are also vulnerable to fail-open attacks; see the IRIX Performance Copilot vulnerability in the references.

HTTP Trojans. The original Back Orifice worked only over a certain port. Blocking access was easy: simply block that port. Since newer versions of Back Orifice (http://www.bo2k.com/), plus many other Trojans such as NetBus (http://www.netbus.org/) and SubSeven (http://subseven.slak.org/), now can be configured to listen on any port, an attacker may attempt to get a port 80–aware Trojan over the firewall. Another example: HTTPtunnel, by Lars Brinkhoff, is a client-server utility that allows for the creation of bi-directional tunneling between two computers using HTTP. It would be a trivial exercise to package the HTTPtunnel client portion inside a Trojan which, when run, would connect to an attacker's machine. Since such an attack would look like a regular outbound web request, it would slip past the firewall unimpeded.

Non-HTTP traffic. Foundstone, a security consulting and education company, has released a tool called FScan. FScan permits scanning from any port the user wishes. By indicating port 80 on the FScan command line, it's possible for someone to scan various hosts on the Internet from behind a firewall.

HTTP has recently become popular as a generic transport protocol. Later I discuss why even the methods I present here may not help address security concerns about this unfortunate development.

Why packet filters aren't the answer

A packet filter simply examines traffic at the protocol level. If a rule permits traffic over a particular protocol and port, the rule will allow all traffic over that protocol and port. Basic packet filters have no way of analyzing the payload to determine whether it makes sense given the characteristics of the protocol. For example, in a typical corporate environment, outbound requests with port 80 as the destination will usually be quite small, while the returned information will be quite large. This describes typical web surfing behavior. If an attacker successfully convinced a user to install HTTPtunnel, thereby exploiting the open port 80 to move private data *out* of the network, traffic patterns would be reversed. No simple firewall will sense that this could present a problem.

Stateful-inspection firewalls don't provide much additional security. A stateful inspection firewall would block unsolicited inbound traffic sourced from port 80, since there is no corresponding initial outbound connection request, but would not block an attack delivered over a Trojan if that Trojan makes the initial connection. Or consider a Trojan like QAZ, the presumed culprit in the November 2000 attack of the Microsoft Corporation's internal network. If QAZ were recompiled to run over port 80 and then successfully installed in a DMZ (where unsolicited inbound requests *are* allowed), an attacker would have complete control of the DMZ.

What, then, is the answer?

As with most security issues, the answer isn't a simple one. There are some steps you can take to significantly reduce your likelihood of falling victim to port 80 attacks, however.

Application-level firewalls. Unlike packet filter firewalls, application-level firewalls (also called *proxies*) don't allow IP connections between networks. Proxies operate at the application layer, essentially disallowing direct connections between the networks it's connected to. Some proxies can be configured to be "context-aware"—they examine the traffic according to various rules and can make more intelligent decisions about what is expected and acceptable based on the actual payload. There can be a performance penalty, since application firewalls more thoroughly examine the traffic, but as hardware continues to improve, this will become less of an issue.

Intrusion detection. No modern IP network should lack some form of intrusion detection. While the human brain will always be the most effective intrusion detection system, automated systems can rapidly detect known intrusion patterns and notify a security team about potential attacks. Early notification provides the security team two critical requirements for successful incident response: evidence and time. An intrusion detection system also complements an application-level firewall's analysis of whether traffic conforms to the expected behavior of its protocol.

Is that all? Of course, even these technological approaches won't work if end users actively or inadvertently circumvent certain desktop security measures. Continued user education is a requirement for all end users—security awareness and how it affects business should be the minimum goal. Disabling anti-virus protection, for example, gives a Trojan its initial access into the network. Dual-homed workstations and remote users connected via split tunnel VPNs can present serious threats to the security of the internal network—if the workstation runs a routing protocol, it advertises a direct route to the internal network; it's only a matter of time before some attacker discovers this rogue gateway.

Lately I've become quite fond of personal firewalls that perform active monitoring of outbound connections, such as ZoneAlarm and Norton Personal Firewall. Again, consider QAZ: it hijacks the NOTEPAD.EXE executable. An infected machine will still run Notepad, but each invocation will make a connection to the Internet. Personal firewalls that monitor outbound connections will raise an alert; seeing a dialog with the notice "Notepad is attempting to connect to the Internet" should arouse anyone's suspicions.

There is also the increasingly popular use of HTTP as a transport for all kinds of data. Microsoft's DCOM currently runs over HTTP; other RPC-based applications may soon have the same capability. SOAP pretty much requires HTTP. One of the primary reasons given for using

HTTP as a transport is "it gets around firewalls." Personally, I remain unconvinced that using HTTP as a firewall-friendly transport is a good idea. How can a DCOM- or SOAP-aware firewall know the intent of the code being transported? Will SOAP become the preferred means for delivering attacks? As the Internet evolves more and more into some kind of universal computing platform (which is a necessary and essential development, I might add), will robust security become less important than improved functionality? As I see it, firewall manufacturers will soon face possibly their most challenging task to date: determining the intent of embedded code and making correct decisions based on an organization's security policy.

References

Brinkhoff, Lars. "GNU httptunnel." 31 August 2000. http://www.nocrew.org/software/httptunnel.html (5 February 2001).

Foundstone, Inc. "Fscan v1.12—command line port scanner." http://www.foundstone.com/resources/fscanbeta.html (5 February 2001).

Pond, Weld. "Netcat." http://www.l0pht.com/~weld/netcat/. (5 February 2001).

Aleph One. "Buffer overflows: a summary." 29 April 1997. http://lists.insecure.org/bugtraq/1997/Apr/0157.html (5 February 2001).

Edwards, Mark Joseph. "Buffer overflows: the developer's bane." 19 April 2000. http://www.windowsitsecurity.com/Articles/Index.cfm?ArticleID=8608&Key=Visual%20InterDev (5 February 2001).

Trend Micro, Inc. "TROJ_QAZ.A." Virus Encyclopedia. http://www.antivirus.com/vinfo/virusencyclo/default5.asp?VName=TROJ_QAZ.A&VSect=T (5 February 2001).

King, Christopher M. "The eight hurdles to VPN deployment." *Information Security Magazine*, March 1999. Originally at http://www.infosecuritymag.com/mar99/cover.htm (3 November 2000); now available at http://www.google.com/search?q=cache:www.infosecuritymag.com/mar99/cover.htm (5 February 2001).

Avolio and Blask. "Application gateways and stateful inspection: A brief note comparing and contrasting." 22 January 1998. Originally at http://www.avolio.com/apgw+spf.html (3 November 2000); now available at http://www.gca.net/solutions/whitepapers/tis/tis_app_gws_stateful_ins.html (5 February 2001).

Allaire Corp. “Allaire security bulletin (ASB99-02): ColdFusion example applications and sample code exposes servers.” 19 May 1999. http://www.allaire.com/handlers/index.cfm?ID=8739&Method=Full (5 February 2001).

SecuriTeam.com. “Performance Copilot for IRIX security vulnerability.” 16 April 2000. http://www.securiteam.com/unixfocus/Performance_Copilot_for_IRIX_security_vulnerability.html (5 February 2001).

Various. “Intrusion detection FAQ,” version 1.37. The SANS Institute. http://www.sans.org/newlook/resources/IDFAQ/ID_FAQ.htm (5 February 2001).

World Wide Web Consortium. “Simple Object Access Protocol (SOAP) 1.1.” 8 May 2000. http://www.w3.org/TR/SOAP/ (5 February 2001).