Securing a Web server

Make your Web server public while keeping it safe

Skill Level: Intermediate

Sean-Philip Oriyano (sean.oriyano@gmail.com)
IT Instructor
Independent Consultant

21 Apr 2009

Web servers are one of the many public faces of an organization and therefore are potentially an easy target. As a public resource, a Web server is like "shark bait" for some. But it doesn't have to be: Learn how a Web server can be public and safe at the same time.

Web servers are one of the many public faces of an organization—and one of the most easily targeted. Web servers represent an interesting paradox—namely, how do you share information about your organization without giving away the so-called store? Solving this dilemma can be a tough and thankless job; but it's also one of the most important.

Before I get too far, though, let's take a look at some of the threats that your server faces by virtue of being one of the "troops" on the front line.

Threats against Web servers

Now, there's a tremendous number of threats facing a Web server, and many depend on the applications, operating system, and environment you have configured on the system itself. What I have assembled in this section are some of the more generic attacks that your poor server may face.

Denial of service

The denial of service (DoS) attack is one of the real "old-school" attacks that a server can face. The attack is very simple, and nowadays it's carried out by those individuals commonly known as *script kiddies,* who basically have a low skill level. In a nutshell, a DoS attack is an attack in which one system attacks another with the intent of consuming all the resources on the system (such as bandwidth or processor cycles), leaving nothing behind for legitimate requests. Generally, these attacks have been relegated to the category of annoyance, but don't let that be a reason to lower your guard, because there are plenty of other things to keep you up at night.

Distributed denial of service

The distributed DoS (DDoS) attack is the big brother of the DoS attack and as such is meaner and nastier (yes, I do have two older brothers: Why do you ask?). The goal of the DDoS attack is to do the same thing as the DoS, but on a much grander and more complex scale. In a DDoS attack, instead of one system attacking another, an attacker uses multiple systems to target a server (your server), and by *multiple systems* I mean (in some cases) not hundreds or thousands, but more on the order of hundreds of thousands. Where DoS is just an annoyance, a DDoS attack can be downright deadly, as it can take a server offline quickly. The good news is that the skill level required to pull a DDoS attack off is fairly high.

Some of the more common DDoS attacks include:

- **FTP bounce attacks.** A File Transfer Protocol (FTP) bounce attack is enacted when an attacker uploads a specially constructed file to a vulnerable FTP server, which in turn forwards it to another location, which generally is another server inside the organization. The file that is forwarded typically contains some sort of payload designed to make the final server do something that the attacker wants it to do.
- **Port scanning attack.** A port scanning attack is performed through the structured and systematic scanning of a host. For example, someone may scan your Web server with the intention of finding exposed services or other vulnerabilities that can be exploited. This attack can be fairly easily performed with any one of a number of port scanners available freely on the Internet. It also is one of the more common types of attacks, as it is so simple to pull off that script kiddies attempt it just by dropping the host name or IP address of your server (however, they typically don't know how to interpret the results). Keep in mind that a more advanced attacker will use port scanning to uncover information for a later effort.
- **Ping flooding attack.** A ping flooding attack is a simple DDoS attack in which a computer sends a packet (ping) to another system with the intention of uncovering information about services or systems that are up or down. At the low end, a ping flood can be used to uncover information covertly, but throttle up the packets being sent to a target or victim so that

now, the system will go offline or suffer slowdowns. This attack is "old school" but still very effective, as a number of modern operating systems are still susceptible to this attack and can be taken down.

- **Smurf attack.** This attack is similar to the ping flood attack but with a clever modification to the process. In a Smurf attack, a `ping` command is sent to an intermediate network, where it is amplified and forwarded to the victim. What was once a single "drop" now becomes a virtual tsunami of traffic. Luckily, this type of attack is somewhat rare.
- **SYN flooding.** This attack requires some knowledge of the TCP/ IP protocol suite—namely, how the whole communication process works. The easiest way to explain this attack is through an analogy. This attack is the networking equivalent of sending a letter to someone that requires a response, but the letter uses a bogus return address. That individual sends your letter back and waits for your response, but the response never comes, because it went into a black hole some place. Enough SYN requests to the system, and an attacker can use all the connections on a system so that nothing else can get through.
- **P fragmentation/fragmentation attack.** In this attack, an attacker uses advanced knowledge of the TCP/IP protocol to break packets up into smaller pieces, or "fragments", that bypass most intrusion-detection systems. In extreme cases, this type of attack can cause hangs, lock-ups, reboots, blue screens, and other mischief. Luckily, this attack is a tough one to pull off.
- **Simple Network Management Protocol (SNMP) attack.** SNMP attacks are specifically designed to exploit the SNMP service, which is used to manage the network and devices on it. Because SNMP is used to manage network devices, exploiting this service can result in an attacker getting detailed intelligence on the structure of the network that he or she can use to attack you later.

Web page defacement

Web page defacement is seen from time to time around the Internet. As the name implies, a Web page defacement results when a Web server is improperly configured, and an attacker uses this flawed configuration to modify Web pages for any number of reasons, such as for fun or to push a political cause.

SQL injection

Structured Query Language (SQL) injections are attacks carried out against databases. In this attack, an attacker uses weaknesses in the design of the database or Web page to extract information or even manipulate information within the database. Although I can't get into the specifics of how to pull this type of attack off, you can look it up if you have SQL knowledge, which—if you're hosting database on your Web server—you should have.

Poor coding

Anyone who has been a developer or worked in information technology has seen the problems associated with sloppy or lazy coding practices. Poor coding problems can result from any one of a number of factors, including poor training, new developers, or insufficient quality assurance for an application. At its best, poor coding can be an annoyance, where features don't work as advertised; at its worst, applications can have major security holes.

Shrink-wrapped code

This problem is somewhat related to the above issues with poor coding, but with a twist: Basically, this problem stems from the convenience of obtaining precompiled or pre-written components that can be used as building blocks for your own application, shortening your development cycle. The downside is that the components you're using to help build your application may not have gone through the same vetting process as your in-house code, and applications may have potential problem areas. Additionally, it's not unheard of for developers who don't really know how to analyze the code and understand what it's actually doing to put so-called "shrink-wrapped" components in applications. In at least one case I can think of, I'm aware of a developer using a piece of shrink-wrapped code to provide an authentication mechanism for an application that was actually authenticating users, but also covertly e-mailing the same credentials to a third-party.

On the ramparts

Okay, now that I've thoroughly scared you about what kinds of threats are out there, I'll take a look at how you can protect yourself from harm.

First, let me detail my six-point plan for protecting Web servers:

1. **Separate Web servers for internal and external use.** This sounds like a no-brainer, but it still bears repeating. Most organizations have Web-based applications or sites used internally, as well as applications and sites used externally. In an ideal situation, these two sets of servers and content should be kept separate, with internal and external sites having their own servers with as little crossover between them as possible. By splitting systems apart like this, you avoid the probability (or at least lessen the risk) of an attacker breaching a server and getting

access to data or even internal systems.

2. **Separate development and production servers.** In my time, I have known several companies that have violated this rule by letting their development team work on production servers to develop their code or tweak existing code. Typically, this is just a case of extreme laziness—one that can lead to catastrophic problems later when an attacker sees your unpolished code and exploits it to his or her own ends. Also, consider that your own developers may compromise security by testing and tweaking code. Do yourself a favor: Implement a development environment.

3. **Regular audits.** Any Web server or Web application worth its salt will have some method of generating logs of activity on the system. After this information is logged, make it part of your regular routine to scan the logs for problems, such as application failures or suspicious activity. Keep in mind that an audit log is like evidence collected at a crime scene: It's essentially worthless unless you intend to examine it later.

4. **Keep your system up to date.** Do I really need to go through this one? Yes, I do. Patching a system is an often-overlooked problem when it really shouldn't be. Ideally, you should keep an eye on whether patches, service packs, updates, or other items become available that can help secure your system. Depending on your hosting platform and other factors, you may have the option of having these updates delivered automatically, or you may have to use the old-fashioned manual delivery method. Also keep in mind that many times, updates are the only way to fix problems such as those related to buffer overflows, network client issues, and so on.

5. **Vulnerability scanning.** In previous articles (see Resources), I covered the topic of vulnerability scanning as a tool for finding problems in your hosting and application infrastructure. Vulnerability scanning can be a very powerful tool in the ongoing struggle to uncover problems relating to software, such as configuration and patching issues. Another advantage is that these scanning tools are regularly updated, so you can use them to find the latest problems that, in a number of cases, include issues that you may not even be aware of, allowing you to address them before they can be exploited. Tools such as the freeware Nessus (see Resources) can be a great asset to administrators regardless of whether you host on Linux®, UNIX®, or some other platform.

6. **Developer training.** This one may be a bit more difficult to pull off, but it reaps a tremendous reward, if undertaken. Educating developers on secure coding practices can result in the elimination or reduction of problems associated with sloppy or lazy coding.

The other details

Aside from these six points, there are plenty of other things you can do to improve security in your organization's hosting structure. Let's take a look at some of the things that tend to cause problems and how you can address them.

Misconfiguration

As hardware and software have become more complex and IT and development teams have gotten smaller, the potential for something to slip under the radar, so to speak, has grown. Fortunately, using vulnerability scanners, automated scanners, and just plain and simple education and due diligence can reduce misconfiguration problems.

Banner information

Banner information can reveal a myriad of information to those who actually know it's there and how to look for it. Banner information can reveal such telling information as version data that may help an attacker.

For example, when you make a request to a Web server to request a static piece of content, such as a basic .html or .htm file, something known as a *content location header* is prepended to the response. In a default configuration, in some Web servers, this content header will reference an IP address and may or may not provide the fully qualified domain name (FQDN). In a worst-case scenario, this header could reveal internal IP address information along with other data. Take a look at the following header in Listing 1.

Listing 1. Sample content location header

```
HTTP/1.1 200 OK
Server: <web server name and version>
Content-Location: http://w.x.y.z/index.htm
Date: Thu, 1 Jan 2009 14:03:52 GMT
Content-Type: text/html
Accept-Ranges: bytes
Last-Modified: Wed, 31 Dec 2008 18:56:06 GMT
ETag: "067d136a639be1:15b6"
Content-Length: 4325
```

This header tells you quite a bit about your server, but never fear: Web servers can quite commonly have this information sanitized or modified to a particular organization's tastes. (Consult your Web server's documentation for more information.)

Permissions

Permissions still tend to be a problem in some circles. Often, they are incorrectly assigned or not properly planned, allowing individuals access to parts of the server or application they should not have. Always check to make sure that those who manage and interact with your server have the appropriate permissions to do what they need to do and nothing more: This is also known as *least privilege.*

Error messages

We have all gotten the dreaded 404 message and its cousins from time to time, telling us that the information we're looking for cannot be found or has been a victim of some other type of foul play. But those error messages could be telling you a lot more if you take a closer look. To an attacker, any error message your Web server or application generates can reveal information about how your application is configured, what libraries you're using, database connection strings, plus a whole lot more. Fortunately, it doesn't have to be this way. You should use all your error messages to your advantage in testing and development to help you catch problems. But when you deploy your application or server into production, your error messages should be disabled or configured to reveal very generic information. (Consult your Web server documentation for information on how to do this.)

Services

When building a system to host your Web server and applications, you should carefully plan what features and functionality you need, and then build the server toward that goal. Basically, when you figure out what your server should be doing, you enable just those services and features needed to support that role and remove or disable everything not needed for the environment. Keep in mind that every service or application that is present and enabled is just another potential hole that an attacker can exploit.

Protocols

Again, much like services, any protocols—such as NetBIOS—that you don't need should be kicked to the curb and disabled or outright removed.

User accounts

It's not uncommon for operating systems and applications to have default user accounts set up when the operating system is installed. To be safe, this is something that should get your attention. Make it a point to look at the user accounts present, disable or remove those you don't need, and strengthen (and change) the passwords for those you do need.

Sample and test files

It's not unheard of for some Web servers and applications to ship with sample files and test components as technology demonstrators to show you what you can do with the product. In a safe and secure environment, these files should be removed from all production Web servers and applications, as they can be manipulated to allow unauthorized access.

Ports

Look at the ports your Web server and applications need to function, and enable (and monitor) those ports. In fact, the ports you're not using should be closed and blocked from a firewall.

Some other protective measures

There are other items you might consider using to make your hosting environment that much stronger:

- **Intrusion-detection systems (host and network-based).** These systems are hardware or software devices that can help you monitor access across the network to and from your server as well as activity on the server itself.
- **Antivirus software.** Yes, you should have one on your Web server.
- **Firewalls.** Plenty of firewalls are available: Choose the one that best fits your needs, and deploy it in front of your Web server.
- **Common sense.** Yes, I had to add this one. In the last decade, when the Internet was making its way into our daily lives—both in the personal and in the business worlds—the idea was to get whatever you could on the Internet, because it was the thing to do. As a result of this "Zerg rush" to get on the Internet quickly to make a big splash before your competitors, a lot of information was put out there that never should have been. Remember, not everything needs to be on the Internet. Also, keep in mind that if you put something on the Internet, you pretty much can't put the genie back in the bottle later. Not convinced? Take a look at www.archive.org and see what your Web site or those of your competitors have had over the past 10 years.

Conclusion

Securing a Web server and your hosted applications is indeed a daunting task, but it's not an impossible one. With some research and a good, healthy dose of hard work (and maybe some late nights with some strong coffee), you can make your hosting environment that much stronger and save yourself some headaches in the long run.

Resources

Learn

- Get the latest security news and information at securityfocus.com.
- The SANS Institute is your one-stop shop for security information, certification programs, and research.
- Learn more about scanning for vulnerabilities in Sean-Philip Oriyano's developerWorks article, "Anatomy of a Web attack" (Feb 2009).
- The developerWorks Web development zone is packed with tools and information for Web 2.0 development.
- IBM technical events and webcasts: Stay current with developerWorks' technical events and webcasts.

Get products and technologies

- The Nessus vulnerability scanner provides several important scanning features, such as high-speed discovery, asset profiling, and vulnerability analysis.

About the author

Sean-Philip Oriyano
Sean-Philip Oriyano has been actively working in the IT field since 1990. Throughout his career, he has held positions such as support specialist to consultants and senior instructor. Currently, he is an IT instructor who specializes in infrastructure and security topics for various public and private entities. Sean has instructed for the U.S. Air Force, U.S. Navy, and U.S. Army at locations both in North America and internationally. Sean is certified as a CISSP, CHFI, CEH, CEI, CNDA, SCNP, SCPI, MCT, MCSE, and MCITP, and he is a member of EC-Council, ISSA, Elearning Guild, and Infragard. You can reach Sean at sean.oriyano@gmail.com.

Trademarks