Running Head: Security Considerations for the Proper Implementation of Webservers and Web Applications

Security Considerations for the Proper Implementation of Webservers and Web Applications

Timothy Simmons

Amridge University

Security Considerations for the Proper Implementation of Webservers and Web Applications

The introduction of *Hacknotes: Network Security Portable Reference* (2003), written by Horten, M. and Mugge, C., starts out by explaining the importance of knowing the tactics used by hackers and other criminals in order to be able to defend against them. This research will use the defense-in-depth strategies to explain the methods by which IT professionals are securing their Webserver and applications running on them. This research takes a look at some threats plaguing two of the most commonly used operating systems Windows Server 2003 and Linux and two Webservers Microsoft IIS and Apache. Next, this research discusses in some detail the considerations that have to be made in the development of secure Web applications that combat current threats that seem to be almost everywhere we look on the Internet today.

Research and analysis was conducted that takes a look at what types of operating systems and Webservers are the most prominent. Different types of sites were tested including famous e-commerce sites like Amazon and Ebay, and government sites like the Veterans Administration. As Figure 1, below illustrates Linux is the most commonly used operating system and Apache is the most common Webserver of the Web sites tested.

Webservers vs. E-Commerce Service

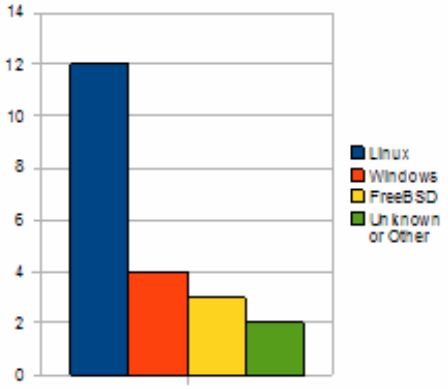
To get a good picture of Webserver uses one must understand the underlying technologies involved. Panko (2004) explains the differences between the two in his security book, *Corporate Computer and Network Security*. He explains that Web service is defined as the basic operations of static file retrieval and the creation of dynamic Web pages (Web pages that are tied to databases and are created after specific queries are made by the user), that are the

main function of HTTP Webservers. Dynamic Web pages use separate programs like PHP, MySQL, and Perl scripting languages that are from different companies or open source projects. Some of these programs like PHP for instance are built into a lot of Webserver software. Other businesses will have Webservers that have even more programs designed specifically for them

Operating Systems

The following is a roll-up of the types of the 21 Operating Systems that was observed in this research:

Linux	Windows	FreeBSD	Unknown or Other
12	4	3	2

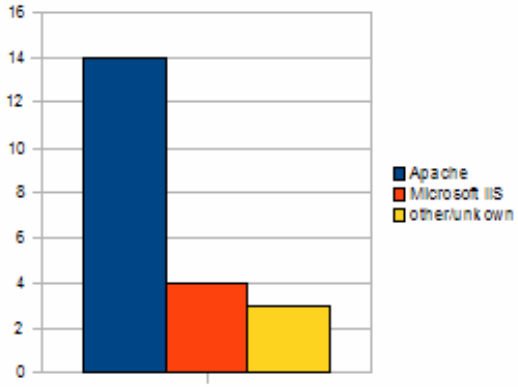


As a by product of this research it was interesting to find out how easy it is for a hacker to learn of a Websites IP address. This makes it easier for them to conduct their initial probing attacks to map out network information. Just as my former research proclaimed, Linux is the clear winner in the OS department.

Web Servers

The following is a roll-up of the 21 types of Web servers that were used by the Websites assessed.

Apache	Microsoft IIS	other/unkown
14	4	3



Again just like my earlier research agrees, Apache takes the cake.

As one can see in this work, Open Source Applications are winning out over the costly vendor's software.

Figure 1. Calculations and results of operating systems and Webservers.

by their own IT specialists. For security reasons good security policies should be in place to test custom programs in a test environment before they are used in production Webservers that help run the company's intranet.

Panko (2004), goes on to explain that e-commerce service should be seen as additional software that enables online catalogs, shopping carts, and payment and checkout functions (see Figure 2). These programs use connections to internal back-end databases and external servers merchant banks and credit card checkers.

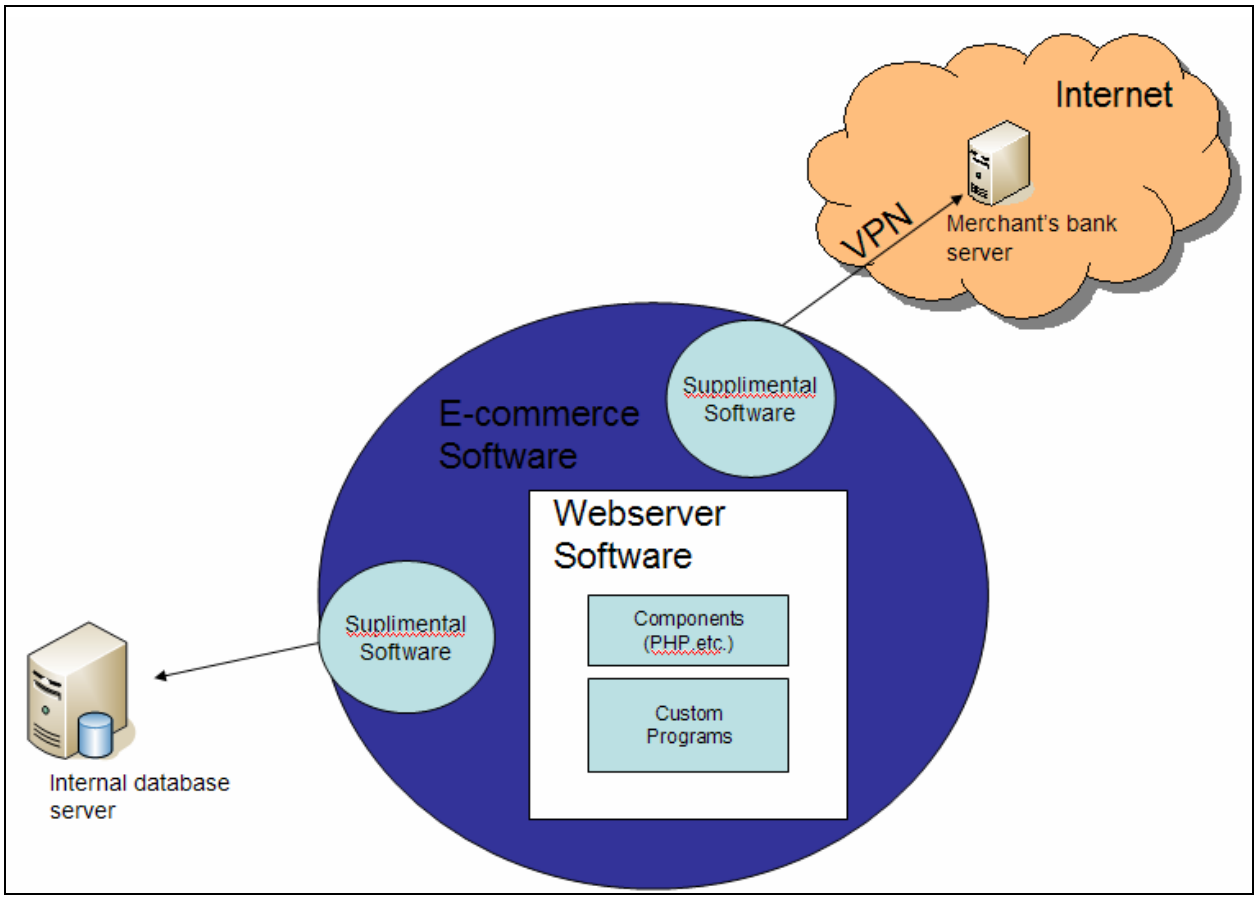


Figure 2. A graphical depiction of the integration of Webserver and E-commerce programs.

Defense-in-Depth

To combat the many threats from outside the network perimeter many businesses are turning to a strategy that puts up layers of defenses to not only prevent security breaches but also buy the organization time to react to a breach if one should occur.

The defense-in-depth can be accomplished by developing a strong management-supported security policy where defensive measures are taken at every level. The use of packet filters, firewall computers, and host firewalls will give a small organization the edge it needs to put up a good defense. However, for an organization that uses extensive networking to conduct business to secure and monitor the entire network the network administrators should consider the use of Intrusion Detection Systems (IDS) and Intrusion Prevention Systems (IPS). The other levels of security that explained and illustrated below are as follows (See Figure 3):

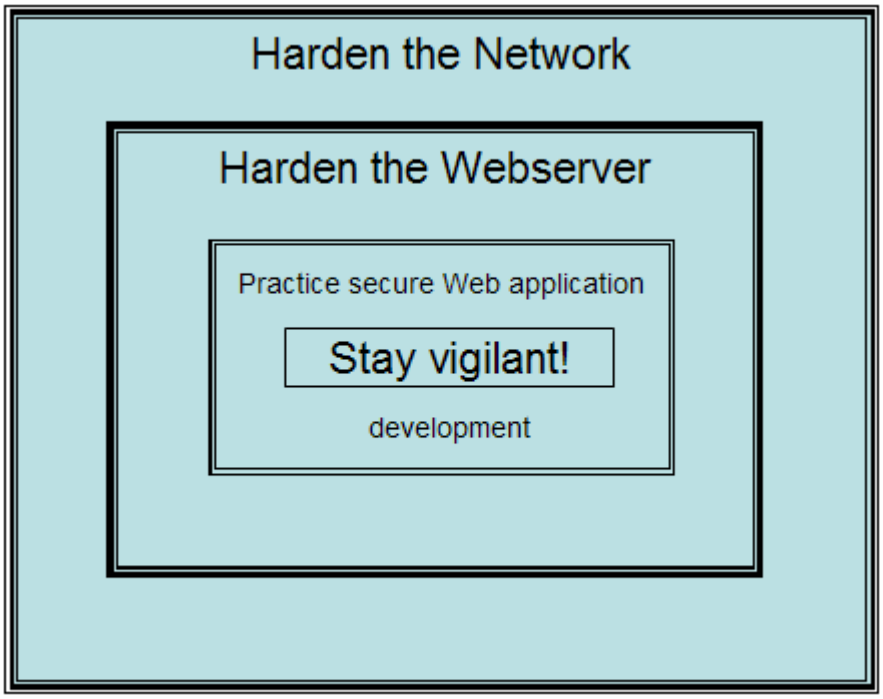


Figure 3. Levels of defense in a Defense-in-Depth strategy

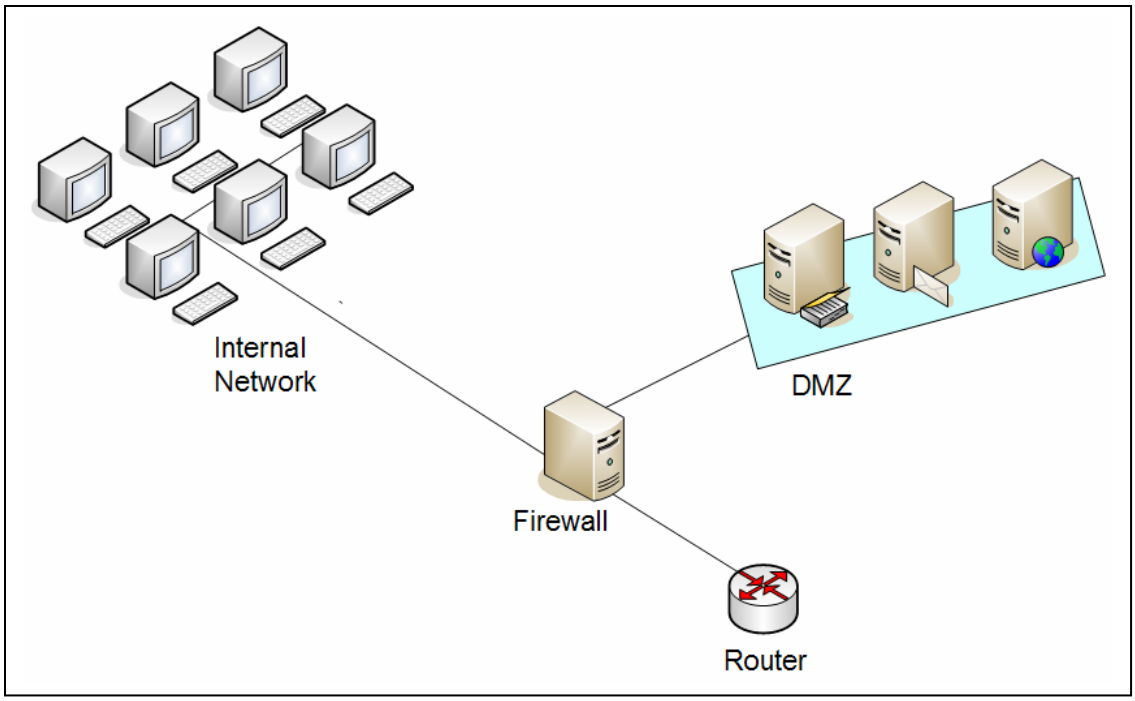
1. Harden the Network.
2. Harden all Servers and all hosts and other perimeter endpoints.
3. Practice secure Web application development.

4. Stay vigilant.

Network Hardening: First level of Defense

For networks that support Web services to the public, Albanese, J. & Sonnenreich, W. (2004) suggest in their book that the network be segmented into a secure internal network and a less secure segment where Webservers, DNS servers, and other types of servers that will be servicing the public reside known as a demilitarized zone (DMZ).

A search with Google on the Internet for a DMZ turns up some interesting information that can be accessed that show pros and cons of using a DMZ and suggestions on setting them up. Wikipedia (2008), has two examples that could be used to set up a DMZ on a simple network for smaller and middle sized organizations, one more secure than another.

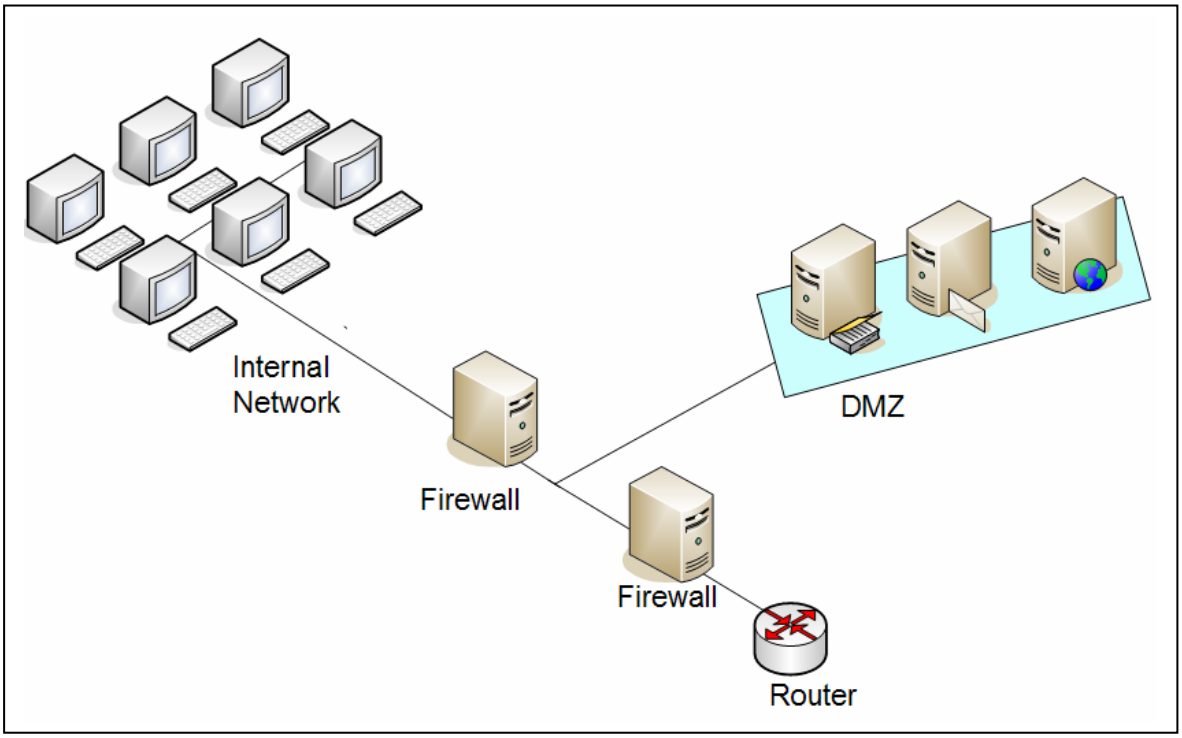


Example 1. An example of a DMZ subnet architecture using only one firewall.

In Example 1 above, a single firewall with at least 3 network interfaces can be used to create a network architecture containing a DMZ. In this approach the network will be broken into

three sections: one interface connecting the firewall to the ISP or the external network, the second interface connecting the firewall to the internal network, and the third interface connecting the firewall to the DMZ.

Example 2 below illustrates a more secure approach where two firewalls are used to create a DMZ. In this approach the first firewall will be configured to allow traffic destined for the DMZ as well as traffic for the internal network. The second firewall will be configured to allow only traffic destined for the internal network, it should stop all traffic originating from the DMZ. The increased amount of protection this approach allows will outweigh the costs of the second firewall.



Example 2. An example of a DMZ subnet architecture with two firewalls, a more secure approach.

Hardening the Endpoints: Operating Systems can be our Worst Enemy

Hardening Operating Systems for use with servers of any type should be left to professionals that have learned the systems inside-out. This is true in a perfect world, but IT

people rarely work in a perfect world. Rather, they are juggling the administration of several different operating systems. The intentions here were to show all of the vulnerabilities associated with the two types of Webserver systems described in the first paragraph, but to do that this research paper would be the size of a book. The best practice for an IT person aspiring to become an effective network manager or systems administrator is to follow the following guidelines from Raymond Panko's textbook, *Corporate Computer and Network Security*:

- □ Provide physical security for the host
- □ Install the operating system with secure configuration (starting from a secure base)
- □ Download and install patches for known vulnerabilities
- □ Turn off all unnecessary services (there should be only a few running for the Webserver)
- □ Harden all remaining applications (Web application hardening is discussed next)
- □ Manage users and groups
- □ Manage access permissions
- □ Conduct regular back-ups
- □ Employ advanced protections
- □ Test for vulnerabilities

There are a number of good books available to help in this process. Two of the books that I found were, *HackNotes: Windows Security Portable Reference* written by Michael O'Dea, and *HackNotes: Linux and Unix Security Portable Reference* written by Nitesh Dhanjani.

Web Vulnerabilities: Know Your Enemy's Tactics

Another Hacknotes series, *Web Security Portable Reference*, written by Mike Shema (2003) proclaims that there are two categories in which Web vulnerabilities can be categorized. The first are the vulnerabilities within the platforms that the Webservers operate from, the

operating systems like Windows Server 2003, and the Webserver Programs like Apache. The second category are the vulnerabilities that are Web application specific, like SQL databases, and applications developed with Perl, PHP, and JavaScript, both from vendors and from organizational internal developers.

Directory Traversal

Panko (2004), describes this type of attack as one that makes use of the knowledge that using "../" in the URL path could give attackers access to sensitive system files by using a Microsoft IIS vulnerability. As is usually the case with these kinds of vulnerabilities vendors (in this case Microsoft) respond with patches that "fix" the problem. Then attackers find another vulnerability using hexadecimal and/or UNICODE values for "../" to circumvent the newest patches. This cat and mouse game is going on as we speak, and this example speaks to the importance of updating vendor programs often.

Buffer overflow

A buffer is a specially allotted pocket of memory allotted for a specific type of data variable. If the data received by the form is larger than the buffer then a buffer overflow results. Again using a HTML form an attacker could input specially crafted form data to cause a server to execute malicious code that overflows the space allotted for valid form input. This is considered a classic attack because there are so many applications, home-grown and commercial programs, that have this vulnerability, affecting both categories of vulnerabilities.

Botnets

Although Peer-to-Peer (P2P) and chat applications are a definite knowledge base multiplier in the corporate settings and for providing personal communications, these

applications unfortunately provide an excellent infection vector for attackers if not configured correctly.

According to Infosecurity's *Threat Analysis* (2008), the Botnet is the highest security threat known to plague the Internet at this time. Infosecurity explains that a botnet is not just a virus but a virus of viruses. The botnet can be seen as a network of compromised computers, so-called zombies, that are controlled by a bot master or botherder. The botnet is modular, meaning that it can stand alone and listen to an IRC channel and download modules that can, for instance, knock-out antivirus software and firewalls, and still another that will scan the system and the network for vulnerable applications and hosts. Even though malicious Bots have been around since the discovery of Pretty Park in May 1999 (Canavan, 2005, p.6), the war on this threat only actually started in 2005-2006 after the connection was made between botnets and the organized crime scene. Although this is not a Webserver specific threat I thought it should be added here because of its significance in the IT security world. If one of the hosts in a network is infected with one of these bots or something newer, as IT is now we have no real tools to find them, much less eradicate them. .

Cross-site scripting

The Infosecurity (2008) analysis describe other types of threats against Webservers and Web applications that are being used often by attackers today called Cross-site scripting attacks. After Netscape fixed this problem with the "same-origin policy," hackers made it a personal challenge to circumvent these restrictions. Script/HTML Injection or cross-site scripting (XSS) was the name of the offspring of this type of attack. Attackers had amused themselves by infecting a Web user's online experience, stealing cookies, proclaiming virus launch warnings, and other hateful digital mischief. It is the trust we have in the Web sites we visit that the

attacker misuses. For the most part this is done by the attacker finding a Web site that echoes the data input by a user back to the user's Web browser. All a hacker has to do then is input malicious code that permanently alters the Web site structure and each time this Web page is called the victim's Web browser executes the malicious code thinking it is valid code from the Webserver.

The cross-site scripting (XSS) is an attack threat that can be used to compromise system integrity, rob sensitive and personal information, hijack user sessions, break into intranets, and opening the internal network up to the world, XSS vulnerabilities have been around a long time and are considered the largest threat to the e-economy. The following attacks are related to and are defined in most literature as being cross-site scripting, or cross-site scripting forgery attacks (Infosecurity, 2008).

Tipton, H. & Krause, M. (2007) add the following to the list of dastardly hacker attacks that can be carried out by Web applications:

Input manipulation

Input manipulation involves using a vulnerability of some Webservers that process HTML forms by a Common Gateway Interface (CGI). For example, the attacker can manipulate data input into the form used by a CGI to mail information to another user and so gain access to the server or delete all files in the system.

Direct access browsing

Direct access browsing describes how attackers use URL manipulation to gain access to a Web site that normally should require authentication. This happens because some Web applications are not configured correctly and allow attackers to directly access URLs that could then lead to the loss of revenue if that page normally requires a fee to view.

Backdoors and debug options

Sometimes Web application developers create a backdoor to allow them to troubleshoot applications and forget about them or leave them in place because of time constraint pressures or poor security policies. Backdoors sometimes allow a user to log in without a password, or a URL will give the user direct access to application configuration. Although this is OK during the development phase it should be cleaned up before the application goes live. Removing backdoors greatly reduce the number of vulnerabilities of this type within Web applications.

URL Parameter tampering

Attackers can manipulate URL strings in many number of ways to inject bogus SQL calls that release user listings, passwords, credit card numbers and any other type of data that is stored in a database. They can manipulate URLs to redirect users to their Web site thus infecting them with malicious code.

Cookie poisoning

Cookie poisoning is a way by which the attacker modifies the data stored on a HTTP cookie. HTTP cookies are commonly used to store user IDs, passwords, and the like so that when a user returns to a Web site they won't have to re-authenticate themselves. Attackers can gain full access to these user's accounts by first stealing the cookies from the user and then manipulating them in such a way that they assume the user's identity. Needless to say, this could be devastating , especially considering how many banks and Web-based e-mail providers use cookies for authentication purposes.

Hidden fields

.Hidden fields are HTML form fields with the "type=hidden" attribute (Cobb, M., 2008). The hidden field will not show up on the Web page, but are non-the-less visible in the page source code. These fields, if used to store sensitive information like system passwords,

merchandise prices, session IDs, or user authentication data could allow malicious users to hack a system, purchase goods at little or no cost, hijack user sessions, or steal a users identity.

With all different benefits of new technologies now available on the Internet, such as Flash, QuickTime, and PDF, there are just as many vulnerabilities associated with them. If Web security considerations for each of these applications aren't considered, there will be a great number of potential insecure protocols that are ignored. Following are the best practices used today to prevent Web application attacks (Infosecurity, 2008, p 111).

According to Tipton, H. et al. (2007) the best way to prevent Web application attacks is through education and vigilance. Vigilance, though, is sometimes easier to talk about than to do.without proper security policy and aggressive adherence. Management needs to know the implications of putting a system online before proper testing, auditing, and code validation has taken place. System administrators, developers, and other IT professionals involved in Web application development should be constantly monitoring Internet sources such as vendor Web sites, CERT, Securityfocus.com, mailing lists and security forums for the latest vulnerability updates for their applications and systems. The following are a few tips that can be read over and over in a number of network and application security literature:

- Developers should only trust input that they can control. There should be a heightened distrust of the end user and all user input data should be considered hostile.
- Add filters and input checks that sanitize input before it is processed; this will reduce the risks of most Web application attacks.
- Developers should integrate security measures as the applications are being built and not get used to running the application in root to save time. Leaving backdoors or debuggers turned on after the application goes online is just asking for trouble.
- Using the principle of 'least privileges' is a must.

- Don't use GET messages to send sensitive data from the client to server. GET requests are logged by the Webserver in cleartext for the world to see. The HTTP POST command with an SSL connection should be used instead.
- Developers should always be aware of the risks that HTML code comments and error messages might have to an application. These could leak information to an attacker that is trying to profile a specific Web application.
- To combat the cross-site scripting attack developers should use HTML encoding to encode input strings that are displayed on the users as encoded text instead of becoming another part of the Web page code.

The Computer Emergency Response Team has started a secure coding initiative that every developer should be aware of. As stated earlier, education and vigilance are key to developing secure Web applications. Security should become an integral part of application development. Procedures for the monitoring and maintenance of Web applications will keep the application up and running.

Tools for the Webserver Administrator

There are many tools that a Web application developer can use to check and validate code before giving it the thumbs-up. Shema (2003), shares his knowledge about a class of tools known as vulnerability scanners. Vulnerability scanners he says have two components- a vulnerability database and a scanning engine- that enable us to analyze information returned by the server to find out if a vulnerability exists. Below is a list of vulnerability scanners:

- Whisker
- LibWhisker
- Nikto
- Nessus

Other tools that Shema (2003) discusses are called assessment tools which create a framework for testing an application's functionality. These tools would help a developer find logical and semantic vulnerabilities, and input validation vulnerabilities. These tools include the following:

- Achilles
- Web Proxy 2.1
- Curl

As one can see here a developers have a number of tools that he could use to develop secure applications. The use of these tools should be added into the normal software development and should be allotted time in project plans.

Conclusion

This research paper has shown what security considerations have to be made when running a Webserver and the applications that support this server. Defense-in-Depth is the watchword in the IT security world, hardening at each level to make an attack more difficult, by challenging hackers and setting up a good strategy to defend against their attack methods. Webserver administrators, network security professionals, and other IT professionals, keep today's security threats from causing undue harm to your organization's IT resources, causing probable financial losses. Stay alert to any new known vulnerabilities, because you-know-who will be alert as well. Stay vigilant!

References

Albanese, J. & Sonnenreich, W. (2004). *Network Security Illustrated.* New York, NY: McGraw-Hill.

Canavan, J. (2005). Symantec Security Response, *The Evolution of Malicious IRC Bots* [Electronic Version]. Retrieved February 20, 2008, from http:// www.symantec.com/ avcenter/reference/the.evolution.of.malicious.irc.bots.pdf

Computer Emergency Response Team. (2008). *Top 10 Secure Coding Practices*. Retrieved February 21, 2008, from https://www.securecoding.cert.org/confluence/display/ seccode/Top+10+Secure+Coding+Practices

Cobb, M. (2005, December 21). *Don't hide sensitive information in hidden form fields.* Retrieved February 20, 2008, from http://searchsecurity.techtarget.com/tip/0,289483,sid14_ gci1153816,00.html

Dhanjani, N. (2003). *Hacknotes: Linux and Unix Security Portable Reference*. Emeryville, CA: McGraw-Hill/Osborne.

Horton, M. & Mugge, C. (2003). *Hacknotes: Network Security Portable Reference*. Emeryville, CA: McGraw-Hill/Osborne.

Infosecurity. (2008). *Threat Analysis.* Burlington, MA: Syngress Publishing.

Kadrich, M. (2007). *Endpoint Security* (1st ed.). Boston, MA: Pearson Education.

Liotine, M. (2003). *Mission-Critical Network Planning.* Norwood, MA: Artech House.

O'Dea, M. (2003). *Hacknotes: Windows Security Portable Reference.* Emeryville, CA: McGraw-Hill/Osborne.

Richards, R. (2006). *Pro PHP XML and Web Services*. New York, NY: Springer-Verlag New York.

Shema, M. (2003). *Hacknotes: Web Security Portable Reference*. Emeryville, CA: McGraw-Hill/Osborne.

Tipton, H. & Krause, M. (2007). *Information Security Management Handbook* (6th ed.). Boca Raton, FL: Auerbach Publications.

Wikipedia. (2008). *Demilitarized zone (computing).* Retrieved March 20, 2008 from http//en.wikipedia.org/wiki/DMZ_host