Global Information Assurance Certification Paper

William Geiger
SANS Security Essentials GSEC Practical Assignment Version 1.2f
Proactively Guarding Against Unknown Web Server Attacks

Introduction

The recent wave of Code Red worms has revealed how vulnerable web servers can be to attacks over port 80, the default TCP/IP port used for HTTP traffic. Expensive firewalls proved ineffective at preventing the worm from infecting vast numbers of web servers through a simple programming bug. While investigating the latest variant, I found that some web servers did not get infected even though they were vulnerable to the programming bug. This led me to the realization that there were ways to protect web servers from future port 80 attacks that were similar in nature to Code Red.

While applying security hot-fixes in a timely fashion is recommended, there is always the chance of being attacked through a newly discovered vulnerability before it can be identified and patched. The premise of this paper is to review various ways of protecting web servers from unknown attacks over port 80. We'll examine the technology, explain why it is effective, and identify areas where further diligence is required.

Signature of a port 80 attack

TCP/IP port 80 is the default port for HTTP traffic. Web servers purposely listen for requests on this port. A conduit for port 80 must exist in any firewalls between the end user's browser, typically on the Internet somewhere, and the web server for it to be able to do its job. Most firewalls do not inspect each packet looking for particular exploits. This makes an attack on port 80 difficult to prevent before it gets to the web server. What are the signs of a typical attack over port 80?

First, they take advantage of programming bugs in code executed by the web server. This code could be an integral part of the web server itself, or extension code that is executed to perform a particular function. Generally, the bug is revealed using a malformed URL, such as one that is overly long or improperly constructed. In the case of Code Red, a buffer overflow in the Microsoft Index Server extension of Internet Information Server was taken advantage of. Some attacks attempt to access a dummy resource in the web server's root directory. Others target some other directory with execute privileges that is installed by default during the web server installation. An example is the /SCRIPTS directory of Microsoft's Internet Information Server. This method is effective because it is not unusual for analysts to install the web server using default values, a process that is familiar from installing other software products.

Once the server is compromised, the attacker can issue root level commands by traversing to the system directory, changing the user context of the web server, and/or uploading trojan code. In the case of Code Red, code was uploaded to the web server that would then attack near by web servers using the same technique that compromised the victim web server. In this way, the worm also created a distributed denial of service attack, which made it difficult to identify and fix the

victimized web servers.

An attack on port 80 is similar to a random shoplifting incident. The perpetrator comes in through the front door just like any other customer. But before you know it, he's taken advantage of some lapse in security and has made off with the goods.

Secure Sockets Layer

Secure Sockets Layer (SSL) is a protocol for encrypting data between the end user's browser and a web server. The web server administrator must obtain a digital certificate for the server from a trusted third party to enabled SSL. To establish an SSL connection, a handshake process is performed between the browser software and the web server.

The handshake starts when the browser tells the server what levels of SSL it can support. The server responds with the SSL level to be used, and a copy of its digital certificate. The browser validates the server's certificate, and creates an initial secret for the session. The secret is encrypted with the server's public key, which was obtained from the server's digital certificate. The encrypted secret is then sent to the server. The server uses its private key to decrypt the initial secret. It uses this secret to generate the master secret. At the same time, the browser also uses the initial secret to independently generate the same master secret. Both the client and the server use the master secret to generate session keys, completing the handshake. Both sides now have securely shared information that is used to encrypt and decrypt all future communications.

How does SSL protect web servers from attacks? A recent survey found less than 122,000 web servers had valid SSL encryption enabled out of over 27 million active servers. That's less than ½ of one percent. SSL enabled web servers may contain sensitive data that a hacker would find interesting, but non-SSL servers are targeted because there are more of them. Automated scanning tools look for susceptible servers by issuing a request, checking the response for the vulnerability, and then moving on to the next target. To scan SSL enabled web servers these tools need to be coded to negotiate the handshake before checking if the server is vulnerable. When the goal is to infect at many servers as possible, as in the Code Red worm, it makes sense to ignore SSL enabled servers.

SSL is easily implemented, requiring only a digital certificate ($349/year from Verisign Corp.) and the appropriate configuration changes enabling and requiring its use. End users simply need to know to use https instead of http when entering the URL. SSL will not protect against the determined hacker. If your SSL enabled server is specifically targeted (i.e. not by an automated scanning tool or worm); it would be as vulnerable as if SSL was not enabled.

SSL is a simple way to protect against most mass scanning attacks. It is easy to implement and relatively inexpensive. In our shoplifting analogy, imagine our store is in a strip mall with many other stores. Our store's front door is locked, but the back door is open. Shoppers would need to know to use the back door to get in. The shoplifter looking for a random store to hit would not be able to get in the front, and so would go on to the next store. Of course if the store is specifically targeted the shoplifter would know to come in the back, just like if your SSL

protected web server was attacked using HTTPS.

End User Authentication

End user authentication requires web server requests to include user information. If this information is not supplied, the request results in a challenge response. HTTP authentication, also known as basic authentication, works like this.

On the first access to an authenticated resource, the server will return an unauthorized status and includes a WWW-Authenticate response header. The browser should then ask the user to enter a username and password. The same resource is requested again, this time including an Authorization header that contains the username and password entered. The server checks the username and password, and if they are valid returns the page.

You can also use a proprietary user authentication product that runs as a web server filter and stores user information in a cookie. In this scenario, it is up to the filter to determine whether the resource requires authentication, and whether the user is allowed access. However, an unauthenticated request still causes a challenge to be returned to the requester.

Requiring end user authentication protects against port 80 attacks in much the same way as SSL. Both mechanisms do not return the resource requested, but instead issue a response that must be replied to in a certain way. Automated scanning tools will see the challenge response as a server that is not vulnerable, and move on.

Even web servers that provide public content can use this technique provided the content isn't served from the web server's root or any default directories that might be targeted by a port 80 attack. If the content were served from specific directories, an attacker would need prior knowledge of the site to mount a successful assault. Unused directories and the root should require user authentication to foil mass vulnerability scanners and worms such as Code Red.

In our shoplifting example, user authentication is equivalent to showing identification before entering the store. If you don't have a valid ID, you can't get in.

While requiring user authentication protects against automated attacks, it is the nature of the challenge response to reveal what type of authentication is required. This gives the sophisticated hacker some knowledge of your system, and consequently some idea on how to attack it directly. Also, some web server vulnerabilities may be exploited before the server checks if the resource is protected, thus defeating this safeguard.

Reverse Proxy

Reverse proxy servers are intermediary servers between the public network and internal web servers. They are also known as application gateways or web publishing servers. They work by accepting incoming HTTP requests, translating the requested URL using a configuration table, and issuing an internal request to the appropriate content web server. The response received

from the content server is then re-translated and sent back to the browser. In this way the proxy server appears to the client to be the internal content server.

A single reverse proxy server can selectively translate and forward requests to multiple Web servers on the internal network. For instance, Internet requests for http://www.mywebsite.com/pages could be translated and routed to an internal server at http://articles.private.net, while requests for http://www.mywebsite.com/pictures could be sent to a different server using http://images.private.net.

Reverse proxies protect against port 80 attacks by adding a layer of interrogation. Much like a firewall, a reverse proxy requires a configuration table of URLs to translate, with anything not matching being rejected. In this way, judicious URL naming standards can prevent generic port 80 attacks from getting through the reverse proxy to the content server.

As effective as reverse proxies can be, they can be completely ineffective if the proxy's configuration table is too generic. A proxy that simply translates public URL names directly to private URLs would not stop attacks, since this configuration puts the onus on the web server to detect invalid requests. For example, the Code Red buffer overflow used a URL like http://www.mywebsite.com/x.ida. If the reverse proxy were configured to forward http://www.mywebsite.com/ to http://internalserver/, "internalserver" would experience the attack. However, if the translation http://www.mywebsite.com/intsrv/ to http://internalserver/ existed, the attack would be rejected since the reverse proxy wouldn't have a translation just for http://www.mywebsite.com/.

Going back to our shoplifting example, a reverse proxy is similar to a store with all of the merchandise in the storeroom. A shopper needs to ask a sales associate for a particular article, and the sales associate goes and gets it from the storeroom. If the shoplifter asks for an item that doesn't exist, the sales associate can't go get it.

Reverse proxies can help prevent mass port 80 attacks, assuming the configuration table is specific enough to avoid passing attacks on to the back-end content servers. This may require additional staff and may be difficult to maintain depending on how many translates are configured. In addition, this may not be effective against a targeted attack against a specific resource that gets passed onto the content server.

Intrusion Detection Systems

An intrusion detection system collects server and/or network information for analysis to determine if an attack or an intrusion has occurred. A network-based intrusion detection system monitors the traffic on its network segment, while a host-based intrusion detection system involves loading software on the server to be protected. Host based intrusion detection systems typically monitor system, event, and security logs on Windows NT and syslog in Unix environments. Intrusion detection systems compare the data with attack signatures to see if there is a match. If so, the system responds with administrator alerts and other calls to action. Some intrusion detection systems can automatically respond to an intrusion. It is these types of

systems we're interested in to prevent port 80 web server attacks.

For the purposes of blocking unknown web server attacks over port 80, it is necessary for the intrusion detection system to do real-time screening of requests. Two ways to accomplish this is though signature identification and acceptable use analysis. Signature identification uses a database of attack patterns. This database is provided by the software vendor, and should be updated with new signatures on a regular basis. The signatures in the database are compared to incoming requests to identify incidents. Once identified the intrusion detection system can log and possibly block the request before the web server acts upon it. Acceptable use analysis requires the intrusion detection system to observe normal operations of the web server and build a profile of acceptable use. Incoming requests are then compared with the usage profile, and wayward requests are logged and rejected.

Intrusion detection is comparable to actively monitoring shopper's actions in our store example. There is a variety of ways to do this. A security guard can observe a shoplifter taking a piece of merchandise, and stop them. Patterns of behavior can be monitored, such as a shopper with an overly large purse. Security tags are routinely placed on merchandise to prevent them from being taken. All of these techniques compare favorably to the various flavors of intrusion detection systems.

Intrusion detection systems can be a very effective way of preventing attacks over port 80, among a host of other attacks. However, there is much variation of intrusion detection systems available each with their own capabilities and drawbacks. For example, intrusion detection systems that rely on matching patterns of specific behavior of known attacks are unable to detect previously unseen attacks with different signatures. Vendor provided generic signatures with patterns of general behavior are required to prevent unknown attacks, but they run the risk of blocking valid web requests that are falsely identified as an attack. It is important that any intrusion detection system you choose with this capability is thoroughly evaluated to reduce any false positive detection.

Implementing and supporting an intrusion detection system requires a broad understanding of computer security. The systems are complex enough to be beyond any one person's ability to understand them, so a competent team of professionals is required, which is a challenge in itself to assemble. Because of this, implementing intrusion detection systems should be part of a long-term security strategy.

Summary

We've reviewed a number of possible ways to prevent unknown attacks over port 80. What have we learned? Clearly, preventing future attacks of the same track as Code Red is not a trivial exercise. It requires a comprehensive survey of how the user accesses the web server application, what content is returned to the user, and how the web server is installed. In addition, security related policies and practices should be in place to handle system changes and security incidents. Here are some recommendations based on my research.

- Apply all vendor security patches as soon as they can be verified in a lab environment.
- Require SSL. The cost is relatively low and the benefit is that your server is segregated from the vast majority of systems targeted by mass attacks.
- Protect the web server root and remove installation default directories. Requiring user authentication for the root protects against generic requests for dummy root resources.
- Consider a reverse proxy as an application firewall. Multiple web servers can be protected behind a single reverse proxy, but the cost of maintaining the proxy's configuration can be high.
- As part of a comprehensive security strategy, implement a complete intrusion detection system. Preventing a single intrusion could offset the very high cost.

References

MacVittie, Lori. " Online Only: The Anatomy of an SSL Handshake ". June 11, 2001.
URL: http://www.networkcomputing.com/1212/1212f415.html

Netcraft. "Web Surver Survey". January 2001.
URL:http://www.netcraft.com/Survey/

Prosise, Chris and Udayan Shah, Saumil. "Double-crossed by proxy". April 12, 2001.
URL: http://builder.cnet.com/webbuilding/0-7532-8-5520092-1.html

Netscape Communications Corporation, "Netscape Proxy Server Administrator's Guide for Windows NT, Chapter 7 : Reverse Proxy" . March 1998.
URL: http://developer.netscape.com/docs/manuals/proxy/adminnt/revpxy.htm

Hudson, Kurt . "An Introduction to Microsoft Proxy Server" . August 2000.
URL: http://www.windowsitlibrary.com/Content/265/1.html

Apache Week . "Using User Authentication" . 1996.
URL: http://www.apacheweek.com/features/userauth

RSA Security Inc. "RSA ACE/Agent v 4.4 for Windows NT Administration Guide". June 2000.

Powell , Eddie. "Network Intrusion Detection for the E-Commerce Environment" . July 10, 2000 . URL:
http://www.securityfocus.com/frames/?focus=ids&content=/focus/ids/articles/idsforecomm.html

Internet Security Systems . "Network- vs. Host-based Intrusion Detection: A Guide to Intrusion Detection Technology" . October 2, 1998.
URL: http://secinf.net/info/ids/nvh_ids/

Allen, Julia et al. "State of the Practice of Intrusion Detection Technologies" . January 2000
URL: http://www.sei.cmu.edu/publications/documents/99.reports/99tr028/99tr028abstract.html

Entercept Security Technologies . “Reviewers Guide for entercept 2.0 “ .

DeJesus, Edmund X. “Application Security: Securing The Final Frontier” . August 2000
URL: http://www.infosecuritymag.com/articles/august00/features2.shtml