Objective:

For our project we will start with an installation of Apache on a Solaris image. Since the default installation of Apache is not secure, we will begin with a version which has been specially compiled and configured with multiple virtual hosts, both of the name based and ip based type. We intend to show how Apache can be made more secure through being specially compiled and carefully configured. We will use this Apache installation as the foundation of our project. We will also implement SSL certification on our web server. Aside from our configurations of Apache, we will aslo integrate with our Apache installation an SQL server which we will attempt to secure against SQL injection attacks. Finally we will use Nikto to determine vulnerabilities in our web server. Once we have identified these vulnerabilities, we will attempt to close them in order to make our web server more secure. We will also show how Nikto can be used to detect vulnerabilities in a weak webserver.

Demonstration Notes:

--Two separate systems:

--First a compiled and configured version of Apache (created in INT525) installed on a Solaris virtual machine (Vmware) – demonstrate how this installation is more secure than a default installation

--Second a default installation of apache and SQL installed on a Fedora Core virtual machine along with Damn Vulnerable Web App – demonstrate why default installations are dangerous

Study Notes:

Default Installations

Default installations are dangerous and should never be used "out of the box". The reason for this is that default installations include a large number of settings and features which are enabled by default which open vulnerabilities to your system – features may been enabled which the user or administrator may not even be aware of; however, this will not stop an attacker from being aware of the vulnerability and exploiting it. It is important to understand the vulnerabilities opened by default installations so that measures can be taken to protect the server against attacks.

Security risks posed by default installations:

--All options are enabled by default – including the options which allow cross site scripting, directory indexing and following of symlinks

--AllowOverride is enabled by default and set to "all"

--Apache may be configured to run with root privileges

How to make Apache more Secure

*Compile rather than default install

--Compiling Apache allows the administrator to have a greater amount of control over the features and modules which are enabled – all functions which will not be used can be disabled at the time of compiling.

--The "Data Footprint" can be obscured – information about the server can be changed, and the server can be configured to answer a certain way to queries about itself – Apache can be made to lie. While this does not make the server secure, it can discourage attackers if the server does not act the way an attacker expects it to.

--Server can be compiled to focus on speed, reliability or *security*

*Disable all options (set Options to "None") or disable options which offer security risks
--Turn off directory indexing – directory indexing can allow an attacker to map the directory structure and contents of your web server which will make it easier for them to attack the server
--Turn off cross site scripting – this allows other websites to make queries to your server – an attacker can use this to exploit your server
--Do not allow Apache to follow symlinks – an attacker can use this ability to gain access to protected files by creating symlinks to them (such as the /etc/passwd or /etc/shadow file)
*Set AllowOverride to "None" - Nothing should be allowed to override the permissions set in the main configuration file
*Configure all sites to run on SSL
*Run Apache in a chrooted environment – "put Apache in Jail" - give Apache only access to the resources which it requires to run and nothing else – *never allow Apache to run as root or have root access*

Virtual Hosts
--Allow more than one website to be hosted on a single web server
--Allows shared websites to make use of private IP address space or share IP addresses in order to take up less IP address space
--Each virtual host can be configured with its own username and home directory in order to separate each website from the other – none of the virtual hosts should be able to access files served by the other virtual hosts
--Two Types:
--Name Based Virtual Hosts
-Share a single IP address and are accessed based on name (IE: www.mom.shop.com)
-More difficult to secure because Apache does not have a way to run name based virtual hosts on SSL by default (because they share the same IP address)
-Can be secured with SSL by using Apache add on "GNU_TLS"
--IP Based Virtual Hosts
-Each have their own individual IP address
-Easy to secure using SSL because they each use separate IP addresses

SSL (Secure Sockets Layer)
--Encrypts segments of Network data
--Used to ensure authenticity and confidentiality
--Used to secure the transaction of sensitive materials such as credit card numbers for online shopping
--SSL is not perfect (as we have seen); as always security should be applied in layers; SSL should not be solely trusted.

How to Create an SSL Certificate and Key
--This is an automated process (no people involved!). *Anyone* can create a self signed certificate (try it!)

Generate a Private Key
Openssl genrsa –des3 –out server.key 1024
Generate Certificate Signing Request
Openssl req –new –key server.key –out server.csr
Remove Passphrase from Key

Openssl rsa –in server.key –out servername.key

Generate Self-Signed Certificate

Openssl x509 –req –days 365 –in server.csr –signkey servername.key –out server.crt

--Be sure to use the domain as the common name when creating the certificate request otherwise the certificate will not be accepted because the domain and the common name will not match

Nikto

Nikto is a web scanner specifically targeted at scanning web servers. Nikto will scan the target web server looking for exploits. It will then output the dangers found on the system and reasons why they are dangerous – IE: how an attacker can use these exploits against your system.

Nikto will also attempt to identify the type of server which is running. For an attacker, identifying the type of server running and the particular build of the server (IE: Apahce 2.1) is useful because the attacker can then predict how the server should react to certain exploits or attempts to communicate. It is also useful if an attacker is looking for a particular type of server and build to attack. This is why it is useful to obscure the data footprint – if the server does not respond the way an attacker expects it to, they may become discouraged and abandon an attempt to exploit the server.

http://cirt.net/nikto2

Our Nikto scans clearly show that the compiled and configured Apache server is by far more secure than the default installation.

Damn Vulnerable Web App

What is it?

It is a web application that is damn vulnerable. It is light weight, easy to use and full of vulnerabilities to exploit.

Who uses it?

It has been developed for the use of information security professionals and students to test the tools in a legal environment.

Some Vulnerabilities it provides?

SQL Injection

- ✓ It is a code injection technique that exploits a security vulnerability occurring in the database layer of an application.

XSS (Cross Site Scripting)

- ✓ It is a type of computer security vulnerability typically found in web applications which enable malicious attackers to inject client-side script into web pages viewed by other users. An exploited cross-site scripting vulnerability can be used by attackers to bypass access controls.

LFI (Local File Inclusion)

- ✓ This is when you find a particular file within a database and uses it against the web server. Such as discovering the faithful /etc/passwd/ username/password file, cracking the MD5 hash, (the format for encryption is {CRYPT}1salt$encrypted_pass) and then logging in via ssh.

.RFI (Remote File Inclusion)

- ✓ The main idea behind it is that the given code inserts any given address, local or public, into the supplied include command. The way it works is that when a web-site is written in PHP,

there is sometimes a bit of inclusion text that directs the given page to another page, file or what you have.

Login Brute Force

- ✓ It consists of trying every possible code, combination, or password until you find the right one. After a successfully brute force attack you are more likely to have achieved a password for an account. For example, a brute-force attack may have a dictionary of all words or a listing of commonly used passwords. So in order to gain access to the account using a brute-force attack, the program would try all the available words it has to gain access to the account.

Damn Vulnerable Web App is damn vulnerable. You should not upload it to your hosting provider's public html folder or any working web server as it will be vulnerable for hacking.
It is always recommended to download and install XAMPP onto a local machine inside your LAN which is used solely for testing.

Website: http://www.dvwa.co.uk/

Dangers of SQL Injection

What is SQL Injection?

A code injection technique that exploits a security vulnerability occurring in the database layer of an Application

When does SQL Injection Occur?

When user input is incorrectly filtered for string literal escape characters embedded in SQL statements this causes unexpected execution of unauthorized code

SQL Injection Examples

- **Incorrectly filtered escape characters**
  statement = "SELECT * FROM users WHERE name = '" + userName + "';"
- **Incorrect type handling**
  statement := "SELECT * FROM data WHERE id = " + a_variable + ";"
- **Conditional responses**
  SELECT student_name FROM student_list WHERE stuId = 'mark14cd' AND 1=1;

Securing MySQL

- Data Clean up

Firstly, all input should be validated. A good way to validate the input would to use regular expressions. Simply define what input is allowable only allow that.
All client-supplied data needs to be cleansed of any characters or strings that could possibly be used maliciously. The best way to filter your data is with a default-deny regular expression. You should make it so that you only include that type of characters that you want.
For example, if the input is first name then only allow you would not need quotes or commas in first name. This following regexp will return only letters and numbers --> s/[^0-9a-zA-Z]//g

- Use minimum privileges.

Secondly, make sure that the application user has specific minimum rights on the database server.
If the application user on the database uses root/administrator on the database then it surely needs to be reconsidered unless the application user really needs such high amount of privileges or can see to it that it be reduced. Do not give the application user permission to access system stored procedures. Only allow access to the ones that the user created.