Lecture 27: Web Security: PHP Exploits, SQL Injection, and the Slowloris Attack

Lecture Notes on "Computer and Network Security"

by Avi Kak (kak@purdue.edu)

April 15, 2017
8:15am

Goals:

- What do we mean by web security?
- PHP and its system program execution functions
- An example of a PHP exploit that spews out third-party spam
- MySQL with row-level security
- SQL Injection Attack
- The Slowloris Attack
- Protecting your web server with `mod-security`

CONTENTS

27.1: WHAT DO WE MEAN BY WEB SECURITY?

- Obviously, practically all of the security-related fundamental notions we have covered so far are relevant to many of our activities on the web. Where would web commerce be today without the confidentiality and authentication services provided by protocols such as TLS/SSL, SSH, etc?

- But web security goes beyond the concerns that have been presented so far. **Web security addresses the issues that are specific to how web servers present their content to web browsers, how the browsers interact with the servers, and how people interact with the browsers.** This lecture takes up some of these issues.

- Until about a decade ago, the web servers offered only static content. This content resided in disk files and security consisted primarily of restricting access to those files.

- Bow now web servers create content dynamically. Newspaper pages and the pages offered by e-commerce folks may, for ex-

ample, alter the advertisements in their content depending on what they can guess about the geographical location and personal preferences of the visitor. Dynamically created content is also widely used for creating wikis, in serving out blog pages with user feedback, in web-hosting services, etc.

- Dynamic content creation frequently requires that the web server be connected to a database server; the information that is dished out dynamically is placed in the database server. This obviously requires some sort of middleware that can analyze the URL received from a visitor's browser and any other available information on the visitor, decide what to fetch from the database for the request at hand, and then compose a web page to be sent back to the visitor. **These days this "middleware" frequently consists of PHP scripts, especially if the web server platform is composed of open-source components, such as Apache for the web server itself and MySQL as the database backend.**

- Although the issues that we describe in the rest of this lecture apply specifically to the Apache+PHP+MySQL combination, similar issues arise in web server systems that are based on Microsoft products. What is accomplished by PHP for the case of open-source platforms is done by ASP for web servers based on Microsoft products.

- For the demonstrations in this lecture, I will make the following assumptions:

  – That you have the Apache2 web server installed on your Ubuntu machine. The installation of Apache2 was addressed earlier in Section 19.4.2 of Lecture 19. In what follows, I will add to the Apache-related comments made earlier in Lecture 19.

  – That your Apache2 server is PHP7 (PHP version 7) enabled. That you can ensure that through the following three steps:

    1. Enter the following two directives at the bottom of your `/etc/apache2/apache2.conf` file:

       ```
       <FilesMatch "\.php$">
       SetHandler application/x-httpd-php
       </FilesMatch>

       <FilesMatch "\.html$">
       SetHandler application/x-httpd-php
       </FilesMatch>
       ```

       The first of these two directives tells the HTTPD server that should there be a browser request for a document whose name carries the "php" suffix, that document must first go through PHP preprocessing and only the output produced by the preprocessor should be sent to the browser. The second directive applies the same rule to browser requests for HTML documents — a rule that needs to be enforced when the web pages (in HTML) hosted by the server contain embedded PHP code.

    2. You'd need to add the Apache module `php7.0` to the set of modules that you see in the directory `/etc/apache2/mods-avaialble/`. This you can do by the following install command:

       ```
       sudo apt-get install libapache2-mod-php7.0
       ```