Website Vulnerability Testing

Developer Documentation

*Revision 0.1*

Table of Contents

[1. Overview 3](#_Toc508107030)

[2. Tools 3](#_Toc508107031)

[a. SSLyze - SSL/TLS Scanner 3](#_Toc508107032)

[b. WPScan - WordPress scanner 3](#_Toc508107033)

[c. OpenSSL – File Encryption 3](#_Toc508107034)

[3. Implementation 4](#_Toc508107035)

[a. Virtual Machine 4](#_Toc508107036)

[b. Scripts 4](#_Toc508107037)

# Overview

This package makes use of two tools called SSLyze and WPScan to conduct vulnerability testing. Both of these packages are a mix of custom testing and other well-vetted tools from the open-source community.

Along with those, OpenSSL is used to encrypt the results of the scans before they are emailed to our collection server.

# Tools

## SSLyze - SSL/TLS Scanner

The SSLyze package is built in Python and can be run on nearly any operating system. When run, it sends one or more requests to the targeted website in order to analyze the connection. Then many tests are conducted against the gathered information, including:

* + If SSL or TLS are enabled
  + Supported encryption algorithms/methods
  + Policies set by the request (such as HSTS)

## WPScan - WordPress scanner

WPScan is built in Ruby and maintained by three developers on GitHub. It uses “black box” testing techniques to passively find potential vulnerabilities in active WordPress installations (“black box” refers to the tool not knowing anything about the website before it begins its scan). It can also be used to determine the login credentials of users, though this is a more advanced and less reliable attack vector.

WPScan will gather information such as the installation’s version, enumerated lists of plugins, themes, thumbnail tools, and users that may be vulnerable to attack.

## OpenSSL – File Encryption

OpenSSL is a command line tool for taking advantage of many different encryption utilities, ranging from certificate generation to file encryption. I have based my implementation off of instructions found [here](http://www.czeskis.com/random/openssl-encrypt-file.html).

The process utilizes a mix of symmetric and asymmetric encryption. The contents to transmit are encrypted symmetrically using a key generated on-machine. Then that key is asymmetrically encrypted using our own public key. The contents and encrypted key are transmitted. Once we receive the files, we decrypt the symmetric key using our private key. Finally, decrypt the contents using the newly gotten symmetric key.

# Implementation

## Virtual Machine

All of the above tools are hosted inside a Kali Linux Light virtual machine. This machine is exported to a portable format distributable on a USB drive. It contains our own security testing tools as well as some popular security testing tools not used in this product.

## Scripts

Scripts have been written to tie all of the tool functionalities together. This allows our end users to only enter their necessary details (such as website address) once, which is then ported to each individual executing of a tool. All such tools are also ensured to be run with optimized settings in place, to ensure effectiveness of the overall test. The scripts included are:

*run.desktop* – Simple way to start our scripts in a shell that novice users know how to use. This also logs the output of *run\_tests.sh* to a log file.

*run\_tests.sh* – Execute each of our individual tests. Then gather up their logs (as well as the log up to this point of my own output), zip, encrypt, and then send to the companies secured endpoint.

*ssl\_test.sh* – Use SSLyze to analyze the client’s website’s SSL implementation, in order to find any common configuration errors or possible improvements.

*wp-scanner.sh* – After updating its vulnerability database, use WPScan to scan the client’s website. This will look for a WordPress installation, its version, installed plugins and themes, thumbnail vulnerabilities, and possible login usernames. Within all but the username scan for known vulnerabilities.