LWASP

Linux Watchful Adaptive Security Profiler

About

- LWASP is a service to analyze and score practice linux images in preparation for cyber security competitions
- Very versatile and open to modification, so that coaches and mentors can customize it specifically for their teams if they choose to do so
- Tested on Ubuntu 12.04, 14.04 and 16.04, but should work for all debian-based linux distributions
- Uses Bash and Python for the backend, Javascript with ReactJS and JQuery for the front end

Initialization

1. Follow the instructions in the LWASP Elements Generator Excel file to create the elements.csv file in the lwasp directory (make sure to allow macros if using excel).
2. If you have your own logo, please replace the default with it in lwasp/ScoringEngine/, and make sure it is named logo.png. This logo is used for the desktop shortcuts and on the scoring report, and is an easy way to personalize this service.
3. Move the lwasp directory onto the Purposefully Vulnerable Linux Image (PVLI) that needs to be scored (recommended in /etc/)
4. Open a command prompt (terminal) on the PVLI and run the following commands:
   - cd /path/to/scoring/engine/lwasp (e.g. cd /etc/lwasp)
   - sudo python initialize.py
5. Answer the prompts:
   - Timed Image: If yes, will count down timer on scoring report and prevent the score from being updated once time is up
   - Email: If a timed image, will email the scoring report once the time is elapsed on the image. If not a timed image, will create a desktop shortcut to send scoring report.
   - Image Name: The general name for the image, will be at the top of the scoring report and in the email
6. Shut down the image, and distribute it to competitors. The scoring engine will start on next boot.

Steps for Competitor to Take

1. Once the image is started and they log in, they double click the "Set ID" shortcut on the desktop to set their unique ID. You will receive this ID and the image ID you set during initialization in the email. This is so that if you have several duplicate images, you can tell them apart.
2. They try to secure the image, gaining points by what was set in the elements file.
3. The scoring report will refresh every time a file is changed and once every minute. If this is not often enough or this does not seem to be working, they can type 'sudo refresh' into a terminal to reload the score manually

Notes

- You can reach me for contact at contact@nsccmanager.com with any questions, concerns, etc
- Forensics questions are in the CyberPatriot model; only checks the text after “ANSWER: “ (can support multiple answers.They try to secure the image, gaining points by what was set in the elements file.

Modifying LWASP

Add Scoring Categories

1. Add a function in analyze.py that can take in the extras as parameters and return True or False as to whether that item should be marked complete according to those parameters
2. Near the bottom of the analyze.py file, where the recording file is read, add your category beneath all of the others, with the same syntax, but use function = myNewFunction(extras...)
3. In initialize.py, add your new category to line 104 where it checks each type for validity.
4. That's it! Add your category in the elements.csv file with the needed extras, and start using it. Contact me with any questions.

Breakdown of the files

(indicates a file that you should provide)**

	//// These are the python files that do all of the grunt work ////
initialize.py	Sets up the scoring engine to run, and puts everything in motion.
analyze.py	Called every minute from a crontab and every time a file that is scored changes. It parses through what should be scored and determines what score the user of the image has achieved, then writes that back to recording.json.
restart.py	Sets up the file watches and does several other tasks that must be run on boot. It is called from cse.bash
	//// These are the files that are referenced/used by initialize.py ////
cron.bash	Called in initialize.py to setup the cron job that calls analyze.py. It is only ever called once
cse.bash	Moved into /etc/init.d from initialize.py and set to run at boot. It simply calls restart.py when run
**** elements.csv**	The user-generated comma-separated-value file that contains what should be scored and what should be penalized
	//// These elements in the ScoringEngine folder make up the UI ////
main.js	The heart of what powers the UI. It is loaded from ScoringReport.html, which is opened when the user double clicks on it from their desktop
jquery.js	Used in main.js to power the ajax calls that retrieves recording.json and settings.json in javascript to do some calculations on and display to the user
transform.js	What powers the use of JSX in main.js. See https://facebook.github.io/react/docs/tooling-integration.html for more information. logo.png is exactly what it sounds like
**** logo.png**	The image that is used on the scoring report and on the desktop images
react.js	Powers the react code in main.js
	//// These store data to be used by the main three python scripts above ////
recording	Stores all of the user-imported elements from elements.csv but in a more computer friendly format, also adding a field for whether each item is completed.
settings.json	Contains some user-generated settings that were entered during initialization.

- Feel free to read through all of my code, I have tried to comment it as much as needed. If you know any amount of python it should be fairly easy to understand and modify to fit your needs.
- main.js is written in javascript using a framework called React, which was created by Facebook in 2013. Feel free to research and learn it, but I have to warn you that just messing with it as plain javascript may not work that well.