6.8. Lab #8: The Fuzzer

Submit Assignment

Objectives

1. Be exposed to software testing techniques.
2. Learn and implement a fuzzer.

Overview

Fuzz testing or fuzzing is an automated software testing technique (black box) throwing invalid, malformed, and unexpected data at a computer program. The goal is to make a program crash, violate invariants, to find potential memory leaks, evidence of user input not handled correctly, and other bugs. Think of fuzzing as poking holes at a computer program. A fuzzer is a program that is used for fuzz testing. In this lab, you will write a fuzzer to find at least reflected Cross-Site Scripting (XSS) bugs.

From OWASP: "Reflected Cross-Site Scripting (XSS) occur when an attacker injects browser executable code within a single HTTP response. The injected attack is not stored within the application itself; it is non-persistent and only impacts users who open a maliciously crafted link or third-party web page. The attack string is included as part of the crafted URI or HTTP parameters, improperly processed by the application, and returned to the victim."

Basic Requirements

At the very least, your fuzzer must be (1) written in Python and (2) must able to determine that the page at [http://www.cs.tufts.edu/comp/120/hackme.php?token=FoodlerLinks to an external site.](http://www.cs.tufts.edu/comp/120/hackme.php?token=Foodler) has a reflected Cross-Site Scripting (XSS) vulnerability.

This lab is worth a total of 10 points. Completing this basic requirement will give you 9 points.

Going Beyond

The way I am structuring this lab is to give you freedom and flexibility to do something cool. The more you put into writing your fuzzer, the more you will be rewarded.

* (+1) Use all the fuzzing lists that Daniel Miessler provided in [https://github.com/danielmiessler/SecLists/tree/master/Fuzzing (Links to an external site.)](https://github.com/danielmiessler/SecLists/tree/master/Fuzzing). NOTE: your fuzzer must prompt user to enter location of where the SecLists folder is on workstation for portability reasons.
* (+2) Extend the fuzzer to test any page for reflected Cross-Site Scripting vulnerabilities.
* BONUS Opportunities (+1):
  + Extend the fuzzed to test for another web security vulnerability or to fuzz not just web pages and web applications (e.g., a remote application running on some remote port --e.g., SMB that is a target for WannaCry).
  + Something way cool not listed or perhaps I did not think of. Please email me your idea.

What's The Point of This Lab?

Burp Suite, OWASP ZAP, and many web application security suites come with a fuzzer. Case-in-point: watch "How to Fuzz Websites for Cross-Site Scripting (XSS) Using Zed Attack Proxy (ZAP)" ([How to Fuzz Websites for Cross-Site Scripting (XSS) Using Zed Attack Proxy (ZAP) (Links to an external site.)](https://www.youtube.com/watch?v=rmbi-VbIK6I)[](https://www.youtube.com/watch?v=rmbi-VbIK6I)). While fuzzers have proven to be valuable, it is important for you to understand how fuzzers work as a good security practitioner. This exercise also provides you more exposure to software testing which isn't emphasized enough (or at all) in a Computer Science curriculum. On a personal note, fuzzing is a topic I still wish I had more exposure to when I started delving into Cyber Security over a decade ago.

The README File

This README file shall describe the work. This description must:

* Briefly explain what this tool is (that is, an overview).
* Briefly explain how the tool works.
* Identify what aspects of the work have been correctly implemented and what have not.
* Identify anyone with whom you have collaborated or discussed the assignment.
* Say approximately how many hours you have spent completing the assignment.
* Be written in either text format (README.txt). No other formats will be accepted. Submission

Your submission shall contain at least two files: the README file and one Python .py file.

References

* [https://www.owasp.org/index.php/Fuzzing (Links to an external site.)](https://www.owasp.org/index.php/Fuzzing)
* [https://www.owasp.org/index.php/Testing\_for\_Reflected\_Cross\_site\_scripting\_(OTG-INPVAL-001) (Links to an external site.)](https://www.owasp.org/index.php/Testing_for_Reflected_Cross_site_scripting_(OTG-INPVAL-001))

[Previous](https://canvas.tufts.edu/courses/28438/modules/items/371930)[Next](https://canvas.tufts.edu/courses/28438/modules/items/371932)

<https://medium.com/@hungry.soul/writing-a-scanner-to-find-reflected-xss-vulnerabilities-part-1-5dd6de7d1a35>

<https://www.thepythoncode.com/article/make-a-xss-vulnerability-scanner-in-python>

<https://code.activestate.com/recipes/496942/>

<https://stackoverflow.com/search?q=xss+in+python>

<https://github.com/matthewdfuller/intellifuzz-xss>

<https://github.com/DavidCJKennedy/MutaFuz>

https://github.com/sohailahmedkhan173/Simple-SQL-XSS-Fuzzing-Tool-PYTHON-