Fuzzing Technique in Web Applications and Beyond

Danyang Zhao
College of Information Security, Chengdu University of Information Technology, Chengdu, 610225, China

Abstract. Web security is an old yet important topic. With the emergence of new applications and new threats, we need new techniques to fight against these threats. Fuzzing technique has been proven as an effective tool and has the potential of evolving with the new threats. In this paper, we discuss the applications of fuzzing technique in both web and other areas. We also present a case study by implementing a fuzzer in this study.

1. Introduction
For web application, security is critical to protect users’ information and understanding the security issues, and security vulnerability is the most important causes of security threats. Recent years, information leakages happen more frequently, including the Huazhu Group information leakage and Weibo information leakage incidents, which have affected at least 5.38 million users. These all related to the lack of security construction on the web applications. Static analysis, dynamic analysis, symbolic execution, fuzzing are traditional vulnerability discovery techniques.

Static analysis relays on source code, by analyzing grammar, lexical, semantics features it can find bugs and don’t need actually executing programs. However, static analysis has a high false rate. Dynamic analysis is analysis performed on programs while they are executing. It relays on heavy people involvement, so it has a low efficiency. Symbolic execution symbolizes the program inputs and after the execution, constraint solvers will solve the constraint and find what inputs cause the execution, but is difficult to use into large applications. Fuzzing is a way of providing invalid inputs to programs to find bugs in software, which can help to find and fix critical bugs. Fuzzing is the most efficient technique to find vulnerabilities on web application and of good scalability and applicability. Fuzzers provide payloads to application, which can produce results for little cost.

| Technique         | Efficiency | Accuracy | Scalability |
|-------------------|------------|----------|-------------|
| static analysis   | relatively high | low      | relatively good |
| dynamic analysis  | low        | high     | uncertain   |
| symbolic execution| low        | high     | bad         |
| fuzzing           | high       | high     | good        |

Fuzzers include three categories: mutation-based, generation-based and evolutionary. How they create input to programs are different. Mutation-based fuzzers are one of the easier types to create and are not aware of the expected input format, Peach [1] can make mutation-based fuzzing. Generation-based fuzzers create input based on the expected input format, which according to the configuration file. Peach can also perform generation-based fuzzing. Evolutionary fuzzers are the newer type, allow the fuzzer to use feedback from each test case to learn the better format of the input over time. Example is AFL [2].

Recent research in fuzzing has focused on applications of new techniques, and the use of fuzzing in
some new areas. New techniques like machine learning [3], new areas like Internet of Things (IoT) [4]. We would give a further discussion in Section 3.

In this paper, we would discuss the usage of fuzzing technique in both the web applications and other applications, especially those which only appear in recent years. We also present a case study by implementing a fuzzer in this study.

2. Fuzzing Technique in Web Applications

In this section, we will introduce some fuzzing techniques used in web application. Unlike traditional fuzzers, most of these web fuzzers rely on dictionaries rather than creating random inputs or mutated data, although part of the generation of the dictionary may depend on the mutation, the more experienced accumulated, which makes more efficient in fuzzing. These fuzzing techniques includes multi-test fuzzing and fuzzing for specific vulnerabilities.

2.1. Multi-test Fuzzing

The general-purpose fuzzers can not only fuzz various vulnerabilities, but also cover the scanning of vulnerable paths, the discovery of logic problems, etc. The fuzz area is very wide, and more problems can be found.

Wfuzz [5] is a multi-test fuzzer that can be used to find sensitive directory, find vulnerability like sql-injection, XSS, XXE. The taster can input target that contains keyword FUZZ in the place you want to fuzz, which will be replaced by given payload or random payload.

![Figure 1. WFuzz.](image1)

Burp Suite [6] is a fully featured web application attack tool, the earliest version was in 2009. It contains the fuzzing part called intruder. Taster can use this type $target$ to choose the place you want to fuzz by using provided payloads.

![Figure 2. The BurpSuite running screenshot.](image2)
2.2. Specific Vulnerability Fuzzing

Although the specific vulnerability fuzzers can fuzz a narrower surface, it is more in-depth in vulnerability detection. Generally, the fuzzers for a specific vulnerability will specify different dictionaries and strategies for different situations of the vulnerability. Using fuzzy strategies for different situations will increase the probability of finding vulnerabilities and improve understanding of the specific issues of vulnerabilities.

2.2.1. Cross-site Scripting (XSS)

One of the most common attacks in web applications is XSS. The impact of such attacks is becoming more and more widespread. The attacker injects javascript statements into the page forms or links of the website to manipulate the content of the page and obtain information such as user’s cookies. XSS generally includes three types: reflex-XSS, storage-XSS and DOM-XSS. When fuzzing vulnerability, in addition to the most commonly javascript payloads, people try to continuously optimize attack statements and attack methods to get higher detection efficiency, such as applying genetic algorithms to the generation of payloads [8], or use ART method to improve detection efficiency [9] The most commonly used tool for detecting XSS is XSSer developed by OWASP.

```
root@kali:~# xsser -h
Usage:

xsser [OPTIONS] [-:all <url> |-u <url> |-i <file> |-d <dork> (options)] [-l ] [-g <get> |-p <post> |-c <crawl> (options)]
[Request(s)] [Checker(s)] [Vector(s)] [Anti-XSS/IDS] [Bypasseral(s)] [Techniqu
e(s)] [Final Injection(s)] [Reporting] [Miscellaneous]

Cross Site "Scripter" is an automatic framework to detect, exploit and report XSS vulnerabilities in web-based applications.

Options:
- -version show program's version number and exit
- -h, --help show this help message and exit
- -s, --statistics show advanced statistics output results
- -v, --verbose active verbose mode output results
- -gtk launch XSSer GTK Interface
- --wizard start Wizard Helper!

*Special Features*:
You can set Vector(s) and Bypasseral(s) to build complex scripts for XSS code embedded. XST allows you to discover if target is vulnerable to 'Cross Site Tracking' [CAPEC-107]:

Figure 3. XSSer.
```

2.2.2. SQL Injection

SQL Injection is one of the most common threat of web-based applications. The attacker inserts the SQL payload in the statement that queries the database to achieve the purpose of obtaining database information. SQL injection can be divided into different levels according to whether there are errors or whether it can respond to time. In the test for SQL injection, many tools will set different payloads for different levels of SQL injection. People optimize the payloads through mutation method to get more available test cases, mutation method includes Value mutations, Decision mutation, Statement mutation.

Sqlmap [7] is an open source penetration testing tool that automates the process of detecting and exploiting SQL injection flaws and taking over of database servers. The tester can provide target and choose the aim and level.
3. Fuzzing Technique in Other Applications

3.1. Fuzzing Technique with Artificial Intelligence

Artificial Intelligence (AI) has gained a great progress in the past decade with the wide usage of deep neural networks. In the famous ImageNet competition, convolutional neural networks become dominant and even outperform humans for image classification. For object detection, several frameworks including SSD, Yolo, etc., have been widely used in surveillance systems. Deep learning is also used for many different other problems, e.g., image segmentation [10], traffic forecasting [11], stock prediction [12], in which it shows a great success. While the deep neural networks have shown a great improvement when compared with traditional machine learning models, they are also more vulnerable when faced with adversarial attacks.

The combination of the fuzzing technique and artificial intelligence is in two aspects. In one hand, the fuzzing technique can be used for tests of AI models, which would help to build more robust models. On the other hand, AI models can help build better fuzzers. In this part, we would review some recent work from these two aspects. For more discussion, we recommend the readers to recent surveys [3, 13-15].

3.1.1. Fuzzing Technique for AI Models

Due to some reasons (such as deviation of training data, over-fitting or under-fitting of the model, etc.), neural networks often exhibit incorrect behaviors under certain conditions. These are incorrect in environments with extremely high safety requirements. Actions often have disastrous consequences. For example, both Google's self-driving cars and Tesla's self-driving cars have been involved in traffic accidents due to their inability to handle some special situations. In view of the complex internal structure of neural networks, it is difficult for traditional software system testing methods to quickly and efficiently detect defects in neural networks. Detecting hidden defects in neural networks has become one of the research hotspots in the field of deep learning. Due to the advantages of high degree of automation, low system consumption, and no dependence on the source code of the target program, fuzzy testing is widely used by testers. In recent years, researchers have improved fuzzers around program features, mutation strategies, and abnormal sample discovery and analysis to improve test efficiency and have achieved initial results for neural networks [16].

Figure 4. Sqlmap.

```
root@kali:~ # sqlmap -h

Usage: python sqlmap [options]

Options:
  -h, --help         Show basic help message and exit
  -hh                Show advanced help message and exit
  --version          Show program's version number and exit
  -v VERBOSE         Verbosity level: 0-6 (default 1)

Target:
  At least one of these options has to be provided to define the target(s)

  -u URL, --url=URL  Target URL (e.g. "http://www.site.com/vuln.php?id=1")
  -g GOOGLEDORK     Process Google dork results as target URLs
```
3.1.2 AI Models for Fuzzing Technique

AI models are used to offer useful tools to overcome challenges in the fuzzing process [3]. One of the most successful applications of AI to fuzzing occurs in the generate inputs stage. Godefroid et al. [17] used LSTMs to create input grammar for PDF files in generation-based fuzzers. As a special type of recurrent neural networks (RNNs), LSTMs have shown promise in many types of sequence generation tasks [18]. In terms of program analysis and vulnerability mining, Pang et al. [19] have successively proposed a method for identifying early vulnerabilities in software components based on support vector machine (SVM) ensemble learning. They use N-Gram model and statistical feature selection to predict the vulnerability. Pang et al. [20] applies deep learning to binary vulnerability prediction and malicious code identification. Wang et al. [21] used deep learning networks to analyze the file structure and guide the generation of seed files for Fuzz. Sablotny et al. [22] proves that RNNs, especially Gate Recurrent Unit (GRU), can create HTML tags, which can then be used in browser fuzzing.

3.2 Fuzzing Technique with Internet of Things

Driven by the rapid development of wireless technology, sensor technology and communication technology, a comprehensive network called the Internet of Things is rapidly developing. The Internet of Things consisting of wireless sensor networks and radio frequency identification sensors covers various fields, such as health, education, transportation, etc. With the development of the Internet of Things, the security problems of the Internet of Things have become increasingly prominent, and the security problems need to be solved urgently.

Fuzzing technique has been proposed as a potential solution to the security problem in the Internet of Things. We would name some of the applications of fuzzing techniques with Internet of Things. Lai et al. [23] used the anti-sample idea implemented a Modbus TCP test case generation model, in view of the development of fuzzing and the vulnerability characteristics of the industrial control protocol. They captured three types of Modbus TCP vulnerabilities with the experimental method successfully, indicating that fuzzers could effectively detect industrial control protocol vulnerabilities. Alhawi et al. [24] investigate software verification techniques to detect security vulnerabilities in typical Unmanned Aerial Vehicles (UAVs). They show that fuzzing and bounded model checking (BMC) techniques can detect various software vulnerabilities, which are of particular interest to ensure security in UAVs.

4. Case Study of Fuzzing Technique

Based on the above understanding of fuzz technology and web vulnerability research, we implement a sample fuzzer, which includes the content of two modules, bypass module and fuzz module.

![Figure 5: Tool modules.](image-url)
4.1. Fuzz Module
The user can select the fuzz module by submitting the parameter type. This module supports two forms of fuzz: a set of given numeric ranges, the location of the specified dictionary file.
- The user needs to provide the url with the keyword "FUZZ"
- Select the fuzz form: -r selects a set of numeric ranges, and -f selects the dictionary.

Compared with the simple fuzzy testing of traditional test software, when testing web applications, a set of regular dictionaries will be more targeted to the vulnerability and have a better effect. The user can directly specify the dictionary that comes with the tool, or using other dictionary.
- The user can decode the payload by input -e
- When fuzz ends, the user can see a table showing the status code and contents length returned by each target to judge the result of the fuzz.

```
python(8) C:\Users\aron\Desktop\fuzzing\python\sfuzz.py -f fuzz-f1.txt -r "https://www.HackerPoa/search/erroneous.html"
```

Figure 6. The fuzz running screenshot.

4.2. Bypass the WAF Module
The user chooses bypass WAF module by submitting the parameter type. The implementation method of this module is similar to the FUZZ module.
- The user can specify the dictionary and don’t needs to submit the URL.
- The module will automatically replace spaces and other places in specific dictionary.

5. Conclusion
In this paper, we discuss the current situation of the fuzzing technique used in both web and other applications. Based on our discussion, we find that fuzzing technique shows a great advantage and has a potential of being applied in new areas. We also implement a sample fuzzer, which could be a part of a larger system in the future.

References
[1] Peach Tech. 2018. Peach. https://www.peach.tech/products/peach-fuzzer/. Accessed on June 6, 2020.
[2] Michal Zalewski.2017.American Fuzzy Lop. http://lcamtuf.coredump.cx/afl/. Accessed on June 6, 2020.
[3] Saavedra G J, Rodhouse K N, Dunlavy D M, et al. A Review of Machine Learning Applications in Fuzzing[J]. arXiv preprint arXiv:1906.11133, 2019.
[4] Alhawi O.M.K., Akinbi A., Dehghantanha A. (2019) Evaluation and Application of Two Fuzzing Approaches for Security Testing of IoT Applications. In: Dehghantanha A., Choo KK. (eds) Handbook of Big Data and IoT Security. Springer, Cham
[5] Wfuzz, https://github.com/xmendez/wfuzz/. Accesssed on June 6, 2020.
[6] PortSwigger. (n.d.). Burp Suite, https://portswigger.net/burp/. Accesssed on June 6, 2020.
[7] Sqlmap, https://github.com/sqlmapproject/sqlmap/. Accesssed on June 6, 2020.
[8] Duchene, Fabien et al. "XSS Vulnerability Detection Using Model Inference Assisted Evolutionary Fuzzing." 2012 IEEE Fifth International Conference on Software Testing, Verification and Validation (2012): 815-817.

[9] Lv, C., Zhang, L., Zeng, F., & Zhang, J. (2019). Adaptive Random Testing for XSS Vulnerability. 2019 26th Asia-Pacific Software Engineering Conference (APSEC), 63-69.

[10] Badrinarayanan V, Kendall A, Cipolla R. Segnet: A deep convolutional encoder-decoder architecture for image segmentation[J]. IEEE transactions on pattern analysis and machine intelligence, 2017, 39(12): 2481-2495.

[11] Jiang W, Zhang L. Geospatial data to images: A deep-learning framework for traffic forecasting[J]. Tsinghua Science and Technology, 2018, 24(1): 52-64.

[12] Jiang W. Applications of deep learning in stock market prediction: recent progress[J]. arXiv preprint arXiv:2003.01859, 2020.

[13] Li J, Zhao B, Zhang C. Fuzzing: a survey[J]. Cybersecurity, 2018, 1(1): 6.

[14] Liang H, Pei X, Jia X, et al. Fuzzing: State of the art[J]. IEEE Transactions on Reliability, 2018, 67(3): 1199-1218.

[15] Manès V J M, Han H S, Han C, et al. The art, science, and engineering of fuzzing: A survey[J]. IEEE Transactions on Software Engineering, 2019.

[16] Cui J, Zhang X, Chen J. Neural network based optimization of fuzzy test algorithm [J]. Journal of Xiamen University of Technology, 2019, 27(5): 41-46. (in Chinese)

[17] Godefroid P, Peleg H, Singh R. Learn&fuzz: Machine learning for input fuzzing[C]//2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE). IEEE, 2017: 50-59.

[18] Graves A. Generating sequences with recurrent neural networks[J]. arXiv preprint arXiv:1308.0850, 2013.

[19] Pang Y, Xue X, Namin A S. Early identification of vulnerable software components via ensemble learning[C]//2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA). IEEE, 2016: 476-481.

[20] Pang Y, Xue X, Namin A S. Predicting vulnerable software components through n-gram analysis and statistical feature selection[C]//2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA). IEEE, 2015: 543-548.

[21] Wen W, Wu B, Jiao Y, et al. Design and Implementation on Malicious Document Detection Tool Based on Machine Learning [J]. Netinfo Security, 2018, 18(8): 1-7.

[22] Sablotny M, Jensen B S, Johnson C W. Recurrent Neural Networks for Fuzz Testing Web Browsers[C]//International Conference on Information Security and Cryptology. Springer, Cham, 2018: 354-370.

[23] Lai Y, Gao H, Liu J. Vulnerability Mining Method for the Modbus TCP Using an Anti-Sample Fuzzer[J]. Sensors, 2020, 20(7): 2040.

[24] Alhawi O M, Mustafa M A, Cordeiro L C. Finding Security Vulnerabilities in Unmanned Aerial Vehicles Using Software Verification[J]. arXiv preprint arXiv:1906.11488, 2019.