WebFuzzer: Automation Testing Using Fuzzing

Akshay Sharma (20900824) and Prikshit Batta(20980469)

University of Waterloo
a483shar@uwaterloo.ca, p2batta@uwaterloo.ca

Abstract. In our day-to-day life, we are reliant on the internet. Nowadays, web security is the biggest challenge in the corporate world. Web security refers to protecting networks and computer systems from damage to or the theft of software, hardware, or data. A lot of the issues that occur over a web application is mainly due to the improper input provided by the client. Being secure and secure within the online world gets to be exceptionally critical each day and it is significant to secure your site and the information it holds. We need some techniques to protect our web applications from these dangers. Fuzz testing is basically a technique that is used for software testing. The main idea of fuzzing is that we provide invalid or undesired inputs for the purpose of causing the error. The software which we are testing will be checked against crashes or information leaks. We try to generate a random set of inputs and check on the inputs which are not generated by another set of testing techniques. Grammar-based fuzzing tools have been effective in finding bugs and generating good fuzzing files. Fuzzing techniques are usually guided by different methods to improve their effectiveness. However, they have limitations as well. A fuzzer is a program that injects automatically semi-random data into a program/stack and detects bugs. The data-generation part is made of generators, and vulnerability identification relies on debugging tools. Generators usually use combinations of static fuzzing vectors (known-to-be-dangerous values) or totally random data. New generation fuzzers use genetic algorithms to link injected data and observed impact. Such tools are not public yet. In this project, we offer the execution of a grammar-based fuzzer which is utilized to discover the security blemishes in a web application.

Keywords: Fuzzing, Grammar-based Fuzzers, SQL Injection, Cross-site-scripting attacks, HTML Parser, Grammer Miner.

1 Introduction

Web-based applications are accessible from anywhere so the possibility of being attacked is greater. Therefore while developing web-based applications we need to pay attention to the security of these applications [1][2]. As we can see that due to COVID-19 Pandemic cybercrimes have increased by up to 600 percent. It is estimated that, worldwide, cybercrimes will cost dollar 10.5 trillion annually by 2025. The global annual cost of cybercrime is estimated to be dollar 6 trillion per year. Cybercrime cost makes up a value worth 1 percent of the Global GDP. On average, a malware attack costs a company over dollar 2.5 million (including the time needed to resolve the attack. Ransomware is 57x more destructive in 2021 than it was in 2015.

There are 30 million SMBs in the USA and over 66 percent of all SMBs had at least 1 incident between 2018-2020. The average cost of a data breach to a small business can range from dollar 120,000 to dollar 1.24 million. Data breach costs rose from dollar 3.86 million to dollar 4.24 million in 2021, the highest average total cost in the 17-year history of this report. The average cost was dollar 1.07 million higher in breaches where remote work was a factor in causing the breach. Security Driven AI had the best cost mitigation, saving up to dollar 3.81 million (80 percent cost difference). Zero trust security policies saved a dollar 1.76 million per breach.10 percent increase in the average total cost of a breach from 2020-2021. It costs a dollar 180 per record with PII that was breached. Over 50 percent of all cyber attacks are done on SMBs.

Enterprises experienced 130 security breaches per year, per organization, on average. Enterprises saw the annual cost of cyber security increase 22.7 percent in 2021. The annual number of security breaches in enterprise organizations increased by 27.4 percent. On average enterprises needed 50 days to resolve an insider's attack and 23 days to recover from a ransomware attack.71.1 million people fall victim to cyber crimes yearly. Individuals lose dollar 4,476 USD on average.

In the security testing of software, multiple techniques can be used in order to find as many vulnerabilities as possible. Web Applications are broadly utilized by companies and organizations. It can give a gigantic assortment of administrations but a larger part of the time, it is interatomic with databases, conveys basic administrations, and handles delicate data. It is within the interface of the clients and proprietors of web applications that it is secure so that its accessibility isn't compromised, the private information that it has isn't stolen, and its security and judgment are not hurt. In arrange to do this, the improvement cycle must address all the security issues that will speak to a risk to the previously mentioned concerns. Software testing is the process of uncovering flaws in program requirements, design, and coding. It is used to identify the correctness, completeness, security, and quality of software products against specifications. Software testing is the process of measuring the quality of developed computer software. It shows all bugs, bugs, and flaws in the developed software. There are many approaches to software tests, but an effective test complex product essentially process is a referral process, not just a matter of creating and following pathway procedures do not have.

2 Software Testing Techniques

The main objective of software testing is to affirm the quality of the software system by systematically testing the software in carefully controlled circumstances, another objective is to identify the completeness and correctness of the software, and finally it uncovers undiscovered errors.[3] Traditional ways to discover the security bugs in a program include Static Analysis, Dynamic Analysis, Symbolic Execution, and Fuzzing [5]. In show disdain toward the truth that static analysis is an invaluable device to recognize bugs in a program that can be manhandled by software engineers, a drawback of static analysis is that it evaluates the application in a non-live state, and it is especially troublesome to consider all the conceivable malicious inputs an aggressor can allow when doing static analysis. Dynamic Analysis can absolutely distinguish the bugs by examining the runtime conduct of the program, but it encompasses a moderate execution speed and is non-scalable. Technically, Symbolic Execution can cover any execution way within the program, but it gives good results as it were for little programs and is indeed less versatile because it endures from the way blast issue. For expansive scale programs, the execution states increment essentially which surpasses the fathoming capacity of the limitation solvers. Compared to all these approaches, fuzzing does not require the source code and offers tall extensibility. In addition, it features a higher precision because it executes the program in real-time. Figure 1 represents the main features of all these techniques[5].

2.1 Types of Testing:

The different types of software testing techniques can be performed to make sure that the software is bug-free and stable. It is categorized as part of a variety of testing activities, including test strategy, defined test objectives, test deliverables, etc. Software testing is the execution of the software to find defects and bugs.

The software testing is mainly categorized into two types:

1. *Automation testing:* Automation testing is the most important aspect of software testing. Without human intervention, it automates manual design test cases using specialized tools. The most effective technique to increase software testing's effectiveness, productivity, and coverage is automation testing. It is used to repeat the manually, swiftly, and repeatedly completed test scenarios.

Technique	Easy to start ?	Accuracy	Scalability
static analysis	easy	low	relatively good
dynamic analysis	hard	high	uncertain
symbolic execution	hard	high	bad
fuzzing	easy	high	good

Fig. 1. Table Representing Different techniques with their important aspects

2. *Manual Testing :* Manual testing is the process of evaluating any piece of software or application in accordance with client requirements without the use of any automation tools. It is a process of verification and validation, to put it another way. The behavior of an application or piece of software in contrast to the requirements specification is verified through manual testing. manual testing can be further classified into different types of testing.

In software testing, manual testing can be further classified into three different types of testing, which are as follows:

- **White Box Testing Technique:** White Box Testing is a software testing technique in which the internal structure, design, and coding of software are tested to verify the flow of input-output and to improve design, usability, and security. The term "White Box" was used because of the see-through box concept. The clear box or WhiteBox name symbolizes the ability to see through the software's outer shell (or "box") into its inner workings. In white box testing, code is visible to testers so it is also called Clear box testing, Open box testing, Transparent box testing, Code-based testing, and Glass box testing. It is a detailed review of internal logic and code structure In a white box test it is required for a tester to have full knowledge of the source code[3].

- **Black Box Testing Technique:** Black Box Testing is a software testing method in which the functionalities of software applications are tested without having knowledge of internal code structure, implementation details, and internal paths. Black Box Testing mainly focuses on the input and output of software applications and it is entirely based on software requirements and specifications. It is also known as Behavioral Testing. It is a technique for testing the inner workings of an application without knowing it. Only basic aspects of the system are examined, with no or very few relevances to the internal logical structure of the system [3].

- **Grey Box Testing Technique:** Grey Box Testing testing is a software testing technique to test a software product or application with partial knowledge of the internal structure of the application. The purpose of grey box testing is to search and identify the defects due to improper code structure or improper use of applications. In this process, context-specific errors that are related to web systems are commonly identified. It increases the testing coverage by concentrating on all of the layers of any complex system.

$$Whitebox + blackbox = Greybox$$

It may be a strategy to test the application with constrained information of the inside working of an application additionally has the information of principal viewpoints of the framework.[3]

2.2 Fuzzing

Fuzz testing is basically a technique that is used for software testing. The main idea of fuzzing is that we provide invalid or undesired inputs for the purpose of causing the error[4] [6] [7]. The software which we are testing will be checked against crashes or information leaks. We try to generate a random set of inputs and check on the inputs which are not generated by another set of testing techniques.

Working process of fuzzing [5] : Figure 2 delineates the most forms of conventional fuzzing tests. The working prepare is composed of four fundamental stages,the testcase generation stage, testcase running stage, program execution state monitoring and analysis of exceptions.

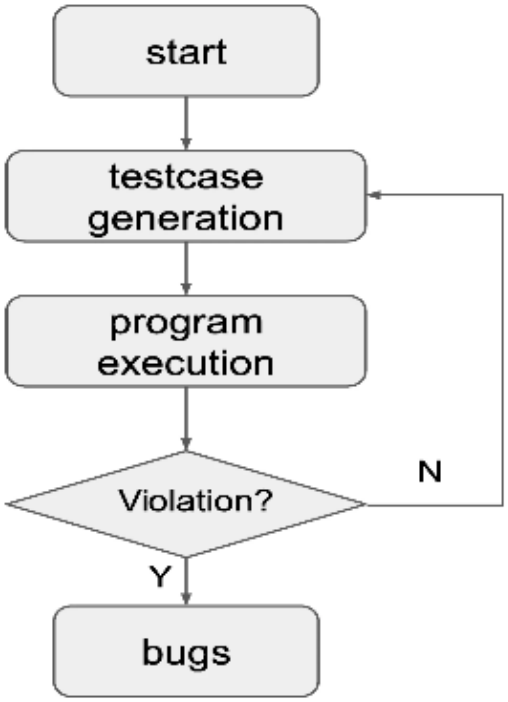

Fig. 2. Working Process of Fuzzing Tests

A fuzzing test begins from the era of a bunch of program inputs, i.e., test cases. The quality of produced test cases specifically impacts the test impacts. The inputs ought to meet the necessity of tried programs for the input arrangement as distant as conceivable. Whereas on the other hand, the inputs ought to be broken sufficiently so that preparing for these inputs would be exceptionally likely to fall flat the program. Concurring to the target programs, inputs may be recorded with diverse record groups, arrange communication information, executable parallels with indicated characteristics, etc. How to create broken sufficient test cases may be a primary challenge for fuzzers.

So, in this project, the fuzzer that's WebFuzz takes the language structure extricated by the miner and makes the input based on the detail characterized by the Grammar Miner. It utilizes the derivation tree and spreads it step by step. The children at each hub of the derivation tree are obtained by utilizing any of the production rules in the grammar. Fuzzers are used heavily to find security vulnerabilities, memory

errors (SIGSEGV fault), integer overflow errors, and SIGABRT errors (divide by zero). Fuzzing has become hugely significant in recent years because of its contributions to the software security discipline. The literature review focuses on providing a great insight into fuzzing tools.

3 Literature Review

Fuzz testing is basically a technique that is used for software testing. The main idea of fuzzing is that we provide invalid or undesired inputs for the purpose of causing the error[4] [6] [7]. The software which we are testing will be checked against crashes or information leaks. We try to generate a random set of inputs and check on the inputs which are not generated by another set of testing techniques. Fuzz testing can be used for the security testing of web applications. Fuzz testing for web applications can be done by sending a set of HTTP demands to the internet application under test. The tester must arrange how input will be generated. After the HTTP request has been sent to the net application beneath the test, the other step is to see if there is any vulnerability present and create testing reports after all these steps[7].In this project, we will develop a grammar-based fuzzer to generate input data for our web application and detect security vulnerabilities like SQL injection, Cross-site-scripting (XSS) attacks.

There are two main methods for fuzzing tools to generate test inputs mutation and generation [10]. Mutation-based works by arbitrarily mutating well-formed input files or using other formerly well-defined mutation approach based on the target program's information that was collected during execution. On the other hand, the generation method generates new test inputs from a grammar rule or a specification. Moreover, test inputs in fuzzing are usually valid that satisfy the specification to pass the parsing stage and invalid to discover vulnerabilities deeply in the target program [7] [10].

For a generation-based fuzzer, knowledge of program input is required. For file format fuzzing, usually, a configuration file that predefines the file format is provided. Test Cases are generated according to the configuration file. With given file format knowledge, test cases generated by generation-based fuzzers are able to pass the validation of programs more easily and could be more likely to test the deeper code of target programs. However, without a friendly document, analyzing the file format is tough work. Thus mutation-based fuzzers are easier to start and more applicable, and widely used by state-of-the-art fuzzers. For mutation-based fuzzers, a set of valid initial inputs are required. Test Cases are generated through the mutation of initial inputs and test cases generated during the fuzzing process[5].

3.1 Mutation based Fuzzers

Mutation-based fuzzers generate different combinations from a given valid input. It is simple to set up and isn't subordinate to program points of interest. The thought behind the mutation technique is to gather information like collect data like network packets, texts, files, etc. then alter or control them arbitrarily or based on a few techniques [11]. A straightforward case of the procedures is supplanting expansive content with littler content or changing the length esteem with a littler or bigger esteem or essentially changing a number with a bigger or a littler number [7]. This method generates test inputs by mutating well-formed seed files randomly or using predefined mutation strategies which can be adjusted based on target-program-oriented information gathered during runtime.

A big advantage of the mutation method is that it needs little or no knowledge about a target program [7]. All that is needed is one or more good samples and a method of fuzzing a target application. However, a major drawback of these types of fuzzers is that most of the inputs generated by them are syntactically invalid and thus are rejected by the processing program.

3.2 Generation based fuzzers

Generation-based fuzzers are also called grammar-based fuzzers. It generates inputs from a specification (e.g., grammar). Like an input from scratch rather than using an initial input and mutating. Here grammars act as the specification of the legal inputs to the program. These fuzzers are complicated to create and we require a distinctive linguistic use for each program. But they have higher accuracy than mutation-based fuzzers. Generation-based fuzzing uses the input information or the vulnerabilities for creating test information. As compared to immaculate random-based fuzzing or mutation-based fuzzing, generation-based fuzzing ordinarily accomplishes a better scope of the program beneath the test, in specific, in case the anticipated input organize is or may be complex. Figure 3 represents the table comparison of Generation based and Mutation based Fuzzers[5].

Technique Name	Easy to start ?	Priori knowledge	Coverage	Ability to pass validation
Generation based	hard	needed, hard to acquire	high	strong
Mutation based	easy	not needed	low, affected by initial inputs	weak

Fig. 3. Comparison of Generation based and Mutation Based Fuzzers

With regard to the reliance on program source code and the degree of program examination, fuzzers may be classified as white box, grey box, and a black box. White box fuzzers are accepted to have got to the source code of programs, and in this way, more data might be collected through investigation on source code and how test cases influence the program running state. Black box fuzzers do fuzzing tests without any knowledge on target program internals. Grey box fuzzers work without source code, either, and gain the internal information of target programs through program analysis[5].

3.3 HTML Parser

The Parser takes the HTML URL as an input and gives a lexicon of all the input areas displayed within the web page as yield. Parsing ordinarily, in common, applies to any computer programming dialect and it is the method of taking the code as content and creating a structure in memory that the computer gadget can recognize and work with. Especially for HTML, HTML parsing is the procedure of taking raw HTML code, reading it, and creating a Document Object Model (DOM) tree object structure from it.

3.4 Grammar-based fuzzers

A compiler has a parser that checks whether an input string is a syntactically valid program or not by checking the grammar of the input to the program. For example, by checking the syntax of C/C++, Python, Java, etc. program language which is defined via context-free grammar. To catch SQL Injection, Cross-site-scripting (XSS) attacks, and regular expressions are used. Assume the program beneath the test has an input validator (parser) or sanitiser like web application and haphazardly transformed input is nourished to this program. Chances are that arbitrarily generated/mutated input will be rejected by the parser.So, to test the program with more noteworthy effectiveness, one needs the control of arbitrary fuzzing but one ought to make beyond any doubt that produced input is well-formed. Program examination fuzzers and grammar-based fuzzers can effectively do this work but program investigation fuzzers are a costly strategy subsequently grammar-based fuzzers are broadly utilized.

Grammar-based methods as a rule take a shorter time and fewer assets to create fuzzing records compared to typical execution. But, the downside of grammar-based procedures is that they generally depend on test input records. Moreover, the quality of test inputs is the central point for any fuzzing apparatuses which impacts the viability and productivity of the device.

3.5 Grammar Miner

The miner takes the set of input fields provided by the parser and creates a grammar out of it. The primary task for the miner is to define a rule for each HTML input type.

3.6 SQL Injection Attacks

SQL Injection attacks are one of the most frequent threats to web-based apps. SQL injection occurs when untrusted user-supplied data is entered into a web application and that data is then used to dynamically create a SQL query to be executed by the database server [1]. SQL injection mostly occurs when instead of providing the required input like user id or username, the hacker provides an SQL statement that the database will run and execute unknowingly which will lead to trouble[13]. SQL injection can be classified into various levels concurring to whether it can react to time or whether there are mistakes. Amid the test of SQL injection, different apparatuses will set various payloads for various levels of SQL injection. Individuals optimize the payloads through the strategy of change to urge extra accessible test cases. [8]. Sometimes there may be a case when a SQL injection attack is successful and through this the hacker can read sensitive data and modify the data thus causing harm to the web application.

3.7 Cross-site-scripting (XSS) attacks

Cross-site scripting(XSS) is additionally a genuine issue of web applications that can be utilized by an attacker.The attacker can embed the pernicious script in a web application through any outside asset. [1] .The effect of such attacks is becoming more and more common. The attacker introduces javascript statements into the page links or forms of the website or web application to alter the content of the page and extract information such as the user's cookies. Cross-site scripting usually includes three types: storage-XSS, DOM-XSS and reflex-XSS [8]. When fuzzing vulnerability, apart from just using the most frequently used javascript payloads, people try to continuously enhance attack methods and attack statements to obtain higher detection efficiency, for instance generating the payloads by applying genetic algorithms or by using different methods to improve detection efficiency.

An attacker can use cross-site-scripting attacks to dispatch a malicious script to an unsuspecting user. The web application of the end-user has no way to know that the malicious script cannot be trusted, and will run or execute the script as the end-user thinks that the script came from a trusted source, the script can access any session tokens, cookies, or any other sensitive information retained by the browser and used with that site Furthermore, in some cases these scripts can even rewrite the content of the HTML page [14].

4 Problem Statement

Vulnerabilities have ended up the root cause of dangers to internet security. A vulnerability is a flaw or weakness in a system's design, implementation, or operation and management that violates the system's security policy. Attack on vulnerabilities, especially on zero-day vulnerabilities, can result in serious damage to the system [5]. Considering the genuine harms caused by vulnerabilities, much effort has been given to vulnerability disclosure strategies for computer programs and data frameworks. Methods counting static analysis, dynamic analysis, symbolic execution, and fuzzing are proposed. Dynamic Analysis can completely recognize the bugs by looking at the runtime conduct of the program, but it includes a moderate execution speed and is non-scalable. Actually, Symbolic Execution can cover any execution way inside the program, but it gives great results because it was for small programs and is undoubtedly less flexible since it perseveres from the impact issue. For expansive scale programs, the execution states increase basically which outperforms the understanding capacity of the limitation solvers.

Compared with other methods, fuzzing requires little information of targets and might be effortlessly scaled up to huge applications, and hence has become the foremost well-known vulnerability disclosure solution, particularly within the industry.[5].

Fuzz testing is basically a technique that is used for software testing. The main idea of fuzzing is that we provide invalid or undesired inputs for the purpose of causing the error[4] [6] [7]. The software which we are testing will be checked against crashes or information leaks. We try to generate a random set of inputs and check on the inputs which are not generated by another set of testing techniques.

In this project, we have created a grammar-based web fuzzer that creates the input information for our web application and distinguishes different security vulnerabilities like SQL injection, Cross-site-scripting (XSS) attacks. Identifying these vulnerabilities and settling them previously will offer assistance to us thus preventing a modern cyber-attack that can take delicate user data or eventually crash the site, driving to an awful trade reputation.

5 Related Works

As of now, there is a number of fuzzers that are utilized to discover the security vulnerabilities within web applications but most of them are not grammar based. Grammar-based fuzzing takes shorter time and fewer assets to produce fuzzing records. In any case, a few confinements incorporate being incapable to handle a few unstructured input records like pictures, not producing redress grammatically input detail, cannot work with undefined grammar specifications for inputs, and being limited to specific file formats [7].

Our thought is novel as we have characterized grammar on top of a fuzzy i.e a language structure is characterized and mined. Unused input values are created and are given to the web fuzzer to test it for diverse security vulnerabilities. Moreover, most of the grammar-based fuzzers actualizes the database control portion of the SQL injection while our Webfuzzer implements a modern thought that's the login bypass.

We have created a grammar-based web fuzzer that creates the input information for our web application and distinguishes different security vulnerabilities like SQL injection, Cross-site-scripting (XSS) attacks. Identifying these vulnerabilities and settling them previously will offer assistance to us thus preventing a modern cyber-attack that can take delicate user data or eventually crash the site, driving to an awful trade reputation.

6 Implementation

The essential thing to implement for the project was to find out a web application that is vulnerable to distinctive security threats like the chance of SQL injection or cross-site scripting attacks. After looking into various web applications all over the internet, the group found a fitting site that can be revealed to SQL injection attacks and cross-site scripting assaults.

The project was created utilizing python programming language and to begin off with the extension, the project was partitioned into two parts that are Execution of SQL attacks and the Implementation of XSS attacks

Firstly, a class named WebFuzzer was developed in which the first step was to parse each and every field of the form present in the web application. We used the combination of HTML parser and the beautiful soup libraries defined in python. Beautiful Soup is a python library that is often used to extract the data out of HTML and XML files. It works with the collaboration of HTML or xml parses which further helps by providing different methods to search or modify the soup object.

The HTML parser is a module in python which serves as the basis for parsing text files formatted in HTML. The output given the beautiful soup is treated as an input for the WebParser method which further gives out the parsed output. The WebParser will give the action and the fields for the html form found in the web application. After that a base grammar file is created in which a base grammar is defined for both the sql injection as well as the cross-site scripting attacks.

Further, a class named GrammarMiner is created to add new rules or mine the grammar on top of the pre-defined grammar. This is done by adding using the dictionary of actions and fields that were extracted using the HTML parser. This class takes in a parameter that was given in the command line which checks whether the type of attack is SQL or XSS. The HTML fields are looped over and depending on the type of fields in the form and the type of the attack, new grammar rules are appended or mined on top of the existing grammar, and further a file named “GeneratedGrammar.py” is created which includes the updated grammar.

The mined grammar is then passed on to an existing gramfuzz which is a grammar-based fuzzer. The file “GeneratedGrammar.py” is loaded to the gramfuzz and is executed. A method called gen is used to create different URLs that will be passed to the run method for the SQL and xss attacks. There an array of URLs is returned by this component. Finally, when the grammar is ready and the urls are created, it is time to show the working of the SQL injection and xss attacks. The attacks are separated by a command line argument called “attacks” and it is implemented under a method run.

6.1 High Level Diagram

In figure no. 4, the architecture of the WebFuzzer is explained below. There are different components in WebFuzzer and the role of each component is defined as:

• *HTML Parser:* The HTML parser takes in the URL of the targeted web application as input, processes it and returns a dictionary of all the input fields present in the web page as output.

• *Grammar Miner* The grammar miner takes in the dictionary of input fields that were returned by the HTML parser as input and creates and mines a grammar out of it. The main task of a grammar miner is to define a rule for each HTML input type.

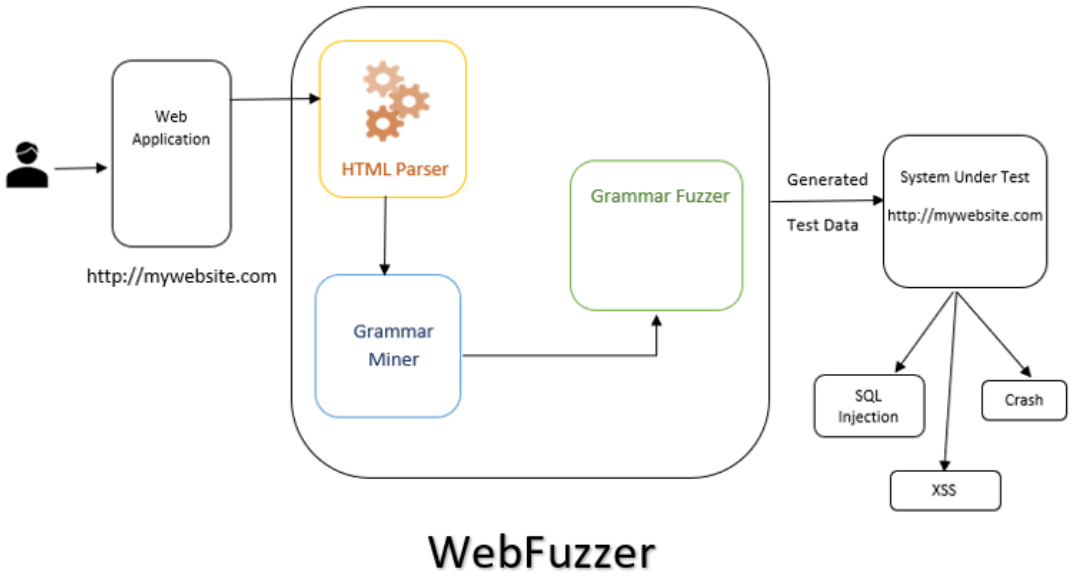

Fig. 4. Overview of WebFuzzer

• *Grammar Fuzzer* The Fuzzer takes the grammar extracted by the grammar miner and generates the input based on the specification defined by the Grammar Miner. It makes use of a derivation tree and expands it step by step. The children at each node are obtained by applying any of the production rules in the grammar. The data that is automatically created by the grammar fuzzer is then sent to the test system which we call a web application and then it is monitored. Thus based on the input of the grammar fuzzer we are able to identify different vulnerabilities in the system. These are part of our calculations. Thus are used to distinguish the SQL injection attacks.

6.2 Working of SQL Injection Attack

Command used to run the SQL injection is:

• python WebFuzzer.py –url=URL –method=METHOD –attack=ATTACK_TYPE –filtercode=CODE

This command is run on the command line and it contain many command line arguments which are defined as follows:

1. –url: it determines the url of the web application.
2. –method: it defines the method i.e GET or POST.
3. –attack: it defines whether the attack is sql or xss.
4. –filtercode: it filters the response code of the response given by the webFuzzer.

A method called “sqlInjectionAttack” is defined for executing the SQL injection. The basic steps of execution are:

The URL generated by the grammar fuzzer is looped one by one. For each URL, a curl is prepared. The request parameters are created using a split method which is further stored in an array. Depending on the type of the request i.e. GET or POST, the request body is created.

For POST requests, the operations that are set are URL, POST, POSTFIELDS. For GET requests, the set opts are URL and HTTPGET. The webfuzzer is executed by using the method perform.

The response code is monitored and if it returns "200", it means the login is bypassed and the SQL injection is done.

6.3 Working of Cross-site Scripting Attacks:

The command for executing the xss attack is:

- python WebFuzzer.py –url=URL –method=METHOD –attack=ATTACK_TYPE

This command is run on the command line and the command line arguments defined in the command are as follows:

1. –url: it determines the URL of the web application.
2. –method: it defines the method i.e GET or POST.
3. –method: it defines the method i.e GET or POST.
4. –attack: it defines whether the attack is SQL or XSS.

A method called "xssAttack" is defined for executing the cross-site scripting attack. All the steps involved in the execution are as below:

1. In the first step, the URLs which are mined, appended, and returned by the grammar fuzzer are looped one at a time.
2. For each URL, a request curl is prepared similar to the one in the SQL injection.
3. Request parameters are created by using python's string method and are stored in an array.
4. Depending on the type of the request i.e. GET or POST, the request body is created.
5. For POST request, the operations that are set are URL, POST, POSTFIELDS.
6. For GET request, the set opts are URL and HTTPGET.
7. The webfuzzer is executed by using the method perform.

Unlike the SQL injection where the response was checked in order to confirm the attack, xss method uses the response's info and compares whether the javascript data that was injected in the request is present in the response or not. If the data matches, it means that the xss attack is done otherwise it is not.

7 Results

After discussing the implementation details of the project. The discussed attacks were performed on the dummy vulnerable website. As described in the implementation we have performed the SQL injection attack and XSS attack and found that the website is vulnerable to these attacks. It can be inferred from the results that testing with the fuzzing technique gave us success as we were able to find out the failures against the attacks on the vulnerable websites.

7.1 Environmental Setup

Our team executed the WebFuzzer on a machine with the specifications as machine company was Lenovo, The processor was Intel(R) Core(TM) i5 CPU having RAM of 8GB on Windows 10 with the 64-bit operating system, x64-based processor using language Python and Visual Studio as IDE.

7.2 Running of the SQL Injection

Command used to run the SQL injection is:

- python WebFuzzer.py –url=URL –method=METHOD –attack=ATTACK_TYPE –filtercode=CODE

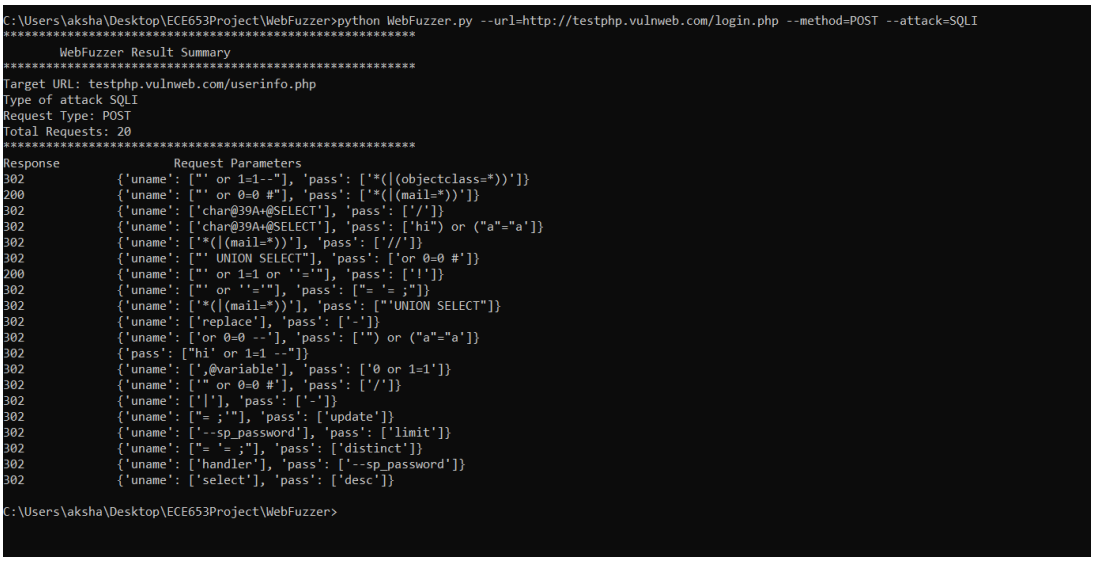

Fig. 5. Sql injection Attack

On running the WebFuzzer, the given URL within the command is focused on and all the areas on the frame of that web application are extricated and passed on to the language structure GrammarMiner for making and upgrading the grammar. The grammar returns the URL and makes the values of the areas of the form.

The WebFuzzer tries to create a log-in endeavor utilizing the values produced by the language structure. By default, 20 URLs are made for each execution. The reaction code decides whether the endeavor made by the WebFuzzer to bypass the login page is fruitful or not. As seen within the figure, the status code 302 characterizes that the WebFuzzer is incapable to enter the net application with a certain input created by the language structure. While the status code 200 delineates the input values for the areas that are dependable for bypassing the login page and building up a SQL injection attack.

Sql Injection attack with filter command

The command used for this case of SQL injection is as:

• python WebFuzzer.py –url=http://testphp.vulnweb.com/login.php –method=POST –attack=SQLI –filtercode=302

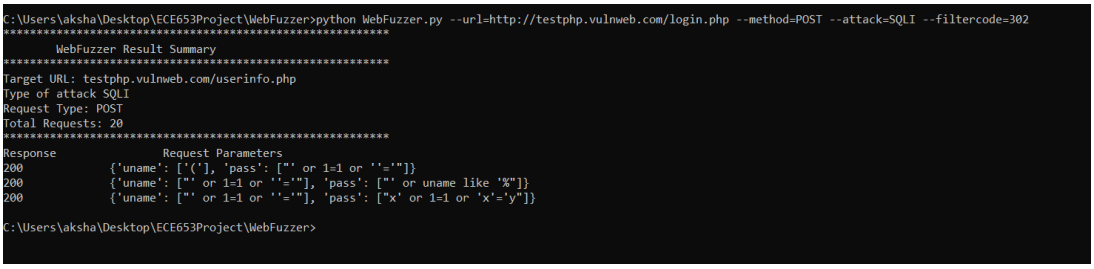

Fig. 6. Sql injection Attack with filter code

The argument –filtercode = 302 filters out the unsuccessful results of the WebFuzzer and displays only those input values for which the SQL injection is made i.e the ones that give response code 200.

7.3 Running of XSS attack

XSS attack was executed by the command as:

- python WebFuzzer.py –url=http://testphp.vulnweb.com/search.php –method=POST –attack=XSS

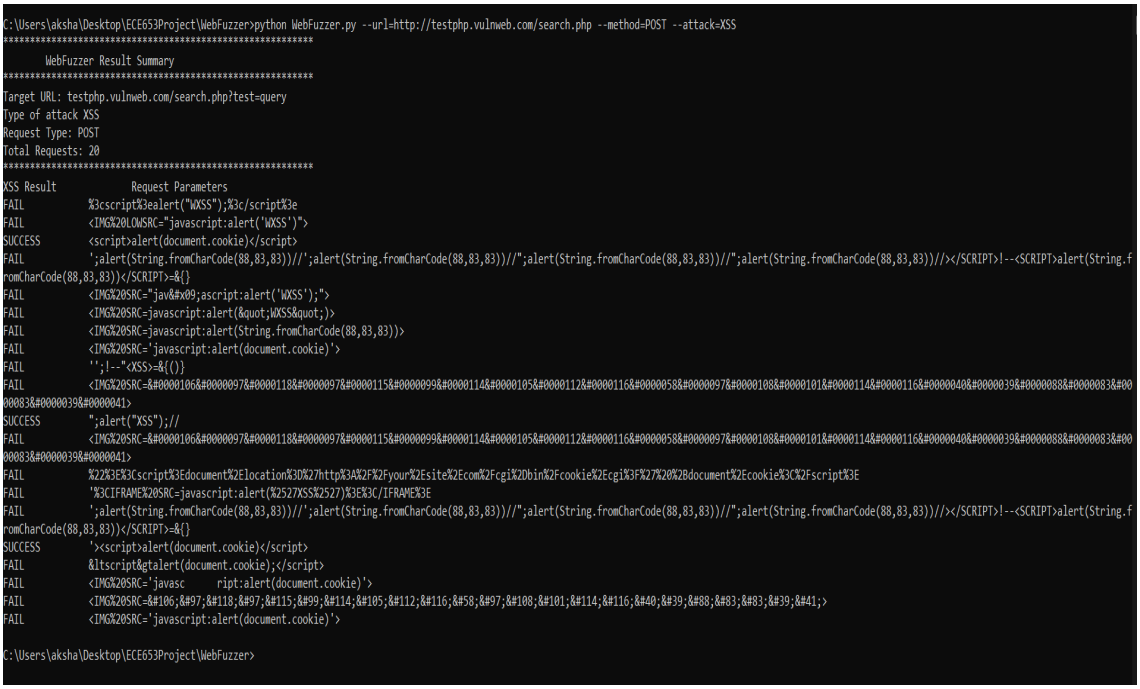

Fig. 7. Cross Site Scripting Attack (XSS attack)

On running the WebFuzzer for cross-site scripting assault, the URL specified within the command in focused on. Comparative to the SQL injection, firstly the areas of the web application are extricated by the WebParser and are bolstered as to the language structure mineworker to include the grammar on the beat of the existing one. By default, 20 URLs are produced by the gramfuzz.

The js values are produced by the GrammarMiner. The request is made to the web application attempting to inject the javascript code into the code of the internet application. The victory of this attack is obtained by comparing the produced values and the values from the reaction of that request. In the event that the reaction contains the created values, it means that the XSS attack was effective in something else, it demonstrates that it may be a disappointment. The figure above shows the victory and disappointment of specific javascript injections.

8 Conclusion

The software Testing is important because if there are any bugs or errors in the software, it can be identified early and can be solved before delivery of the software product. Properly tested software product ensures reliability, security, and high performance which further results in time-saving, cost-effectiveness and customer satisfaction. Fuzzing is the validation technique used in testing practices. Therefore, by testing with fuzzing technique gave us successful results as we were able to find out the failures against the attacks on the vulnerable websites. This idea can be extended further by using different kinds of attacks and testing on the website whether a particular website is vulnerable to the attacks or not. We can also use a different type of grammar used in the fuzzing and with all these techniques, a particular website can be tested against the attacks. It is a novel idea to test and detect these kinds of failures with the help of a fuzzing mechanism to improve the security and the quality of the code.

References

1. Author, Sandeep Kumar : A Study on Web Application Security and Detecting Security Vulnerabilities, https://www.researchgate.net/publication/324935252_A_study_on_web_application_security_and_detecting_security_ vulnerabilities

2. Author, Abdalla Wasef Marashdih : Cross Site Scripting: Detection Approaches in Web Application, https://pdfs.semanticscholar.org/8a72/06d9411fd81506fba7f36079c6e2579e62ba.pdf

3. Author, Mohd. Ehmer Khan : CA Comparative Study of White Box, Black Box and Grey Box Testing Techniques , https://www.researchgate.net/publication/270554162_A_Comparative_Study_of_White_Box_Black_Box _and_Grey_Box_Testing_Techniques

4. Author, WANG Tao : Research and Application of a New Fuzz-test Framework, https://www.sciencedirect.com/science/article/pii/S1876610212002676

5. Author, Jun Li : Fuzzing: a survey, https://cybersecurity.springeropen.com/articles/10.1186/s42400-018-0002-yTab3

6. Author, Ivan Andrianto : Web Application Fuzz Testing, https://www.researchgate.net/publication/323135624_Web_application_fuzz_testing

7. Author, Hamad Al Salem: A Review on Grammar-Based Fuzzing Techniques, https://www.cscjournals.org/manuscript/Journals/IJCSS/Volume13/Issue3/IJCSS-1481.pdf

8. Author, Danyang Zhao : Fuzzing Technique in Web Applications and Beyond, https://iopscience.iop.org/article/10.1088/1742-6596/1678/1/012109/pdf

9. Author, IEEE : Violating assumptions with fuzzing, https://ieeexplore.ieee.org/document/1423963

10. Author, IEEE : Fuzzing: State of the Art, https://ieeexplore.ieee.org/document/8371326

11. Author, Charlie Miller : Analysis of Mutation and Generation-Based Fuzzing, https://www.ise.io/wp-content/uploads/2018/04/analysisfuzzing.pdf

12. Author, Hongliang Liang : Fuzzing: State of the Art, https://wcventure.github.io/FuzzingPaper/Paper/TRel18_ Fuzzing.pdf

13. Author, : Stephanie Reetz : SQL Injection, https://www.cisecurity.org/wp-content/uploads/2017/05/SQL-Injection-White-Paper2.pdf

14. Cross site scripting (XSS), https://owasp.org/www-community/attacks/xss/

15. Hamilton, T.: What is white box testing? techniques, example amp; types, https://www.guru99.com/white-box-testing.html

16. Hamilton, T.: What is black box testing? techniques, example amp; types, https://www.guru99.com/black-box-testing.html.

17. Hamilton, T.: What is grey box testing? techniques, example, https://www.guru99.com/grey-box-testing.html.