Journal of Computer Science and Engineering Research and Development (JCSERD)

ISSN 2248-9304 (Print)
ISSN 2248-9312 (Online)
Volume 3, Number 1, January - December (2015), pp. 01-05

JCSERD

© PRJ PUBLICATION

IMPLEMENTING AN AUTOMATIC TEST CASE GENERATION FOR DHCP PROTOCOL

Chandu P.M.S.S
Research Scholar, Department of CSE, Sathyabama University, Chennai, India

Dr. T. Sasikala
Principal, SRR Engineering College, Chennai, India

ABSTRACT

Software testing is taking place while implementing new software. There are different types of testing techniques available in software engineering. In network protocols like Dynamic Host Configuration Protocol, testing a data packet is not easy. Random Based Feedback Directed Approach and Rule Based Specification concept can be used to test a DHCP based software applications effectively. A group of test cases are generated using Rule Based Specifications method. Next, a Random Based Feedback Directed Approach is using these generated test cases for testing the application's methods in random order also which produce set of inputs then these inputs also used to test the remain methods. This approach ensures better code coverage.

Keywords: Rule Based, Random Testing, Automatic Test Case Generation, DHCP Protocol

1. INTRODUCTION

The Dynamic Host Configuration Protocol is a standardized network protocol used on Internet Protocol networks for dynamically distributing network configuration parameters, such as IP addresses for interfaces and services [7]. Several different implementations are available for DHCP protocol. So it is important to make sure that the protocol implementation conforms to its specifications. This calls for testing the protocol implementation for conformance and interoperability. Various testing methods are in practice including manual testing, random testing, and symbolic execution. In these testing methods, creation of test sequences i.e. pair of input and output packets plays a key role. These inputs can be applied to the implementation and the output should be monitored. The DHCP servers act as agents for network administrators and automate the process of network address allocation and parameter configuration [5]. But with existing approaches such as random testing the code coverage is low. With symbolic execution it is infeasible to make the entire input packet as symbolic. For larger codes it is inappropriate.

2. LITERATURE SURVEY

System models can be used in the generation of test cases [2].this can be automated or manual generation. A system model is developed based on the implicit knowledge of the tester. But it does not guarantee uniformity in the generation of test suites. An automated tool called TaRGeT is used in model based tests. Separate manual tests are carried out and compared against manual tests. Manual testing gives more freedom for the people involved in testing because it is easy to test unpredictable test scenarios. But for use cases which are of complex nature Model Based Testing (MBT) is effective. It also evaluates system documentation properly. The automatic test case generation can cover all possible scenarios in a precise manner.

Fuzzy testing [3] is a type of testing in which a random data or malicious data is given as input to the software implementation. This testing method involves the generation of test cases which are semi valid in order to bypass certain checks and verification .In model based fuzz testing a model of the system is used in input generation. The grammar driven approach for fuzz testing starts with building a model of the system which is followed by syntax tree construction. Then the syntax tree nodes are traversed and modified. This results in test case generation. It is efficient in reducing redundant and invalid test cases.

In the early development stage of a protocol high code coverage and automatic generation of test cases are the core challenges for testing [4] .This leads to insufficient manual testing. The common issues in protocol testing are low code coverage, Automatic test case generation missing code etc. Symnet (Symbolic Network) is a testing environment for protocol implementation. It tests the protocol instances in separate operating system. Execution control is in a central unit of a network.

A protocol will have different implementations. Test sequences are generated from protocol specification. The aim of testing is to verify whether the test sequences conform to the specification. Four different formal methods (T, U, D, W) are used for developing test sequences[6]. These methods except T method use a characterizing entity for generating test cases. This method make use of finite state model (FSM).This approach has better fault detection capabilities.

Testing is important in assessing the quality of software. Protocol implementations will have lot of vulnerabilities, interoperability issues and conformance issues. Rules can be derived manually from the specification of the protocols [4]. Symbolic execution will execute the set of input test cases and find the variation from the rules if any. It is difficult to follow sequential execution for larger codes. Instead of supplying the normal inputs to a program one supplies symbols representing arbitrary values. The execution proceeds as in a normal execution except that values may be symbolic formulas over the input symbols [1].

3. PROPOSED METHOD

Rule based feedback driven random testing can be applied for testing a protocol implementation like DHCP implementation in a client-server environment. The main idea of this approach is to generate a set of test packets as input and execute them to identify the potential flaws in the specific implementation. Here execution of both single as well as multi-packets are explored. Any host machine can become a server if appropriate information is configured on to the system. In such a client-server environment packet will be sent to and from client. Such an application with a protocol implementation can be tested effectively by creating a pool of test cases.

Set of rules are derived from the standard protocol specification. The implementation should not violate the rules specified manually. From the test packets pool, certain packets have been selected randomly and executed. When the execution generates output packets, it is compared against input packets to check the violation in expected behavior. Feedback is obtained from each execution which in turn used to modify the test suite.

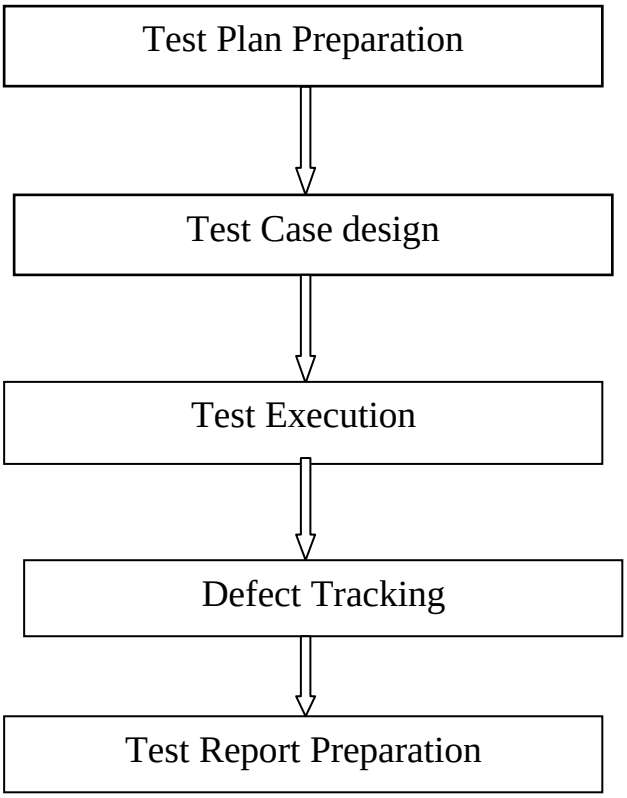

Figure 2.2: Testing process

Testing process is repeated with the aim of maximum code coverage. It also provides deep exploration of the code. This is an effective way for checking the compliance of the particular protocol implementation with its specification and interoperability of multiple implementations. The design is simple but efficient enough to discover various flaws and bugs since it validate the protocol implementation against its specification.

The testing process is depicted in figure 2.2.The process starts with a test plan preparation. It is followed by test case design. The tester should aware of carious situation that may arise in the execution of the software. After developing valid test cases, execution of test cases will be carried out. Based on the result of execution, various defects in the software is found out and corresponding test report is prepared.

4. EXPERIMENTAL ANALYSIS

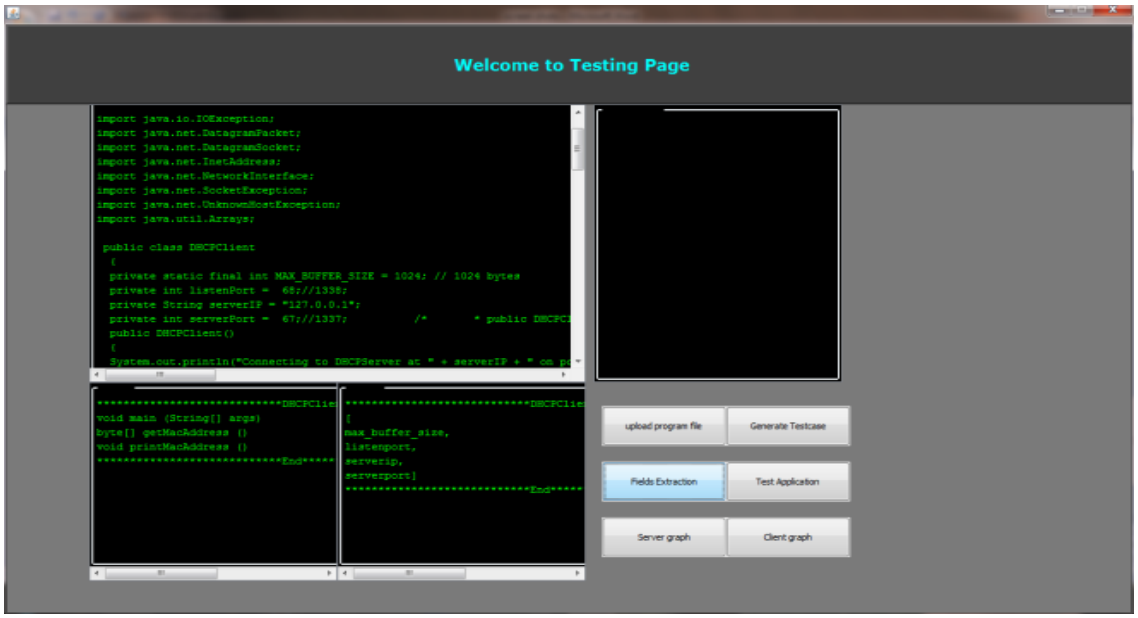

Fig. 4.1 Field extraction

A group of test case is generated by using Rule Based Specifications approach. A set of rules are generated manually from protocol specification and it is used to create the network packets. The every packet is including some specifications such as size, set of attributes and their values.

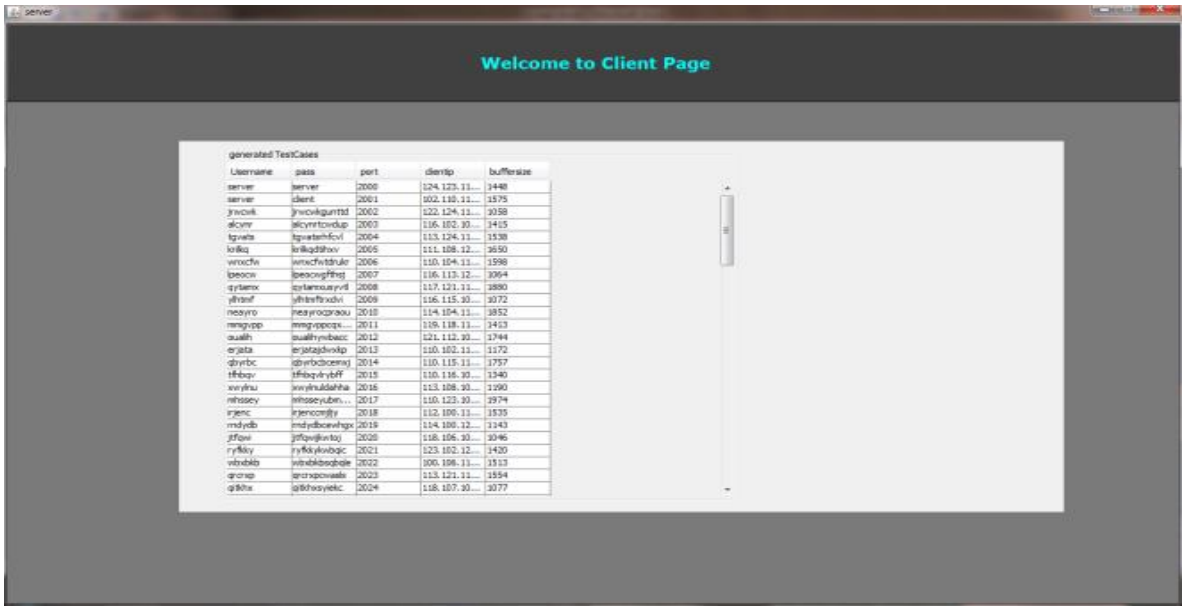

Fig 4.2 Generated test cases

In this approach, Test cases are given as input to the function and get the output. Test cases for execution are selected on a random basis. The generated output is again given as an input to the next function by using iterative method.

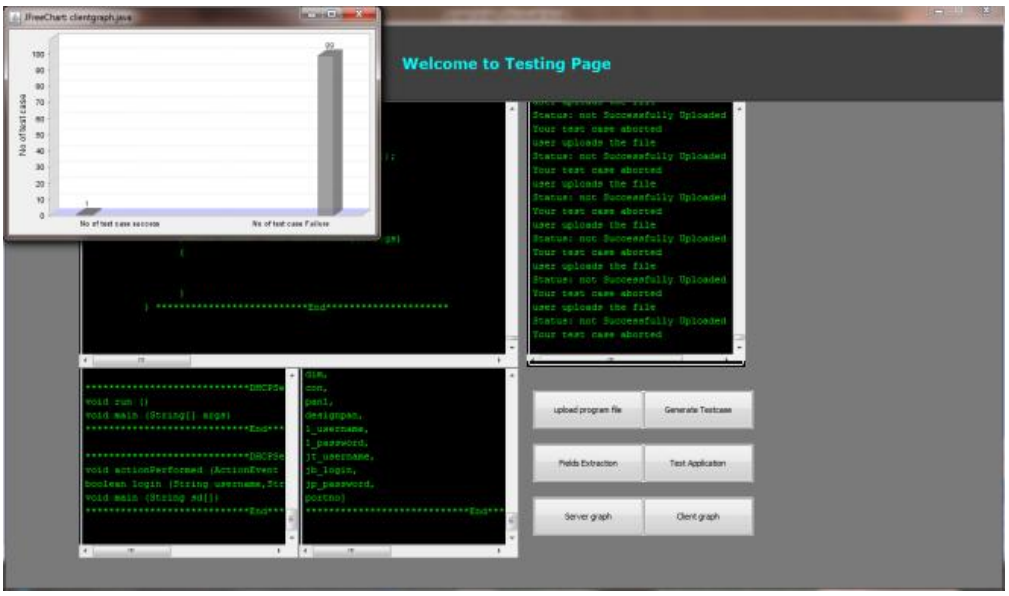

Fig 4.3 Client side graph

Fig 4.3 shows the client side test case graph. It shows the number of test case success and failure.

5. CONCLUSION

The DHCP protocol implementation is tested for its conformance with the specification. Automatic test case generation is done from rule based specification. Test case generation follows a random behavior. This method is suitable for testing DHCP implementations effectively. The same approach with certain modifications can be used to verify other communication protocols.

REFERENCES

1. D. Vivek, V. Venkatesa Kumar, 'Application of Symbolic Execution, Rule Based Learning and Explicit State Model Checking For Optimizing Protocol Conformance Testing', IJARTET, Vol. II, Issue I, March 2015
2. Arthur Marques, Franklin Ramalho, Wilkerson L. Andrade, 'Comparing Model-Based Testing with Traditional Testing Strategies: An Empirical Study', 2014 IEEE Seventh International Conference on Software Testing, pp: 264-273 March 2014.
3. Efficient Model-based Fuzz Testing Using Higher-order Attribute Grammars Fan Pan, Ying Hou, Zheng Hong,Lifa Wu, Haiguang Lai, Journal Of Software, Vol. 8, No. 3, March 2013.
4. Raimondas Sasnauskas, Philipp Kaiser, Russ Lucas Jukic, Klaus Wehrle, "Integration testing of protocol implementations using symbolic distributed execution", ICNP, 2012, 20th IEEE International Conference on Network Protocols (ICNP), pp. 1-6, 2012.
5. Droms, R., "Automated configuration of TCP/IP with DHCP," Internet Computing, IEEE, vol.3, no.4, pp.45, 53, Jul/Aug 1999.
6. Deepinder P Sidhu,Ting kau leung, ' Formal methods of protocol testing:a detailed study', IEEE transactions on software engineering,vol 15,no. 4 april 1989.
7. https://en.wikipedia.org/wiki/Dynamic_Host_Configuration_Protocol.
8. P.Vigneshwaran and Dr. R. Dhanasekaran, "A Novel Protocol To Improve TCP Performance – Proposal" International journal of Computer Engineering & Technology (IJCET), Volume 3, Issue 2, 2012, pp. 372 - 377, ISSN Print: 0976 – 6367, ISSN Online: 0976 – 6375.