Security Testing of Network Protocol Implementation

Pradnya Kawade[1]

[1]*Student, Dept. of IT Engineering, K. J. Somaiya College of Engineering, Mumbai Maharashtra 400077, India;*

---***---

Abstract – *Inaccurate Network protocol implementation can bring various consequences. Therefore, security testing of Network protocol implementation is a hot topic for research in information security. Design and implementation of secure network protocol is very important nowadays. Any security flaw in network protocol implementation leads to making the whole network vulnerable. This paper includes security testing methods of implemented network protocol. We use network scanning, fuzzing for verification and exploration of network protocol. To check suitability of network protocol we use ESBMC, Map2Check and KLEE as software verifiers. Paper proposes a new FuSeBMC network verification framework model to effectively detect security vulnerabilities related to network protocol implementation.*

Key Words: Network protocol, Security testing, Fuzzing, FuSeBMC, Runtime testing

1. INTRODUCTION

Implementation of Network protocol is one of the challenging task. The software bugs which were introduced during implementation of network protocol can lead to security vulnerabilities. Even a small point of flaw can make the whole network vulnerable. Thus developers need to implement strict end-to-end security to maintain the secured network. Network testing involves testing vulnerabilities in network devices, servers, DNS, TCP and FTP are hard to detect because the protocol software state-space is too large to explore. Validation of possible events such as packet access, packet loss, and timeout must have to check during protocol implementation.

The network protocol implementation need to be verified because of various reasons like, large state-space exploration of protocol implementation, finding semantic error need a machine readable specification to check whether the implementation meets specification automatically, another reason is since many bugs manifest themselves after a long period of time until then they remained hidden. Therefore, due to these problems developers needs to develop tools to identify and verify the network protocol implementation. It's very challenging because there are multiple manufacturers lead to different protocol implementation. Many errors can introduced during implementation and can be detected when service is in real use. Therefore, to reduce such errors developed by programmer which can cause many high risk vulnerabilities in network protocol, we need to develop a reliable and accurate verification method.

Fuzzing, symbolic execution, static code analysis, taint tracking are most common techniques to verify network security vulnerabilities and possible threats to the network. Here, proposed method is a combination of fuzzing and symbolic execution to determine the security vulnerability in the network protocol implementation. For symbolically verifying network protocol implementations, we use two approaches. Path exploration which is symbolic executor explores each branch separately, thereby making a copy of the current state and other is bounded model checking BMC. We also use fuzzing to produce random inputs to locate security vulnerabilities in network protocols. All though fuzzing and symbolic execution not able to go deep in protocol implementation. Fuzzing is not able to create various inputs for all paths in the network protocol implementation and symbolic execution cannot achieve high path-coverage because of the state-space explosion problem. Thus we have to use combination of both for better coverage.

This combination is used to generate automatically high-coverage test packets from the network protocol implementations. Used to detect various implementation errors. Then we used FuSeBMC framework to verify security vulnerability in network protocol implementation. Paper also proposed testing of protocol at runtime in an online way which is more complex and challenging work because tester have to undergo a large amount of nonstop traces.

2. NETWORK PROTOCOL DESIGN

Network is defined as a group of computer devices that are interconnected for sharing and exchanging the information. This sharing and exchanging of information within the network should be based on certain predefined rules and these set of rules are called as Network protocol. The network is implemented based on OSI and TCP/IP network model and each layer has different protocols.

2.1 Protocol Definition

Protocols are defined by their properties. Structure of PDU that is protocol data unit and behavioral description is the main property. For proper processing of PDU transmission should be carried out on binary data. Data should be serialized at the sender and parsed at the receiver. PDU can be divided by separating metadata and actual data needs to be transmitted that is header and payload. Header includes all information required by protocol to perform its function properly. It is divided into packet fields that can represent different in formation for protocol setting. It can include information on communication link and on payload. The payload is at the end of PDU and can present at different position of PDU.

Protocol also includes description of its behavior like how to establish and maintain a connection, determination of its different packet fields. Protocol also includes a state transition diagram in which states and transition between states can be represented. For example, UDP and TCP are transport layer protocols in which UDP is simple and stateless protocol and doesn't have any connection status established on transport layer. TCP is more complex than UDP it's a state-full and provide flow and congestion control.

2.2 Protocol goal and features

Network protocol have different properties to achieve secured communication goals:

- Flow and congestion control: network protocol can adjust the flow of sending rate according to its capacity.
- Accurate sharing of information: protocol make sure that information gets delivered without any loss.
- Duplicate filtering: packets can be duplicates during transmission. These can be automatically identified and dropped by network protocol
- Application separation: ensure that even though data transmitted through same channel, data from different application should stay separated.
- Large message support: if a sender wants to transmit large message than its limit then, protocol should split the message s in smaller ones and reassemble them at the receiver.
- Maintaining the order: protocol should make sure that data is receiving in its original order.

Protocols should be implemented according to these feature and properties at different layers of protocol stack. This stack can be used in different layers to achieve high rate adaptability. After the designing of these protocols, security testing of network protocol implementation is very important.

3. SECURITY TESTING METHODS

There are numerous of methods for network protocol testing but each possesses some limitations. There is not any technique which is reliable and accurate.

3.1 Fuzzing

Fuzzing is a black box software testing technique to exploit introduced vulnerabilities during implementation. Fuzzing uses malformed and semi-malformed inputs injection to the target network protocol. It automatically injects data into program and detect bugs. Common method for fuzzing is to define list of "known-to-be-dangerous values" and injects them over a target protocol. Protocol fuzzer send forged packets to the tested application or acts as proxy, also it modify the request and replay them. Protocol fuzzing is type of protocol abuse which is generally used to test robustness of the target or any security vulnerability like remote code execution or crashes due to any reasons. There are numerous of factors to check in network protocol implementation like structure, state, buffer or integer overflow etc., Although fuzzing is not able to create various inputs for all paths in the network protocol implementation within a reasonable time.

3.2 Symbolic Exploration

Symbolic execution is widely adopted to find security vulnerabilities in network protocol implementation. It overcomes the problem with fuzzing because it uses symbolic inputs instead of randomly generated concrete inputs. In this program memory and output values are represented as symbolic expression. For network protocol first extract the message formats from the protocol specification of the target network protocol implementation. Then, use these message formats to construct a concrete packet, which is used to mark the ID field of this packet as symbolic values to form a symbolic packet. Although, symbolic execution have main limitations like certain queries can be slow or unsolvable and symbolic execution cannot achieve high path-coverage because of the state-space explosion when analysing large problems.

4. TESTING VULNERABILITIES IN NETWORK PROTOCOL IMPLEMENTATION

Network protocol implementation is very complicated task contain high risk to be prone to vulnerabilities like buffer overflow, Denial of service, memory leak. Thus, we need a proper tool to verify these vulnerabilities at the implementation stage only to avoid extreme loss.

4.1 FuSeBMC Approach

We have seen that fuzzing and symbolic execution both are not sufficient techniques for finding vulnerabilities in network protocol implementation. Combination of both can give great results and coverage over vulnerabilities. There is not any tool exists that is developed in the field of network protocol implementation, which require dealing with packets in the network. On the other hand some tools that are available don't have a combination of these two technologies. They face problems such as path explosion or achieved low coverage. Therefore, this paper propose the approach called FuSeBMC used for detecting security vulnerabilities in network protocol implementations using fuzzing and symbolic execution. FuSeBMC uses fuzzing to generate set of test input packets and these inputs will guide the symbolic execution and BMC engine to reach to the parts at which fuzzing was unable to reach. Then, by using symbolic execution and BMC we can achieve high-code coverage and replay them against an implementation, it helps in observing potential violations of rules derived from the protocol specification.

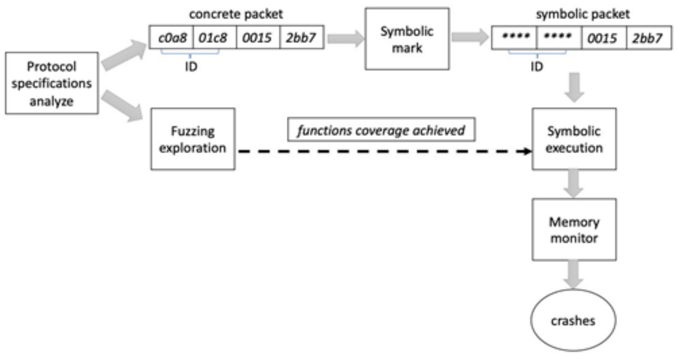

Fig -1: FeSuBMC Framework

FeSuBMC framework is illustrated in above figure 1. There are five steps for this verification framework

1. Protocol specification analyzer: produces concrete packets and that are captured by using wireshark;
2. Fuzzing exploration: In this we fuzz the software for exploring the function and then compute the coverage achieved by the fuzzer
3. Symbolic packet: marks the input packet as a symbolic packet that results in many paths. The network packet consists of multiple fields, which are part of the packet header. Therefore, FuSeBMC uses these fields as symbolic variables instead of entire input packets.
4. Symbolic execution: used to reach those function that were uncovered by fuzzer. More time can be allocated to deep functions.
5. Symbolic marker: used to convert the concrete packet to a symbolic packet by marking some bytes of the packet as symbolic values.
6. FuSeBMC prototype builds on top of Map2Check as a path-based symbolic execution engine combined with fuzzing and ESBMC as a state-of-the-art BMC engine. These tools explores all program paths of the network protocol software and produce a concrete packet while the memory monitor module reports and records the crashes.

5. EXPERIMENTAL EVALUATION

FuSeBMC approach has an ability to detect bugs and low verification time required to find security vulnerabilities compared to other methods. We examined the vulnerable FTP server which contains vulnerabilities. To test the vulnerabilities we have used ESBMC, KLEE, Map2Check tools. To compare vulnerability detection we have used Spike which is based on generational fuzzer. The main goal of this evaluation is to check the performance and reliability of these tools and requirement to be further developed to detect security vulnerabilities in network protocol implementation within FuSeBMC.

We have applied ESBMC, KLEE, and Map2Check to the vulnerable FTP server which has known buffer overflow vulnerability. Buffer overflow vulnerability found by ESBMC, KLEE and spike and not by Map2Check. Then we have compared these tools for verification time, found that ESBMC can detect the "buffer overflow" vulnerability in less than one second, while SPIKE took about 8sec and KLEE took 2sec to find that vulnerability. This states that existing methods are not reliable for secured network protocol implementation. Thus, proposed FuSeBMC approach will be efficient for security testing of all kind of network protocol vulnerabilities.

6. CONCLUSIONS

This paper proposes an approach to reduce gaps in protocol implantation. It's a hybrid technique involving fuzzing and symbolic execution may achieve better function coverage than fuzzing or symbolic execution in isolation by dealing with network packets. As a result vulnerabilities related to deep state can be identified. It includes injection of symbolic packets into the network so that one packet can generate various packets to test the target protocol, which is the advantage of this FuSeBMC approach.

REFERENCES

[1] Fu, Y. L., & Xin, X. L. "A model based security testing method for protocol implementation" Scientific World Journal 2014. https://doi.org/10.1155/2014/632154

[2] Pfrang, S., Giraud, M., Borcherding, A., Meier, D; Beyerer, J. "Design of an example network protocol for security tests targeting industrial automation systems." ICISSP 2019 - Proceedings of the 5th International Conference on Information Systems Security and Privacy, 727–738. https://doi.org/10.5220/0007704907270738

[3] Scarfone, K. A., Souppaya, M. P., Cody, A; Orebaugh, A. D. "Technical guide to information security testing and assessment." https://doi.org/10.6028/NIST.SP.800-115

[4] Ahmad, N. Habib, M. K. (2010). "Analysis of Network Security Threats and Vulnerabilities by Development & Implementation of a Security Network Monitoring Solution." www.bth.se

[5] Alshmrany, K. Cordeiro, L. "Finding Security Vulnerabilities in Network Protocol Implementations." http://arxiv.org/abs/2001.09592