Analysing IoT Protocol Implementation based on Fuzzing

Fengyu Chen. Author
School of Information Technology and Electrical Engineering
The University of Queensland, Qld., 4072, Australia

Abstract

As IoT becomes popular in many domains, security of IoT protocol has become an important research topic. IoT protocols include application-layer protocols like CoAP and MQTT, and communication-layer protocols like Bluetooth and Zigbee. These protocols may be implemented differently by various manufacturers, who may realize their own requirements in the implementations. Nonetheless, failing to follow the protocols' official specifications may introduce bugs and potential security problems. In this project, our goal is to analyze IoT protocol implementations using a fuzzing framework.

Fuzzing has shown its effectiveness in analysing binaries, OS kernels, and many other programs, but has not been widely used in analysing IoT protocols, due to some non-trivial IoT-specific challenges. Our fuzzing framework is proposed to bridge this gap. We take discrepancies and rules from RFCs as the oracles of our testing to address the ineffectiveness of crash-based oracle that current fuzzing techniques heavily rely on. We have applied our methodology to analyse several implementations of MQTT.

1 Introduction

Internet of Things (IoT) has become an essential technology in many areas. It allows various devices to connect to the network and exchange data, such that people can better control, monitor, and utilize environments and devices. IoT has been applied to agriculture [1], industrial control [2], smarthome [3, 4], wearable devices [5], and many other areas. However, as IoT becomes popular, security has become one of the main concerns for IoT deployment. In these years, IoT security incidents happen in all trades and professions. For example, Colonial Pipeline paid $4.4 million ransom to the attackers in 2021 [6, 7]. This company operates the main gasoline pipeline along the American east coast. Hackers' attacks can also target personal properties. In Black Hat USA 2015, researchers showed how they remotely hacked a Jeep and controlled its speed [8]. IoT hacking can also target national defense security. Stuxnet worm took out the key Iranian nuclear facility in 2010 [9]. This is believed to be the first worm that can spy and reprogram industrial systems [10]. From these incidents, we can see those attacks on IoT systems will cause huge influence and loss, for infrastructure, industry, and personal devices.

IoT security greatly relies on the protocols used for secure communications and operations, including applications layer protocols such as MQTT [11], CoAP [12], and AMQP [13] and communication protocols such as Bluetooth [14], Zigbee [15], and 6LoWPAN [16]. Security of IoT protocols relies on implementations in practice. However, protecting implementations of IoT protocols is a non-trivial task. One reason is that IoT protocols work in a special environment. Many IoT devices have limited resources, including power and computation capability. Some security operations like encryption require much energy, which may be simplified or removed by manufacturers. This may result in weak security protection. Another reason is that although most manufacturers operate according to official protocol specification, many of them may add their own requirements. This can be found in their implementations, which cause some security problems [17, 18, 19, 20, 21]. Compared to the rapid development of IoT, the security of IoT protocol implementations has not gotten enough attention. Systematic analysing and guaranteeing are highly desirable.

There mainly exists three methods to analyse the security of IoT protocols and their implementations. The first one is formal analysis [22]. It extracts an abstract model from protocol specification and then uses some mathematical methods to verify security properties. Formal analysis can prove the correctness and security properties of protocols, but it is hard to build the formal link between implementations and abstract models. The second method is static analysis [23]. Its main concept is automatically analysing the programs without executing them. It can locate the vulnerabilities and bugs quickly at the development stage, but it has a high percentage of false positives and ignores runtime interaction with other software components of external environment. The third method is fuzzing [24], which delivers malformed or unexpected inputs to the target program to see if failure happens. Although fuzzing can be blind and random, it can detect problems in run time with nearly no false-positive cases. Programs passing fuzzing tests are convinced to have higher robustness. With proper guidance, fuzzing can be highly efficient.

Fuzzing has shown its effectiveness in analysing binaries [25, 26], OS kernels [27, 28], and many other programs, but it has not been widely used in analysing IoT protocols. In this thesis, we aim to analyse implementations of IoT protocols based on fuzzing. We will build a fuzzing framework to analyse implementations of IoT protocols. Currently, there exist some challenges in IoT protocol analysis and fuzzing. First, many current fuzzing approaches take crash as the oracle of testing, but they may not be suitable in the IoT context. Second, during fuzzing, the state space and possible fuzzing inputs explosively grow. It is a non-trivial task to find bugs in huge state space within limited time and computation power.

2 Literature Review

Our literature review consists of three parts, corresponding to three methods of analysing the security of IoT protocols and their implementations: formal analysis, static analysis, and fuzzing. We introduce the general concepts of each method and review some works applying each method to IoT protocols analysis.

Formal analysis was first proposed to prove the security properties of protocols in cryptography [22]. It has been widely used to prove the correctness of general protocols. There exist some well-developed formal analysis tools, such as PRISM [29], Tamarin [30], ProvVerif [31]. General formal analysis consists of four steps: (1) analysing the protocol's specification, (2) constructing an abstract model based on the protocol's specification, (3) translating the abstract model into the model checker input language, (4) checking security properties and giving suggestions for standard changes.

Some researchers have used formal analysis to analyse IoT protocols. Duflot et al. [32] apply formal analysis to the device discovery phase of Bluetooth. Richard et al. [33] perform formal analysis on authentication properties of Bluetooth device pairing based on ProVerif. Kim et al. [34] formally analyse some popular IoT protocols including MQTT and CoAP. Coman et al. [35] analyse some Proof-of-Concept (PoC) attacks targeting LoRaWAN [36], Sigfox [37], and NB-IoT [38] and find some vulnerabilities in protocol specification.

Among these research, formal analysis improves IoT protocols security by proof of functional correctness and security properties in IoT protocols. This proof is strong and reliable because it depends on mathematics. Formal analysis can detect weaknesses and vulnerabilities at the protocol design stage. Finding weaknesses earlier can reduce the impact and give proper instructions to implementations. However, it also has some drawbacks. It is hard to build the formal link between implementation and abstract model because the programming language and library details cannot be fully represented by the abstract model. In addition, problems in compilation and runtime are not covered by it. Another drawback is it requires researchers to have deep professional knowledge, especially in mathematics, which can be difficult for many researchers without related background to apply this method.

Static analysis aims to automatically analyse the behaviors of computer programs without executing them. Its most common usage is source code analysis in the development stage. Many successful tools have been proposed for different programming languages, such as PC-lint Plus [39], Jlint [40], and Coverity [41].

Concerning IoT protocols security, there exist some research using static analysis to analyse the implementations of IoT protocols. Ferrara et al. [42] present an extension based on static analyzer Julia [43] and show its feasibility to detect some of the vulnerabilities mentioned in OWASP IoT Top 10 2018. Celik et al. [44] propose a static analysis system called SOTERIA. It can check properties violation in the model extracted from IoT implementations. Sachidananda et al. [45] propose a static analysis framework for IoT software.

Through analysing the source code of IoT implementations, these works try to find some security issues or properties violations. Using static analysis, detection of bugs and vulnerabilities at the development stage becomes feasible. In addition, it can locate the weaknesses in the code at the exact location. However, large amounts of false positives are inevitable in static analysis. These false positives require manual inspections, which can be a lot of effort. Static analysis usually focuses on one program, without analysing the interactions in runtime between different software. IoT is a complex system with interactions among multiparty. Ignoring these interactions leads to a restricted analysis of IoT protocols.

Fuzzing is a popular automatic testing method. It generates different inputs and delivers them to the target program to see if failure happens. There exist many general fuzzing tools that can be used to test different programs, such as American Fuzzy Lop (AFL) [46], AFLFast [47], and FairFuzz [48]. Fuzzing has many research directions, and in this work, we mainly focus on guided fuzzing and differential testing. Guided fuzzing determines how to fuzz efficiently. Differential testing gives a new oracle for fuzzing. Both are important methods used in this thesis. We will also introduce some works applying fuzzing to analyse implementations of IoT protocols.

Different from general fuzzing which takes failures like crashes as the oracle, differential testing takes discrepancies as the oracle. It generates inputs and sends them to different programs with similar functions, to see their difference in behaviours and outputs, which is called discrepancies. Here we review some impressive works using differential testing. Chen et al. [52] propose a guided differential testing method called Mucert. They convert the state exploration problem into an optimization problem. They set code coverage as an optimization goal and use the Metropolis-Hastings algorithm [53] to search for better inputs that can increase the code coverage. This approach is proved to achieve higher code coverage with fewer inputs. Tian et al. [54] propose a novel differential testing approach based on the Request for Comments (RFC). They extract rules from the RFC and generate malformed certificates according to the rules. They send these malformed certificates to various SSL/TLS implementations to see if these programs accept them. Their approach shows a better performance combined with Mucert. Petsios et al. [55] propose a differential testing framework called NEZHA. NEZHA proposes a new measure called δ -diversity to represent the path coverage among different PUTs. By keeping all inputs that can improve δ -diversity as seeds to generate new inputs, NEZHA tends to have more significant seeds that can improve fuzzing efficiency. NEZHA finds 26 times more discrepancies than Mucert on average. It also finds some vulnerabilities confirmed by developers. Currently, differential testing is mainly used to test SSL/TLS certificate verification, but it is capable of testing other programs. The discrepancy is not only a novel oracle for fuzzing, but also gives a new view for guiding fuzzing. In this thesis, we will adopt differential testing as one of our main methodologies for IoT protocol analysis.

3 Motivation and Problem Statement

As IoT become popular in many areas, IoT security and IoT protocol security have become essential research topics. There mainly exists three methods to analyse the security of IoT protocols and their implementations, which are formal analysis, static analysis, and fuzzing. We use fuzzing to address the analysis of IoT protocol implementation for three reasons. First, fuzzing can find problems in run time. In this project, we focus on the implementations of IoT protocols. Finding run-time problems is more relevant to our objective than proving the correctness of protocols' specifications or security properties. Second, fuzzing gives nearly no false-positive cases. Through analysing the results, we can efficiently identify security problems. Finally, with fuzzing strategies and differential testing, we are confident to build a well-structured fuzzing framework that can be applied to various IoT protocols.

Based on the above discussion, the objective of this thesis is to analyse implementations of IoT protocols based on fuzzing. We will build a fuzzing framework to analyse implementations of IoT protocols. The research questions considered in this thesis are listed below. We discuss our methodology for each research question in the next section.

4 Methodology

We divide our approach into 4 phases. In the first phase, we extract functionalities and rules defined in MQTT Version 5.0 OASIS Standard. In the second phase, we map the functionalities to modules in MQTT libraries, and write test cases for each functionality if it is provided by the library. We also capture the MQTT messages transmitted in test cases as the seeds for mutation and fuzzing. In the third phase, we generate mutants from seeds as the input of fuzzing. In the final phase, we analyse the discrepancies based on the rules we extracted from RFC.

5 Experiments

By parsing the MQTT version 5 standard, we create test cases in Fig 1.

Functionality	Corresponding RFC Section	Test case
CONNECT	MQTT Version 5 (Section 3.1)	4
CONNACK	MQTT Version 5 (Section 3.2)	1
PUBLISH	MQTT Version 5 (Section 3.3)	3
PUBACK	MQTT Version 5 (Section 3.4)	1
PUBREC	MQTT Version 5 (Section 3.5)	1
PUBREL	MQTT Version 5 (Section 3.6)	1
PUBCOMP	MQTT Version 5 (Section 3.7)	1
SUBSCRIBE	MQTT Version 5 (Section 3.8)	1
SUBACK	MQTT Version 5 (Section 3.9)	1
UNSUBSCRIBE	MQTT Version 5 (Section 3.10)	1
UNSUBACK	MQTT Version 5 (Section 3.11)	1
PINGREQ	MQTT Version 5 (Section 3.12)	1
PINGRESP	MQTT Version 5 (Section 3.13)	1

DISCONNECT	MQTT Version 5 (Section 3.14)	1
AUTH	MQTT Version 5 (Section 3.15)	1

Fig 1. Test cases table

We are deploying the following libraries. Although some of them only provide clients or servers, they are all active and have high scores.

Libreries	language	Github start
mosquitto	C	6174
MQTTX	Javascipt	1749
MQTT.fx	Java	
Vert.x MQTT	Java	143
vernemq	Erlang	2741
rabbitmq-server	C	9501
MQTTnet	C	2800

We will run the test file, make the client communicate with the server normally, and use Wireshark to capture the pcap file. These files will be used as seeds as input into the mutation program we write. Our mutation program will send each mutated packet to a different IoT protocol implementation. Wireshark will be started when our program starts sending mutated packets.

The results obtained from fuzzing are analysed using the rule information obtained from the RFC. In addition to comparing the differences between them and the RFC, compare the differences between the different implementations.

6 Results and Discussion

We analyse multiple implementations of MQTT. Some unusual behavior was found. Some of the challenges faced in this project. Because we read the RFCs manually, we cannot guarantee that our test cases cover all functionality, and when analyzing test results, there may be misunderstandings. Even so, the abnormal behavior of the MQTT implementation can still be found using this method.

7 Conclusion

There are many IoT protocols, corresponding to many RFCs. As IoT develops, we cannot deny that more IoT protocols may appear in the future. To better use fuzzing to analyze IoT protocol implementations, we need an automated tool to help us read RFC. At present, NLP technology is expected to solve this problem.

Acknowledgment

I would like to thank my supervisor, Dr. Guangdong Bai, for his guidance through each stage of the process. His expertise was invaluable in formulating the research questions and methodology, insightful feedback pushed me to sharpen my thinking and brought my work to a higher level.

References

[1] J. Lee, “The top 5 iot applications for agriculture,” https://blog.particle.io/the-top-5-iot-applications-for-agriculture/, 2021, accessed: 2021-10-22.
[2] S. Ranger, “What is the iiot? everything you need to know about the industrial internet of things,” https://www.zdnet.com/article/what-is-the-iiot-everything-you-need-to-know-about-the-industrial-internet-of-things/, 2019, accessed: 2021-10-22.
[3] “Alexa,” https://developer.amazon.com/en-AU/alexa, accessed: 2021-10-06.
[4] “Device access — google developers,” https://developers.google.com/nest/device-access, accessed: 2021-10-06.
[5] B. Marr, “The biggest wearable technology trends in 2021,” https://www.forbes.com/sites/bernardmarr/2021/03/05/the-biggest-wearable-technology-trends-in-2021/?sh=1e7382a73092,2021, accessed: 2021-10-22.
[6] “Colonial pipeline hit by communications outage,” https://www.cbc.ca/news/business/colonial-pipeline-outage-1.6031067, 2021, accessed: 2021-10-22.
[7] S. Morrison, “How a major oil pipeline got held for ransom,” https://www.vox.com/recode/22428774/ransomeware-pipeline-colonial darkside-gas-prices, 2021, accessed: 2021-10-22.
[8] A. Drozhzhin, “Black hat usa 2015: The full story of how that jeep was hacked,” https://www.kaspersky.com.au/blog/blackhat-jeep-cherokee-hack-explained/9493/, 2015, accessed: 2021-10-22.
[9] M. Saunokonoko, “How stuxnet worm took out key iranian nuclear facility in 2010,” https://www.9news.com.au/world/how-stuxnet-cyberattack-took-out-natanz-nuclear-facility-in-iran/37694f90-c2e1-454c-9597-e7de8f54e495, 2021, accessed: 2021-10-22.
[10] R. McMillan, “Siemens: Stuxnet worm hit industrial systems,” https://www.computerworld.com/article/2515570/siemens--stuxnet-worm-hit-industrial-systems.html, 2010, accessed: 2021-10-22.
[11] “Mqtt: The standard for iot messaging,” https://mqtt.org, 2021, accessed: 2021-10-06.
[12] “The constrained application protocol (coap),” https://datatracker.ietf.org/doc/html/rfc7252, 2014, accessed: 2021-10-06.
[13] “Oasis advanced message queuing protocol (amqp) version 1.0,” http://docs.oasis-open.org/amqp/core/v1.0/os/amqp-core-complete-v1.0-os.pdf, 2012, accessed: 2021-10-06.

[14] “Bluetooth technology website,” https://www.bluetooth.com/, accessed: 2021-10-06.
[15] “Zigbee - connectivity standards alliance,” https://zigbeealliance.org/solution/zigbee/, accessed: 2021-10-06.
[16] “Ipv6 over low power wpan (6lowpan) - documents,” https://datatracker.ietf.org/wg/6lowpan/documents, accessed: 2021-10-06.
[17] “CVE-2020-12883.” Available from MITRE, CVE-ID CVE-2020-12883., May 15 2020. [Online]. Available: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2020-12883
[18] “CVE-2020-12884.” Available from MITRE, CVE-ID CVE-2020-12884., May 15 2020. [Online]. Available: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2020-12884
[19] “CVE-2020-12885.” Available from MITRE, CVE-ID CVE-2020-12885., May 15 2020. [Online]. Available: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2020-12885
[20] “CVE-2020-12886.” Available from MITRE, CVE-ID CVE-2020-12886., May 15 2020. [Online]. Available: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2020-12886
[21] “CVE-2020-12887.” Available from MITRE, CVE-ID CVE-2020-12887., May 15 2020. [Online]. Available: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2020-12887
[22] D. Dolev and A. Yao, “On the security of public key protocols,” IEEE Transactions on Information Theory, vol. 29, no. 2, pp. 198–208, 1983.
[23] B. Chess and G. McGraw, “Static analysis for security,” IEEE Security Privacy, vol. 2, no. 6, pp. 76–79, 2004.
[24] M. Sutton, A. Greene, and P. Amini, Fuzzing: brute force vulnerability discovery. Pearson Education, 2007.
[25] Y. Li, B. Chen, M. Chandramohan, S.-W. Lin, Y. Liu, and A. Tiu, “Steelix: Program-state based binary fuzzing,” in Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering, ser. ESEC/FSE 2017. New York, NY, USA: Association for Computing Machinery, 2017, p. 627–637. [Online]. Available: https://doi.org/10.1145/3106237.3106295
[26] S. Dinesh, N. Burow, D. Xu, and M. Payer, “Retrowrite: Statically instrumenting cots binaries for fuzzing and sanitization,” in 2020 IEEE Symposium on Security and Privacy (SP), 2020, pp.1497–1511.
[27] S. Schumilo, C. Aschermann, R. Gawlik, S. Schinzel, and T. Holz, “kafl: Hardware-assisted feedback fuzzing for {OS} kernels,” in 26th {USENIX} Security Symposium ({USENIX} Security 17), 2017, pp. 167–182.
[28] D. R. Jeong, K. Kim, B. Shivakumar, B. Lee, and I. Shin, “Razzer: Finding kernel race bugs through fuzzing,” in 2019 IEEE Symposium on Security and Privacy (SP), 2019, pp. 754–768.
[29] M. Kwiatkowska, G. Norman, and D. Parker, “PRISM 4.0: Verification of probabilistic real-time systems,” in Proc. 23rd International Conference on Computer Aided Verification (CAV’11), ser. LNCS, G. Gopalakrishnan and S. Qadeer, Eds., vol. 6806. Springer, 2011, pp. 585–591.
[30] S. Meier, B. Schmidt, C. Cremers, and D. Basin, “The tamarin prover for the symbolic analysis of security protocols,” in Computer Aided Verification, N. Sharygina and H. Veith, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 696–701.
[31] B. Blanchet, Automatic Verification of Security Protocols in the Symbolic Model: The Verifier ProVerif. Cham: Springer International Publishing, 2014, pp. 54–87. [Online]. Available: https://doi.org/10.1007/978-3-319-10082-1 3
[32] M. Duflot, M. Kwiatkowska, G. Norman, and D. Parker, “A formal analysis of bluetooth device discovery,” International Journal on Software Tools for Technology Transfer, vol. 8, no. 6, pp. 621–632, Nov 2006. [Online]. Available: https://doi.org/10.1007/s10009-006-0014-x
[33] C. Richard and S. Vitaly, “Formal analysis of authentication in bluetooth device pairing,” in Proceedings of LICS/ICALP workshop on foundations of computer security and automated reasoning for security protocol analysis (FCS-ARSPA ’07), 2007.
[34] J. Y. Kim, R. Holz, W. Hu, and S. Jha, “Automated analysis of secure internet of things protocols,” in Proceedings of the 33rd Annual Computer Security Applications Conference, ser. ACSAC 2017. New York, NY, USA: Association for Computing Machinery, 2017, p. 238–249. [Online]. Available: https://doi.org/10.1145/3134600.3134624
[35] F. L. Coman, K. M. Malarski, M. N. Petersen, and S. Ruepp, “Security issues in internet of things: Vulnerability analysis of lorawan, sigfox and nb-iot,” in 2019 Global IoT Summit (GIoTS), 2019, pp. 1–6.
[36] “Lorawan technology,” https://www.lora-alliance.org, accessed: 2021-10-06.
[37] “Sigfox technology,” https://www.sigfox.com/en, accessed: 2021-10-06.
[38] “Nb-iot technology,” https://www.gsma.com/iot/narrow-band-internet-of-things-nb-iot, accessed: 2021-10-06.
[39] “Gimpel software - the leader in static analysis for c and c++ with pc-lint plus,” https://gimpel.com, accessed: 2021-10-06.
[40] “Artho software - jlint,” http://artho.com/jlint/, accessed: 2021-10-06.
[41] “Coverity scan - static analysis,” https://scan.coverity.com, accessed: 2021-10-06.
[42] P. Ferrara, A. K. Mandal, A. Cortesi, and F. Spoto, “Static analysis for discovering iot vulnerabilities,” International Journal on Software Tools for Technology Transfer, vol. 23, no. 1, pp. 71–88, Feb 2021. [Online]. Available: https://doi.org/10.1007/s10009-020-00592-x
[43] “Julia static analyzer,” https://juliasoft.com, acce

ssed: 2021-10-06.
[44] Z. B. Celik, P. McDaniel, and G. Tan, “Soteria: Automated iot safety and security analysis,” in 2018 USENIX Annual Technical Conference (USENIX 1 8), 2018, pp. 147–158.
[45] V. Sachidananda, S. Bhairav, and Y. Elovici, “O ver: Overhauling vulnerability detection for iot throu gh an adaptable and automated static analysis framew ork,” in Proceedings of the 35th Annual ACM Sympo sium on Applied Computing, ser. SAC ’20. New Yor k, NY, USA: Association for Computing Machinery, 2020, p. 729–738. [Online]. Available: https://doi.org /10.1145/3341105.3373930
[46] “american fuzzy lop (2.52b),” https://lcamtuf.cor edump.cx/afl/, accessed: 2021-10-06.
[47] M. B ̈ohme, V.-T. Pham, and A. Roychoudhury, “Coverage-based greybox fuzzing as Markov chain,” IEEE Transactions on Software Engineering, vol. 45, no. 5, pp. 489–506, 2019.
[48] C. Lemieux and K. Sen, “Fairfuzz: A targeted m utation strategy for increasing greybox fuzz testing co verage,” in Proceedings of the 33rd ACM/IEEE Inter national Conference on Automated Software Enginee ring, ser. ASE 2018. New York, NY, USA: Associati on for Computing Machinery, 2018, p. 475–485. [Onl ine]. Available: https://doi.org/10.1145/3238147.323 8176
[49] Y. Chen, Y. Jiang, F. Ma, J. Liang, M. Wang, C. Zhou, X. Jiao, and Z. Su, “Enfuzz: Ensemble fuzzing with seed synchronization among diverse fuzzers,” in 28th USENIX Security Symposium (USENIX Securi ty 19), 2019, pp. 1967–1983.
[50] M. Rajpal, W. Blum, and R. Singh, “Not all byte s are equal: Neural byte sieve for fuzzing,” arXiv pre print arXiv:1711.04596, 2017.
[51] T. Su, G. Meng, Y. Chen, K. Wu, W. Yang, Y. Yao, G. Pu, Y. Liu, and Z. Su, “Guided, stochastic m odel-based gui testing of android apps,” in Proceedin gs of the 2017 11th Joint Meeting on Foundations of S oftware Engineering, ser. ESEC/FSE 2017. New Yor k, NY, USA: Association for Computing Machinery, 2017, p. 245–256. [Online]. Available: https://doi.org /10.1145/3106237.3106298
[52] Y. Chen and Z. Su, “Guided differential testing o f certificate validation in ssl/tls implementations,” in Proceedings of the 2015 10th Joint Meeting on Found ations of Software
Engineering, ser. ESEC/FSE 2015. New York, NY, U SA: Association for Computing Machinery, 2015, p. 793–804. [Online]. Available: https://doi.org/10.1145 /2786805.2786835
[53] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbl uth, A. H. Teller, and E. Teller, “Equation of state cal culations by fast computing machines,” The journal o f chemical physics, vol. 21, no. 6, pp. 1087–1092, 19 53.
[54] C. Tian, C. Chen, Z. Duan, and L. Zhao, “Differ ential testing of certificate validation in ssl/tls implem entations: An rfc-guided approach,” ACM Trans. Soft w. Eng. Methodol., vol. 28, no. 4, Oct. 2019. [Online]. Available: https://doi.org/10.1145/3355048
[55] T. Petsios, A. Tang, S. Stolfo, A. D. Keromytis, and S. Jana, “Nezha: Efficient domain-independent di fferential testing,” in 2017 IEEE Symposium on Secu rity and Privacy (SP), 2017, pp. 615–632.
[56] L. G. Araujo Rodriguez and D. Mac ̂edo Batista, “Program-aware fuzzing for mqtt applications,” in Pr oceedings of the 29th ACM SIGSOFT International S ymposium on Software Testing and Analysis, ser. ISS TA 2020. New York, NY, USA: Association for Com puting Machinery, 2020, p. 582–586. [Online]. Avail able: https://doi.org/10.1145/3395363.3402645
[57] S. Hern ́andez Ramos, M. T. Villalba, and R. La cuesta, “Mqtt security: A novel fuzzing approach,” W ireless Communications and Mobile Computing, vol. 2018, p. 8261746, Feb 2018. [Online]. Available: http s://doi.org/10.1155/2018/8261746
[58] B. K. Aichernig, E. Mu ̌skardin, and A. Pfersche r, “Learning-based fuzzing of iot message brokers,” i n 2021 14th IEEE Conference on Software Testing, V erification and Validation (ICST), 2021, pp. 47–58.
[59] S. Kwon, S.-J. Son, Y. Choi, and J.-H. Lee, “Pro tocol fuzzing to find security vulnerabilities of rabbit mq,” Concurrency and Computation: Practice and Ex perience, vol. 33, no. 23, p. e6012, 2021, e6012 CPE-20-0718.R1. [Online]. Available: https://onlinelibrary. wiley.com/doi/abs/10.1002/cpe.6012
[60] Y. Zeng, M. Lin, S. Guo, Y. Shen, T. Cui, T. Wu, Q. Zheng, and Q. Wang, “Multifuzz: A coverage-bas ed multiparty-protocol fuzzer for iot publish/subscrib e protocols,” Sensors, vol. 20, no. 18, 2020. [Online]. Available: https://www.mdpi.com/14248220/20/18/5 194

Biography

In this project, I deeply understand how difficult it is to build a system and the importance and necessity of protecting the security of the system. At the same time, in addition to the use of fuzzing in this project and the knowledge of the Internet of Things, I also have a strong interest in NLP. NLP is a good assistant for automation, which can standardize the experimental process and reduce errors caused by human factors.