Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.35, NO.1, Feb. 2025 ISSN 2288-2715(Online)
https://doi.org/10.13089/JKIISC.2025.35.1.79

고효율 고신뢰 펌웨어의 최적화된 퍼징을 위한 이벤트 데이터 레코딩 기법*

지 승 하,[1†] 한 동 임,[2] 전 소 은,[3] 이 일 구[4‡]
[1,3,4]성신여자대학교 (학생, 대학원생, 교수), [2]백석대학교 (학생)

Event Data Recording Technique for High-Efficiency and Reliable Fuzzing of Firmware*

Seung-Ha Jee,[1†] Dong-Im Han,[2] So-Eun Jeon,[3] Il-Gu Lee[4‡]
[1,3,4]Sungshin Women's University (Student, Graduate student, Professor),
[2]Baekseok University (Student)

요 약

IoT(Internet of Things)가 일상생활과 산업에 널리 사용되면서 대규모 IoT 네트워크 환경에서 데이터를 효율적으로 처리하고 보호하는 것이 중요해졌다. 그리고 무선 네트워크 장치의 펌웨어 취약점이 IoT의 보안 문제와 직결됨에 따라 취약점을 적시에 발견하고 패치하는 것이 중요해졌다. 그러나 종래의 IoT 보안 매커니즘은 무작위 퍼징 방식으로 취약점을 탐지하여 발견한 취약점을 패치하기 위해 무선 펌웨어 업데이트를 수행하므로 취약점을 신속하고 효율적으로 탐지 및 제거하지 못한다. 본 논문에서는 무선 커넥티비티 장치의 펌웨어 패치 여부를 효율적으로 검증하기 위한 EDR(Event Data Recording) 기반의 퍼징 기법을 제안한다. 실험 결과에 따르면, 종래모델 대비 취약점 탐지율은 평균 90% 증가하였으며, 취약점 탐지 속도는 89.15% 개선되었다.

ABSTRACT

As the Internet of Things (IoT) continues to be widely used in daily life and industry, efficiently processing and protecting data in large-scale IoT networks has become increasingly important. Furthermore, vulnerabilities in the firmware of wireless network devices are directly linked to IoT security issues, making it critical to promptly identify and patch these vulnerabilities. However, conventional IoT security mechanisms rely on random fuzzing techniques to detect vulnerabilities and perform wireless firmware updates to patch them. This prevents the swift and efficient detection and resolution of vulnerabilities. This paper proposes an event data recording (EDR)-based fuzzing technique to efficiently verify firmware patch status in wireless connectivity devices. Experimental results show that, compared to conventional models, vulnerability detection rates increased by an average of 90%, and vulnerability detection speed improved by 89.15%.

Keywords: Fuzzing, Firmware, Patch mechanism, IoT security

Received(10. 04. 2024), Modified(12. 27. 2024), Accepted(12. 27. 2024)

* 본 논문은 2024년도 한국정보보호학회 하계학술대회에 발표한 우수논문을 개선 및 확장한 것임.

* 본 연구는 2024년도 국가R&D 리얼챌린지 프로그램(국가과학기술인력개발원 주관)의 재원으로 지원, 산업통상자원부 및 한국산업기술진흥원의 산업혁신인재성장지원사업 (RS-2024-00415520) 지원, 과학기술정보통신부의 정보보호핵심원천기술개발 사업(RS-2024-00437252)과 과학기술정보통신부 및 정보통신기획평가원의 ICT혁신인재4.0 사업(No. IITP-2022-RS-2022-00156310)의 연구결과로 수행되었음

† 주저자, 20210640@sungshin.ac.kr

‡ 교신저자, iglee@sungshin.ac.kr(Corresponding author)

I. 서 론

무선 커넥티비티 기술은 여러 IoT(Internet of Things) 장치를 연결하는 스마트 팜, 스마트 시티, 스마트 팩토리 등 주요 기반 시설의 핵심 구성 요소로 일상생활과 여러 분야에 널리 활용되고 있다[1,2,3,4,5,6]. 무선통신 네트워크의 효율성과 보안성이 중요해졌지만, 기술이 발전하면서 시스템의 복잡성이 증가하였고, 사용자가 파악하지 못한 취약점의 수도 증가하고 있다[7,8,9,10,11]. 이에 따라 공격자가 장치에 내재된 취약점을 악용하여 피해 파급력이 큰 공격을 수행하는 사례가 증가하고 있다[12]. 취약한 펌웨어를 사용하는 무선 커넥티비티 장치가 해킹되면 이 장치와 연결된 IoT 장치들의 보안 위협이 커진다. 이러한 문제를 해결하기 위해 무선 업데이트를 통해 펌웨어의 취약점을 패치하고 있으나, 이는 피해가 발생한 후에 패치하는 방식으로 취약점을 적시에 해결하기 어렵다[13]. 이러한 문제를 해결하기 위해 소프트웨어의 안전성을 검증하기 위한 테스트 기법인 퍼징(fuzzing) 기술이 활발하게 연구되고 있다[14,15,16,17]. 퍼징은 무작위 테스트케이스를 생성하여 검증 대상인 소프트웨어에 주입하여 잠재된 취약점 및 결함을 탐지한다[18]. 그러나 기존의 퍼저(fuzzer)는 테스트 대상에 사용할 테스트케이스를 무작위로 생성하여 테스트 프로그램에 대한 코드 커버리지를 향상하는 데에만 초점을 두고 있다[19]. 또한, 기존의 퍼징은 테스트 종료 시점이 정해져 있지 않기 때문에, 사용자가 무한히 퍼저를 실행한다면 수많은 무작위 테스트케이스를 사용하여 퍼징 테스트를 수행한다. 따라서 하나의 프로그램을 퍼징 테스트하기 위해서는 많은 시간과 리소스가 소모되는 한계점이 존재하며, 이는 효율적인 펌웨어 패치 검증을 수행하기에 부적절하다.

본 논문에서는 무선통신 장치의 펌웨어 취약점의 패치 여부를 효율적으로 검증하기 위한 EDR(Event Data Recording) 기법을 제안한다. 제안하는 기법은 패치되기 전, 취약점이 존재하는 펌웨어에 대한 퍼징 테스트를 수행한다. 테스트 과정에서 크래시를 유발하는 테스트케이스를 수집하고, 이를 패치된 펌웨어에 대한 퍼징 테스트를 수행할 때 테스트케이스로 사용한다. 펌웨어 패치 전 퍼징 테스트 과정에서 크래시를 유발했던 테스트케이스를 사용하여 효과적으로 펌웨어의 패치 여부를 검증하고 펌웨어에 대한 보안 신뢰성을 강화한다.

본 논문의 주요 기여점은 다음과 같다.

- 종래의 무작위 퍼징 기법 대비 취약점 탐지 시간을 단축하고 취약점 탐지율을 높인 고효율, 고신뢰 EDR 기법을 제안하였다.
- 퍼징 기반의 펌웨어 패치 성공 여부를 검증할 수 있는 메커니즘을 제안하였다.

본 논문은 다음과 같이 구성된다. 2장에 관련 연구를 분석하고, 3장에서 EDR 기반의 퍼징 기법을 제안한다. 그리고 4장에서 제안한 방법과 종래방법의 성능을 비교 평가하고, 5장에서 결론을 맺는다.

II. 선행연구

본 장에서는 퍼징 기법과 관련된 선행연구를 분석한다. Xiaogang Zhu 등[14]은 코드 변경이 발생한 부분을 집중적으로 퍼징하여 취약점 탐지 효율성을 높이는 기법을 제안한다. 해당 기법은 코드가 변경된 부분에 대해 반복적인 퍼징을 수행하여, 테스트 시간을 단축하면서 자원의 효율성을 높인다. 그러나 변경되지 않은 코드에 잠재된 취약점을 탐지하기에는 한계점이 존재한다. Stanislav Abaimov 등[15]은 프로그램 코드 변경으로 인해 발생하는 취약점을 효율적으로 탐지하는 RGF(Regression Greybox Fuzzing) 기법을 제안한다. 이 기법은 프로그램이 수정된 영역을 중점적으로 퍼징하여 기존의 방대한 퍼징 대상의 범위를 축소했다는 점에서 효율적이나, OSSFuzz의 버그 보고서에 기반하여 취약점을 탐지하므로 사전에 보고되지 않은 취약점은 탐지하기 어렵다는 한계점이 있다. Jisoo Jang 등[16]은 USB 드라이버의 입력 시퀀스를 기록하고, 이를 리플레이하여 테스트를 반복 수행하면서 취약점을 발견하는 ReUSB 기법을 제안한다. ReUSB는 USB 드라이버의 여러 입력 조건을 효율적으로 테스트하여 새로운 취약점을 발견할 수 있으나, 리플레이 데이터에 의존하기 때문에 탐지 성능이 떨어진다는 한계가 있다. T. Scharnowski 등[17]은 펌웨어 퍼저가 테스트케이스 생성을 안정적이고 효과적으로 변형할 수 있도록 입력되는 값을 유형화하고, 여러 데이터 스트림으로 분할하는 HOEDUR 기법을 제안한다. HOEDUR은 펌웨어의 최적화된 테스트케이스 분석 및 뮤테이션 기법을 통해 무의미한 무작위 테스트케이스를 줄이고, 코드 커버리지를 향상시켜서 퍼징 시간을 단축했으나, 펌웨어의 패치 여부를 검증하는 과정에서 수많은 테스트케이스를 사용해야 한다는 한계

점이 있다.

공통적으로 선행연구들은 코드 커버리지를 향상시키는 것에 중점을 두었기 때문에, 부가적으로 발생하는 취약점 탐지를 위해 소모되는 오버헤드와 복잡성이 증가했다. 또한 펌웨어의 패치 여부를 검증하기에는 많은 테스트케이스와 퍼징 테스트 시간이 소모되는 한계점이 존재했다. 따라서 본 논문에서 제안하는 기법은 낮은 오버헤드로, 효율적으로 펌웨어 패치 성공 여부를 검증할 수 있는 EDR 기법을 제안한다.

III. EDR Technique

본 장에서는 효율적인 펌웨어 패치 여부를 검증하기 위한 EDR 기법의 동작 방식을 서술한다. Fig.1은 EDR 기법의 동작 메커니즘을 나타낸 것이다.

EDR 기법은 패치 전 펌웨어에 무작위 테스트케이스를 삽입하여 퍼징 테스트를 수행하고, 이때 크래시를 유발했던 테스트케이스를 기록한다. 이후 패치가 적용된 펌웨어에 대해서 패치 검증 테스트를 수행할 때, 기록된 테스트케이스를 다시 퍼징의 테스트케이스로 사용하는 방식으로 동작한다. 패치된 펌웨어에 크래시를 발생시킨 취약점이 여전히 존재한다면, EDR이 기록한 테스트케이스가 해당 취약점을 다시 유발하여 크래시가 발생한다. 반면 취약점이 패치됐다면, 테스트케이스는 크래시를 발생시키지 않고 펌웨어를 통과하게 된다. 따라서, 수많은 테스트케이스

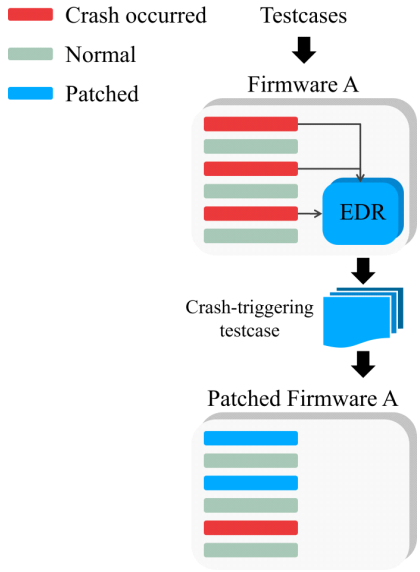

Fig. 1. Operation mechanism of EDR technique

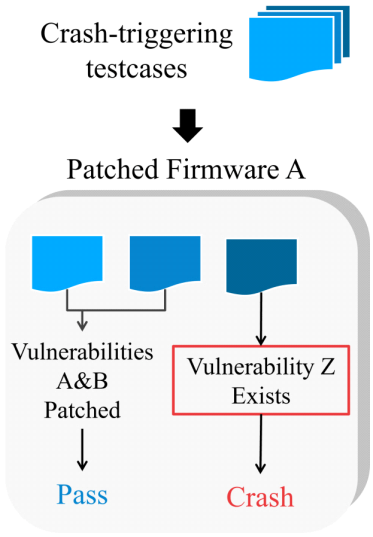

Fig. 2. Process of fuzzing patched firmware using testcases generated by the EDR technique

를 펌웨어에 주입하지 않고 패치의 성공 여부를 효과적으로 검증할 수 있다.

Fig.2는 EDR 기법으로 생성한 테스트케이스와 이를 패치된 펌웨어에 적용했을 때의 세부 동작 방식을 나타낸 것이다.

패치 전 펌웨어 취약점 A,B,Z이 있을 때, EDR이 퍼징 테스트를 과정에서 취약점에 대한 크래시를 유발하는 테스트케이스를 수집한다. 이후 취약점 패치가 완료된 펌웨어를 EDR이 수집한 테스트케이스를 사용하여 퍼징 테스트를 수행한다. 취약점 A,B는 성공적으로 패치되었으므로, 테스트케이스는 해당 취약점에 대해 크래시를 유발하지 않고 펌웨어를 통과한다. 반면, 취약점 Z는 패치되지 못했으므로, 이에 대한 크래시를 발생시킨 테스트케이스가 다시 한번 크래시를 발생시킨다. 따라서 사용자는 수많은 무작위 테스트케이스를 생성하지 않고 펌웨어의 패치 성공 여부를 효과적으로 검증할 수 있다.

IV. 성능 평가 및 분석

4.1 성능 평가 환경

본 장에서는 제안하는 EDR 기법의 성능을 검증하기 위한 실험 환경과 실험 방법을 서술한다. EDR의 성능을 비교 분석하기 위해 무작위로 테스트케이스를 생성하여 퍼징하는 RIM(Random Input

Model)[14,15,16,17]을 종래모델로 선정하였다.

EDR과 RIM 모델의 성능을 비교하기 위해 펌웨어 퍼징 시뮬레이터를 구현하였다. 펌웨어에 존재하는 취약점은 실제 무선통신 장치의 펌웨어에서 발견되는 10개의 버퍼 오버플로우(Buffer Overflow) 취약점이 존재하는 환경으로 구현했다. Table 1은 펌웨어 퍼징 시뮬레이터에 구현한 취약점에 대한 설명이다.

패치된 펌웨어는 10개의 취약점 중 5개의 취약점이 성공적으로 패치된 환경으로 가정하였다. 시뮬레이션은 1,000회 반복을 통해 실험을 진행하였으며, EDR과 RIM의 취약점 탐지율과 취약점 탐지 시간을 평가하였다. 취약점 탐지율은 패치가 적용되지 않은 펌웨어 버전의 전체 취약점 10개 중 각 모델이 탐지한 취약점 개수를 백분율로 환산하였다. 취약점 탐지 시간은 EDR과 RIM 모델이 퍼징 테스트 시작 시점부터 각각의 취약점을 탐지하기까지 소요된 시간을 의미하며, 단위는 밀리 초(millisecond, ms)이다.

Table 1. Previous Studies on Fuzzing Techniques

Vulnerability	Vulnerability Description	CVE
Heap buffer overflow	Occurs when software writes data to a buffer in the heap that exceeds its capacity, overwriting adjacent memory locations	CVE -2019-13581
Stack-based buffer overflow	Occurs when a local variable overwrites the return address stored in the stack, often due to excessive recursion	CVE -2018-4023
Integer overflow	Occurs when trying to store a value larger than the maximum storable in an integer	CVE -2021-22675
Integer underflow	Occurs when trying to store a value smaller than the minimum storable in an integer	CVE -2018-3926

4.2 성능 평가 결과 및 분석

Fig.3은 테스트케이스 비율에 따른 제안모델 EDR과 종래모델 RIM의 취약점 탐지율을 비교한 결과이다.

Fig.3에 따르면, EDR과 RIM 모두 테스트케이스가 증가함에 따라 취약점으로부터 크래시를 유발할 수 있는 인풋 값이 증가함으로 취약점 탐지율이 증가했다. EDR 기법은 크래시를 유발했던 인풋 값으로 퍼징하기 때문에 패치되지 못한 취약점만을 빠르고 정확하게 탐지할 수 있다. 반면, RIM은 무작위로 테스트케이스를 생성하여 퍼징을 수행하며, 이때 사용되는 전체 테스트케이스 대비 실제 취약점으로부터 크래시를 유발하는 테스트케이스의 비율이 20%로 제안하는 EDR보다 낮은 취약점 탐지율을 보였다. 이는 무작위 테스트케이스를 사용하더라도, 취약점에 대해 크래시를 유발할 수 있는 테스트케이스가 낮은 확률로 생성됨을 의미한다.

Fig.4는 테스트케이스 비율에 따른 EDR과 RIM의 취약점 탐지 시간을 비교한 결과이다. 실험 결과에 따르면, 제안하는 EDR과 종래 기법인 RIM 모두 테스트케이스가 증가할수록 취약점 탐지 시간도 비례하여 증가했다. EDR은 패치 전 펌웨어에 내재된 취약점을 유발한 인풋 값을 수집하였고, 이를 퍼징의 테스트케이스로 사용하므로 퍼징 과정에서 크래시를 유발할 가능성이 높다. 따라서 모든 테스트케이스를 사용하더라도 약 0.001초 이내로 취약점 탐지 시간이 종료되었다. 반면, 종래 기법인 RIM은 무작위로 테스트케이스를 생성하여 퍼징을 진행하므로,

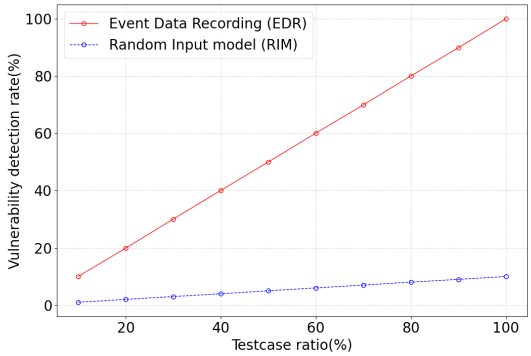

Fig. 3. Evaluation results of vulnerability detection rate

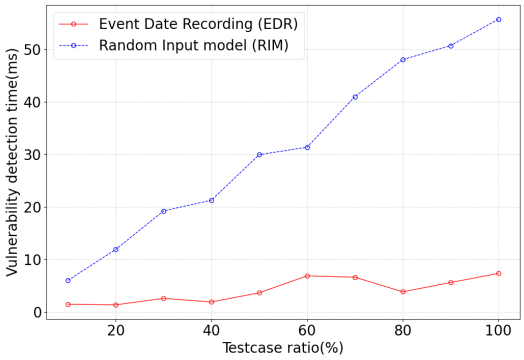

Fig. 4. Evaluation results of vulnerability detection time

취약점을 유발할 수 있는 테스트케이스 또한 무작위 확률로 생성된다. 그러므로, 각각의 취약점을 탐지하기까지 제안하는 EDR보다 취약점 탐지 시간이 더 오래 걸렸다. 결과적으로 EDR 대비 취약점 탐지 시간이 89.15% 증가했다.

V. 결 론

소프트웨어 복잡성이 증가하면서 잠재적인 취약점 또한 증가하고 있다. 특히, 와이파이 펌웨어의 경우 방대한 기능을 포함하는 대규모 시스템의 특성을 반영하고 있으므로 펌웨어의 전반적인 로직을 일일이 관리하기 어렵다. 결과적으로, 펌웨어 내의 잠재한 취약점을 발견하고 적시에 패치하고 효율적으로 관리하지 못하고 있다. 이를 해결하는 방안으로 소프트웨어의 안전성 검증을 위한 퍼저가 활발히 연구되고 있다. 그러나, 종래의 퍼징 기법은 주로 코드 커버리지 향상에 집중하고 있어서 성공적인 취약점 패치 적용 여부를 정확하게 검증하기에는 종래의 퍼징 기법이 적절하지 않았다. 따라서 본 연구에서는 무선 커넥티비티 장치 펌웨어의 취약점 패치 여부를 효율적으로 검증하기 위한 기법인 EDR을 제안하여 패치된 펌웨어에 대한 보안 신뢰성을 높이고자 하였다. 실험 결과에 따르면 제안 기법은 종래 기법 대비 취약점 탐지율은 평균 90% 증가하였으며, 취약점 탐지 속도는 89.15% 개선됨을 확인하였다.

본 연구에서는 실험 환경의 단순화를 위해 하나의 시스템에서 테스트케이스를 수집하는 환경을 가정하여 구현했다. 향후 연구에서는 다수 개의 무선 커넥티비티 장치 펌웨어가 정상 동작하는 과정에서 펌웨어의 오류 등의 이벤트를 유발한 인풋 값을 다수의 EDR 장치가 협력하여 기록하고, 퍼징에 활용하는 기법을 연구할 계획이다. 또한, 현실적인 테스트베드를 구축하여 성능 평가를 진행하고자 한다.

References

[1] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," Future Generation Computer Systems, vol. 29, no. 7, pp. 1645-1660, Sept. 2013.

[2] M. A. Rahim, M. A. Rahman, M. M. Rahman, A. S. M. Kamal, K. Cho, and S. Lee, "Evolution of IoT-enabled connectivity and applications in automotive industry: A review," Vehicular Communications, vol. 27, 100285, Jan. 2021.

[3] Q. V. Khanh, N. V. Hoai, M. L. Dao, A. N. Le, and G. Jeon, "Wireless Communication Technologies for IoT in 5G: Vision, Applications, and Challenges," Wireless Communications and Mobile Computing, vol. 2022, 3229294, Feb. 2022.

[4] S. Zeadally and O. Bello, "Harnessing the power of Internet of Things based connectivity to improve healthcare," Internet of Things, vol. 14, 100074, Jun. 2021.

[5] R. A. Mouha, "Internet of things (IoT)," Journal of Data Analysis and Information Processing, vol. 9, no. 2, 108574, Apr. 2021.

[6] D. K. Singh and R. Sobti, "Wireless Communication Technologies for Internet of Things and Precision Agriculture: A Review," 2021 6th International Conference on Signal Processing, Computing and Control (ISPCC), pp. 765-769, Nov. 2021.

[7] M. Ibrahim, A. Continella, and A. Bianchi, "AoT - Attack on Things: A

security analysis of IoT firmware updates," Proceedings of the 2023 IEEE 8th European Symposium on Security and Privacy (EuroS&P), pp. 1047-1064, Jul. 2023.

[8] C. Xenofontos, I. Zografopoulos, C. Konstantinou, A. Jolfaei, M. K. Khan, and K. R. Choo, "Consumer, Commercial, and Industrial IoT (In)Security: Attack Taxonomy and Case Studies," IEEE Internet of Things Journal, vol. 9, no. 1, pp. 199-221, Jan. 2022.

[9] X. Liang and Y. Kim, "A Survey on Security Attacks and Solutions in the IoT Network," Proceedings of the 2021 IEEE 11th Annual Computing and Communication Workshop and Conference (CCWC), pp. 853-859, Jan. 2021.

[10] P. Kumari and A. K. Jain, "A comprehensive study of DDoS attacks over IoT network and their countermeasures," Computers & Security, vol. 127, 103096, Apr. 2023.

[11] N. Abosata, S. Al-Rubaye, G. Inalhan, and C. Emmanouilidis, "Internet of things for system integrity: A comprehensive survey on security, attacks and countermeasures for industrial applications," Sensors, vol. 21, no. 11, 103096, May 2023.

[12] Ö. Aslan, M. Samet, and B. Özkan, "A comprehensive review of cyber security vulnerabilities, threats, attacks, and solutions," Electronics, vol. 12, no. 6, 1333, Mar. 2023.

[13] Y. He, Z. Zou, K. Sun, Z. Liu, K. Xu, Q. Wang, C. Shen, Z. Wang, and Q. Li, "RapidPatch: Firmware Hotpatching for Real-Time Embedded Devices," Proceedings of the 31st USENIX Security Symposium (USENIX Security 22), pp. 2225-2242, Aug. 2022.

[14] X. Zhu and M. Böhme, "Regression greybox fuzzing," Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security, pp. 2169-2182, Nov. 2021.

[15] S. Abaimov, "Understanding and Classifying Permanent Denial-of-Service Attacks," Journal of Cybersecurity and Privacy, vol. 4, no. 2, pp. 324-339, Apr. 2024.

[16] J. Jang, M. Kang, and D. Song, "ReUSB: Replay-Guided USB Driver Fuzzing," Proceedings of the 32nd USENIX Security Symposium (USENIX Security 23), pp. 2921-2938, Aug. 2023.

[17] T. Scharnowski, N. Bars, M. Schloegel, E. Gustafson, M. Muench, and G. Vigna, "Hoedur: Embedded Firmware Fuzzing using Multi-Stream Inputs," Proceedings of the 32nd USENIX Security Symposium (USENIX Security 23), pp. 3799-3816, Aug. 2023.

[18] X. Zhu, S. Wen, S. Camtepe, and W. Xiang, "Fuzzing: a survey for roadmap," ACM Computing Surveys (CSUR), vol. 54, no. 11s, pp. 1-36, Sept. 2022.

[19] H. Liang, X. Pei, X. Jia, W. Shen, and J. Zhang, "Fuzzing: State of the art," IEEE Transactions on Reliability, vol. 67, no. 3, pp. 1199-1218, Sept. 2018.

〈저 자 소 개〉

지 승 하 (Seung-ha Jee) 학생회원
2021년 3월~현재: 성신여자대학교 융합보안공학과 재학
〈관심분야〉 정보보안, 융합보안, IoT 보안

한 동 임 (Dong-Im Han) 학생회원
2021년 3월~현재: 백석대학교 첨단IT학과 재학
〈관심분야〉 정보보호, 정보통신, IoT 보안

전 소 은 (So-Eun Jeon) 학생회원
2021년 8월: 성신여자대학교 융합보안공학과 졸업
2023년 2월: 성신여자대학교 미래융합기술공학과 석사 졸업
2023년 3월~현재: 성신여자대학교 미래융합기술공학과 박사과정
〈관심분야〉 융합보안, 통신네트워크보안, IoT 보안

이 일 구 (Il-Gu Lee) 종신회원
2016년 2월: 한국과학기술원 정보보호대학원 박사
2005년 2월~2017년 2월: 한국전자통신연구원 선임연구원
2014년 2월~2017년 2월: 뉴라텍/뉴라컴 책임연구원/프로젝트리더
2017년 2월~현재: 성신여자대학교 융합보안공학과/미래융합기술공학과 부교수
2024년 3월~현재: 성신여자대학교 융합보안공학연구소 소장
〈관심분야〉 정보보호, 융합보안, 정보통신