# 리눅스 커널 취약점 세대별 트렌드와 탐지기술 발전 조사

장수혁*, 최상훈[1], 박기웅†

*세종대학교 SysCore Lab. (대학원생*, 연구교수[1])

**세종대학교 정보보호학과 (교수† )

# Generational Trends in Linux Kernel Vulnerabilities and Advances in Detection Techniques

Su-Hyeok Jang*, Sang-Hoon Choi[1], Ki-Woong Park†

*SysCore Lab., Sejong University
Dept. of Computer and Information Security, Sejong University
(Graduate Student*, Research Professor[1], Professor† )

**요 약**

리눅스 커널은 다양한 시스템 환경에서 핵심적인 운영체제로 사용되며, 최근에는 고성능 기능을 위해 eBPF, io_uring 등 새로운 인터페이스들이 지속적으로 도입되고 있다. 그러나 이러한 구조적 진화는 커널 내부의 복잡도를 증가시켜 다양한 보안 취약점이 반복적으로 발생하는 원인이 되고 있다. 본 논문에서는 2005년부터 최근까지 보고된 리눅스 커널 보안 취약점을 기반으로, 탐지 기술의 발전 흐름을 총 3세대로 구분하여 정리한다. 1세대는 정적 분석 중심, 2세대는 퍼징 기반 자동화, 3세대는 정상 흐름 오용 기반 탐지 방식으로 정의되며, 각 기술의 특징과 한계점을 사례 중심으로 분석하였다. 이를 통해 커널 보안 기술이 어떤 방식으로 진화해왔는지를 고찰하고, 향후 구조 기반 위협 대응을 위한 연구 방향성을 제시한다.

## I. 서론

리눅스 커널은 서버, 클라우드, 모바일, IoT와 같은 다양한 환경에서 사용되며 시스템 자원 및 권한 제어를 담당하는 핵심 구성 요소이다. 특히 최근에는 성능 향상과 유연성을 위해 eBPF, io_uring과 같은 고급 기능이 도입되고 있으며, 이러한 기능들은 동시에 커널 내부 구조를 복잡하게 만들고 있다. 이로 인해 새로운 형태의 구조적 취약점이 반복적으로 발견되고 있으며, 기존 보안 기술만으로는 이를 완전하게 대응하기 어려운 상황이다. 실제로 eBPF는 사용자 정의 프로그램을 커널에서 직접 실행할 수 있도록 허용하지만, 이 과정에서 발생할 수 있는 메모리 보호 우회나 JIT 영역 훼손과 같은 문제가 반복적으로 보고되고 있다[7][8]. io_uring 역시 고속 비동기 처리를 제공하는 장점이 있지만, 그 내부 동기화 과정에서 발생하는 취약점은 여전히 보안 상의 이슈로 지적된다[9]. 이러한 구조적 위협은 단순히 코드 상의 실수나 논리적 결함을 넘어서, 시스템 설계 수준의 문제로까지 확장되고 있다. 이에 본 논문은 리눅스 커널 취약점의 발생 양상을 시기별로 정리하고, 이를 탐지하기 위해 사용된 기술의 발전 흐름을 세대별로 구분하여 분석하였다. 각 세대의 기술적 특성과 대표적인 사례를 중심으로 탐지 기법의 한계와 대응 방안을 고찰하고, 향후 보안 기술이 나아가야 할 방향을 제안하고자 한다.

† 교신저자: 박기웅 (세종대학교 정보보호학과 교수)

본 논문은 과학기술정보통신부의 재원으로 정보통신기획평가원 (IITP)의 정보보호핵심원천기술개발(Project No. RS-2024-00438551,30%), 한국연구재단(NRF) 중견후속연구사업(Project No. RS-2023-00208460, 30%), 실감콘텐츠핵심기술개발(Project No. RS-2023-00228996,40%)의 지원을 받아 수행된 연구임.

## II. 관련 연구

리눅스 커널 보안에 대한 연구는 활발히 진행되고 있으며, 각 연구는 고유한 기여와 분석 방향을 가지고 커널 보안 기술 발전에 중요한 기초 자료를 제공하고 있다.

### 2.1. 1세대 - 정적 분석 기반 연구

리눅스 커널 보안 탐지 기술의 첫 번째 세대에서는 정적 분석 기법에 중점을 둔 연구들이 주를 이루었다.

#### 2.1.1. Vulnerability Prediction Models: A Case Study on the Linux Kernel

2011년 Matthieu Jimenez 외 2명은 “Vulnerability Prediction Models: A Case Study on the Linux Kernel”을 발표하며 소프트웨어 메트릭과 함수 호출 분석을 결합한 예측 모델을 제안하였다[1]. 이 연구는 커널 소스 코드에서 추출한 정적 특성으로 취약점 발생 가능성을 사전에 파악할 수 있음을 입증했다.

#### 2.1.2. An Empirical Study of Static Analysis Tools for Secure Code Review

2012년 Wachiraphan Charoenwet외 3명은 “An Empirical Study of Static Analysis Tools for Secure Code Review”를 통해 주요 SAST 도구인 Cppcheck, Coverity, Flawfinder의 경고 정확도를 비교 평가하였다[2]. 실험결과, 평균 오탐률은 35% 수준이었으나, Coverity는 실제 취약점 식별에서 70% 이상의 검출률을 기록해 도구별 장단점을 분석하였다.

### 2.2. 2세대 - 퍼징 기반 연구

두 번째 세대 관련 연구로는 퍼징 기법의 효율성과 정확도를 높이기 위한 시도가 주목받았다.

#### 2.2.1. Tuning Configuration Selection for Continuous Kernel Fuzzing

2018년 Sanan Hasanov외 2명은 “Tuning Configuration Selection for Continuous Kernel Fuzzing”을 발표하며 Syzkaller 기반 퍼징의 테스트 구성 최적화 전략을 제안하였다[3]. 이 전략을 적용한 결과, 코드 커버리지가 기존 대비 25% 확대되고, 충돌 탐지 속도가 30% 향상됨을 보였다.

#### 2.2.2. SyzScope: Revealing High-Risk Security Impacts of Fuzzer-Exposed Bugs in Linux Kernel

2019년 Xiaochen Zou외 4명은 “SyzScope”를 제시하여 퍼징으로 생성된 10만 건 이상의 충돌 로그 중 보안적으로 의미 있는 사례만을 자동 분류,필터링하였다[4]. SyzScope는 필터링 정확도를 85%까지 높여 연구자의 후속 분석 부담을 줄였다.

#### 2.2.3. Structurally Aware Fuzzing of the Linux Kernel

2022년 S. Park 외 2명은 “Structurally Aware Fuzzing of the Linux Kernel”을 발표하며 커널의 구조 인식을 기반으로 하는 퍼징 기법을 제안하였다[10]. 이 연구는 커널 내부 데이터 구조의 흐름을 반영하여 퍼징의 효율성과 취약점 노출 가능성을 크게 향상시켰다.

### 2.3. 3세대 - 정상 흐름 오용 기반 연구

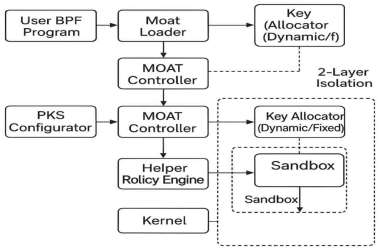
세 번째 세대에서는 공급망 침투와 정상 흐름 오용 공격을 분석한 연구들이 있다.

#### 2.3.1. Wolves in the Repository: A Software Engineering Analysis of the XZ Utils Supply Chain Attack

2024년 Piotr Przymus외 1명은 “Wolves in the Repository”에서 XZ Utils 프로젝트에 은밀히 삽입된 백도어(CVE-2024-3094)를 분석하고, 오픈소스 유지보수 권한 악용 방식의 위협 모델을 제시하였다[5].

#### 2.3.2. MOAT

MOAT는 eBPF 인터페이스 보안성 강화를 위해 Intel MPK(Memory Protection Key)를 활용한 격리 프레임워크를 제안하였다[7]. eBPF는 사용자 정의 로직이 커널에 삽입될 수 있다는 점에서 매우 위험한 인터페이스이며, verifier의 허점을 통해 JIT 영역이 손상될 경우 커널 내 임의 코드 실행이 가능해진다. MOAT는 eBPF 프로그램 실행 시 verifier와 실행 영역을 서로 다른 MPK로 분리하며, 실행 중 권한 전환 시 키 스위칭을 통해 접근을 통제한다. 또한 커널 패치 없이 동작 가능하도록 설계되었으며, 실험에서는 기존 공격 코드가 MPK 경계에서 차단됨을 입증하였다. (그림1)은 MOAT의 기본 작동 구조를 나타낸 도식이다.

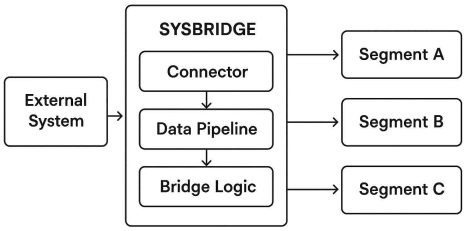



(그림 1) MOAT 내부 구조

#### 2.3.3. Sysbridge

SyzBridge는 2024년 NDSS에서 발표된 연구로, 퍼

징 기반 도구가 탐지한 수많은 충돌 중 실제로 코드 실행으로 이어지는 익스플로잇 가능한 취약점을 선별하는 시스템이다[6]. 연구팀은 Syzkaller로 수집된 191개의 리눅스 커널 충돌 중 약 132건이 실제 제어 흐름 탈취 또는 권한 상승으로 이어질 수 있음을 입증하였다. 이 시스템은 커널 로그와 스택 트레이스를 기반으로 취약점을 분석하며, 취약점 유형 분류(UAF, OOB 등), 트리거 조건, 취약 함수 패턴을 자동으로 태깅한다. 이를 통해 보안 연구자가 실제 우선 대응해야 할 취약점을 정량적으로 분류할 수 있는 기초 자료를 제공하였다. (그림 2)는 Sysbridge의 내부 구조를 나타낸 도식이다.




(그림 2) Sysbridge 내부 구조

## III. 커널 보안 위협 및 탐지 기술의 세대별 흐름

리눅스 커널 보안 위협에 대응하기 위한 탐지 기술은 정적 분석, 퍼징, 구조 흐름 기반 분석 등 다양한 방식으로 발전해왔다. 이들 기술은 단절적으로 교체되기보다는, 특정 시기에 중심적으로 활용되던 방식이 있었고 이후 새로운 접근법이 발견되면서 병행,보완되는 형태로 진화해왔다. <표1>은 그 중심 축의 변화에 따라 탐지 기술의 발전 흐름을 정리한 것이다.

<표 1> 커널 취약점 탐지 기술의 세대별 정리

| 세대 | 특징 | 기술/도구 | CVE | 공격유형 |
|---|---|---|---|---|
| 1 | -수작업 코드리뷰 및 정적 분석 도구 사용 | -Sparse<br>-Coverity<br>-Coccinelle | CVE-2009-2692 | -널 포인터 역참조<br>-Use-After-Free |
| 2 | -Syzkaller 기반 자동화 퍼징 | -Syskaller<br>-Trinity | CVE-2016-5195 | -Race Condition |
| 3 | -정상 흐름 조합 및 공급망 악용 | -MOAT<br>-SyzBridge<br>-Beyond Control | CVE-2024-3094 | -공급망 공격<br>-흐름 오용 |

### 3.1. 1세대: 정적 분석 중심 도입 (2005 ~)

이 시기의 탐지 기술은 주로 정적 분석 도구와 수작업 코드 리뷰에 의존하였다. 커널 코드의 API 흐름과 구조를 기반으로 결함을 추론하는 방식이 중심이었으며, 널 포인터 역참조나 Use-After-Free와 같은 비교적 명확한 취약점이 주요 탐지 대상이었다. 대표적인 사례로 CVE-2009-2692는 널 포인터 역참조를 이용해 루트 권한을 탈취한 공격으로, 당시 정적 분석 기반 도구들이 얼마나 중요한 역할을 했는지를 보여준다. Jimenez 외 연구진[1]은 소스코드 메트릭을 기반으로 취약점 발생 가능성을 예측하는 모델을 제안하였고, Charoenwet 등[2]은 Cppcheck, Coverity, Flawfinder와 같은 주요 정적 분석 도구의 탐지 정확도를 비교하여 도구 간 차이를 분석하였다.

### 3.2. 2세대: 퍼징 기반 자동화 확대 (2015 ~)

2015년 이후부터는 Syzkaller와 같은 퍼징 도구가 등장하면서 자동화된 취약점 탐지 기법이 본격적으로 활용되기 시작했다. 퍼징은 시스템 콜 경로와 파라미터 조합을 무작위로 생성하여 예상치 못한 경로에서의 충돌을 유발하고 이를 기반으로 결함을 탐지하는 방식이다. Hasanov 외 연구진[3]은 퍼징의 효율성을 높이기 위한 테스트 구성 최적화 방안을 제안하였으며, Zou 등이 발표한 SyzScope[4]는 수많은 충돌 중 실제 보안 위협으로 이어질 수 있는 취약점만을 선별해내는 시스템을 구현하였다. 이 시기 대표적인 사례로는 CVE-2016-5195, 일명 Dirty COW 공격이 있으며, 경쟁 조건을 이용해 읽기 전용 파일을 변조할 수 있는 심각한 취약점이었다.

### 3.3. 3세대: 정상 흐름 오용 기반 정밀 탐지 (2022 ~)

최근에는 단순한 결함이나 충돌보다는 정상적인 흐름을 교묘하게 조작하거나 공급망을 통해 은밀하게 침투하는 공격이 증가하고 있다. 대표적으로 XZ Utils의 백도어 사건(CVE-2024-3094)은 유지보수 권한을 악용해 오픈소스 소프트웨어에 악성 코드를 삽입한 사례로, 기존의 정적 분석이나 퍼징만으로는 탐지가 어려웠다[5]. 이러한 흐름에 대응하기 위해 MOAT는 eBPF의 verifier와 실행 영역을 MPK 기반으로 분리하여 JIT 영역의 공격을 차단하는 구조적 방식을 제안하였고[7], SyzBridge는 퍼징 도구에서 발생한 충돌 중 실제 익스플로잇으로 이어질 수 있는 취약점을 선별 분석하는 시스템을 제안하였다[6].

## IV. 도전 과제 및 향후 연구 방향

최근 커널 보안 위협은 정상적인 API 호출의 조합이나 업데이트 경로 은닉 등으로 탐지를 회피하며, 기존의 정적 분석이나 퍼징 기반 기법으로는 이러한 공격을 실시간으로 식별하는 데 한계가 존재한다. 특히, 동일한 공격 기법이라도 시스템 설정값이나 보안 모듈(AppArmor, SELinux 등)에 따라 탐지 가능성이 달라지는 현상이 보고되고 있으며, 이는 커널 내부의 복잡한 인터페이스 상호작용이 기존 탐지 도구에서 충분히 고려되지 않았기 때문이다. 본 논문에서는 이러한 문제를 해결하기 위해, 커널 인터페이스 호출 흐름과 환경 설정의 조합을 모델링하고, 이를 기반으로 정책 기반 행위 분석 및 위협 시뮬레이션이 가능한 자동화 분석 체계를 제안한다. 이 체계는 다양한 시스템 상태를 반영한 호출 경로 시나리오를 생성하고, 정책 위반 가능성을 자동 탐지함으로써, 환경 변화에 따른 위협 발생 조건을 동적으로 식별할 수 있을것으로 예상한다.

## V. 결 론

본 논문에서는 리눅스 커널의 보안 취약점 탐지 기술을 세대별로 구분하여, 1세대(정적 분석 기반), 2세대(퍼징 기반 자동화), 3세대(정상 흐름 오용 기반) 각각의 주요 특징과 대표 사례를 조사하였다. 1세대는 정적 분석 도구 및 수작업 코드 리뷰를 기반으로 명백한 결함 탐지에 주력하였으며, 2세대는 자동화된 퍼징 도구를 활용해 충돌 로그로부터 복잡한 동시성 취약점을 효과적으로 식별하였다. 3세대에서는 공급망 공격이나 정상 흐름 오용 등 은밀한 공격 유형에 대응하기 위한 구조적 분석 기법이 도입되었다. 이러한 세대별 기술 발전 양상을 통해, 리눅스 커널 보안 탐지 기술이 점차 구조 기반 자동화 중심으로 고도화되고 있음을 확인할 수 있었다. 본 논문은 이러한 흐름을 종합적으로 정리하고자 하였으며, 향후 연구에서는 커널 내 상호작용 모델링, 환경 조합 기반 이상 탐지, 정책 기반 동작 분석 등의 기술이 더욱 주목받을 것으로 예상된다. 본 조사가 커널 보안 연구의 기초 자료로 활용되기를 희망한다.

## [참고문헌]

[1] J. Jimenez, M. Elazab, and Y. Le Traon,Vulnerability Prediction Models: A Case Study on the Linux Kernel, ORBilu, 2016.

[2] Z. Yang, Y. Jiang, H. Wang, Y. Li, and K. Chen,An Empirical Study of Static Analysis Tools for Secure Code Review, arXiv:2407.12241, 2024.

[3] L. Chen, Z. Qian, Z. Feng, C. Zhang, and K. Chen,Tuning Configuration Selection for Continuous Kernel Fuzzing, FuTURES³ Lab, University of Utah, ICSE 2025 (to appear).

[4] Z. Li, H. Zhang, Y. Jiang, S. Wei, X. Bai, and X. Wang,SyzScope: Revealing High-Risk Security Impacts of Fuzzer-Exposed Bugs in Linux Kernel, arXiv:2111.06002, 2021.

[5] A. L. Garcia, M. Howard, J. Bonneau, and R. Anderson,Wolves in the Repository: A Software Engineering Analysis of the XZ Utils Supply Chain Attack, arXiv:2504.17473, 2024.

[6] Z. Li, Y. Chen, Z. Zhou, J. Zhang, Y. Feng, and T. Wei, "SyzBridge: Bridging the Gap Between Fuzzing and Exploitation in the Linux Kernel," in Network and Distributed System Security Symposium (NDSS), 2024.

[7] Y. Xu, J. Wang, L. Yang, and K. Wang, "MOAT: Securing eBPF Execution with Memory Protection Keys," arXiv preprint arXiv:2301.13421, 2023.

[8] A. Aristizábal, J. Aumasson, and T. Kipf,"Beyond Memory Safety: Exploiting eBPF Verifier Logic,"Proceedings of Black Hat USA, 2023.

[9] J. Wang, T. Wei,"Curing: io_uring-based Rootkit Attacks and Mitigations,"arXiv preprint arXiv:2403.04102, 2024.

[10] S. Park, D. Kim, K. Woo,"Structurally Aware Fuzzing of the Linux Kernel,"USENIX Security Symposium, 2022.