JDCS
디지털콘텐츠학회논문지
Journal of Digital Contents Society
Vol. 27, No. 8, pp. 2287-2296, Aug. 2026

Check for updates

저사양 포렌식 환경을 위한 SLM 기반 부트킷 감염 사후조사 프레임워크

장 혁 수[1] · 손 태 식[2*]
[1]아주대학교 사이버보안학과 학사과정
[2]아주대학교 사이버보안학과 교수

A Small Language Model-Based Bootkit Post-Incident Investigation Framework for Resource-Constrained Forensic Environment

Hyeoksu Jang[1] · Taeshik Shon[2*]

[1]Undergraduate Student, Department of Cyber Security, Ajou University, Gyeonggi 16499, Korea
[2]Professor, Department of Cyber Security, Ajou University, Gyeonggi 16499, Korea

[요 약]

본 연구에서는 운영체제 장악 여부와 관계없이 독립적인 분석이 가능한 Small Language Model 기반의 부트킷 사후조사 프레임워크를 제안한다. 기존의 Windows API 의존형 탐지의 한계 극복을 위해, 본 연구에서는 MBR의 원시 데이터를 직접 해석하여 분석의 객관성을 확보하고자 한다. 특히, 리소스가 제한된 환경을 고려하여 QLoRA 파인튜닝을 적용한 Phi-3.5-mini 모델을 활용한다. 본 연구에서는 프레임워크의 핵심인 MBR 분석기를 대상으로 실증적 검증을 수행하였다. 실험 결과, HEX 기반 입력이 ASM 기반 입력 대비 현저히 높은 탐지 성능을 보였으며, 이는 암호화 및 난독화된 MBR의 낮은 디스어셈블리 커버리지에서 기인함을 분석하였다. 본 연구는 OS 무결성이 파괴된 극한의 상황에서도 저사양 하드웨어만으로 신뢰할 수 있는 부트킷 분석이 가능함을 입증하였으며, 향후 지능형 사고 대응 체계 실효성 증진에 기여할 것으로 기대된다.

[Abstract]

This study introduces an advanced forensic framework based on a Small Language Model for independent bootkit analysis, functioning regardless of OS compromise. To address the vulnerabilities of traditional API-dependent detection and secure forensic objectivity, this research directly interprets raw MBR data. The underlying Phi-3.5-mini model is optimized using QLoRA fine-tuning for efficient deployment in resource-constrained environments. Empirical verification on the framework's core MBR analyzer demonstrates that HEX-based inputs achieve superior detection performance compared to Assembly data because HEX inputs effectively mitigate low disassembly coverage triggered by sophisticated MBR encryption and obfuscation. The study proves that robust bootkit analysis can be reliably executed on low-specification hardware, contributing to the efficacy of intelligent incident response systems.

색인어 : 경량 언어 모델, 디지털 포렌식, 원시 데이터 분석, 부트킷, QLoRA
Keyword : Small Language Model, Digital Forensics, Raw Data Analysis, Bootkit, QLoRA

http://dx.doi.org/10.9728/dcs.2026.27.8.2287

Received 21 May 2026; **Revised** 22 June 2026
Accepted 06 July 2026

***Corresponding Author; Taeshik Shon**

Tel: +82-31-219-3321
E-mail: tsshon@ajou.ac.kr

Ⅰ. 서 론

사이버 위협이 지능화됨에 따라 시스템의 제어권을 근본적으로 탈취하려는 시도가 지속되고 있다. 특히 Master Boot Record(MBR)를 변조하는 Bootkit(부트킷)은 운영체제가 로드되기 전 단계에서 실행되므로, 커널 수준의 보안 솔루션이 가동되기 전에 시스템을 장악할 수 있는 강력한 은닉성을 가진다. MBR은 시스템 부팅 시 가장 먼저 참조되는 물리적 영역이라는 점에서, 이곳에 삽입된 악성 코드는 시스템 전체의 신뢰의 기점을 파괴한다. 따라서 침해사고 발생 시 MBR의 변조 여부를 신속하고 정확하게 판별하는 사후조사 과정은 사고 대응 및 피해 확산 방지를 위한 필수적인 단계이다.

기존의 악성코드 탐지 연구는 주로 운영체제가 제공하는 Windows API 호출 패턴이나 파일 시스템 이벤트 등의 동적 데이터를 수집하여 분석하는 방식에 의존한다. 그러나 부트킷에 의해 운영체제가 장악된 경우, API 호출 결과가 변조되었을 가능성이 있어, 보안 솔루션의 신뢰도가 떨어지는 한계가 존재한다.

이러한 한계를 극복하기 위해, 본 연구에서는 원시 바이너리 데이터에 직접 접근하여 분석한다. 원시 데이터는 인간이 이해하기 어려운 16진수(HEX) 혹은 어셈블리어(ASM)의 형태를 갖기 때문에 해석 과정이 필요하다. 이때, 전문 인력이 수동으로 분석하는 방식은 침해사고 발생 시 실시간 대응에 심각한 시간적 한계를 가지기 때문에 이를 극복할 자동 분석 방식이 요구된다.

최근 대규모 언어 모델(LLM)을 활용한 악성코드 분석 연구가 활발하나, 국내 중소기업의 침해사고대응 현장이나 군·경찰의 기밀 조사 환경은 고성능 하드웨어 인프라의 부재뿐만 아니라, 보안 유지를 위해 외부 인터넷 연결이 전면 차단되는 폐쇄망 환경을 띤다. 이로 인해 인프라 의존성이 높고 대량의 연산 자원을 요구하는 클라우드 기반의 대규모 언어 모델을 실시간으로 활용하는 것은 현실적으로 불가능하다. 이러한 제한된 환경에서는 온-디바이스 방식의 경량 분석 솔루션이 요구된다.

본 연구는 이러한 제약 조건을 핵심 설계 원칙으로 수용하여, RTX 3080 수준의 소비자용 GPU에서 동작 가능한 Small Language Model(SLM) 기반 부트킷 분석 프레임워크를 제안한다. 구체적으로는 Phi-3.5-mini 모델에 QLoRA(Quantized Low-Rank Adaptation) 파인튜닝을 적용하여 특정 도메인 지식을 주입하며, 특히 데이터 제공 양식(HEX vs ASM)에 따른 모델의 성능차이를 실험적으로 입증한다.

본 논문의 구성은 다음과 같다. 2장에서는 관련 선행 연구를 고찰하고, 기존 API 의존형 분석 방식의 한계를 짚어 본 연구의 차별성을 명확히 한다. 3장에서는 제안하는 다영역 통합 부트킷 탐지 프레임워크를 소개하고, 본 연구의 실험적 검증 대상인 MBR 분석기를 위한 데이터 수집 파이프라인, 의미 보존 mutation 기반의 데이터셋 생성 전략을 상세히 기술한다. 4장에서는 실험 결과와 정량적 평가 지표를 제시하며, 특히 HEX와 ASM 입력 표현 방식 및 한국어와 영어 도메인 간의 탐지 성능을 비교 분석한다. 5장에서는 실험 결과를 바탕으로 원시 바이너리 분석에서 HEX 입력의 우월성, 입출력 언어별 안정성 차이, 그리고 현장 장비로서의 저사양 환경 적합성에 대해 고찰한다. 6장에서는 본 연구의 결론과 향후 연구 방향을 제시한다.

Ⅱ. 관련 연구

2-1 MBR(Master Boot Record) 구조 및 역할

MBR은 저장 장치의 첫 번째 섹터에 위치하며, 시스템 부팅 시 바이오스(BIOS)에 의해 가장 먼저 메모리에 로드되어 실행되는 512 바이트 크기의 핵심 데이터 구조이다. MBR은 크게 시스템 부팅을 제어하는 부트 코드 영역, 디스크의 구획 정보를 담고 있는 파티션 테이블 영역, 그리고 유효성을 검증하는 부트 시그니처의 세 가지 영역으로 구성된다.

부트 코드 영역은 활성화된 파티션을 찾아 해당 파티션의 부트 섹터(VBR)에 제어권을 넘겨주는 기계어 명령어가 포함되어 있다. 부트킷 악성코드는 주로 이 영역의 정상 코드를 변조하거나 자체 악성 코드를 삽입하여 실행 권한을 선점한다.

파티션 테이블은 디스크의 파티션 정보를 저장하는 공간으로, 최대 4개의 주 파티션 정보를 각각 16 바이트씩 할당하여 관리한다.

부트 시그니처는 MBR의 마지막에 위치하는 2 바이트(0x55 0xAA)의 매직 넘버로, 해당 섹터가 유효한 MBR임을 바이오스에 알리는 역할을 수행한다. 이 값이 변조될 경우 시스템은 부팅 가능한 장치로 인식하지 못한다.

2-2 부트킷 위협과 MBR 분석

부트킷은 운영체제가 로드되기 이전 단계에서 실행되는 악성코드로, MBR 또는 Volume Boot Record(VBR)를 변조하여 시스템의 부팅 과정을 장악한다. Kleissner는 Black Hat USA 2009에서 Stoned Bootkit을 발표하며, MBR 감염을 통한 전체 디스크 암호화 우회 및 루트킷 은닉 기법을 실증적으로 보여주었다[1]. 이후 Petya, Satana, NotPetya 등 MBR 감염형 랜섬웨어가 대규모 피해를 야기하면서, MBR 영역에 대한 포렌식 분석의 중요성이 크게 부각되었다[2].

2-3 언어 모델 기반 악성코드 분석

LLM을 활용한 보안 분석 연구는 최근 활발히 진행되고 있으나, 대부분 GPT-4, Claude 등 클라우드 기반 대형 모델에

의존하며 폐쇄망 환경에서의 운용을 전제하지 않는다. 이에 대한 대안으로, 최근 선행 연구[3]에서 LoRA로 최적화된 경량 언어 모델(SLM)을 활용하여 폐쇄망 환경에서 악성코드를 탐지하고 행동을 분석하는 프레임워크를 제안하였다. 해당 연구는 네트워크 격리 환경에서도 LLM 수준의 분석 능력을 확보할 수 있음을 보여주었으며, 본 연구와 가장 직접적으로 관련된 선행 연구에 해당한다.

그러나, 선행연구[3]에는 다음의 한계가 존재한다. 첫째, 분석 대상이 일반 악성코드에 한정되어 있어, MBR과 같은 운영체제 로드 이전 단계의 부트 영역 바이너리에는 직접 적용할 수 없다. 둘째, 입력 데이터로 Windows API 호출 시퀀스를 사용하므로, 부트킷에 의해 OS 자체가 장악된 환경에서는 API 반환 값이 변조되었을 가능성이 있어 입력의 신뢰성이 보장되지 않는다.

본 연구는 선행연구[3]의 접근과 유사하게 폐쇄망 환경에서 운용 가능한 SLM 기반 프레임워크를 목표로 하되, 위 두 가지 한계를 극복하기 위해 분석 대상을 MBR 영역의 원시 바이너리로 특화하고, Windows API 의존성을 완전히 배제하여 운영체제 무결성이 파괴된 상황에서도 신뢰할 수 있는 분석이 가능하도록 한다.

2-4 LoRA 및 QLoRA 파인튜닝

사전학습된 대규모 모델을 특정 도메인에 적응시키기 위해서는 파인튜닝이 필요하나, 전체 파라미터를 재학습하는 방식은 막대한 연산 자원을 요구한다. LoRA(Low-Rank Adaptation)는 사전학습된 가중치를 동결한 채 저순위(low-rank) 행렬 쌍을 주입하여 학습 가능한 파라미터 수를 대폭 감소시키는 기법이다[4]. 이를 통해 전체 파라미터 학습 대비 수십 배 적은 자원으로도 우수한 성능을 낼 수 있다.

Dettmers 등은 LoRA에 4-bit 양자화를 결합한 QLoRA를 제안하여, 기존 LoRA의 자원 효율성을 한층 더 개선하였다[5]. QLoRA는 베이스 모델의 가중치를 4-bit NormalFloat(NF4)로 양자화하여 메모리 사용량을 크게 줄이면서도, 저순위 어댑터는 16-bit 정밀도로 학습하여 성능 저하를 최소화한다. 본 연구는 이 QLoRA 기법을 Phi-3.5-mini에 적용하여, RTX 3080(10GB VRAM)급 소비자용 GPU에서 전체 학습 파이프라인이 안정적으로 동작하는 것을 확인하였다.

Ⅲ. SLM을 이용한 MBR 분석 기반 부트킷 감염 사후조사 프레임워크 제안

3-1 프레임워크 구조 및 구성요소

제안하는 프레임워크는 디스크 이미지를 입력을 받아 부트킷 감염 여부 및 행위를 진단하는 4단계 파이프라인으로 구성된다. 그림 1은 전체 프레임워크의 흐름을 나타낸다.

먼저, 섹터 비교 추출기(Sector Diff Extractor)는 변조가 발생한 영역의 원시 데이터(512B)를 추출한다. 동시에 변조 영역을 목록화한 Diff Summary JSON을 생성하여 최종 보고서 생성기에 전달한다.

다음으로, 데이터 전처리기(Data Preprocessor)는 추출된 512바이트 원시 데이터를 모델이 인식할 수 있는 최적의 입력 형태로 정제하고, 모델의 공식 챗 템플릿(Chat Template)에 맞춰 구조화하는 역할을 수행한다.

다음으로, 경량 언어 모델 추론 엔진(SLM Inference Engine)은 영역별로 특화된 4개의 분석기(MBR Analyzer, VBR Analyzer, MFT Analyzer, Slack Analyzer)를 포함하며, 각 분석기는 증거 기반 분석 보고서(Evidence-Grounded Insight Report)를 반환한다.

마지막으로, 섹터 비교 추출기의 Diff Summary JSON과 경량 언어 모델 추론 엔진의 증거 기반 분석 보고서들을 전달받은 최종 보고서 생성기(Report Generator)가 최종 판정 및 보고서를 생성한다.

본 연구에서는 위 프레임워크의 전체 구조를 설계하되, 실험적 검증은 MBR 분석기에 집중한다. 이는 MBR이 부트킷의 가장 빈번한 1차 감염 대상이며[6], 해당 분석기의 유효성이 입증될 경우 동일한 학습 파이프라인을 VBR·MFT·Slack 영역에 확장 적용할 수 있기 때문이다.

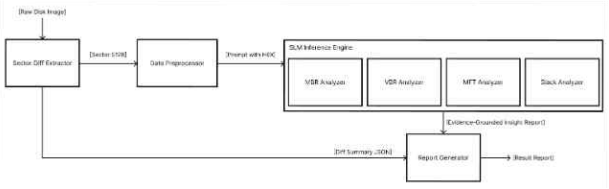

그림 1. 프레임워크 흐름도
Fig. 1. Framework workflow

1) 섹터 비교 추출기(Sector Diff Extractor) 상세

섹터 비교 추출기는 정상 기준 이미지와 분석 대상 이미지 간의 섹터 단위 비교를 수행하여, 변조가 발생한 영역의 원시 데이터를 추출한다. 이후 모델에 입력을 위해 추출된 데이터를 전처리기로 전달한다. 그와 동시에, 변조된 섹터 목록, 해당 오프셋, 소속 영역(MBR, VBR, Post-MBR Gap 등)을 기록한 Diff Summary JSON을 생성한다. 해당 JSON은 어떤 영역에서 몇 개의 섹터가 변조되었는지를 구조화하여, 후속 모듈이 분석 범위를 결정하는 데 활용되며, 최종 분석기의 진단 보고서 생성시에도 활용된다.

이러한 방식은 방대한 전체 디스크 데이터를 전수 조사하는 대신, 부트킷이 침투하는 핵심 섹터에만 분석 자원을 집중함으로써 지능형 사후조사의 효율성을 극대화한다.

2) 데이터 전처리기(Data Preprocessor)

데이터 전처리기는 원시 512바이트 바이너리를 Phi-3.5-mini가 처리할 수 있는 구조화된 입력으로 변환하는 두 단계 처리를 수행한다.

첫 번째 단계는 HEX 데이터 변환으로, 512바이트를 공백으로 구분된 16진수 문자열로 변환한다. 이 방식은 난독화 및 암호화 여부에 관계없이 전체 바이트를 정보 손실 없이 전달할 수 있다. 일부 부트킷은 난독화 기능을 포함하므로, 디스어셈블러가 바이트를 코드로 해석하지 못하는 상황이 발생할 수 있다. 실제로, 본 연구에서 ASM 도메인 모델의 한계는 그러한 점에서 기인한다. 반면, HEX 데이터는 이러한 난독화에 영향을 받지 않고 일정한 입력을 보장한다.

두 번째 단계는 챗 템플릿 구조화로, MBR 데이터 및 프롬프트를 Phi-3.5-mini의 공식 챗 템플릿 형식에 맞게 조립하여 최종 입력을 구성한다. 구체적으로는 역할 정의 및 출력 형식을 지시하는 '<|system|>' 토큰과 HEX 데이터를 담고 있는 '<|user|>' 토큰을 결합하고, '<|assistant|>' 토큰으로 모델의 응답 생성 위치를 지정한다.

한편, 입력 표현 방식의 선택이 탐지 성능에 미치는 영향을 실험적으로 검증하기 위해, Capstone 디스어셈블러를 사용한 ASM 변환도 대조군으로 포함하며, 그 결과는 4장에서 다룬다.

3) 경량 언어 모델 추론 엔진(SLM Inference Engine)

경량 언어 모델 추론 엔진은 제안된 프레임워크의 핵심 모듈로,각 영역(MBR, VBR, MFT, Slack)의 분석기는 동일한 Phi-3.5-mini 베이스 모델을 공유하되, 시스템 프롬프트와 파인튜닝 데이터셋이 영역별로 분리되어 해당 영역의 구조적 특성에 맞는 행동 패턴을 학습한다.

분석기는 QLoRA 기법으로 파인튜닝된 Phi-3.5-mini 모델을 사용하며, 프롬프트와 함께 입력된 HEX 데이터를 분석하여 BEHAVIOR, STRINGS, METADATA, VERDICT의 네 가지 항목으로 구조화된 보고서를 출력한다.

추론 시에는 결정론적 출력을 유도하기 위해 temperature를 0.1로 설정하고, 반복 생성 억제를 위해 repetition_penalty를 1.1로 적용한다.

모델의 학습 설정 및 데이터셋 구축 방법론은 3.3절과 3.4절에서 다루며, 보고서의 VERDICT 정확도에 대한 평가 방법은 4장에서 다룬다.

4) 보고서 생성기(Report Generator)

보고서 생성기는 경량 언어 모델 추론 엔진의 각 영역별 분석 보고서와 Diff Summary JSON을 종합하여 최종 결과 보고서를 생성한다. Diff Summary JSON에 기록된 변조 섹터 분포와 각 분석기의 VERDICT를 교차 검증함으로써, 단일 영역 분석의 오탐지(FP, False Positive)를 보완하고, 다영역에 걸친 감염 체계를 통합적으로 진단한다.

3-2 데이터 수집 파이프라인

1) 정상 MBR 샘플 수집

정상 MBR 샘플은 Digital Corpora에서 공개한 2009-m57-patents 시나리오의 E01 포렌식 이미지로부터 추출하였다. 해당 데이터셋은 디지털 포렌식 학계 및 공인 연구 기관에서 표준 벤치마크로 널리 인정받는 표준 데이터셋이다[7]. 특히 이 시나리오는 인위적으로 생성된 단일 샘플이 아닌, 가상의 기업 환경에서 사용자들이 수 주간 PC를 구동하며 발생시킨 정밀한 디스크 변화와 파일 시스템 구조를 포함하고 있다. 따라서, 정상 운영체제 MBR 상태를 정밀하게 대변한다는 장점이 있다.

해당 시나리오에서 내려받은 각 이미지를 pytsk3 라이브러리를 사용하여 파싱한 뒤 MBR 영역을 추출하여 총 19개의 정상 MBR 원시 데이터를 확보하였다.

2) 악성 MBR 샘플 수집(8종 부트킷)

악성 샘플은 VMware 환경에서 실제 악성코드를 실행하여 감염된 '.vmdk' 이미지를 'qemu-img'로 RAW로 변환한 뒤, MBR 영역 데이터를 추출하였다.

표 1은 수집된 8종 악성 샘플의 메타데이터 및 구조적 특징을 정리한 것이다. 엔트로피(Entropy) 열은 MBR 512바이트의 Shannon 엔트로피로, 값이 높을수록 데이터의 무작위성이 크며[8] 이는 데이터가 암호화 또는 난독화되었음을 시사한다. ASM 커버리지(ASM Coverage) 열은 Capstone 디스어셈블러가 코드로 해석할 수 있는 바이트의 비율이다. 핵심 행위 패턴(Core Behavioral Pattern) 열은 증거 추출 과정에서 해당 샘플에서 탐지된 주요 악성 행위 지표를 나타낸다.

표 1. 부트킷 샘플의 메타데이터 및 구조적 특징
Table 1. Metadata and structural features of bootkit samples

Malware	Entropy	ASM Coverage	Core Behavioral Pattern
Petya	6.11	100%	INT 13h(x6), XOR loop, RETF, Self-Relocation
Petya_2	2.65	100%	INT 13h(x2), INT 10h, CLI, HLT, Multi-stage sector loading
Satana	6.32	9.9%	PUSHAL → Self-Relocation →XOR decryption loop, High entropy
Mamba	5.97	3.4%	XOR AX, AX entry, High entropy code, Embedded error string
RedBoot	4.13	99.8%	INT 10h(x2), Ransom message output, Partition table destruction
Unknown_6B	5.73	99.8%	XOR 0x42 decoding loop, INT 13h(x2), Self-Relocation
Unknown_B6	6.03	99.8%	Multiple XOR loops(x6), INT 13h(x2), Partition table destruction
Unknown_F4	5.56	99.8%	XOR 0x42 decoding loop, INT 13h(x2), Self-Relocation

표 1과 같이, 8종 중 Petya, Petya_2, RedBoot, Unknown_6B, Unknown_B6, Unknown_F4 6종은 ASM 커버리지가 99.8~100%로 원시 바이트의 대부분이 코드로 해석된다. 반면, Satana와 Mamba는 ASM 커버리지가 각각 9.9%와 3.4%에 불과하여 디스어셈블러가 대부분의 MBR 바이트를 유효한 코드로 해석하지 못한다. 이러한 극단적인 커버리지 차이는 부트킷이 사용하는 난독화 및 암호화 수준의 다양성을 보여주며, 본 연구에서 ASM 방식 대신 HEX 기반 입력 방식을 제안하는 핵심적인 근거가 된다.

3-3 증거 기반 데이터셋 생성

1) 의미 보존 mutation 전략

8종의 악성 샘플과 19개의 정상 샘플에 mutation을 적용하여 총 12,816개의 데이터로 증강했다. 실제 부트킷은 시그니처 기반 탐지를 우회하기 위해 감염 시마다 코드를 변형하는 다형성(polymorphism) 및 메타모피즘(metamorphism) 기법을 사용하므로[9],[10], 학습 데이터에도 이러한 변형을 반영해야 모델의 탐지 능력을 확보할 수 있다. 악성코드의 기능을 보존하면서 바이너리를 변형하는 의미 보존 뮤테이션(semantics-preserving mutation) 방식은 기계학습 기반 탐지 모델의 강건성 향상에 효과적이다[11]. 이에 본 연구는 표 2의 4가지 mutation 기법을 적용하였다.

'NOP/Junk Insertion'은 코드 내 NULL 패딩 영역에 무의미한 바이트를 추가하는 기법으로, 악성코드가 실행 흐름을 변경하지 않으면서 바이너리 시그니처를 변형하는 dead-code insertion[9],[10]에 대응된다. 'Equivalent Opcode Substitution'은 동일한 연산 결과를 생성하는 다른 명령어(예: XOR EAX,EAX ↔ SUB EAX,EAX)로 대체하는 기법으로, 메타모픽 악성코드의 핵심 변형 기법인 Instruction Substitution[9]에 대응된다. 'JMP Offset Adjustment and Code Relocation'는 상대 점프 명령어의 오프셋을 변경하고 코드를 재배치하여 제어 흐름 그래프를 다변화하는 기법으로, 악성코드가 명령어 순서를 뒤트는 code transposition[9]에 대응된다. 'ASCII Case Toggling'은 MBR에 내장된 에러 메시지 문자열의 대소문자를 무작위로 치환하여 문자열 시그니처를 다변화하는 기법이다.

표 2. 적용된 mutation 기법
Table 2. List of applied mutation techniques

Mutation Technique	Target	Effect
NOP/Junk Insertion	Null padding areas	Padding-Level Diversification
Equivalent Opcode Substitution	XOR↔SUB, MOV(reg)↔MOV(mem)	Instruction-Level Diversification
JMP Offset Adjustment and Code Relocation	0xEB-series instructions and code blocks	Control-Flow-Level Diversification
ASCII Case Toggling	Embedded strings	Static-Data-Level Diversification

모든 변형에서 부트 시그니처(0x55 0xAA)는 보존하며, 파티션 테이블 영역(0x1BE~0x1FD)은 변형 대상에서 제외한다. SHA-256 해시를 통한 중복 검사를 수행하여 각 변종의 고유성을 보장하며 데이터셋을 구축한다.

2) 증거 추출 및 레이블링

데이터셋을 생성하는 데 있어서, 코드에 없는 행위가 정답지에 포함되어선 안 된다. 따라서, 각 데이터의 바이너리에 대해 Capstone 디스어셈블러를 적용하여 실제로 탐지 가능한 패턴만을 추출한다. 탐지 대상 패턴은 표 3과 같이 분류한다.

표 3. 디스어셈블 기반 탐지 패턴 분류
Table 3. Disassembly-based detection patterns

Classification Pattern	Pattern Characteristics
XOR_DECODE_LOOP	A pattern where LOOP/JNZ/JNE/DEC instructions exist within 5 instructions after an XOR instruction
INT_13H / INT_10H / INT_18H	BIOS interrupts for disk I/O, video, and boot failure
SELF_RELOCATE	Memory self-relocation utilizing REP MOVSB/MOVSW instructions
FAR_JUMP	Long-distance jumps using the LJMP / RETF / LRET instruction families
HIGH_ENTROPY_LOW_CODE	Entropy > 5.9 and disassembly coverage < 20%
UNUSUAL_ENTRY	Non-standard startup instructions that deviate from the standard MBR entry point (CLI/XOR/JMP)

'XOR_DECODE_LOOP'는 XOR 명령어 이후 5개 명령어 이내에 LOOP, JNZ, JNE, DEC 등의 반복 제어 명령어가 존재하는 패턴이다. 부트킷은 자신의 페이로드를 암호화하여 저장한 뒤, 실행 시점에 XOR 연산을 반복 수행하여 원본 코드를 복원하는 자기 복호화(self-decryption) 루틴을 사용한다[12]. 이는 정적 분석 및 시그니처 매칭을 우회하기 위한 핵심 기법으로, 본 연구의 8종 악성 샘플 중 Petya, Satana, Mamba, Unknown_6B, Unknown_B6, Unknown_F4의 6종에서 탐지되었다.

'INT_13H / INT_10H / INT_18H'는 BIOS 인터럽트를 호출하는 패턴이다. INT 13h는 BIOS 디스크 I/O 서비스로, 부트킷이 추가 섹터를 로딩하거나 디스크 데이터를 변조할 때 호출한다. 정상 MBR도 VBR 로딩을 위해 INT 13h를 사용하지만, 부트킷은 비정상적으로 많은 횟수로 호출하거나 비표준 파라미터를 사용하는 점에서 구분된다. INT 10h는 비디오 출력 인터럽트로, RedBoot처럼 랜섬 메시지를 화면에 출력하는 부트킷에서 탐지된다. INT 18h는 부트 실패 시 ROM BASIC을 호출하는 인터럽트로, 정상 부팅 흐름에서는 의도적으로 호출되지 않는다. 각 인터럽트의 호출 횟수와 위치를 함께 기록함으로써, 단순 존재 여부를 넘어 호출 맥락까

지 정답지 보고서에 반영한다.

'SELF_RELOCATE'는 REP MOVSB 또는 REP MOVSW 명령어를 사용하여 자기 자신의 코드를 다른 메모리 영역으로 복사하는 패턴이다. Legacy BIOS 환경에서 MBR은 0x7C00에 로드된 후, VBR을 동일한 주소에 적재하기 위해 이와 같은 재배치 메커니즘을 사용한다. 부트킷 역시 시스템의 제어권을 탈취한 뒤 정상적인 부팅 과정을 이어가기 위해 원본 부트 코드를 백업하고 이를 메모리에 로드하여 재사용하는 특징을 보인다[13]. 따라서, 'SELF_RELOCATE' 패턴은 그 자체로 악성 행위가 아닌, 타 패턴과 종합적으로 분석하기 위한 보조 지표로써 기능한다.

'FAR_JUMP'는 LJMP, RETF, LRET 계열의 원거리 점프 명령어를 사용하는 패턴이다. 일반적인 정상 MBR에서도 원거리 점프 패턴이 보이지만, 그 목적지가 정적으로 노출되어 있다. 반면 부트킷은 정적 분석기의 제어 흐름 그래프 구성을 방해하기 위해 명시적인 JMP 명령어 대신, 스택에 목적지 세그먼트와 오프셋을 동적으로 계산하여 Push한 뒤 RETF 명령어를 호출하는 기법을 활용한다. 이러한 스택 기반의 제어권 전환은 정적 분석 도구가 점프의 타겟을 특정하지 못하게 하여, 악성 페이로드의 실행 흐름을 은닉하는 난독화 수단으로 작용한다[14]. 따라서, 해당 패턴은 악성 영역으로 제어권을 넘기거나 정적 분석을 우회하려는 부트킷의 고의적인 은닉 시도를 탐지하는 주요 지표로 기능한다.

'HIGH_ENTROPY_LOW_CODE'는 MBR 512바이트의 Shannon 엔트로피가 5.9를 초과하면서 동시에 Capstone 디스어셈블리 커버리지가 20% 미만인 패턴이다. 높은 엔트로피는 데이터가 암호화 또는 압축되었음을 의미하고, 낮은 디스어셈블리 커버리지는 해당 바이트가 명령어로 해석되지 않음을 의미한다. 본 연구의 악성 샘플 중 Satana(엔트로피 6.32, 커버리지 9.9%)와 Mamba(엔트로피 5.97, 커버리지 3.4%)에서 해당 패턴이 관찰된다.

'UNUSUAL_ENTRY'는 MBR의 첫 번째 명령어가 CLI, XOR, JMP 등 표준 진입점 패턴이 아닌 비표준 명령어로 시작하는 패턴이다. 정상 MBR은 BIOS로부터 제어권을 인계받은 직후 인터럽트를 비활성화(CLI)하고 세그먼트 레지스터를 초기화(XOR)하는 것이 표준 절차이다[15],[16]. 이러한 표준 패턴에서 벗어난 비표준 진입 패턴은 부트킷이 자체적인 초기화 루틴을 실행하고 있음을 시사한다.

3) 데이터셋 구성

모델의 학습 데이터셋은 입력 표현 방식(HEX/ASM)과 지시문 및 출력 언어(KR/EN 프롬프트) 환경별로 구성된다. 이 중 'HEX KR' 데이터셋이 본 프레임워크의 핵심 학습 데이터이며, 'ASM KR'은 입력 표현 방식별 성능 비교를 위한 대조군으로, 'ASM EN'은 입출력 언어별 성능 비교를 위한 대조군으로 각각 활용된다.

이때, 본 실험 설계에서는 입력 표현 방식(HEX vs ASM)과 프롬프트 언어 환경(KR vs EN)이라는 두 가지 핵심 대조축을 효과적으로 검증하는 데 집중하기 위해, 'HEX EN' 데이터셋은 제외하였다. 기계어 코드 자체의 불변성을 고려할 때, 입력 방식의 우월성 검증은 국문 환경(HEX KR vs ASM KR)만으로 충분히 유효하며, SLM의 언어별 프롬프트 제어 특성과 포맷 안정성은 ASM 대조군(ASM KR vs ASM EN)을 통해 검증이 가능하기 때문이다. 따라서 불필요한 실험의 중복성을 배제하고 자원 및 실험 설계의 효율성을 극대화하기 위해, HEX 도메인은 'HEX KR'으로 통합 구축하였다.

각 도메인(HEX KR, ASM KR, ASM EN)의 학습 데이터는 감염과 정상 데이터의 비율을 1:1로 설정하여 각각 6,408건씩 구성한다. 모델의 성능 평가를 위한 검증 데이터 역시 1:1 비율로 각각 160건씩 학습 데이터와 중복 없이 구성한다.

표 4. 데이터셋 구성
Table 4. Dataset composition

Domain	Infected (Training)	Normal (Training)	Infected (Validation)	Normal (Validation)
HEX KR	6,408	6,408	160	160
ASM KR	6,408	6,408	160	160
ASM EN	6,408	6,408	160	160

또한, 모델이 그림 2와 같이 구조화된 출력을 갖도록 학습 데이터를 구성한다. 출력은 네 개의 섹션으로 구성되며, 각 섹션의 역할은 다음과 같다.

'BEHAVIOR' 섹션은 탐지된 행동 패턴을 열거한다. 각 항목은 패턴 유형과 해당 패턴이 관찰된 주소를 함께 기술한다.

'STRINGS' 섹션은 MBR 영역에서 발견된 6바이트 이상의 ASCII 문자열을 나열한다. 일부 부트킷은 공격자의 이메일 주소 등을 출력하기 위해 MBR 영역에 메시지를 저장한다. 이러한 MBR 영역의 메시지를 통해 코드의 의도를 추론하는 등 보조 증거로 활용이 가능하다.

'METADATA' 섹션은 바이너리의 구조적 메타데이터를 요약한다. 부트 시그니처(0x55 0xAA)의 유효 여부, 파티션 테이블에 등록된 엔트리 수, 코드 영역의 Shannon 엔트로피 값이 포함된다. 이 중 부트 시그니처 무효나 파티션 테이블 부재는 MBR 변조의 직접적 증거에 해당한다.

'VERDICT' 섹션은 최종 판정으로, 위 섹션에서 식별된 증거를 종합하여 '[TAMPERED]' 또는 '[CLEAN]'의 이진 태그를 부여한다.

```
BEHAVIOR:
    - [Detected Behavioral Pattern + Memory Address]
STRINGS:
    - [Discovered ASCII Strings]
METADATA:
    - Boot Signature: Valid(55 AA) / Invalid
    - Partition Table: N Entries
    - Code Entropy: X.XX
VERDICT: [TAMPERED|CLEAN]
```

그림 2. 모델 출력 구조
Fig. 2. Model output structure

이러한 4단 구조를 학습 데이터의 정답지에 포함하여, 모델은 해당 구조를 출력하도록 학습하며, 추론 시에도 동일한 형식의 보고서를 생성한다.

3-4 모델 학습

1) SLM 시리즈 비교 분석

본 연구에서는 네트워크가 차단된 폐쇄망 및 저사양 컴퓨팅 자원 등 제한된 환경에서의 포렌식 분석을 위해 Microsoft의 경량 언어 모델인 Phi 시리즈를 검토한다. MBR 바이너리 데이터는 구조가 매우 조밀하고 복잡한 논리 구조를 포함한다. 따라서, 단순한 분류 성능을 넘어 높은 추론 능력을 갖춘 모델 선정이 필수적이다. 표 5는 Phi 시리즈의 'mini' 클래스 모델별 주요 스펙을 비교한 것이다. Phi-3.5 모델이 Phi-3.0과 동일한 파라미터 규모를 유지하면서도 아키텍처 개선을 통해 추론 능력이 향상 되었다[17]. 그럼에도 8.0 GB 내외의 vRAM으로 학습이 가능하다. 이는 소비자용 GPU인 NVIDIA GeForce RTX 3080에서도 충분히 파인튜닝과 추론이 가능한 수치로, 본 연구의 목표인, 자원이 제한적인 환경에서의 포렌식 프레임워크에 적합하다.

표 5. Phi 시리즈 스펙 비교
Table 5. Comparison of Phi series specifications

Item	Phi-2.0-mini	Phi-3.0-mini	Phi-3.5-mini
Parameters	2.7B	3.8B	3.8B
Context Length	2K	4K / 128K	128K
Required vRAM	6.0 GB	8.0 GB	8.0 GB
Key Features	High-performance logical reasoning baseline	Cost-effective inference capability	Multilingual and enhanced reasoning

2) QLoRA 파인튜닝 설정

표 6은 대표적인 파인튜닝 기법의 스펙을 비교한 것이다. Full Fine-tuning은 모델 전체 가중치를 업데이트하므로 표현력이 가장 높으나, 3.8B 파라미터 모델 기준으로도 20GB 이상의 VRAM을 요구하여 본 연구의 RTX 3080(10GB) 환경에서는 적용이 불가하다. LoRA는 사전학습된 가중치를 동결하고 저순위 행렬만을 추가 학습하여 학습 파라미터를 전체의 1~3%로 축소하지만, 원본 모델을 16-bit로 유지해야 하므로 여전히 12~16GB의 VRAM이 필요하다. QLoRA는 LoRA에 4-bit NormalFloat 양자화와 이중 양자화를 결합하여, 모델 가중치의 메모리 점유를 약 1/4로 추가 절감한다. 이를 통해 동결된 가중치는 4-bit로 저장하면서 LoRA 어댑터만을 16-bit로 학습이 가능하여, 10GB VRAM 환경에서도 안정적인 파인튜닝이 실현된다. 따라서, 본 연구는 QLoRA 기법을 채택한다.

표 6. 파인튜닝 기법 스펙 비교
Table 6. Comparison of fine-tuning methods specifications

Item	Full Fine-tuning	LoRA (Low-Rank Adaptation)	QLoRA (Quantized LoRA)
Principle	Updates all weights	Trains low-rank matrices only	Applies LoRA + 4-bit quantization (NF4) and double quantization
Trainable Parameters	100% (All)	1% – 3%	1%
vRAM Usage	20 GB+	12 GB ~ 16 GB	10 GB

하이퍼파라미터 설정 시 RTX 3080(10GB VRAM) 환경에서의 제한된 자원과 모델 성능 간의 균형을 최우선으로 고려한다. LoRA Rank는 8로 설정하며, 대상 모듈은 Attention 계열(q, k, v, o)뿐만 아니라, Feed-Forward Network(gate, up, down) 전체로 확장한다. 이는 모델의 모든 선형 레이어에 LoRA를 적용하는 것이 Full Fine-tuning 성능에 가까워지기 때문이다[16]. 마지막으로 'train_on_responses_only' 옵션을 활성화하여 HEX/ASM 데이터는 Loss 계산에서 제외하고 보고서를 작성하는 방법을 학습하는 데 모든 자원을 집중시킨다.

Ⅳ. 모델 평가

4-1 입력 표현 방식별 성능 비교

1) 정량적 결과

표 7은 도메인별 이진 분류 평가 결과를 나타낸다. 본 실험에서는 MBR 데이터의 변조 여부를 기준으로 악성을 Positive로, 정상을 Negative로 정의하였다. 이에 따라 모델이 변조된 MBR을 악성으로 올바르게 식별한 경우를 True Positive(TP), 정상 MBR을 정상으로 판정한 경우를 True Negative(TN)로 분류하여 성능 지표를 산출하였다. 아울러, 모델이 악성 여부를 명확히 판단하지 못하였거나 지정된 구조로 응답하지 않은 사례는 'OTHER' 클래스로 분류하였다. Precision은 모델이 악성으로 판정한 MBR 중 실제 악성 MBR의 비율을 의미한다. Precision이 높을수록 정상 MBR을 악성으로 오탐지(False Positive, FP)하는 경우가 적음을 의미한다. Recall은 실제 악성 MBR 중에서 모델이 올바르게 악성으로 탐지해낸 비율을 의미한다. Recall이 높을수록 악성 MBR을 탐지하지 못하고 놓치는 미탐지(False Negative, FN)가 적음을 의미한다. F1-Score는 Precision과 Recall의 조화 평균으로, 모델의 전반적인 신뢰성을 나타낸다. Overall Accuracy는 악성 및 정상 전체 샘플에 대한 판정 정확도이다. OTHER 사례도 분모에 포함하여, 모델의 출력 안정성까지 반영한 종합 지표로 활용하였다.

표 7. 입력 데이터 도메인별 성능 비교
Table 7. Performance evaluation metrics by input data domains

Domain	TP	FP	FN	TN	OTHER	Precision	Recall	F1	Overall Accuracy
HEX KR	159	5	1	153	2	0.970	0.994	0.981	97.5%
ASM KR	115	37	45	108	15	0.757	0.719	0.737	69.7%
ASM EN	134	23	26	40	97	0.854	0.838	0.845	54.4%

HEX KR 도메인은 F1-Score 0.981, Overall Accuracy 97.5%로 세 도메인 중 가장 높은 성능을 기록했다. ASM KR은 F1 0.737, Overall Accuracy 69.7%로 중간 수준이며, ASM EN은 악성 탐지 자체의 F1-Score는 0.845로 준수하나 정상 샘플에 대한 포맷 붕괴로 인해 Overall Accuracy가 54.4%에 불과하다.

2) HEX 입력 우월성의 원인

HEX 입력이 ASM 대비 우수한 성능을 보이는 주된 원인은 난독화 및 암호화된 MBR 샘플에서의 ASM 커버리지 차이다. 표 1에서 확인했듯, Satana와 Mamba 샘플은 대부분의 바이트가 코드로 해석되지 않아 ASM 코드가 공백에 가까운 상태로 모델에 입력된다. 반면, HEX 데이터는 난독화 여부와 무관하게 512바이트 전체를 그대로 입력하므로, 특이한 바이트 분포를 모델이 직접 관찰할 수 있다.

3) 언어별 성능 비교

ASM 입력 도메인에서의 한국어(KR)와 영어(EN) 출력 모델을 비교하면, EN이 KR보다 악성 탐지 성능이 높은 반면, 정상 샘플 처리에서는 EN이 심각한 포맷 붕괴를 보인다. 표 8은 각 도메인별 정상 샘플에 대한 모델 출력 길이를 나타낸 것이다.

표 8. 정상 샘플에 대한 도메인별 출력 길이
Table 8. Comparison of output lengths by input domains for clean samples

Domain	OTHER	OTHER Avg. Output Length (Characters)	CLEAN Avg. Output Length (Characters)
HEX KR	2	336	343
ASM KR	14	491	550
ASM EN	92	1,005	710

ASM EN의 OTHER 사례가 정상적으로 CLEAN을 출력한 사례보다 유의미하게 길게 생성된 것을 확인할 수 있다. ASM KR의 OTHER 응답은 CLEAN 응답과 유의미한 길이 차이가 없으며, 대부분이 정상적으로 CLEAN을 출력했다.

ASM EN의 OTHER 응답을 분석한 결과, BEHAVIOR 또는 STRINGS 항목을 기술한 이후 의미 없는 대소문자 혼합 텍스트(예: "OpeRATin GEC₩₩nLOading OperatiNG sysTem")를 반복적으로 생성하다 최대 토큰 한도에 도달하여 결론을 도출하지 못하고 종료되는 패턴이 확인됐다. 반면, ASM KR의 OTHER 14건 중 다수는 METADATA 항목 이후 환각적 문장이 짧게 삽입된 뒤 종료되어, 환각의 규모에서 뚜렷한 차이를 보인다.

V. 고 찰

5-1 HEX 입력의 우월성

본 연구는 SLM을 이용한 원시 바이너리 학습에서, 전처리 과정을 거치지 않은 HEX 데이터가 디스어셈블리를 거친 ASM 데이터보다 효과적인 입력임을 실험적으로 확인했다. HEX KR은 F1 0.981을 기록한 반면, ASM KR은 F1 0.737에 머물렀다. 이 차이의 근본 원인은 디스어셈블리 과정에서 발생하는 입력 데이터의 정보 손실에 있다.

디스어셈블리는 바이너리를 인간이 읽을 수 있는 표현으로 변환하는 손실 과정이다. 3.2.2절에서 확인하였듯, Satana와 Mamba는 MBR 자체가 암호화되어 있어, 디스어셈블러가 대부분의 바이트를 코드로 해석하지 못하여, ASM Coverage가 낮게 나온다. 결과적으로 ASM 입력 시 공백에 가까운 시퀀스가 모델에 전달되며, 모델이 데이터에서 관찰할 수 있는 특징이 부족해진다. 이 경우 고엔트로피 패턴(HIGH_ENTROPY_LOW_CODE)을 명시하더라도, 원시 바이트 정보의 부재를 해결할 수 없다.

반면, HEX 데이터는 난독화 및 암호화 여부에 관계없이 512바이트 전체를 손실 없이 모델에 전달하므로, 특이한 바이트 분포를 모델이 직접 관찰할 수 있다.

5-2 저사양 환경 적합성 검증

본 연구의 목표는 하드웨어 자원이 제한된 환경에서도 독립적으로 구동 가능한 온-디바이스 기반의 부트킷 감염 사후 조사 프레임워크를 실현하는 것이다. 실험 결과, 제안하는 프레임워크는 모델 학습 및 추론 과정에서 약 8GB 내외의 VRAM만을 점유하여 10GB 환경에서도 안정적인 동작이 가능함을 입증하였다.이는 단순 추론을 위해서만 A100(80GB) 4장 이상의 막대한 서버 인프라를 요구하는 GPT-4급 대규모 언어 모델과 극명한 대조를 이룬다. 결과적으로 본 연구는 외부 클라우드 API 의존에 따른 보안 데이터 유출 우려를 원천 차단하는 동시에, 저비용 환경에서도 즉각적이고 신뢰할 수 있는 실무 포렌식 분석이 가능함을 확인하였다.

Ⅵ. 결론 및 향후 연구

본 연구는 RTX 3080급 소비자용 GPU에서 동작 가능한 SLM 기반 부트킷 감염 사후조사 프레임워크를 설계하고, 프레임워크의 핵심 모듈인 MBR 분석기를 구현 및 평가하였다. HEX 기반의 한국어로 구성된 데이터를 QLoRA 기법으로 학습한 Phi-3.5-mini 모델이 F1-Score 0.981과 준수한 포맷 안정성을 보여줬다. 그러나, 본 연구의 실제 구현과 실험적 검증은 핵심 모듈인 MBR 분석기에 한정하여 진행되었다. 실제 고도화된 부트킷은 디스크의 다양한 영역에 페이로드를 분산 은닉하므로, 본 연구에서 제안한 전체 통합 아키텍처의 유기적인 작동 신뢰성과 MBR 외 타 영역 분석 모듈의 실효성을 입증하는 데에는 한계가 있다.

본 연구의 한계점을 극복하고 프레임워크를 실전 수준으로 발전시키기 위해, MBR에 한정된 검증 범위를 부트킷 침투 가능성이 있는 디스크 영역 전반(MBR, VBR, MFT, Slack 등)으로 확장할 계획이다. 또한, 디스크 전반 분석이 적용된 경우, 개별 영역 분석 보고서를 입력으로 받아 최종 악성 판정을 내리는 Stage 2 진단기를 구현할 계획이다. 해당 진단기는 각 보고서의 VERDICT와 BEHAVIOR 항목을 교차 분석하여, 단일 영역 분석기의 오탐지를 상호 보완하는 방식으로 동작하는 것을 목표로 한다.

감사의 글

본 연구는 2026년 과학기술정보통신부 및 정보통신기획평가원의 SW중심대학사업의 연구결과로 수행되었음(2022-0-01077)

참고문헌

[1] P. Kleissner, “Stoned Bootkit,” in *Proceedings of Black Hat USA 2009*, Las Vegas: NV, July 2009. https://blackhat.com/presentations/bh-usa-09/KLEISSNER/BHUSA09-Kleissner-StonedBootkit-PAPER.pdf

[2] Wired. The Untold Story of NotPetya, the Most Devastating Cyberattack in History [Internet]. Available: https://www.wired.com/story/notpetya-cyberattack-ukraine-russia-code-crashed-the-world/.

[3] S. Jang, M.-R. Lee, and T.-S. Shon, “A LoRA-Optimized SLM-Based Framework for Malware Detection and Behavior Analysis in Closed Network Environments,” *Journal of Digital Forensics*, Vol. 19, No. 4, pp. 1-20, 2025.

[4] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, ... and W. Chen, “LoRA: Low-Rank Adaptation of Large Language Models,” in *Proceedings of the 10th International Conference on Learning Representations*, Virtual Event, April 2022. https://doi.org/10.48550/arXiv.2106.09685

[5] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer, “QLoRA: Efficient Finetuning of Quantized LLMs,” in *Proceedings of the 37th International Conference on Neural Information Processing Systems*, New Orleans: LA, pp. 10088-10115, December 2023. https://doi.org/10.48550/arXiv.2305.14314

[6] B. Grill, C. Platzer, and J. Eckel, “A Practical Approach for Generic Bootkit Detection and Prevention,” in *Proceedings of the 7th European Workshop on Systems Security*, Amsterdam, Netherlands, pp. 1-6, April 2014. https://doi.org/10.1145/2592791.2592795

[7] S. Garfinkel, P. Farrell, V. Roussev, and G. Dinolt, “Bringing Science to Digital Forensics with Standardized Forensic Corpora,” *Digital Investigation*, Vol. 6, pp. S2-S11, September 2009. https://doi.org/10.1016/j.diin.2009.06.016

[8] C. E. Shannon, “A Mathematical Theory of Communication,” *The Bell System Technical Journal*, Vol. 27, No. 3, pp. 379-423, July 1948. https://doi.org/10.1002/j.1538-7305.1948.tb01338.x

[9] I. You and K. Yim, “Malware Obfuscation Techniques: A Brief Survey,” in *Proceedings of the 2010 International Conference on Broadband, Wireless Computing, Communication and Applications*, Fukuoka, Japan, pp. 297-300, November 2010. https://doi.org/10.1109/BWCCA.2010.85

[10] W. Wong and M. Stamp, “Hunting for Metamorphic Engines,” *Journal in Computer Virology*, Vol. 2, No. 3, pp. 211-229, 2006. https://doi.org/10.1007/s11416-006-0028-7

[11] M. D. Wong, E. Raff, J. Holt, and R. Netravali, “Marvolo: Programmatic Data Augmentation for Practical ML-Driven Malware Detection,” in *Proceedings of the KDD Workshop on AI4Cyber*, Washington, DC, August 2022. https://doi.org/10.48550/arXiv.2206.03265

[12] M. Christodorescu and S. Jha, “Static Analysis of Executables to Detect Malicious Patterns,” in *Proceedings of the 12th Conference on USENIX Security Symposium*, Washington, DC, pp. 169-186, August 2003.

[13] B. Grill, A. Bacs, C. Platzer, and H. Bos, “"Nice Boots!" - A Large-Scale Analysis of Bootkits and New Ways to Stop Them,” in *Proceedings of the 12th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, Milan, Italy, pp. 25-45, July 2015. https://doi.org/10.1007/978-3-319-20550-2_1

[14] C. Linn and S. Debray, “Obfuscation of Executable Code to

Improve Resistance to Static Disassembly," in *Proceedings of the 10th ACM Conference on Computer and Communications Security*, Washington, DC, pp. 290-299, October 2003. https://doi.org/10.1145/948109.948149

[15] D. Starman. Examination of the Standard Master Boot Record [Internet]. Available: https://thestarman.pcministry.com/asm/mbr/STDMBR.htm#CODE.

[16] D. Starman. Windows 7, 8 or 10 Master Boot Record (MBR) [Internet]. Available: https://thestarman.pcministry.com/asm/mbr/W7MBR.htm#CODE.

[17] Microsoft. Discover the New Multi-Lingual, High-Quality Phi-3.5 SLMs [Internet]. Available: https://techcommunity.microsoft.com/blog/azure-ai-foundry-blog/discover-the-new-multi-lingual-high-quality-phi-3-5-slms/4225280.

장혁수(Hyeoksu Jang)

2022년～현　재: 아주대학교 사이버보안학과 학사과정

※관심분야 : Cyber Security, Digital Forensics, Network 등

손태식(Taeshik Shon)

2000년 : 아주대학교 정보및컴퓨터공학부 졸업(학사)

2002년 : 아주대학교 정보통신전문대학원 졸업(석사)

2005년 : 고려대학교 정보보호대학원 졸업(박사)

2004년～2005년: University of Minnesota 방문연구원

2005년～2011년: 삼성전자 통신·DMC 연구소 책임연구원

2017년～2018년: Illinois Insitute of Technology 방문교수

2011년～현　재: 아주대학교 정보통신대학 사이버보안학과 교수

※관심분야 : Digital Forensics, ICS/Automotive Security