국민대학교
소프트웨어융합대학
소프트웨어학부

캡스톤 디자인 I

종합설계 프로젝트

프로젝트 명	Platform GIPS - 생성형 침입 방지 기술기반 보안 플랫폼
팀 명	34조
문서 제목	수행결과보고서

Version	1.3
Date	2024-05-21

팀원	엄석현 (조장)
	김태경
	김태윤
	박준서

문서 정보 / 수정 내역

Filename	T34_결과보고서.docx
원안작성자	엄석현, 김태경, 김태윤, 박준서
수정작업자	엄석현, 김태경, 김태윤, 박준서

수정날짜	대표수정자	Revision	추가/수정 항목	내 용
2024-05-18	엄석현	1.0	최초 작성	최초 작성
2024-05-19	김태윤	1.1	내용 추가	가이드 추가
2024-05-20	김태경	1.2	내용 추가	결과물 내용 추가
2024-05-21	박준서	1.3	내용 추가	시스템 구성도 내용 추가

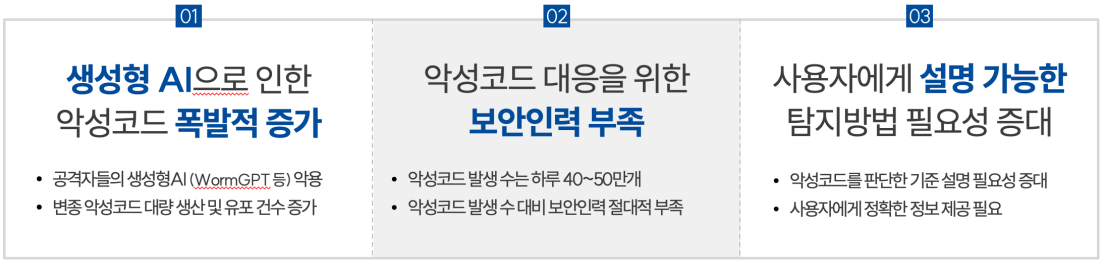
목 차

1 개요

1.1 프로젝트 개요

생성형 침입 방지 기술 (GIPS: Generative Intrusion Detection and Prevention on data Stream)[1] 알고리즘을 기반으로 악성 PE파일에서 시그니처를 추출하여 탐지 규칙 중 하나인 Yara rule을 생성합니다. 이러한 Yara Rule을 이용하여 다양한 분석 툴에 활용할 수 있도록 합니다. 또한 새로운 파일에 대해서 기존에 있던 시그니처와 비교하여 악성과 정상을 판별하고 이 데이터를 저장하여 시각화하여 Web 플랫폼을 구성하여 보안 전문가들이 파일을 분석하는데 시간을 줄일 수 있게 합니다. 더 나아가 보안 전문 기업인 (주)누리랩과 협력하여 연구 및 개발을 진행했습니다.

1.2 추진 배경 및 필요성



현재 생성형 AI의 발전으로 wormGTP, fraudGPT와 같은 프로그램을 통해 손쉽게 악성 코드를 만들고 유포할 수 있게 되었습니다. 그로인해 악성코드의 수와 종류가 급증하고 있는 반면 악성코드를 대응하기 위한 보안 인력은 절대적으로 부족한 상황입니다. 현재 AI를 활용한 악성코드 탐지를 많이 시도하려고 하고 있지만 아직 사용자에게 악성코드를 탐지한 이유를 설명할 수 없기에 사용 가능한 탐지 방법이 아닙니다. 그렇기 때문에 악성코드의 판단 기준을 제시하고 어떠한 이유로 탐지가 되었는지 사용자에게 정보를 제공해야하기 위해서 이 프로젝트를 진행하게 되었습니다.

2 개발 내용 및 결과물

2.1 목표

1. 다중의 악성 파일에서 GIPS를 이용하여 시그니처를 뽑아내는 기술을 개발합니다.
2. 시그니처를 가지고 Yara Rule을 만들어 웹페이지에서 생성된 룰을 보여주고 이 후 작업으로 DB에 업로드 하거나 PC에 다운로드를 제공합니다.
3. 만들어진 Yara Rule을 가지고 새로운 파일이 들어왔을 때 악성 시그니처를 탐지합니다.
4. 분석 할 파일이 들어왔을 때 어떤 시그니처가 나왔는지, 몇 개의 공격 시그니처들이 들어있는지 등의 자세한 정보를 보여줍니다.
5. 저장된 결과들을 통해 어떤 시그니처들이 공격에 자주 사용되었는지, 공격 파일과 정상 파일의 비율은 어떠한지와 같은 정보를 시각적으로 보여줍니다.

2.2 연구/개발 내용 및 결과물

2.2.1 연구/개발 내용

1. 연구/알고리즘 개발

PE파일에서 문자열 집합을 추출하기 위해서 파일을 바이트 단위로 읽고 그 중에 아스키 코드 값으로 바뀌었을 때 사람이 알아볼 수 있는 문자열로 변환했습니다. 그중에서 최소 6바이트 즉 6글자이상인 문자열만 사용하고 대소문자를 구분하지 않기 위해서 전부 소문자로 처리했습니다.

기존 GIPS의 입력은 긴 문자열이었기 때문에 청킹을 이용하여 시그니처를 생성하는 반면에 현재 기술에서는 먼저 시그니처가 될 수있는 후보 문자열을 집합으로 만들어져 입력이 들어오기 때문에 GIPS 청킹 부분을 삭제하고 집합을 처리할 수 있게 알고리즘 수정했습니다.

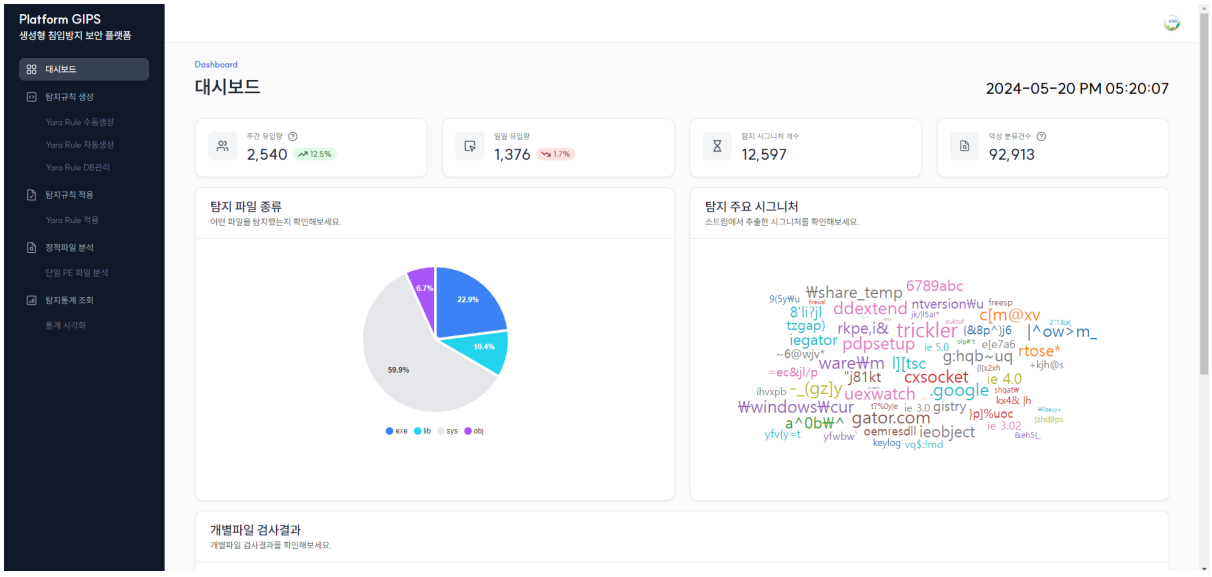
기존의 GIPS는 stopword를 이용하여 시그니처가 되지 않는 문자열을 처리했습니다. 하지만 현재 기술에서는 문자열 집합을 입력으로 받기 때문에 미리 정상 파일에서 등장하는 문자열을 이용하여 화이트 리스트를 만들어놓고 GIPS알고리즘이 적용 되기 전 단계에서 전처리로 시그니처가 되지 않는 문자열을 처리했습니다.

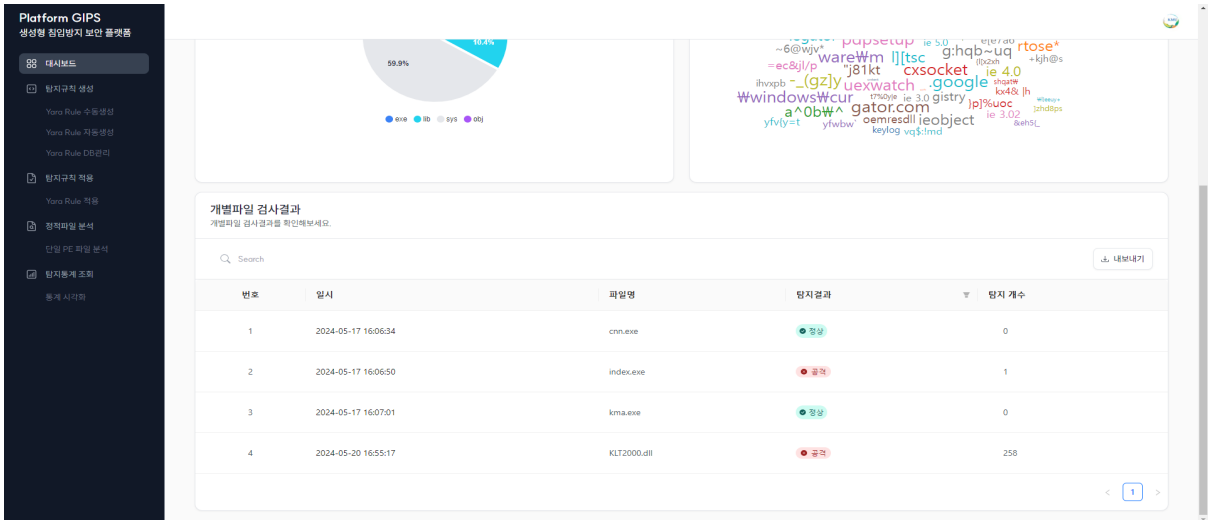
2. 프론트엔드

2.1. 로그인 화면

- 사이트 최초 접속시 로그인을 할 수 있는 화면
- /로 접속할 수 있다.

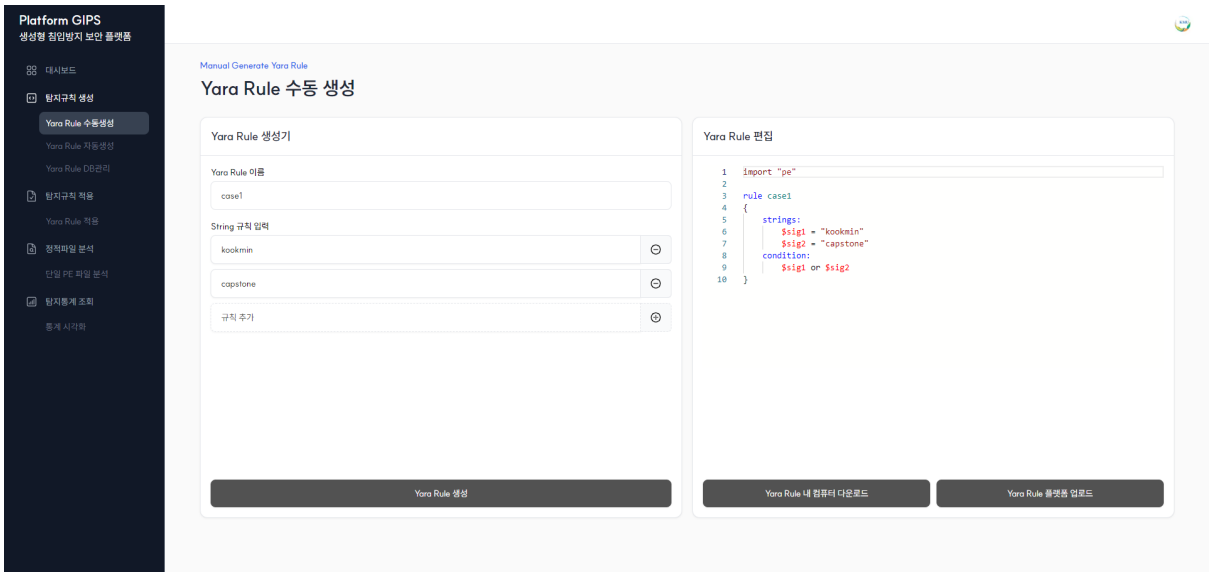
2.2. 대시보드 화면





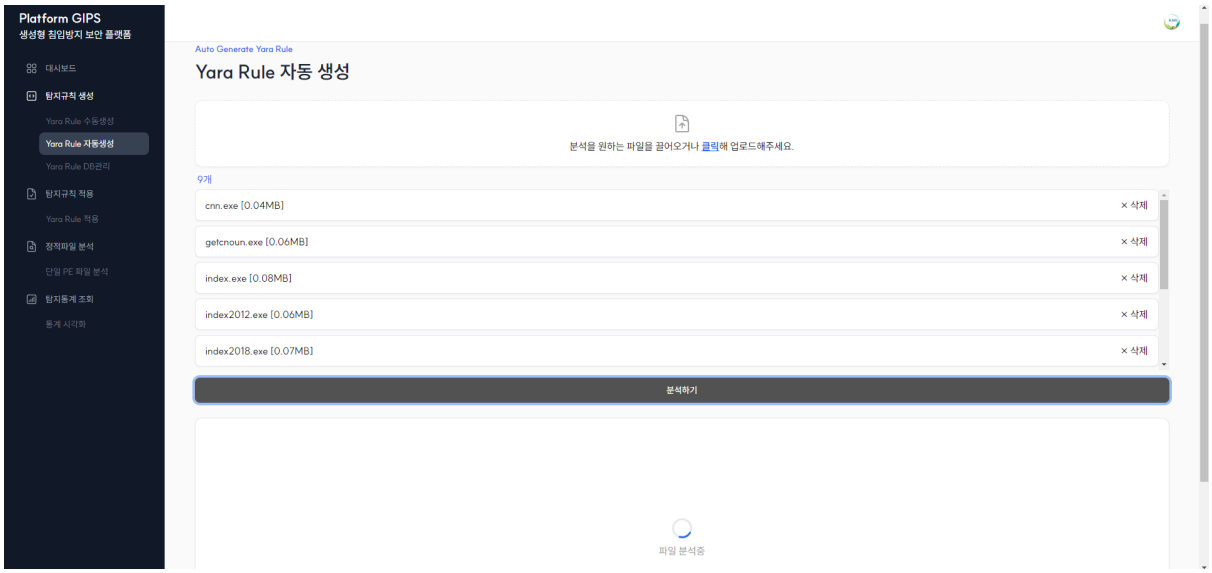
- /dashboard 주소로 접속 가능하다.
- 사이트의 모든 페이지에서 좌측 상단의 로고를 클릭 시 접속 가능하다.
- 사이트의 유입량과 서버에 저장된 시그니처 개수 및 분류 건수를 확인 가능하다.
- 서버에 저장된 단일 파일 분석 결과들을 확인할 수 있다.

2.3. Yara Rule 수동 생성 화면

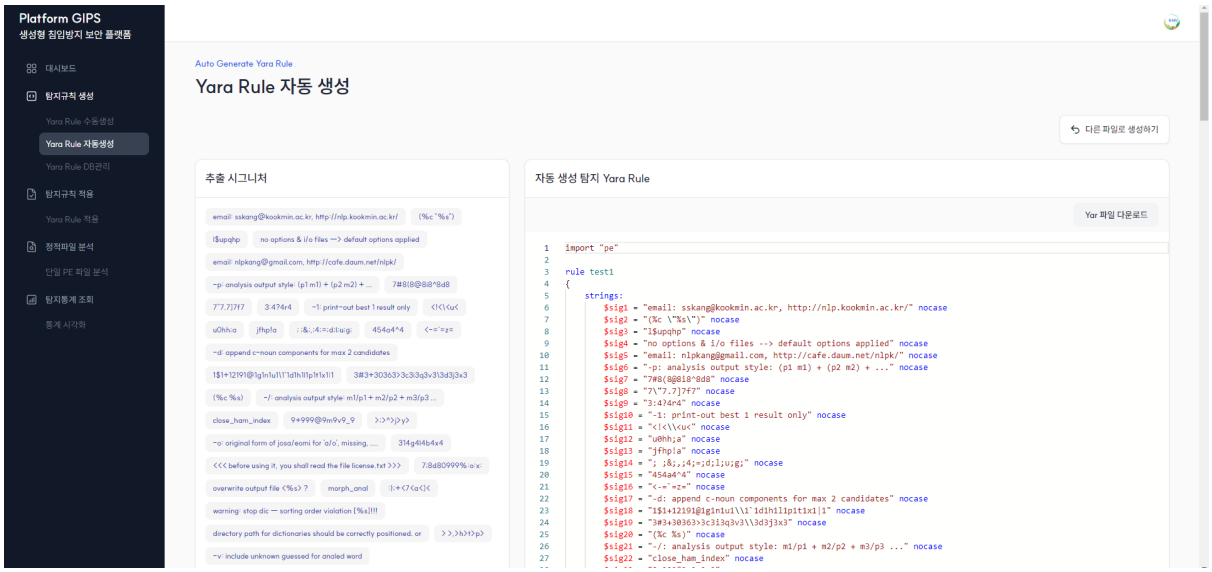


- /generate/rule/yara/manual 주소로 접속 가능하다.
- 사용자가 생성하고 싶은 이름과 규칙을 적고 생성시 새로운 Yara Rule을 만들 수 있다.
- 생성된 Yara Rule은 사용자의 컴퓨터에 다운받거나 서버에 저장할 수 있다.

2.4. Yara Rule 자동 생성 화면

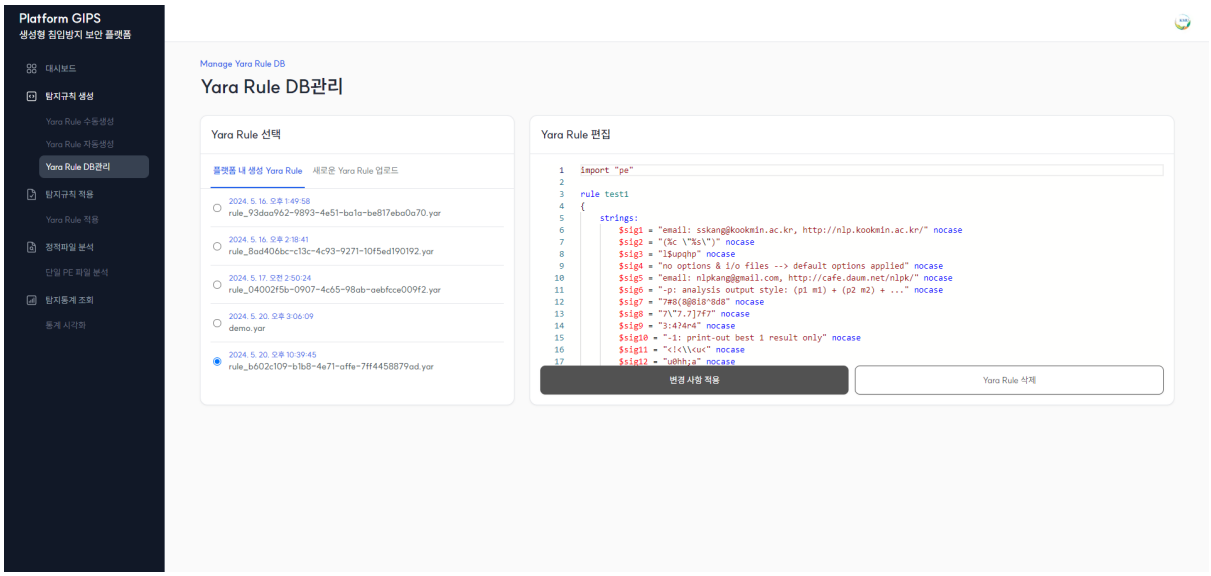


- /generate/rule/yara/auto 주소로 접속 가능하다
- 사용자가 원하는 다수의 파일을 업로드해 파일들의 공통된 시그니처를 사용해 새로운 Yara Rule을 생성할 수 있다.



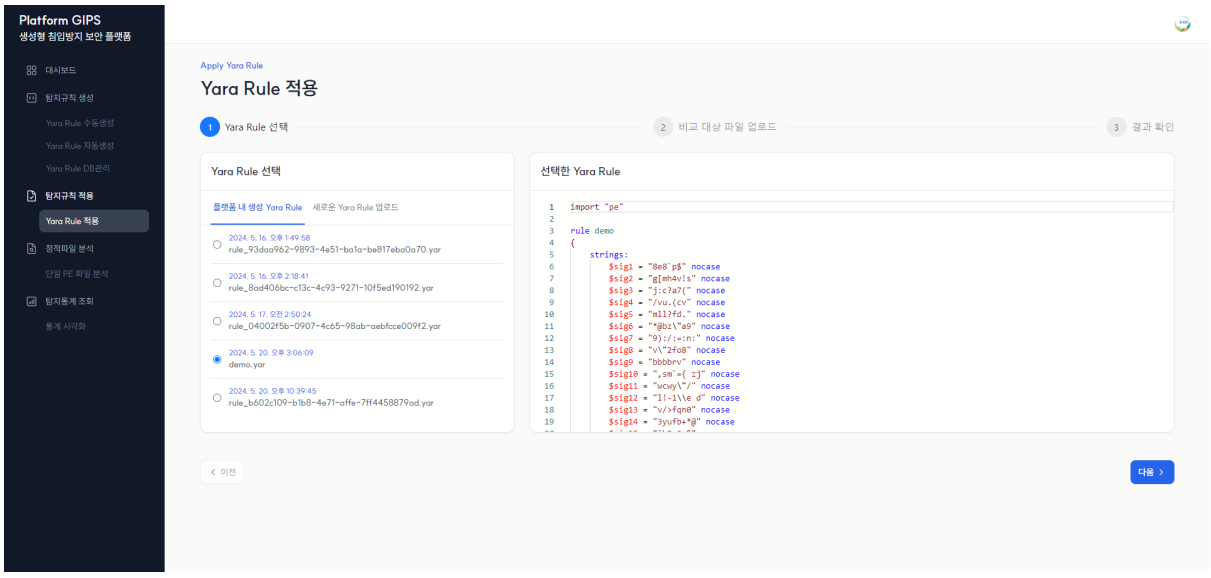
- 생성된 Yara Rule에 적용된 추출 시그니처들을 확인할 수 있다.
- 생성된 Yara Rule의 코드를 확인하고 다운로드 받을 수 있다.
- 생성된 결과는 자동으로 DB에 저장된다.

2.5. 탐지규칙 DB 관리 화면



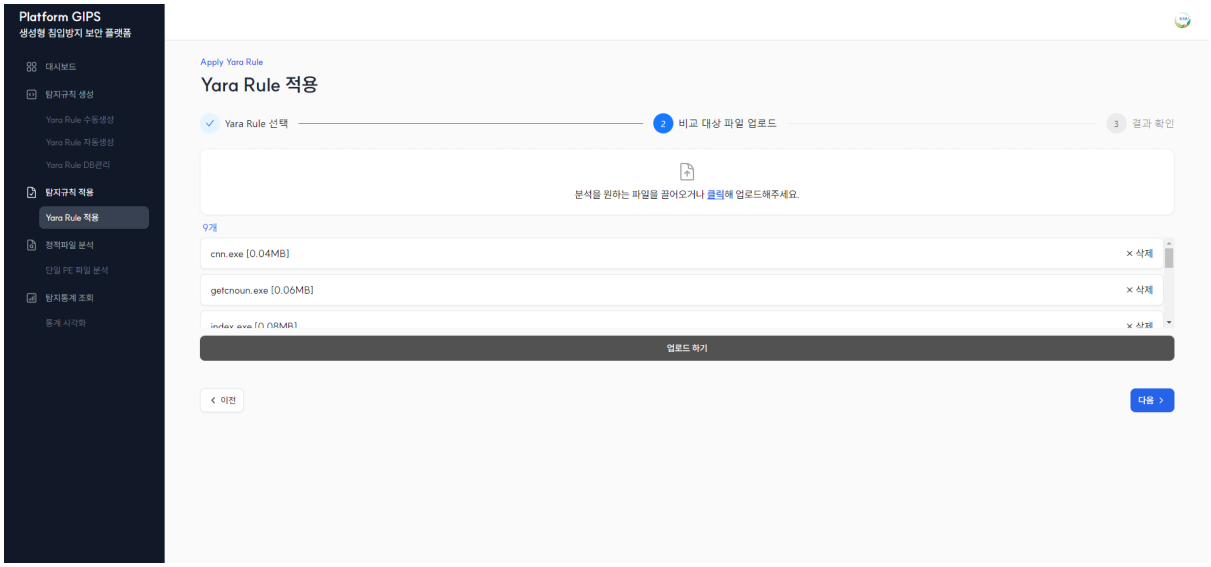
- /generate/rule/yara/manage 주소로 접속할 수 있다.
- 서버에 저장된 Yara Rule의 목록을 확인할 수 있다.
- 선택된 Yara Rule을 서버에서 변경하거나 삭제할 수 있다.
- *.yar파일을 사용하여 서버에 직접 Yara Rule을 업로드 할 수 있다.

2.6. 탐지규칙 적용 화면

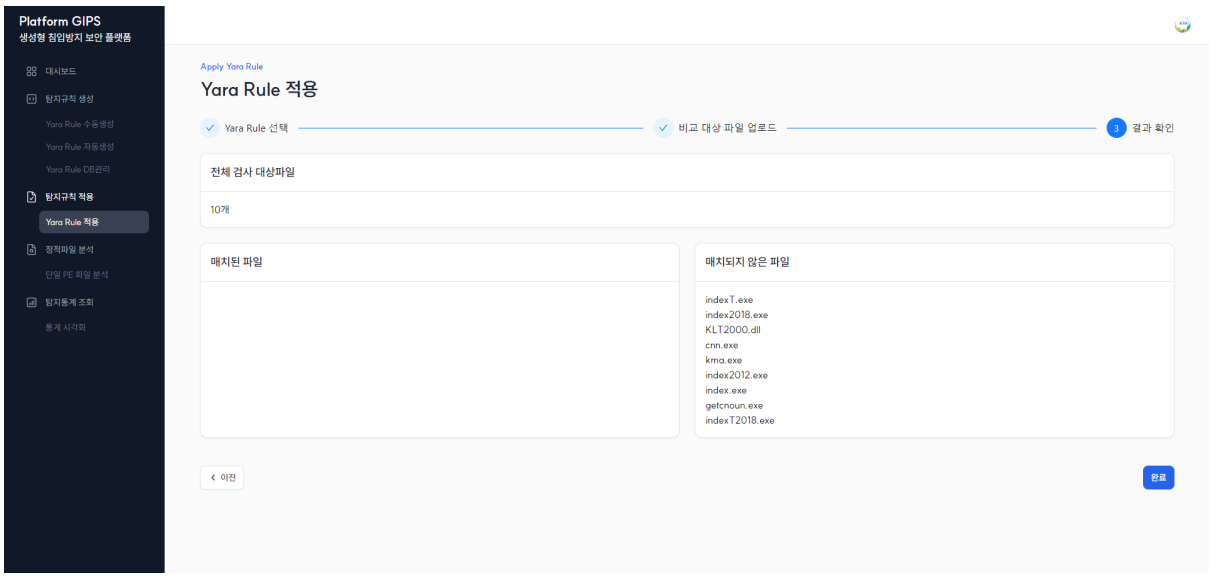


- /apply/rule/yara 주소를 통해 접속할 수 있다.

- 사용자가 서버에 저장되어있는 Yara Rule을 선택하고 세부 사항을 확인할 수 있다.
- 각 단계는 완료시 다음으로 넘어갈 수 있으며, 이전으로 돌아가 전 단계를 다시 수행할 수 있다.

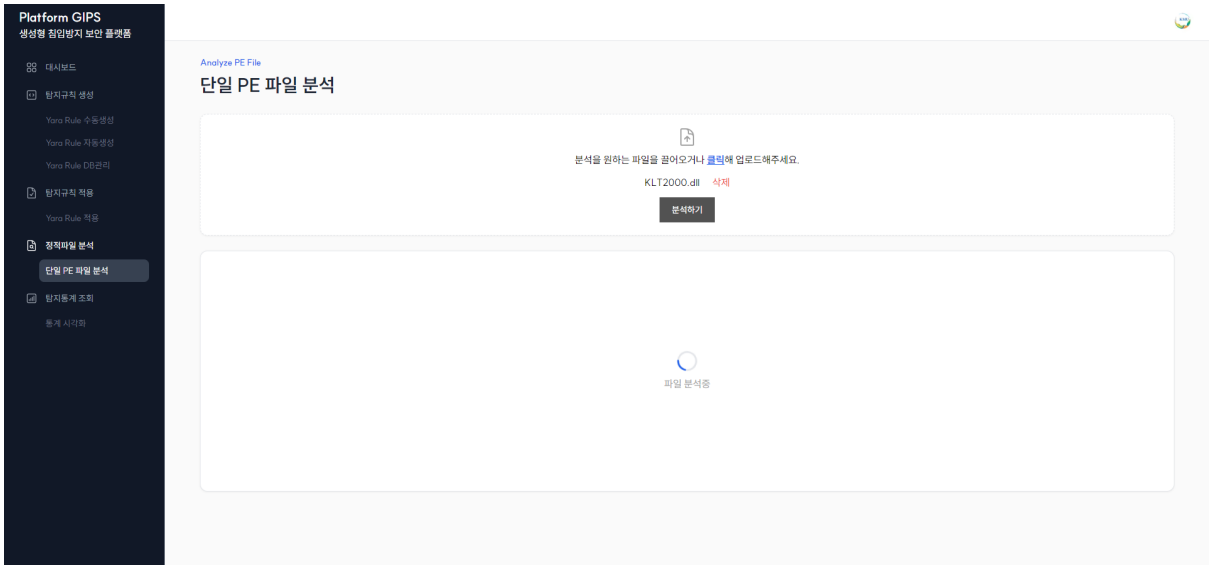


- 사용자가 원하는 다수의 파일을 업로드하여 선택한 Yara Rule에 분석할 수 있다.

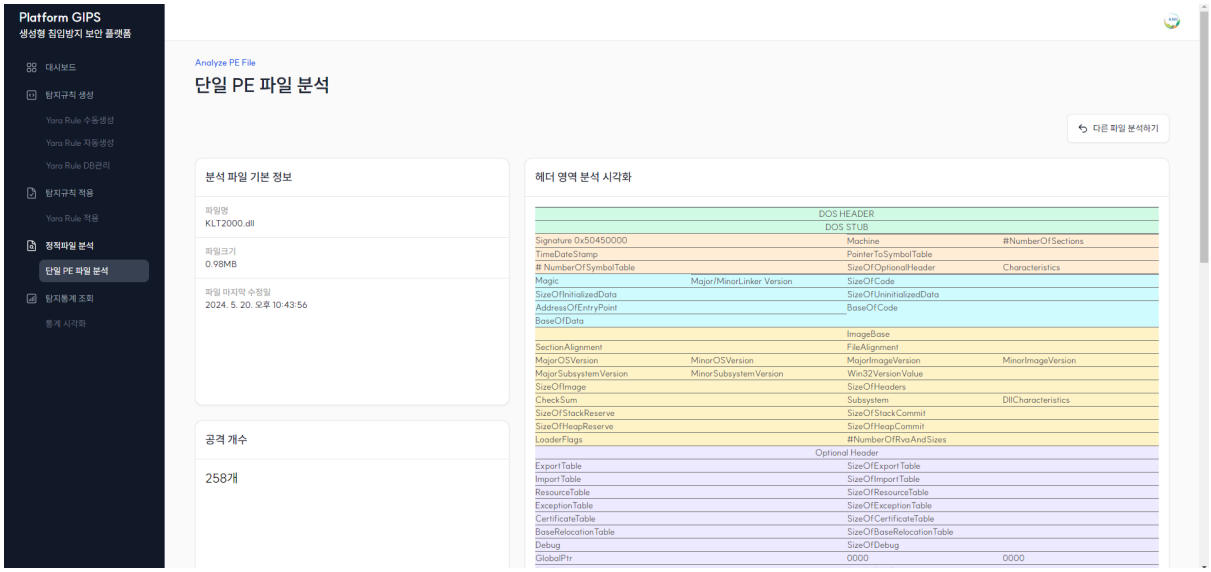


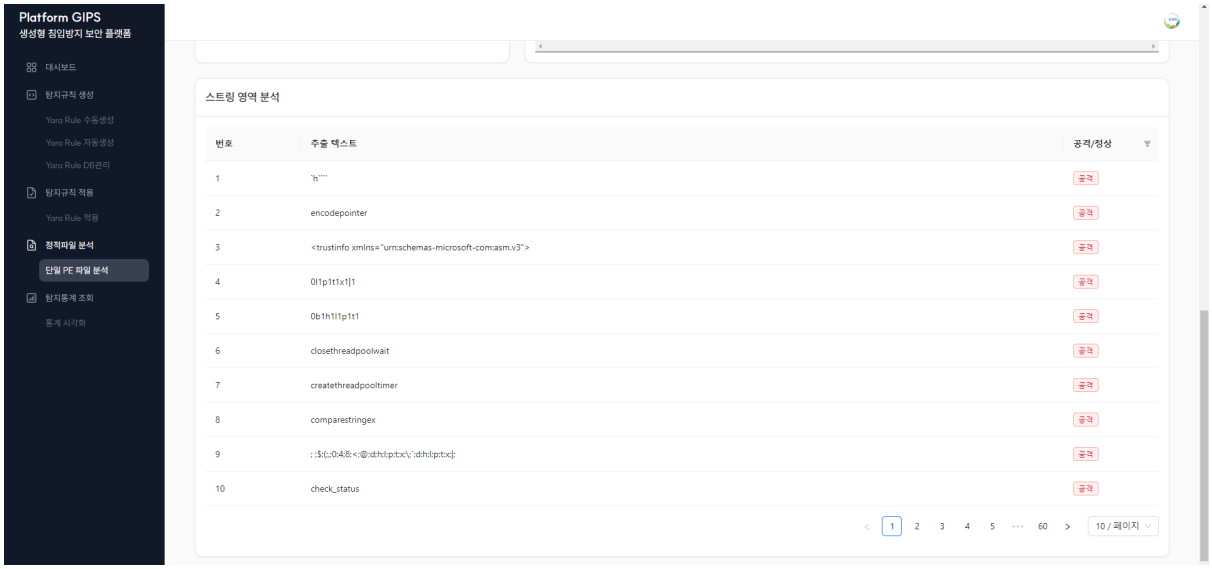
- 분석된 파일들의 결과를 확인할 수 있다.
- 결과 화면은 사용자가 선택한 규칙에 매칭되는 파일 목록과 매칭되지 않는 파일 목록으로 구분된다.

2.7. 단일 PE 파일 분석 화면



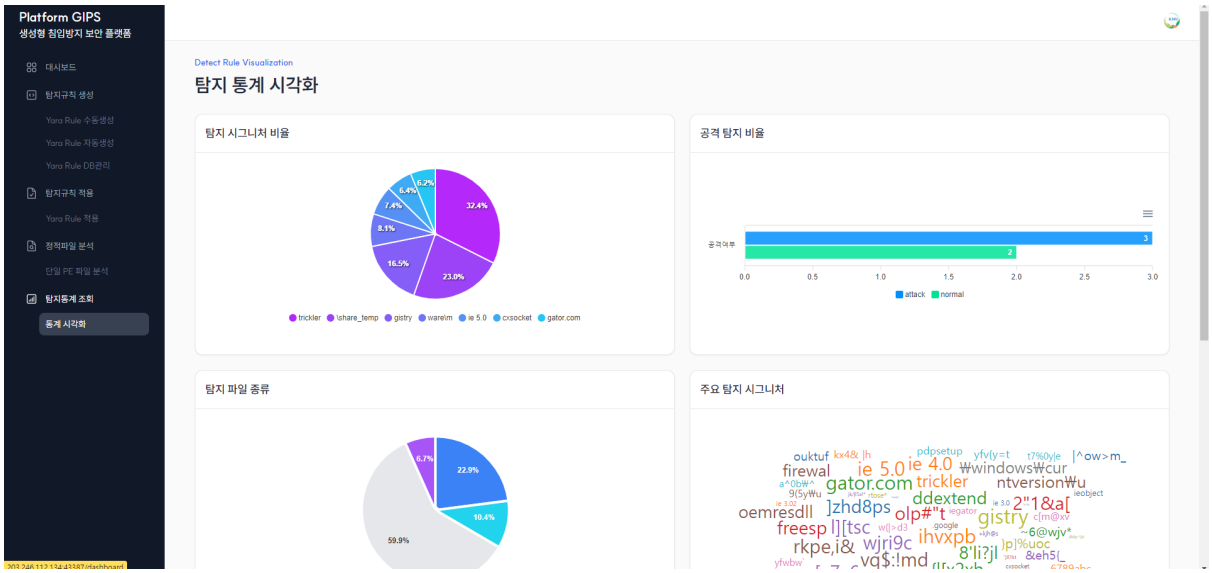
- /analyze/file/pe 주소로 접속 가능하다.
- 사용자가 분석을 원하는 단일 파일을 업로드해 결과를 확인할 수 있다.
- 다른 파일 분석을 원할 시 “다른 파일 분석하기” 버튼을 통해 이전 화면으로 돌아갈 수 있다.

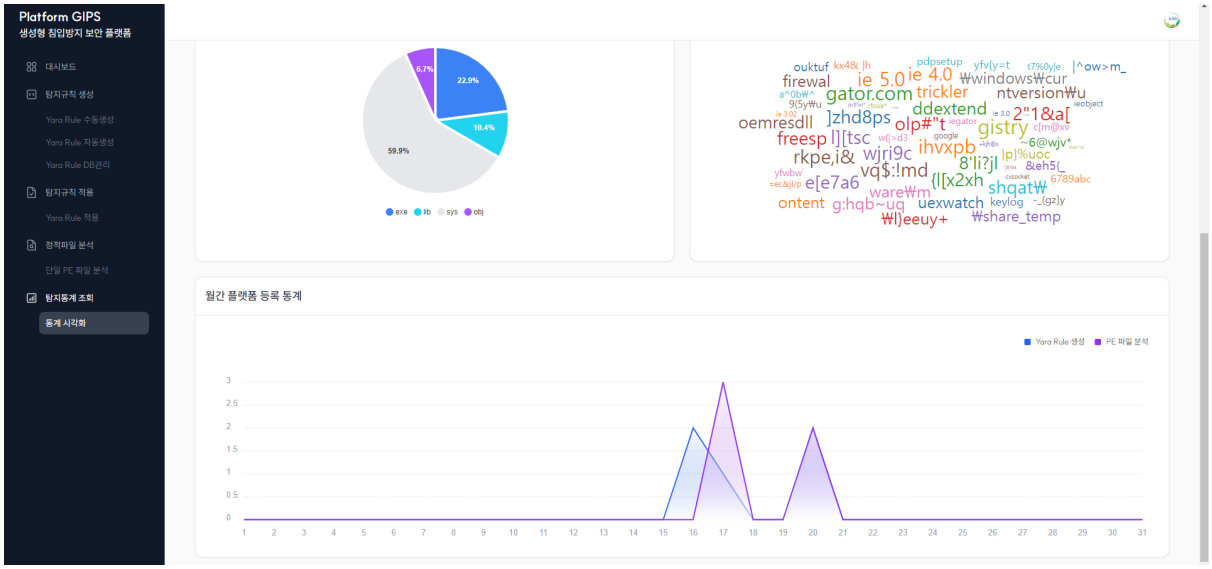




- 분석이 완료된 파일은 파일의 기본 정보, 파일의 pe헤더 정보, 탐지된 공격 시그니처 개수, 분석된 모든 스트링 시그니처와 공격 여부를 확인할 수 있다.
- 파일 분석 결과는 DB서버에 저장된다.

2.8. 통계 시각화 화면





- /stats/chart 주소로 접속 가능하다.
- 사이트를 통해 분석된 파일들의 공격/정상 비율을 확인할 수 있다.
- 공격에 사용된 대표 시그니처들의 종류와 비율을 볼 수 있다.
- 분석에 사용된 파일의 확장자 별 분포를 확인할 수 있다,
- 사이트 이용자가 분석/생성한 파일의 수를 일 단위로 확인할 수 있다.

3. 백엔드

- prisma와 SQLite를 사용하여 구현하였음
- 분석된 단일 파일의 정보를 저장하는 Analysis 테이블, 자동/수동 생성된 Yara Rule을 저장하는 YaraRule 테이블, 유저 정보를 저장하는 User 테이블, 탐지에 사용되는 시그니처를 저장하는 Signature 테이블로 구성되어 있다,
- Analysis 테이블과 YaraRule 테이블은 사용자가 사이트의 기능을 이용시에 자동/수동으로 결과를 저장하여 보여주는 역할을 하고있다.
- User 테이블에 저장되는 사용자의 비밀번호는 해쉬화 되어 저장되어 보안성을 높이고 있다.

2.2.2 시스템 기능 요구사항

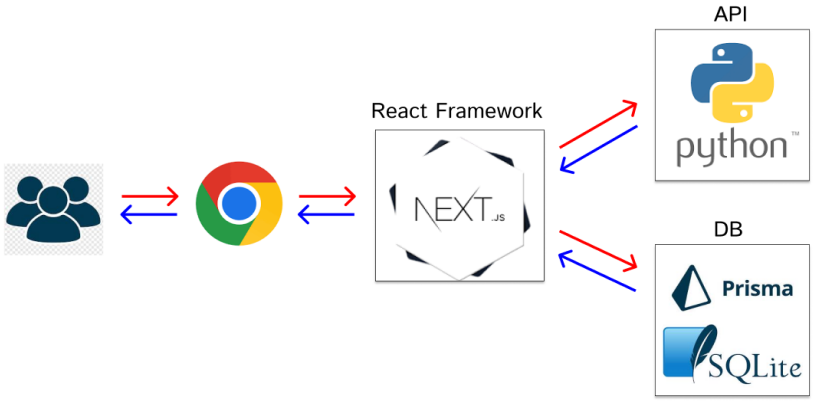
기능 요구사항	내용	적용 여부
청킹 방식 수정	청킹 방식 사용 방식에서 스트링 감지 방식으로 변경	변경
GIPS 알고리즘 적용	새로운 파일이 업로드 되었을 때 해당 파일을 구성하는 시그니처들을 추출하는 알고리즘을 적용함	완료
탐지 규칙 생성(수동)	사용자가 임의로 Yara Rule을 생성할 수 있도록 기능을 구현함	완료
탐지 규칙 생성(자동)	사용자가 다수의 파일들을 업로드해 새로운 규칙을 생성할 수 있도록 기능을 구현함	완료
파일 분석(단일, 복수)	사용자가 원하는 파일을 업로드 했을 경우 해당 파일(파일들)의 공격 유무와 근거를 보여주는 기능을 구현함	완료
공격 시각화(시그니처)	분석에 사용된 시그니처들과 구성 비율을 차트 형식으로 시각화함	완료
공격 시각화(탐지 비율)	분석된 파일들의 공격/비공격 비율을 차트 형식으로 시각화함	완료
공격 시각화(사용률)	사이트 통해 만들어진 Yara Rule과 분석한 파일의 수를 시각화함	완료

2.2.3 시스템 비기능(품질) 요구사항

비기능 요구사항	내용	적용 여부
사용성	프로젝트의 주요 기능들을 사이드 바를 통해 나열하여 쉽게 찾을 수 있도록 나열하고, 현재 사용 중인 기능을 강조해 가시성과 가독성을 높임	달성
사용성	모든 페이지에서 로고를 클릭 시 메인 화면으로 돌아갈 수 있으며, 각 기능마다 이전, 다음 등 사용자의 의도에 따라 쉽게 페이지를 이동할 수 있도록 조직성을 높임	달성
성능	모든 API 서버 응답 속도가 5초 이내여야 함	달성
성능	데이터베이스 질의 중 80% 이상이 2초 이내에 완료되어야함	달성
안정성	로그인에 사용되는 모든 비밀번호는 해쉬화 하여 데이터베이스에 저장함	달성
이식성	프로젝트 기능들을 웹에서 구동하도록 제작하여 플랫폼에 관계 없이 작동함	달성

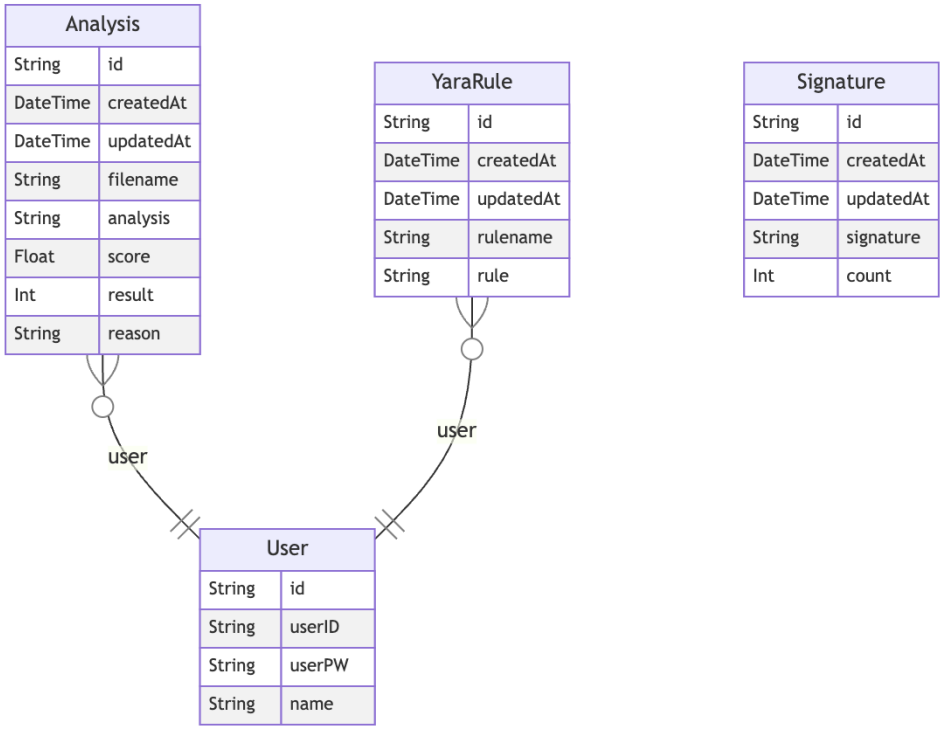
2.2.4 시스템 구조 및 설계도

- 시스템 구조도



NEXT.JS 기반의 Web 플랫폼 환경으로 프론트엔드, 백엔드 모두 통합으로 구축하였다. 파이썬 라이브러리가 필요할때만 파이썬을 subprocess 형태로 호출해 사용해 효율적으로 구현하였다.

- ER 다이어그램



2.2.5 활용/개발된 기술

GIPS는 여러 패킷의 페이로드에서 공통된 문자열을 추출하는 알고리즘입니다. 이러한 기술을 개발하게 된 이유는 실제 공격들은 유사한 형태를 가지고 있기 때문에 실제 벡터화를 하면 잘 뭉쳐있기 때문입니다. GIPS는 알고리즘 진행 흐름은 크게 2가지 파트로 나뉩니다. 첫번째 파트로는 비슷한 문자열을 가지는 페이로드끼리 묶어주는 빅그룹 식별 그 다음 파트로는 시그니처 추출입니다.

첫번째 빅그룹 식별 파트에서는 AE chunking과 minhash를 이용하여 각 페이로드들을 벡터로 만듭니다. 이 때 minhash를 사용하는 이유는 아무리 긴 페이로드가 와도 고정된 크기의 벡터로 만들 수 있기 때문에 메모리 효율적입니다. 이렇게 생성된 벡터들을 하나씩 더해주면서 특정 차원의 값이 높다면 그 벡터가 만들어진 페이로드를 빅그룹으로 식별하게 됩니다.

시그니처 추출 파트에서는 빅그룹에서 비슷한 페이로드를 식별 하기 위해서 DBSCAN을 이용해 비슷한 페이로드끼리 클러스터를 생성하게 됩니다. 그 다음으로는 각 클러스터별로 THH를 이용하여 시그니처 추출하게 됩니다. THH는 각 페이로드를 N-gram 단위로 쪼갠 다음 n-gram 중에서 많이 나오는 스트링을 추출하는 알고리즘입니다. 마지막으로는 빅그룹으로 식별되지 않은 패킷들에서도 똑같이 THH를 이용하여 스트링들을 뽑아냅니다. 이는 정상과 공격 모두에서 많이 나오는 문자열을 시그니처로 추출 되지 않게 하는 일종의 필터링 역할을 하고 이를 stopword라고 부릅니다. 이렇게 빅그룹으로 식별된 페이로드에서 나오는 스트링에서 stopword를 빼서 최종 시그니처를 생성하게 됩니다.

2.2.6 현실적 제한 요소 및 그 해결 방안

1. 현재 사용하는 프로젝트에서 사용하는 피처는 파일을 바이트 단위로 읽어서 아스키 문자열 중 사람이 읽을 수 있는 부분만 가져오기 때문에 파일의 특징을 잘 나타낸다고 할 수 없습니다. 이는 산학 프로젝트로 진행 중인 누리랩에서 만든 기술은 PE scanner를 이용하여 실제 유의미한 피처들을 가져올 수 있습니다. 하지만 이러한 피처들이 개발한 알고리즘의 입력에 맞는 피처들이 아니기 때문에 피처를 재가공하는 연구가 선행한 후에 개발한 알고리즘에 적용해야합니다.

2. 만들어지는 Yara Rule은 시그니처를 포함하는지 확인하는 간단한 룰입니다. 그렇기 때문에 이전에

보지 못했던 스트링이 추출되게 하거나 탐지 시그니처를 아는 경우에는 이를 회피하는 공격을 만들어 내기 쉽습니다. 그렇기 때문에 앞선 1번 설명과 같이 누리랩 협력에서 누리랩이 가지고있는 Yara Rule을 위한 생성형 AI를 이용하면 더욱 정교한 Yara Rule을 생성할 수 있고, 이를 통해 견고한 탐지 규칙을 생성한다면 기존에 보지 못했던 공격이나 회피하는 공격에 대해서도 잘 탐지할 수 있을 것입니다.

2.2.7 결과물 목록

상세	기능	경로
로그인 페이지	사용자 계정 로그인	/
대시보드 페이지	사이트 사용량, 저장되어있는 시그니처 수, 분석한 파일의 결과 등의 수치를 시각적으로 표시	/dashboard
Yara Rule 수동생성 페이지	사용자가 직접 규칙의 이름과 상세 내용들을 적어 새로운 Yara Rule을 생성하여 사이트에 올리거나 사용자의 컴퓨터로 다운로드 받을 수 있음	/generate/rule/yara/manual
Yara Rule 자동생성 페이지	다수의 파일들을 분석하여 공통된 시그니처들을 추출하고 추출한 것들을 바탕으로 새로운 Yara Rule을 생성. 생성한 룰은 자동으로 서버에 올라가고 사용자가 다운로드 받을 수 있음	/generate/rule/yara/auto
Yara Rule DB 관리 페이지	서버에 있는 Yara Rule을 선택하여 삭제/수정할 수 있음. Yara Rule이 적혀있는 파일을 서버에 업로드 할 수 있음	/generate/rule/yara/manage
Yara Rule 적용 페이지	서버에 있는 Yara Rule 중 하나를 선택하여 사용자가 원하는 다수의 파일들을 검사하고 각 파일들의 매칭 여부를 보여줌	/apply/rule/yara

단일 PE 파일 분석 페이지	사용자가 분석을 원하는 파일을 업로드하면 해당 파일의 정보 및 공격 여부와 PE header 영역, 분석된 시그니처들의 목록과 공격 여부를 보여줌	/analyze/file/pe
통계 시각화 페이지	프로젝트를 통해 분석된 파일들의 공격 비율, 탐지된 시그니처와 비율, 사이트 사용률 등의 시각화	/stats/chart

2.3 기대효과 및 활용방안

1. 생성 된 Yara rule를 가지고 사용자들의 자발적 참여와 공유를 통해 확장성 있는 위협 인텔리전스 기반을 구축하여 SaaS 제공한다면 새로운 공격이 들어왔을 때 더 발 빠르게 대처할 수 있을 것입니다. 또한 더 나아가 이러한 소프트웨어를 제공하여 다양한 보안 솔루션 회사 협업 및 제휴를 이끌어 내 지속적으로 업데이트되는 서비스를 제공할 수 있을 것 입니다.

2. 실제 공격량의 증가가 보안 전문 인력들이 처리하는 속도보다 훨씬 빠르게 증가하고 있기 때문에 보안 전문 인력들이 도와줄 수 있는 툴 혹은 플랫폼의 중요성이 올라가고 있는데 이러한 프로젝트에서 만들어진 플랫폼을 이용하면서 악성 파일 데이터셋을 빠르게 식별 및 처리하여 시간을 절약한다면 추 후 실제 악성 파일에 대한 심층 분석이나 클러스터링을 이용하거나 탐지 결과로 부터 만들어진 통계 자료를 보고 파일을 분석하는데 더 많은 시간을 할애할 수 있습니다. 이는 곳 작업 속도와 작업량 자체를 늘려 보안을 강화하는데 도움을 줄 것입니다.

3. 보안이 다소 부족한 기업들의 약점 혹은 기업의 특정 인원을 공격하는 사회공학적 기법들을 이용하여 악성 코드들을 만들어 공격을 하는 추세입니다. 그렇기 때문에 실제 기업 이러한 기업들에게 들어오는 공격 파일을 수집하여 기업 환경에 맞는 탐지 시그니처를 뽑아낸다면 탐지 규칙의 정확도를 향상시킬 수 있을 것입니다.

3 자기평가

엄석현: 악성 PE파일에서 시그니처를 뽑아내는 알고리즘을 만들어 내는 것에 중점을 두고 프로젝트를 진행했습니다. 현재 약 30000개의 악성파일을 train(80%), test(20%) + 정상 파일(6000개)로 나누어서 실험을 진행할 결과 precision은 약 0.98 recall은 0.82 f1-score는 0.9로 절대 무시할 수 없는 수준의 결과를 만들어 냈습니다. 또한 탐지 규칙을 강화하게 된다면 오탐 부분에서는 대략 6000개의 파일 중에서 1개만 나올정도로 매우 성능을 올릴 수 있습니다. 또한 test데이터셋에 일부에 대해서는 실제 오픈 소스 백신 소프트웨어 clamAV에서는 탐지하지 못하는 경우로 보아 실제 산업체에서도 사용 가능성이 높다고 봅니다.

김태경: 이번 프로젝트를 통해 Next.js 프레임워크를 사용하여 프론트엔드 개발을 진행하면서 개발에 있어서 프레임워크의 필요성을 알 수 있었습니다. 특히 서버 사이드 렌더링(SSR)과 정적 사이트 생성(SSG)을 활용해 효율적으로 작업을 수행할 수 있었습니다.
다음으로 백엔드와 데이터베이스를 설계하고 연동해 사용하면서 웹 개발에 있어서 데이터가 어떻게 상호작용하는지에 대한 이해를 높일 수 있었고, 마지막으로 데이터 시각화까지 제작하면서 전반에 걸쳐 균형 잡힌 개발 경험을 할 수 있었습니다.

김태윤: 프로젝트 전반에 걸쳐 성능을 향상시키기 위해 다양한 기법들을 적용하는 과정에서 많은 도전과 어려움이 있었습니다.. 특히, 최적의 결과를 도출하기 위해 여러 실험을 반복하며, 성능 개선을 위한 방법론에 대해 깊이 연구하고 테스트하는 데 많은 시간을 할애하는 경험을 하였습니다. 단순히 정보를 보여주는 것이 아니라 사용자 경험을 최우선으로 생각하며 UI/UX 디자인에 있어서도 사용자의 편의성과 접근성을 개선하기 위한 방안을 모색하고, 이를 실현하기 위한 다양한 접근법을 시도해 기술뿐만 아니라 사용자 경험을 설계하는 시간이라 의미 있는 시간이었다고 생각됩니다.

박준서: Git과 Next.js 프레임워크를 사용해, 프론트엔드와 백엔드개발을 동시에 하면서 협업의 중요성을 다시한번 느꼈습니다. 팀원의 코드를 서로 리뷰하는 과정에서 발견된 버그나 개선사항을 효율적으로 수정할 수 있어 프로젝트 개발에 큰 도움이 되었습니다.
또한, Next.js 프레임워크를 사용하면서 Server-Side Rendering을 활용해 사용자 경험을 향상시킬 수 있었습니다. SSR과 CSR에 대해 깊게 생각해 볼 수 있는 기회가 되었습니다.

4 참고 문헌

번호	종류	제목	출처	발행년도	저자	기타
1	논문	Generative intrusion detection and prevention on data stream.	32nd USENIX Security Symposium (USENIX Security 23)	2023	HyungBin Seo, and MyungKeun Yoon.	

5 부록

5.1 사용자 매뉴얼

Yara Rule 수동생성

1. 사용자 입력
   - 좌측 모달에 .yar 파일명, 적용할 시그니처 입력
2. 결과 확인
   - 생성 후 보여지는 모달에서 수정 및 저장 가능

Yara Rule 자동생성

1. 파일 업로드
   - 실행파일 업로드
2. 결과 확인
   - 모달 및 결과카드 UI에서 확인 가능

Yara Rule DB 관리

1. Yara Rule 선택
   - Default Rule 및 User Rule 중 적용할 파일 선택
2. 수정 및 저장

- 선택 후 보여지는 모달의 .yar 파일 수정 및 저장
- 사용자 컴퓨터로 다운로드 및 DB 저장 가능

다중 PE 파일 분석

1. Yara Rule 선택
   - 기본 룰 및 사용자 룰 중 적용할 파일 선택
2. 파일 업로드
   - 실행파일 업로드
3. 결과 확인
   - 모달 및 결과카드 UI에서 확인 가능

단일 PE 파일 분석

1. 파일 업로드
   - 실행파일 1개 업로드
2. 결과 확인
   - 모달 및 결과 카드 UI에서 확인 가능

5.2 운영자 매뉴얼

개발 환경 설정

```
git clone https://github.com/kookmin-sw/capstone-2024-34
cd capstone-2024-34/web
pip install -r requirements.txt
npm install
npm run dev
```

5.3 배포 가이드

프로덕션 환경 빌드

```
git clone https://github.com/kookmin-sw/capstone-2024-34
```

```
cd capstone-2024-34/web
pip install -r requirements.txt
npm install
npm run build
npm run start
```

5.4 테스트 케이스

대분류	소분류	기능	테스트 방법	기대 결과	테스트 결과
Web	공통	로그인	ID/PW로 사용자 계정 로그인	발급된 계정으로 로그인 성공	정상
	대시보드	메인 페이지	다른 서브페이지에서 작업한 내용이 통계 UI에 반영여부 확인	실제 플랫폼 내 사용 결과와 일치	정상
	탐지규칙 생성 및 관리	Yara Rule 수동생성 페이지	직접 규칙의 이름과 상세 내용들을 적어, 새로운 Yara Rule을 생성되는지 확인. 생성후 사이트 업로드 및 DB 정상 저장	사용자가 입력한 내용에 따른 룰 생성 및 저장 성공	정상
		Yara Rule 자동생성 페이지	다수의 파일들을 분석하여 공통된 시그니처들을 추출하고 추출한 것들을 바탕으로 새로운 Yara Rule을 생성. 생성한 룰은 자동으로 서버에 올라가고 사용자가 다운로드	GIPS 모듈 호출 및 결과 저장 성공	정상
		Yara Rule DB 관리 페이지	서버에 있는 Yara Rule을 선택하여 삭제/수정할 수 있음. Yara Rule이 적혀있는 파일을 서버에 업로드	사용자의 수정 내용이 DB에 정상적으로 반영	정상
	탐지규칙 적용	Yara Rule 적용 페이	서버에 있는 Yara Rule 중 하나를 선택하여 사용자가 원하는 다수의	업로드된 파일의 룰 적용 결과 출력	정상

		지	파일들을 검사하고 각 파일들의 매칭 여부		
	정적파일 분석	단일 PE 파일 분석 페이지	사용자가 분석을 원하는 파일을 업로드하면 해당 파일의 정보 및 공격 여부와 PE header 영역, 분석된 시그니처들의 목록과 공격 여부	파일 업로드후 분석 결과 출력	정상
	탐지통계 조회	통계 시각화 페이지	프로젝트를 통해 분석된 파일들의 공격 비율, 탐지된 시그니처와 비율, 사이트 사용률 등의 시각화	사용자의 이용량, 탐지량에 따른 데이터 정상 반영	정상
Python	API	feature extractor	각 파일을 바이트 단위로 읽은 다음에 읽을 수 있는 문자열 추출	기존 청킹 알고리즘을 대체하여 토큰 생성	정상
		GIPS	문자열 집합을 이용하여 클러스터링을 진행 각 클러스터 별로 시그니처 생성	공격에서 나오는 시그니처로 사용하여 탐지 기능	precision: 0.98 recall: 0.82 fl-score: 0.9
		탐지 규칙 생성	생성된 시그니처와 Yara를 이용하여 탐지 규칙 생성	실제 악성파일을 입력으로 받았을 때 탐지 가능한지	정상
	파일 생성	Whitelist	정상 파일에서 등장하는 문자열을 이용하여 whitelist 파일 생성하여 시그니처 생성할 때 필터링을 걸어서 정상과 악성 모두에서 많이 나오는 문자열이 시그니처가 되는 것을 방지	시그니처를 필터링한 효과로 성능 향상	정상적으로 동작