

MIC

Mobile Integrity Checker

Contents

프로젝트 개요

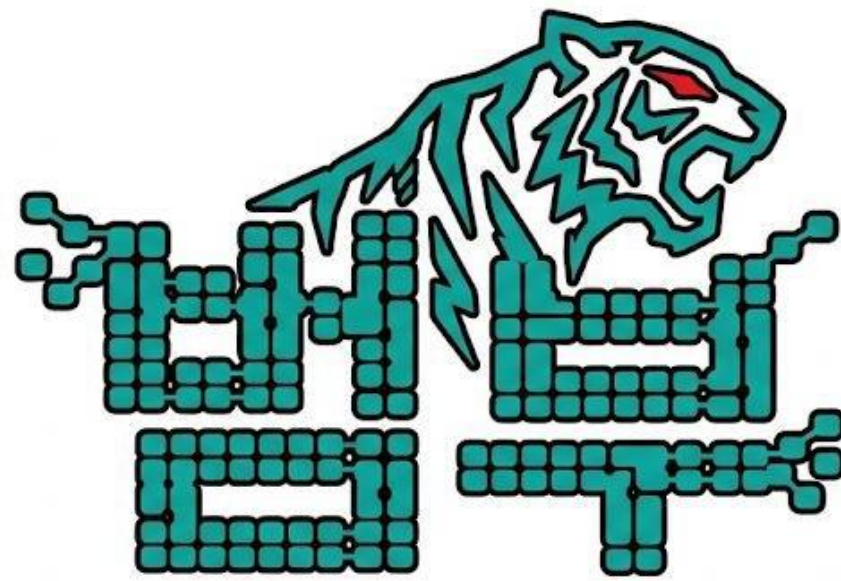
시스템 구성

프로그램 사용법

고객 지원 및 사후 관리

Intro

팀 소개



BUMBU: Small Steps, Mighty Leap
APK Integrity & Threat Dashboard

“개개인의 작은 힘이 모여 강력한 범(호랑이)을 이룬다”

Intro

| 팀원 소개

**김용진**

PM / Mobile

**송채린**

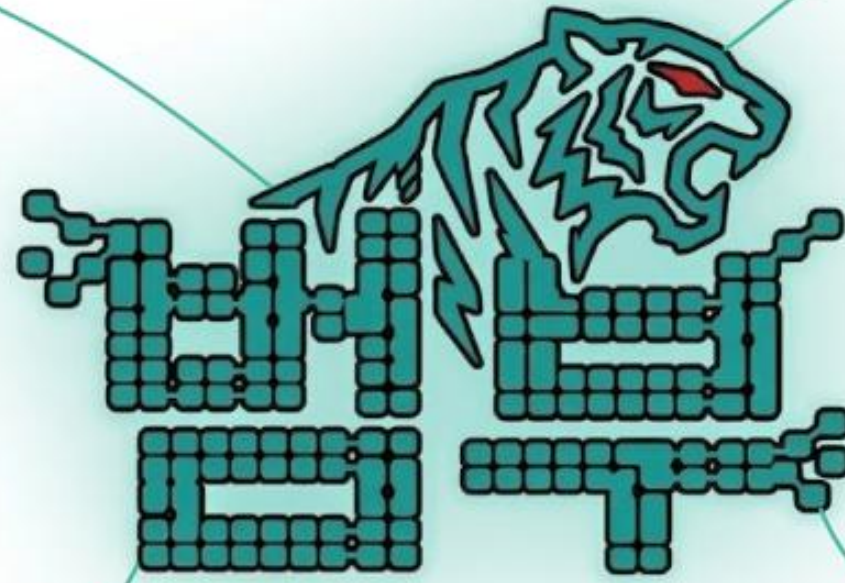
DB / 저장 / 관리

**강원찬**

비교 / 위험도 로직

**김병민**

서버 API / 프론트엔드



Project Overview

프로젝트 개요

“이 앱이 정말 안전할까?”

프로젝트 개요

일상의 평온을 위협하는 위조 앱: 보이스피싱의 덫



프로젝트 개요

■ 시장 조사



N H N CLOUD
NHN AppGuard

- 실시간 위협 탐지
- 무중단 변조탐지 설정 제어가 가능

- 웹 콘솔을 이용해 적용
- 사용한 만큼만 지불하는 경제성

프로젝트 개요

아이디어

MIC 보안 서버: 기본을 넘어서는 운영의 진화



프로젝트 개요

프로젝트 제안 - 모바일 앱 무결성 검증 시스템

MIC가 잡아내는 8가지 보안 위험 신호 (The 8 Security Threat Signals Captured by MIC)

1. APK 파일의 해시값
(가짜 앱 검사)



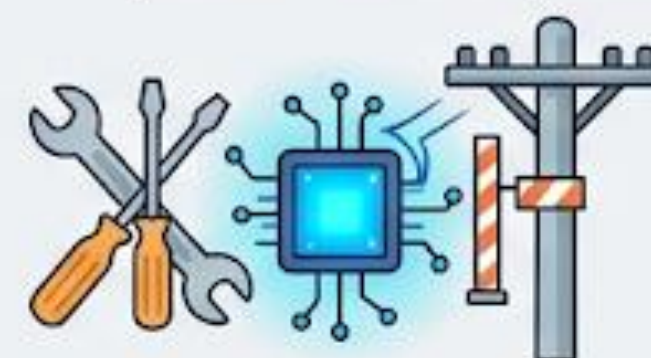
2. APK 서명 인증서 해시값
(지문 대조)



3. Frida (동적 분석 도구)
(투명인간 감시)



4. Xposed Framework
(시스템 조작 차단)



5. 미등록된 앱 탐지
(불청객 확인)



6. 단말기 루팅 여부
(잠금장치 해제)



7. 앱 디버그 모드 활성화
(현미경 관찰 차단)



8. 에뮬레이터 실행 여부
(가상 환경 탐지)



프로젝트 개요

프로젝트 제안 - 모바일 앱 무결성 검증 시스템

실시간 위협 탐지

모바일 환경의 무결성을 보장하는 실시간 위협 탐지 시스템 구축

웹 콘솔 이용

클라이언트의 탐지 결과를 대시보드를 통해 관리자에게 전달

위험도별 차등 데이터 생명주기(TTL) 관리

위험도에 따라 로그 보관 기간을 다르게 설정하여, 효율적 관리 추구



SDK화로 인한 편리성

가벼운 리소스로 어떤 프로젝트에도 간편하게 사용 가능

프로젝트 개요

프로젝트 제안 - 모바일 앱 무결성 검증 시스템

```
//INIT INFOs
BeombuSecurityEngine.init( url = "http://192.168.200.131:8888")
val btn = findViewById<Button>(R.id.btnCheck)
btn.setOnClickListener {

    findViewById<Button>(R.id.btnCheck).setOnClickListener {
        BeombuSecurityEngine.verify( context = this) { isSafe, message, score ->
            runOnUiThread {
                Toast.makeText( context = this, text = " ○○서버에 성공적으로 전송했습니다.", duration = Toast.L
            }
        }
    }
}
```

이 두줄로

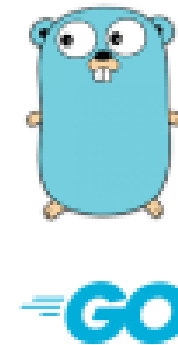


다 된다고?

System Overview

시스템 구성

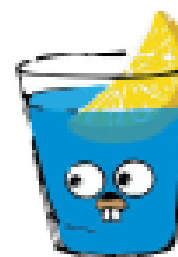
시스템 구성 기술 스택



주 사용 프로그래밍 언어



데이터 변환 및 가공을 위한
DBMS



GIN

웹 서비스를 위한 프레임워크



클라이언트 APK/SDK 개발

시스템 구성

시스템 아키텍처

유기적 아키텍처 시스템 구성도



시스템 구성

사용자 시나리오

비정상 도구 탐지 및 대응 시나리오



시스템 구성

메뉴별 화면 구성 (UI/UX Interface)

DASHBOARD V1.0

APK 무결성 검증 대시보드

Mobile Integrity Checker API를 웹에서 간편하게 호출하고 응답을 확인할 수 있습니다. 모델 구조체에 맞는 정확한 JSON을 전송합니다.

Endpoints	API Method	서버 상태
5개	GET / POST	● 연결됨

🔗 서버 URL

http://localhost:8888

연결 테스트

● 연결됨

서버 연결 테스트

연결 상태 확인

시스템 구성

메뉴별 화면 구성 (UI/UX Interface) - 등록

새로운 앱 해시값 등록

- 패키지 이름
- APP version
- APK Hash
- SIG Hash

POST
/api/baseline — 기준값 등록
Baseline

요청 파라미터

Go 모델 Baseline 구조체에 맞는 JSON 필드를 입력합니다.

Package Name
packageName
예: com.example.bankapp

Version Code
versionCode
예: 21

APK Hash (SHA-256)
apkHash
예: e3b0c44298fc1c149afb4c8996fb924...

Signature Hash
signatureHash
예: a1b2c3d4e5f6...

전송될 JSON 미리보기

```

{
  "packageName": "",
  "versionCode": 0,
  "apkHash": "",
  "signatureHash": ""
}

```

기준값 등록
샘플 데이터 채우기

시스템 구성

메뉴별 화면 구성 (UI/UX Interface)

1초마다 갱신

검증
등록
조회
정리
실시간

LOG 실시간 히스토리

실시간 기록

com.example.bankapp
2026. 4. 3. 오전 10:38:41

UUID: 550e8400-e29b-41d4-a716-446655440000 | Score: 100 (Critical)

Rooted: false
Frida: true
Xposed: false
Debug: false
Emulator: false

com.example.bankapp
2026. 4. 3. 오전 10:38:39

UUID: 550e8400-e29b-41d4-a716-446655440000 | Score: 30 (Warning)

Rooted: false
Frida: false
Xposed: false
Debug: true
Emulator: true

com.example.bankapp
2026. 4. 3. 오전 10:38:36

UUID: 550e8400-e29b-41d4-a716-446655440000 | Score: 40 (Warning)

Rooted: true
Frida: false
Xposed: false
Debug: false
Emulator: false

com.example.bankapp
2026. 4. 3. 오전 10:38:34

UUID: 550e8400-e29b-41d4-a716-446655440000 | Score: 80 (Danger)

Rooted: false
Frida: false
Xposed: true
Debug: false
Emulator: false

시스템 구성

메뉴별 화면 구성 (UI/UX Interface)

GET
/api/results — 지목 조회
VerifyDataSet[]

요청 파라미터

UUID (기기 고유 식별자) *

예: 550e8400-e29b-41d4-a716-446655440000

조회 기간 (시간)

24

지목 조회

응답 결과

조회 결과가 여기에 표시됩니다.

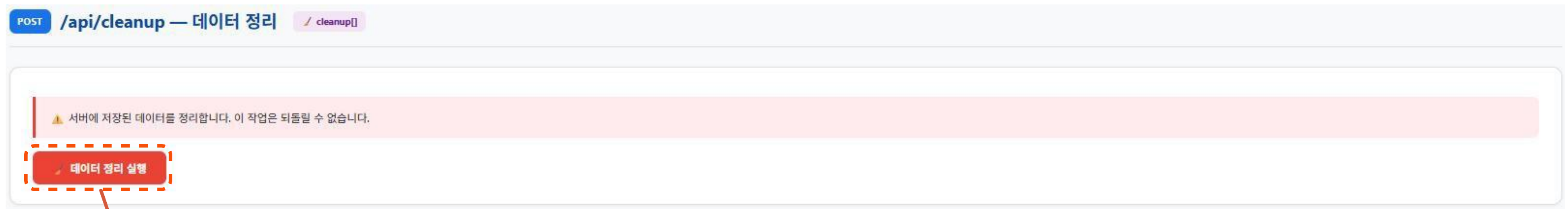
검증결과 조회

• UUID를 기준으로 DB에 저장된 검증 기록 및 결과 조회

20

시스템 구성

메뉴별 화면 구성 (UI/UX Interface)



데이터 정리 버튼

- VerifyDataSet 테이블의 등급별 데이터 정리
- Safe/Low(7일), Warning / Danger(1개월), Critical(3개월) 경과 데이터를 삭제

시스템 구성

메뉴별 화면 구성 (UI/UX Interface) - 검증

검증

등록

조회

정리

실시간

POST

/api/verify — 무결성 검증 요청

VerifyRequest

검증 필요한 데이터 입력란

패키지 이름,hash,버전,SDK 버전 등

요청 파라미터

Go 모델 VerifyRequest 구조체에 맞는 JSON 필드를 입력합니다.

UUID

예: 550e8400-e29b-41d4-a716-446655440000

Package Name

예: com.example.bankapp

Version Code

예: 21

APK Hash (SHA-256)

예: e3b0c44298fc1c149afb4c8996fb924...

Signature Hash

예: a1b2c3d4e5f6...

보안 탐지 플래그

Rooted rooted false

Debug debugEnabled false

Frida isFrida false

Xposed isXposed false

Emulator isEmulator false

전송될 JSON 미리보기

```
{
  "uuid": "",
  "packageName": "",
  "versionCode": 0,
  "apkHash": "",
  "signatureHash": "",
  "rooted": false,
  "debugEnabled": false,
  "isFrida": false,
}
```

검증 요청 보내기

샘플 데이터 채우기

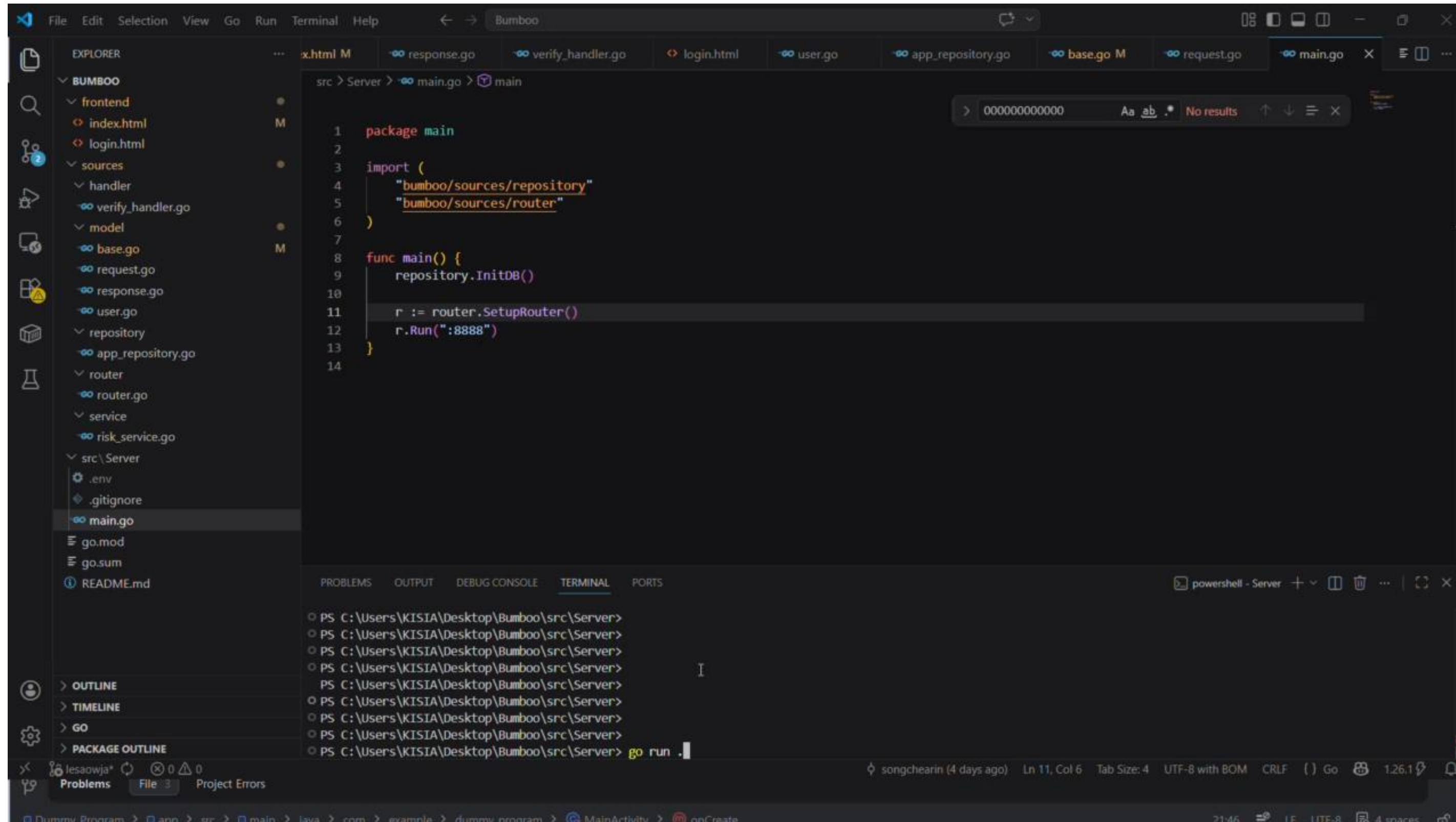
위험 샘플

How to use MIC

프로그램 사용법

시스템 구성

모바일 앱 무결성 검증 시스템 - 시연



The screenshot shows a Visual Studio Code editor with a Go project named 'Bumboo'. The Explorer panel on the left shows the project structure, including folders like 'frontend', 'sources', 'handler', 'model', 'repository', 'router', 'service', and 'src'. The main editor window displays the 'main.go' file in the 'src/Server' directory. The code in 'main.go' is as follows:

```
1 package main
2
3 import (
4     "bumboo/sources/repository"
5     "bumboo/sources/router"
6 )
7
8 func main() {
9     repository.InitDB()
10
11     r := router.SetupRouter()
12     r.Run(":8888")
13 }
14
```

The terminal window at the bottom shows the command 'go run .' being executed in the 'src/Server' directory. The output of the terminal is as follows:

```
PS C:\Users\KISIA\Desktop\Bumboo\src\Server>
PS C:\Users\KISIA\Desktop\Bumboo\src\Server>
PS C:\Users\KISIA\Desktop\Bumboo\src\Server>
PS C:\Users\KISIA\Desktop\Bumboo\src\Server>
PS C:\Users\KISIA\Desktop\Bumboo\src\Server>
PS C:\Users\KISIA\Desktop\Bumboo\src\Server>
PS C:\Users\KISIA\Desktop\Bumboo\src\Server>
PS C:\Users\KISIA\Desktop\Bumboo\src\Server>
PS C:\Users\KISIA\Desktop\Bumboo\src\Server>
PS C:\Users\KISIA\Desktop\Bumboo\src\Server>
PS C:\Users\KISIA\Desktop\Bumboo\src\Server>
PS C:\Users\KISIA\Desktop\Bumboo\src\Server>
PS C:\Users\KISIA\Desktop\Bumboo\src\Server> go run .
```

모바일 앱 무결성 검증 시스템 - 시연

탐지 항목	가중치
App Integrity	100점
Sign Integrity	100점
Frida	100점
Xposed	80점
Unregistered Package	50점
Rooted	40점
Debug	20점
Emulator	10점

Risk Score 범위	등급
0 ~ 9	Safe
10~19	Low
19 ~ 79	Warning
80 ~90	Danger
90~100	Critical

Customer support & after-sales service

고객 지원 및 사후 관리

Customer support & after-sales service

공식 릴리즈 노트 주요 지원 내용



최신 보안 위협 탐지 내역의 공개

최신 보안 위협에 대한
패치 반영 시점 제공



정량적 성능 개선 지표 공개

백엔드와 DB 등 보이지 않는 시스템
내부 '최적화' 지표 제공



고객 지원 및 사후 관리

기술 지원 및 담당자

1. 프로젝트 관리 및 모바일 클라이언트 (PM & Android)

담당자: 김용진

지원 범위: 클라이언트 측 보안 탐지 모듈 연동 중 발생하는 앱 충돌(Crash) 및 예외 상황 트러블슈팅

2. 보안 분석 엔진 및 등급 산출 (Risk Logic & Analysis)

담당자: 강원찬

지원 범위: 가중치 기반 위험도 산출 알고리즘, 패키지 및 해시값 기반 Baseline 비교 로직, 위험 지수 커스터마이징 기술 문의.

3. 서버 아키텍처 및 인터페이스 (Server API & Front-end)

담당자: 김병민

지원 범위: 서버-클라이언트 간 통신 지연(Latency) 분석 및 API 호출 장애 대응.

4. 데이터베이스 및 스토리지 관리 (Data Management Support)

담당자: 송채린

지원 범위: 대용량 보안 검증 로그 누적으로 인한 대시보드 조회 속도 저하(Slow Query) 시 인덱싱 최적화 튜닝.

Q&A

감사합니다

■ 송채린, 강원찬, 김용진, 김병민

■ 출처

- LIAPP 로고 : <https://liapp.lockincomp.com/ko/liapp-learn-more>
- NHN CLOUD 로고 : <https://www.raonsecure.com/ko/solution/appiron>
- Go 로고 : <https://go.dev/>
- Gin 로고 : <https://github.com/gin-gonic/gin>
- PostgreSQL 로고 : <https://www.postgresql.org/>
- 김성모 짤 생성기: <https://sungmo.jjong.co.kr/>
- 앱 위변조 관련 피해 사례 기사 :

<https://www.edaily.co.kr/News/Read?newsId=02122166642368016&mediaCodeNo=257&OutLnkChk=Y>