사물인터넷융복합논문지 Vol. 11, No. 5, pp. 139-144, 2025 https://doi.org/10.20465/KIOTS.2025.11.5.139

내부망 기반 보안 메신저 시스템 설계 및 정보 유출 방지 방안

주수민[1], 이근호[2*]
[1]백석대학교 컴퓨터공학부 학생, [2]백석대학교 컴퓨터공학부 교수

Schema of a Secure Messaging System on Intranet Networks with Information Leakage Mitigation

Su-Min Joo[1], Keun-Ho Lee[2*]
[1]Student, Division of Computer Engineering, Baek-Seok University
[2]Professor, Division of Computer Engineering, Baek-Seok University

요 약 디지털 정보화 사회가 발전하면서 인터넷 사용이 활발해짐에 따라 보안의 중요성은 날로 증가하고 있다. 2021년 한 기업에서 외부 메신저인 이메일이 해킹되어 서버가 침해당하는 사고가 발생했다. 이로 인해 많은 기업들이 업무 효율성과 보안을 강화하기 위해 사내 메신저를 도입하게 되었다. 그러나 일부 중소기업은 여러 가지 이유로 사내 메신저 도입이 어려워 여전히 외부 메신저를 사용하는 경우가 많다. 특히 대한민국의 핵심 산업인 반도체 관련 중소기업은 해킹 피해 발생 시 기술 유출로 이어질 위험이 크다. 본 논문은 이러한 중소기업을 대상으로 내부망에 서버를 구성하여 외부망과 분리된 환경에서만 접속할 수 있는 메신저 시스템을 제안한다. 이 시스템은 동일 네트워크 대역을 가진 인원만 접속 가능하도록 설계되어 외부 해킹 위험을 물리적으로 차단하며 적은 비용으로 정보 유출을 예방할 수 있는 효율적인 보안 솔루션을 제공한다.

주제어 : 망 분리, 내부망, 보안 메신저, 정보 유출 방지, 접근제어

Abstract As the digital information society advances and Internet use becomes more widespread, the importance of security continues to grow. In 2021, a company suffered a server breach due to the hacking of its external messenger, email. This incident led many companies to adopt in-house messengers to enhance work efficiency and security. However, some small and medium-sized enterprises (SMEs) still rely on external messengers due to implementation difficulties. In particular, SMEs in the semiconductor industry—a key sector in South Korea—face high risk of technology leaks during a cyberattack. This paper proposes a messenger system for SMEs that configures a server within the internal network, allowing access only in an environment isolated from the external network. This system is designed to allow access only to those with the same network bandwidth, physically blocking the risk of external hacking and providing an efficient security solution that can prevent information leaks at low cost.

Key Words : Network Separation, Private Network, Secure Messenger, Data Leakage Prevention, Access Control

이 논문은 2025학년도 백석대학교 학술연구비 지원을 받아 작성되었음
*교신저자 : 이근호(root1004@bu.ac.kr)
접수일 2025년 10월 10일 수정일 2025년 10월 13일 심사완료일 2025년 10월 19일

1. 서론

오늘날 인터넷이 보급되면서 이메일과 메신저 애플리케이션의 사용이 급증하였고, 이에 따라 많은 기업들의 업무 효율성이 증가하였다. 이메일과 메신저를 활용해 기업의 중요한 문서와 정보를 주고받는 업무가 이루어지며, 해커들에게는 매우 매력적인 공격 대상이 되었다. 2021년, 한 기업의 외부 이메일이 해킹되어 내부 서버까지 침해되는 사고가 발생했으며, 이와 같은 사례들이 여러 차례 보고되었다. 이에 따라 많은 기업은 보안 인력에 투자하고 사내 메신저를 도입하는 등 정보 유출을 방지하기 위한 보안 투자를 적극 확대하였다.

하지만 자본력이 부족한 중소기업은 보안에 대한 투자가 어렵고, 여전히 외부 메신저를 사용하는 경우가 많다. 이러한 기업들은 해커들의 표적이 되어 정보 유출의 위험에 직면하게 된다. 사내 메신저를 외부망과 분리된 내부망에서 구동함으로써, 동일 대역폭을 가진 인원만이 접속할 수 있도록 하고, 외부 접근을 물리적으로 차단하는 시스템을 구현하면, 중소기업도 적은 자본으로 외부 공격으로 인한 정보 유출 피해를 예방할 수 있다.[1-6].

본 논문에서는 이러한 중소기업을 위한 보안 솔루션으로, 외부망과 분리된 내부망 로컬에서 운영되는 사내 메신저 시스템을 제안한다[7-11].

2. 관련 연구

2.1 중소기업 보안 현실

사이버 보안은 해커의 공격으로부터 보호하기 위해 기술적 및 비기술적 조치를 취하는 중요한 관행이다. 대부분의 기업은 급변하는 디지털 환경에 맞춰 사이버 보안을 강화하고 있으며, 이를 위해 위협을 감지하고 대응할 수 있는 장비와 관련 전문 지식을 갖춘 인력을 채용하는 등 방대한 투자가 필요하다. 그러나 이러한 투자는 많은 자본을 요구하기 때문에, 자본력이 부족한 중소기업은 사이버 보안 투자에 있어 보수적인 태도를 취하게 되어 많은 위협에 노출된다. 특히, 대한민국의 핵심 산업인 반도체 관련 중소기업들 중 상당수는 사이버 보안에 충분히 투자할 여력이 부족한 상황이다. 이처럼 국가의 핵심 산업을 담당하는 중소기업이 해킹 피해를 입을 경우, 관련 정보가 타 기업이나 타국으로 유출되는 피해를 초래할 수 있다[1-6].

따라서, 중소기업의 사이버 보안을 강화하기 위해서는 적은 자본으로도 효과적으로 위협으로부터 보호할 대책을 마련해야 한다[1].

2.2 내부망 분리

외부 해커의 공격으로부터 내부의 중요한 문서 및 자원을 보호하기 위한 전략은 다양하다. 그 중 망 분리는 외부 해커의 공격을 원천적으로 차단하기 위해 기관의 네트워크를 내부망과 외부망으로 분리하는 방법이다. 이러한 망 분리는 「국가 망 보안 체계 보안 가이드라인」을 기준으로 각 기관에 적합한 망 분리 방법을 적용할 수 있도록 「국가 정보보안 기본지침」에 따라 제시한다. 해당 가이드라인은 각 기관에 필요한 망 분리 구성 기준과 보안 고려사항 등을 제공한다[7,8].

망 분리는 외부로 노출되면 안 되는 민감한 정보를 내부망에서 운영하여 인터넷 등을 통한 불법적인 접근이나 해커로 인한 정보 유출 등 다양한 사이버 위협을 원천적으로 차단하는 기술이다. 이러한 기술은 많은 자본을 요구하지 않으며, 국가 핵심 산업을 담당하는 중소기업의 경우 대부분 업무망 분리가 이루어져 있다. 따라서, 이를 활용하여 보안 시스템을 구축하면 적은 자본으로 사이버 보안 위협으로부터 효과적으로 보호할 수 있다[7-11].

2.3 React

React는 컴포넌트 기반 아키텍처를 사용하여 독립적인 UI를 재사용 가능한 컴포넌트로 나누어 개발할 수 있다. 이를 통해 코드 모듈화가 가능해지며, 유지 보수가용이하고, 다양한 프로젝트에서 컴포넌트를 재사용할 수 있어 개발 효율성이 높아진다. 또한, React는 복잡한 상태 관리가 필요한 경우 Redux와 같은 상태 관리 라이브러리를 쉽게 통합할 수 있어, 애플리케이션의 상태를 중앙 집중적으로 관리하고 여러 컴포넌트 간의 데이터 공유 시 일관성을 유지할 수 있다. 그 결과, 컴포넌트 간의 상태 전달 및 관리가 간편해진다[5].

React는 또한 지연 로딩을 지원하여 개발자가 필요할 때만 컴포넌트를 로드할 수 있다. 이를 통해 불필요한 자원을 지연 로드함으로써 성능을 최적화하고 초기 페이지 로딩 속도를 향상시킬 수 있다. 이러한 특징은 초기 로딩 시간을 줄여 사용자 경험을 개선하며, 많은 자본을 투자해 전문 개발 인력을 채용하는 대신, 기존 자원을 활용하여 적은 인적 자원과 시간으로 프로젝트를 효율적으로 수행할 수 있게 해준다[12, 13].

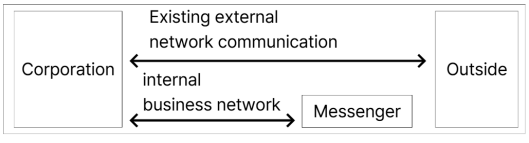
2.4 Flask

Flask는 경량 프레임워크로 개발 서버와 디버거를 내장하고, 데이터베이스·인증 등은 확장(Extension)으로 연동한다. 특히 Flask는 단순하고 경량화된 미니멀리스트 구조를 채택하고 있어 개발자가 필요한 기능만 선택적으로 추가할 수 있으므로, 복잡한 설정 없이 빠르게 애플리케이션을 만들 수 있다. 기본 웹 서버는 개발·테스트용으로 제공되며, 확장을 통해 데이터베이스 관리, 폼 처리, 인증 등 다양한 기능을 추가하거나 필요한 기능만 구현할 수 있어 유연한 개발이 가능하다. Flask는 내장 디버거와 테스트 클라이언트를 제공해 실행 중 오류를 신속히 추적하고 단위 테스트를 작성하기 쉽다. 또한 가벼운 서버 구성과 빠른 배포가 가능해 추가 서버 장비 설치에 들어가는 비용을 절감할 수 있어, 중소기업이 적은 자본으로 보안 메신저 시스템을 설계하는 데 적합한 프레임워크이다[14, 15].

3. 내부망 기반의 보안 메신저 구현

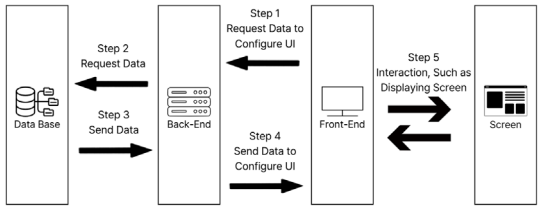
3.1 메신저 구현

대다수 중소기업은 자본력이 제한적이므로, 보안 메신저 개발 시 비용 효율적이고 가벼운 도구를 활용할 필요가 있다. 본 논문에서 제안하는 메신저 시스템은 효율적인 구현을 위해 백엔드와 프론트엔드를 분리하였다. 백엔드는 Flask를 활용하여 서버에서 웹 서비스 기능과 연산을 처리하도록 설계하였으며, 프론트엔드는 React를 사용하여 사용자에게 제공되는 UI를 구성하고, 불필요한 요소를 제외한 필요한 부분만 로드하여 화면을 신속하게 표시하도록 구현하였다. 이러한 도구 활용은 전체 개발 시간을 단축하고, 제한된 자원 환경에서도 자본 사용을 최소화할 수 있게 한다. 본 연구에서는 Flask와 React의 특성을 높이 평가하여 두 도구를 기반으로 시스템을 구현하였다[1, 12-15].



[Fig. 1] Internal Network Security Messenger System Isolated from External Network

[Fig. 1]과 같이 메신저를 외부망과 격리하여 내부 업무망을 통해 구동하면, 외부 통신이 차단되어 네트워크 지연이 감소하고, 통신 속도가 빠르며 안전하다. 또한, 내부망 기반으로 구동하면 많은 자본을 요구하지 않으면서도, 보다 손쉽게 사이버 보안 위협으로부터 효과적으로 보호할 수 있다[7, 11].



[Fig. 2] Messenger System Flowchart

메신저 시스템은 [Fig. 2]와 같은 흐름으로 동작하며, Flask와 React를 활용하여 구현할 것이다.

3.1.1 Flask를 활용한 백엔드 구현

Python Flask를 활용하여 [Fig. 2]와 같이 데이터베이스와 통신하고, 사용자가 볼 수 있는 UI로 데이터를 전송하는 기능을 구현할 것이다. 이러한 기본적인 서버 구현 내용은 〈Table 1〉과 같으며, 데이터베이스와 연결하여 세션 관리, 모델 정의, 사용자 API 엔드포인트 정의 등 백엔드에서 수행해야 할 기본 작업과 함께, 실시간 메시지 전송을 위한 WebSocket 연결 관리와 같은 부가적인 기능도 포함된다[14]. 나아가 보안 성능 강화를 위해 HTTPS 프로토콜 사용 등 추가적인 보안 서버 기능도 적용할 수 있다[14-16].

〈Table 1〉 Default Flask Code Flow

code	Explanation
app = Flask(__name__)	Create Flask app instance.
@app.route('/')	Define root path route.
def hello_world():	Handle root path request.
return "Hello, World!"	Return greeting message.
@app.route('/api')	Define /api route.
def api():	Handle /api request.
return jsonify({"message": "Hello, API!"})	Return JSON response.
if __name__ == '__main__':	Ensure script is run directly.
app.run(debug=True)	Start app in debug mode.

3.1.2 React를 활용한 프론트엔드 구현

[Fig. 2]와 같이 백엔드에서 수신·가공한 정보를 UI로 렌더링하는 계층이 프론트엔드이다. Flask와 React를 기반으로 백엔드와 프론트엔드를 구성하면 핵심 기능 구현이 가능하며, 본 연구의 목적은 이 두 도구를 활용해 보안 메신저를 구현하는 데 있다. 정상적인 메신저 기능을 위해서는 다수 사용자 간 실시간 통신이 필수이므로, 〈Table 2〉와 같이 WebSocket 서버에 연결하여 백엔드와 프론트엔드가 원활히 연동되도록 설계한다. 이때 Flask·React 기반 구성에서 WebSocket을 통해 양방향 실시간 송수신을 수행하고, JWT 토큰으로 서버 인증을 처리하여 사용자 식별 절차를 수행한다. 이를 통해 사용자 간 실시간 메시지 전송을 위한 기반을 마련한다[12, 16].

〈Table 2〉 Basic Real-time Messaging Code in React

code	Explanation
socket.connect();	Connect to WebSocket.
socket.emit('authenticate', { token });	Authenticate with JWT token.
socket.on('chat', handleIncomingMessage);	Listen for chat messages.
handleSend = async () => { ... };	Send message to WebSocket.
await axios.post(${API_BASE}/api/messages, msg);	Save message to server.
const res = await axios.get(${API_BASE}/api/users);	Fetch user list from server.
setMessages(prev => [...prev, msg]);	Update UI with new message.
scrollToBottom = () => { ... };	Scroll chat window to bottom.

3.2 메신저 서버 구동

상기 과정을 통해 구현된 메신저 시스템은 Flask 기반 서버에서 구동된다. Flask를 활용하면 전용 장비 없이 일반 노트북에서도 실행할 수 있고, 경량 서버 특성상 기존 서버에 통합해도 무리 없다. 내부망 배포를 전제로 사설 IP와 포트만 지정해 간단히 기동하며, 운영 중 리소스 부담이 낮아 유지관리도 용이 하다. 이에 따라 추가 자본 없이 시스템을 운영할 수 있다. 서버 구동 과정은 다음과 같다[12, 13].

3.2.1 Flask 서버 구동

Flask로 기능을 구현한 뒤 Python·Gunicorn 명령으로 애플리케이션을 구동한다. 실시간 비동기 통신은 WebSocket 워커 기반 비동기 이벤트 루프로 처리한다. [Fig. 3]에 보이듯 Gunicorn으로 서버를 기동하고 127.0.0.1:8000(루프백 인터페이스)에 바인딩하면, 로그에 로컬 환경 전용 리스닝이 확인되어 외부 접근이 차단된다. 이를 통해 외부와 격리된 로컬 구동 기본 설정을 완료할 수 있다[14, 16].

```
(base)              -MacBook-Pro backend % gunicorn -k geventwebsocke
t.gunicorn.workers.GeventWebSocketWorker app:app -bind 0.0.0.0:5050
[2025-09-26 20:10:39 +0900] [46314] [INFO] Starting gunicorn 23.0.0
[2025-09-26 20:10:39 +0900] [46314] [INFO] Listening at: http://127.
0.0.1:8000 (46314)
[2025-09-26 20:10:39 +0900] [46314] [INFO] Using worker: geventwebso
cket.gunicorn.workers.GeventWebSocketWorker
[2025-09-26 20:10:39 +0900] [46315] [INFO] Booting worker with pid:
46315
```

[Fig. 3] Flask Backend Server Execution Log

3.2.2 서버 IP 지정

[Fig. 4]는 React와 Flask의 베이스 IP를 외부 접속을 차단하고 로컬에서만 접속을 허용하기 위해 로컬 IP로 설정한 예시이다.

```
REACT_APP_API_BASE=http://127.0.0.1:5050
REACT_APP_REA_BASE=http://127.0.0.1:3000
FLASK_SECRET_KEY=super-secret
```

[Fig. 4] Specify Base IP as Local IP

[Fig. 4]에서 작성된 파일을 〈Table 3〉과 같이 React 시작 스크립트에 추가하면, 실행 시 해당 설정 파일이 React 실행 폴더에 복사되어 로컬 IP에서만 접속 가능하도록 설정된다[7, 12, 13].

〈Table 3〉 Applying IP Configuration at Startup

code	Explanation
"prestart": "cp ../.env .env"	Copy .env file from parent directory to current directory.
"start": "react-scripts start"	Start development server.
"build": "react-scripts build"	Build app for production.
"test": "react-scripts test"	Run React app tests.
"eject": "react-scripts eject"	Eject React setup for full build configuration control.

이어서 [Fig. 4]와 같이 IP 설정 후 [Fig. 5, 6]과 같이 Flask 서버와 WebSocket에 연결해 주는 작업을 통해 Flask 구동 시 [Fig. 4]에서 설정한 IP로 동작하게 설정한다.

```
base_url = os.environ.get('REACT_APP_REA_BASE')
```

[Fig. 5] Applying an IP Configuration File to Flask

```
const socket = io(process.env.REACT_APP_API_BASE, {
  transports: ['websocket'],
  autoConnect: false
```

[Fig. 6] Applying an IP Configuration File to WebSocket

상기 과정을 거치면 Flask와 React 서비스는 사설 IP에 바인딩 되고 방화벽/라우팅을 차단하여 외부 사용자 접근이 원천적으로 제한돼, 로컬 네트워크에서만 접근 가능해진다. 다만 [Fig. 4]와 같이 로컬 IP로 설정하면 해당 장치에서만 접속할 수 있다. 메신저는 다수 사용자의 접속이 필요하므로, [Fig. 7]과 같이 공유기가 할당한 사설 IP로 설정 파일을 변경하면 동일 네트워크 대역 사용자만 접근하도록 제한된다[7-9].

```
REACT_APP_API_BASE=http://192.168.0.4:5050
REACT_APP_REA_BASE=http://192.168.0.4:3000
FLASK_SECRET_KEY=super-secret
```

[Fig. 7] Change IP Configuration File to Router IP

3.3 메신저 접속 및 인증

상기 과정을 통해 IP 설정을 완료하면 지정한 IP와 포트 번호로 서비스 접속이 가능해진다. 이 구성에서는 외부(공인) IP를 통한 접근이 차단되고 내부(사설) IP에서만 접속과 로그인이 가능하다. 절차가 단순하고 추가 비용이 거의 들지 않으므로 제한된 예산의 중소기업 환경에서도 구현할 수 있다. 또한 구현 난이도가 낮아 전담 인력 없이도 단기간에 구축 가능하며 웹 서버·프론트엔드 UI·시큐어 코딩 등 고급 역량을 대규모로 투입하지 않아도 된다. 따라서 본 방식은 인력과 시간 투자가 어려운 환경에서도 적용 가능한 정보 유출 대비 최소 보안 조치로 기능한다[1].

3.4 도입 효과

대한민국 천안시 소재 반도체 장비 제조 중소기업의 약 20명을 대상으로 파일럿을 수행한 결과, 기존 메신저 대비 보안성과 전송 지연 측면에서 우수하다는 긍정적 피드백이 다수 확인되었다. 파일럿 테스트는 시스템이 격리된 내부망에 도입되는 특성상 개발을 외부에서 수행하고, 외부 환경에서 준비한 테스트 웹 서비스와 관련 라이브러리·툴을 HDD에 담아 이관한 뒤 해당 HDD를 내부망 서버에 장착하여 설치·구성하는 방식으로 진행하였다. 〈Table 4〉는 총 약 200회 측정 결과를 제시하며, 제안 시스템은 기존 메신저 대비 안전성 98%, 사용자 만족도 92%를 기록하였다. 또한 도입 시 정기 보안 감사 대비 소요 시간을 약 40% 단축할 수 있음을 확인하였다. 이로써 대규모 투자 없이도 내부망 기반 메신저가 사이버 보안 위협에 대한 실효적 보호 수단이 될 수 있음을 실증하였다.

〈Table 4〉 Comparative Effects of the Proposed System

Speed up transmission	safety	User Satisfaction	Reduce security audit time
35%	98%	92%	40%

4. 결론

본 연구는 자본력이 충분하지 않은 중소기업을 대상으로, 외부 메신저에 대한 해킹 공격으로부터 기업의 기밀 자료를 보호할 수 있는 내부망 기반 보안 메신저 시스템을 제안한다. 기존 외부 메신저는 인터넷을 통한 통신을 수행하므로 지속적으로 공격에 노출된다. 반면 제안 시스템은 다수 중소기업이 운용 중인 분리된 업무용 내부망과 경량 Flask 서버를 활용하여, 별도 자본 투입 없이 외부 접근을 차단하고 주요 문서를 안전하게 보호하며 파일을 신속히 교환할 수 있도록 설계하였다. 특히 내부망 사용을 통해 외부 공격을 원천 차단하고, 내부 사고 발생 시 접근 로그·세션 추적으로 신속한 원인 규명이 가능하다.

제안 시스템은 Flask와 React 등 개발자 친화적 도구를 활용하여 개발 시간을 단축하였으며, 사용자 평가 결과 외부망과 격리된 내부망 사용이 안정적인 보안 성능과 메신저 기능 제공에 효과적임이 입증되었다. 또한 보안 감사 대비 시간 단축과 업무 효율성 향상을 확인하였다. 아울러 Flask와 같은 프레임워크는 환경에 맞춘 다양한 보안 설정을 적용할 수 있어, 환경별 맞춤형 보안 구성이 가능하다.

향후에는 사용자 수가 많은 기업을 대상으로 메신저 성능과 보안 성능을 추가 검증하고, 실제 외부 공격에 대한 안전성을 확인하여 시스템을 지속적으로 보강할 계획이다. 더불어 IP 기반 화이트리스트 도입, 사용자 접근 정책 강화 등 운영적 보안 절차를 고도화하여, 환경별 맞춤형 관리와 정책 적용이 가능하도록 할 예정이다. 이를 통해 사이버 위협으로부터 더욱 안전한 시스템을 구현하고, 적은 자본으로 기업 정보를 보호할 수 있는 실용적 방안을 제시하고자 한다.

REFERENCES

[1] J. Afolabi, "Cybersecurity challenges and solutions for

small businesses," 2024.

[2] "SMB Cybersecurity Report: Small and Medium Business (SMB) Cyberattacks Are Frequent and Costly," Microsoft, 2024.

[3] U. Ejaz and S. Iseal, "Cost-effective cybersecurity solutions for SMEs: Balancing security needs and budget constraints," 2024.

[4] U. Ejaz, M. Gimah, and S. Iseal, "Cybersecurity talent shortage in SMEs: Innovative approaches to recruitment and retention," 2024.

[5] S. Iseal, L. Matthews, and U. Ejaz, "The evolution of cybersecurity roles in SMEs: From IT support to strategic security management," 2024.

[6] U. Ejaz, M. Gimah, and S. Iseal, "Developing a cybersecurity incident response framework for SMEs: Workforce roles and responsibilities," 2024.

[7] W. Chi, "Web browsing technique using web scraping in a network separation environment," Ph.D. Dissertation, Dept. of Convergence Information Security, Jeju National University, Jeju, Korea, 2022.

[8] National Intelligence Service, "National network security system security guidelines (Draft)," National Intelligence Service, Jan. 2025.

[9] National Intelligence Service, "Basic guidelines for national information security," National Intelligence Service, Jan. 31, 2023.

[10] N.-E. Park, S.-H. Park, Y.-S. Oh, J.-H. Moon, and I.-G. Lee, "Distributed authentication model for secure network connectivity in network separation technology," *Sensors, Vol.*22, No.2, p.579, 2022.

[11] "Implementing Network Segmentation and Segregation," Commonwealth of Australia, 2021.

[12] H.A. Jartarghar, G.R. Salanke, A.K.A.R., S.G.S., and S.D. Dalali, "React apps with server-side rendering: Next.js," *JTEC, Vol.*14, No.4, pp.25-29, 2022.

[13] E. Hellquist, "Evaluating security for JavaScript-based frontend frameworks," M.S. Thesis, Dept. of Interaction Technology and Design, Umeå University, Sweden, 2024.

[14] M. Singh, A. Verma, A. Parasher, N. Chauhan, and G. Budhiraja, "Implementation of database using Python Flask framework," *Int. J. Eng. Comput. Sci.,* Vol.8, No.12, pp.24894-24899, 2019.

[15] M. Kang, H. Kim, D. Im, and H. Choi, "A study on weaknesses that can be introduced and secure coding for them in Flask web application framework," *in Proceedings of the Korea Information Science Society Conference (KIISE),* pp.1685-1687, 2020.

[16] J.-P. Erkkilä, "WebSocket security analysis," *Seminar on Network Security,* Aalto University, Finland, 2012.

주 수 민(Su-Min Joo) [준회원]

■ 2021년 3월 ~ 현재 : 백석대학교 컴퓨터공학부

〈관심분야〉
융합 보안, 블록체인, 머신러닝, 보안 엔지니어링, 취약점 분석

이 근 호(Keun-Ho Lee) [종신회원]

■ 2006년 8월 : 고려대학교 컴퓨터학과(이학박사)
■ 2006년 9월 ~ 2010년 2월 : 삼성전자 DMC연구소 기술전략팀 과장
■ 2010년 3월 ~ 현재 : 백석대학교 컴퓨터공학부 교수

〈관심분야〉
융합보안, 블록체인, 개인정보보호, 이동통신 보안