심재빈 SIM JAEBIN

iOS Team Lead · 스틸리언

iOS 보안 SDK를 5년간 설계·운영하며 다진 추상화 감각을, 사내 로컬 RAG·평가 체계와 프레임워크 없는 멀티에이전트 런타임에 적용하고 있습니다.

온라인 이력서 **ai.iruyo.com**

영문명	Sim Jaebin	생년월일	2001.04.24 (만 25세)
성별	남	이메일	simjabin@gmail.com
GitHub	github.com/je-empty	거주지	서울 동작구 노량진동
병역	산업기능요원 복무 완료 (2022.01 – 2023.12)		

경력	학력	자격증	포트폴리오
스틸리언	**학점은행제**	**총 10건**	**총 54건**
총 5년 1개월 · 재직중	재학중	정보처리기사 외	경력 44 · 개인 10

핵심 역량

iOS·앱 보안 SDK 5년 팀 리드

앱 바이너리(Mach-O) 가공, 코드 난독화, 위·변조 방어를 6개 제품·100+개사에 적용

SDK로 다진 추상화 설계

100여 개사가 내부를 몰라도 붙여 쓰는 보안 SDK를 만들며 익힌 인터페이스·계층 분리 감각을, 이제 AI에 적용: 검색·에이전트를 재사용 가능한 도구·커널로 추상화

사내 로컬 RAG·평가 체계 구축

임베딩·검색·생성을 직접 운영하고, 골든셋·LLM-as-judge로 품질을 측정·개선

프레임워크 없는 멀티에이전트 런타임

LangGraph·AutoGen 없이 단독 설계, HITL·Fallback·관측성 내장, 품질을 git에 추적(EDD)

기술 스택

언어	`Python` `Objective-C` `Swift` `Shell` `C`
기술 · 라이브러리	`RAG` `LLM-as-judge` `Multi-Agent` `Embeddings` `Reranker` `Mach-O` `XCFramework` `Binary Instrumentation` `Code Signing` `LIEF` `Ollama` `iOS Dynamic Library Injection` `Frida` `MVVM` `Fat Binary` `Jenkins` `vLLM` `iBeacon`
환경 · 도구	`Xcode` `PyCharm` `Hopper Disassembler` `Mach-O Viewer` `VSCode` `Claude Code` `Hex Edit`

경력

2021.06 – 재직중 · 5년 1개월 · 서울

스틸리언

iOS Team Lead

iOS 보안 솔루션 AppSuit 제품군 설계·개발·운영 총괄, 100+개사 적용

학력

2025.10 – 2026.08

학점은행제

컴퓨터공학 학사 (학위 수여 예정)

2020.03 – 2025.08

고려대학교 세종캠퍼스

컴퓨터융합소프트웨어

2017.03 – 2020.02

퇴계원 고등학교

이과계열

자기소개서

보안 개발자가 되기까지

중1 때 겪은 DDoS 사고에서 시작해, 5년차 iOS 보안 솔루션 팀 리드가 되기까지

중학교 1학년 때 직접 운영하던 마인크래프트 서버가 DDoS 공격으로 마비된 적이 있습니다. 누군가의 악의 하나에 제가 만들어 둔 공간이 그대로 무너지는 걸 보면서, 그날 처음으로 컴퓨터가 '지켜야 하는 것'이 됐습니다. 막연히 좋아하던 대상이 지켜야 할 대상으로 바뀐 순간이었고, 그때 보안을 업으로 삼겠다고 마음먹었습니다.

인문계 고등학교를 다녔지만 전공으로 삼고 싶은 분야의 활동은 학교 밖에서 스스로 찾아다녔습니다. 그렇게 쌓은 전공 역량을 인정받아 고려대학교 세종캠퍼스에 특기자 전형으로 입학했습니다. 다만 코로나로 온라인 수업이 길어지면서, 학교에 더 머무르는 것보다 빠르게 변하는 개발 시장에서 실무로 직접 부딪히는 편이 저에게 맞다고 판단했습니다. 그래서 군 대체복무가 가능한 보안 기업 스틸리언에 입사해 일찍 현장에 들어갔습니다. 학사는 학점은행제로 2026년 8월에 졸업할 예정입니다.

입사 후 처음 맡은 일부터 '되어 있는 것을 쓰는' 작업은 거의 없었습니다. 그때까지 Objective-C 앱에서만 동작하던 보호 기능을 Swift 앱까지 확장해야 했는데, 문자열 암호화와 클래스 난독화를 직접 연구해 적용 범위를 넓혔습니다. iOS가 기본으로 제공하지 않는 화면 캡처 방지 기능은 레퍼런스가 없어 처음부터 직접 만들었고, 그 결과로 특허를 한 건 출원했습니다. 이후로도 여러 보안 SDK 제품을 연구·개발했고, macOS 보안 솔루션은 신규로 만들었습니다. 그러다 보니 개발자가 SDK나 프로젝트 설정을 건드리지 않고 IPA 파일만 넣으면 암호화 및 난독화 등 보호 기능이 바로 적용되도록, 그동안 쌓은 보안 기능을 가장 쓰기 쉬운 형태로 다시 묶은 솔루션까지 만들게 됐습니다.

그렇게 AppSuit 제품군의 iOS 파트 전반을 책임지게 됐고, 100여 개사에 적용된 iOS 보안 솔루션을 운영하고 있습니다. 2025년 7월부터는 iOS 팀의 리드를 맡고 있습니다.

5년을 돌아보면 제 일하는 방식은 한 문장으로 정리됩니다. 필요한 것이 없으면 직접 만들어 현실에서 끝까지 굴린다는 것입니다. 기존 라이브러리나 플랫폼의 기능에 기대기보다 없는 부분은 정면으로 파고들어 만들어 냈고, 만든 것은 운영까지 책임졌습니다. 보안은 남이 만든 것을 그대로 믿기 어려운 분야라 내부 동작을 이해하고 직접 쌓아 올려야 합니다. 동작하는 구조를 밑바닥부터 직접 만들어 온 이 경험이 지금 제 일하는 방식의 중심이 되었습니다.

일하는 방식과 협업

팀이 나 없이도 굴러가는 구조와 검증 가능한 절차를 만드는 플레이어형 팀 리드

팀을 이끌게 된 뒤 가장 먼저 한 일은 제 머릿속에만 있던 운영 방식을 팀이 함께 볼 수 있는 체계로 옮기는 것이었습니다. 제품별로 메인·보조 담당을 정해 책임이 비지 않게 했습니다. 목표와 분장, 보드와 할 일은 노션과 지라로 표준화해, 누가 무엇을 어디까지 하고 있는지를 묻지 않아도 보이게 했습니다.

가장 공들인 부분은 '검증 가능한 절차'였습니다. 여러 고객사에서 오는 이슈는 상황도 환경도 제각각이라 말로만 주고받으면 같은 문제를 매번 처음부터 다시 풀게 됩니다. 그래서 모든 이슈를 재현 가능한 샘플로 좁히고, 원인 분석·패치·회귀 검증을 산출물로 남기게 했습니다. 담당이 바뀌어도 확인한 흔적이 이어져 품질이 사람에 묶이지 않습니다. QA도 빌드–설치–실행–크래시까지 자동화해 손으로 반복하던 확인을 절차로 바꿨습니다. 그 결과 6개월 주기의 메이저 릴리스를 안정적으로 유지하고 있습니다.

저는 관리만 하는 리드가 되고 싶지 않았습니다. 지금도 설계와 구현을 함께 하고 있으며, 문제를 '시스템으로 푸는' 감각을 유지하려 합니다. 같은 질문이 반복되고 불필요한 왕복이 생기는 구조는 본질에서 멀어집니다. 좋은 절차는 같은 질문이 다시 나오지 않게 하고, 일이 막힘없이 흘러가게 합니다. 제가 지향하는 것은 손이 덜 가는 구조이고, 그렇게 아낀 시간을 팀과 제가 정말 풀어야 할 문제에 쓰는 것입니다.

이렇게 '검증 가능한 절차로 만든다'는 습관이 몸에 배고 나니, 이후 전혀 다른 분야의 문제 앞에서도 같은 방식으로 접근하게 되었습니다.

문제 해결 경험

사내에 흩어진 정보를 직접 만든 로컬 RAG로 풀어 팀이 쓰게 만든 경험

팀을 맡고 보니 일을 막는 건 어려운 기술이 아니라 정보를 찾는 시간이었습니다. 노션·컨플루언스·지라에 흩어진 기록을 찾느라 같은 검색을 반복하는 일이 잦았습니다. 답은 사내 어딘가에 분명히 있는데, 찾는 데 시간이 너무 오래 걸렸습니다. 이를 줄이고 싶었고, 마침 취미로 파고들던 AI가 떠올랐습니다. 업무에서 마주친 문제였기에 두 분야를 붙여 보기로 했습니다.

그렇게 만든 것이 로컬 LLM 기반의 사내 검색 RAG입니다. 보안 회사 특성상 외부 API에 의존하지 않고 민감한 문서를 밖으로 내보내지 않도록 했습니다. 임베딩·리랭킹·생성 모델과 벡터 저장소는 기성 구성 요소를 썼지만, 그 위의 검색·재정렬 조합과 평가 루프, 좁은 RAM에서 여러 모델을 동시에 운용하는 제약 제어, 코드와 문서를 한 엔진에서 다루는 부분은 RAG 프레임워크에 얹지 않고 직접 설계·구현했습니다. 좁은 메모리에서 모델 경합을 통제하고 코드 검색과 문서 검색을 통합하려면 세밀한 제어가 필요했기 때문입니다.

검색 품질은 감으로 판단하지 않았습니다. 자체 골든셋 30문항으로 recall@5와 MRR을 계산하며 개선했습니다. 초기 recall@5는 86.7%였고, 문항별 오답을 분석해 관련 자료를 추가한 뒤 재색인을 반복하여 recall@5 96.7%, MRR 0.83까지 끌어올렸습니다. 회귀로 향상 폭을 확인했으며, 코드 검색 정확도는 거의 100%였습니다. 현재 10개 코퍼스, 약 3,200문서, 2만여 청크가 색인돼 있습니다.

지금은 팀에서 사용 중이며 본부 확장이 진행 중이고, 이후 전사 도입도 예정돼 있습니다. 같은 엔진 구조를 코드 리뷰 봇이 재사용하고 있어, 검색 인프라가 문서와 코드 검토 전반에 연결되는 형태로 확장되었습니다.

결과적으로, 정보를 찾는 시간을 줄이려던 시도로 시작했지만 실무 문제 해결에서 AI 적용과 운영을 한데 잇는 경험이 되었습니다. AI를 '적용'이 아니라 '실제 사용하는 구조'로 만든 이 경험이 이후 제 일하는 방식을 바꾸어 놓았습니다.

앞으로 되고 싶은 개발자

실생활의 꼭 필요한 문제를 AI로 응용해 풀어, 사용자에게 쉽게 닿게 하는 개발자

자취방의 귀찮음이 출발점이었습니다. 퇴근 후 손에 짐을 든 채 불을 켜고 에어컨을 맞추는 몇 분이 번거로웠습니다. 그래서 음성 명령 한마디로 집 기기들이 제 생활 리듬에 따라 움직이게 하는 'IntentCP'를 만들었습니다. "다녀왔어" 하면 불이 켜지고, "영화 보기 좋게 바꿔줘" 하면 조명이 어두워졌습니다. 처음엔 장난처럼 시작했지만, 기술이 생활의 속도를 따라줄 때의 편안함을 느꼈습니다. 이 경험을 유튜브 '자취남' 채널에 소개했을 때, 코드를 모르는 사람들이 흥미로워하는 모습을 보며 AI가 일상을 직접 바꿀 수도 있겠다는 생각이 들었습니다. 그 자리에서 저는 "AI가 개발자 영역까지 넘어오는 만큼, 그 AI를 잘 다루는 사람이 되고 싶다"고 말했습니다.

그 확신은 회사에서도 이어졌습니다. 흩어진 사내 정보를 찾아주는 로컬 RAG를 만들어 팀이 실제 사용하게 되었고, AI를 단순한 도구가 아니라 업무 흐름에 녹여내는 방식을 배웠습니다.

이후 'Glimi'라는 런타임 오픈소스를 직접 만들고 있습니다. LangGraph·AutoGen 같은 멀티에이전트 프레임워크 없이 각 에이전트의 기억을 데이터베이스에 영속화해 대화가 리셋되지 않도록 한 구조를 직접 설계했습니다. 메인 에이전트가 제 맥락과 선호를 반영해 역할별 담당 에이전트에게 일을 나누고, 저는 잠시 자리를 비워도 제 방향으로 시스템이 움직입니다. 개발뿐 아니라 글쓰기나 협업 등에서도 저와 같은 맥락을 가진 'AI 팀원'을 두는 구조로 확장 가능합니다. 이 과정은 기술을 쌓는 일이라기보다 혼자 일하는 시간을 줄이는 법을 찾는 과정이었습니다.

현재는 비개발 직군도 AI를 다룰 수 있도록 실용서를 집필 중입니다. 혼자 만든 것을 나누어 다른 사람의 일에도 닿게 하려는 같은 마음에서입니다.

저는 머신러닝 모델을 깊게 파는 사람은 아닙니다. 하지만 자취방의 귀차니즘, 사내 RAG, Glimi는 같은 선 위에 있습니다. 모두 실생활의 불편을 이미 있는 AI를 응용해 풀고, 사용자에게 자연스럽게 닿게 하려는 시도였습니다. 그 바탕에는 늘 같은 생각이 있었습니다. 반복되는 일은 AI가 대신하게 만들고, 사람은 정작 중요한 판단과 결정에 집중하면 된다는 것입니다. 앞으로도 사람의 일과 기술을 자연스럽게 잇는 개발자가 되고 싶습니다.

자격증

2025.06	정보처리기사	한국산업인력공단
2024.06	정보처리산업기사	한국산업인력공단
2021.12	정보처리기능사	한국산업인력공단
2021.03	1종보통 운전면허	경찰청 (도로교통공단)
2019.04	네트워크관리사 2급	한국정보통신자격협회
2019.02	3D 프린터조립전문가 2급	3D프린팅산업협회
2018.06	리눅스마스터 2급	한국정보통신진흥협회
2016.07	정보기술자격 (ITQ) 한글엑셀 (한셀) A등급	한국생산성본부
2011.12	정보기술자격 (ITQ) 한글파워포인트 (한쇼) B등급	한국생산성본부
2011.10	정보기술자격 (ITQ) 인터넷 A등급	한국생산성본부

수상

2024.08

let us:Go! 2024 해커톤 대상 — Vision Desk (Apple Vision Pro AR) · let us:Go! 운영위원회

let us:Go! 2024 해커톤 대상 (4인 팀) — Vision Desk 개발

Vision Desk는 Apple Vision Pro의 가상 작업대 위에서 캘린더, 리마인더, 사진, 지도를 AR로 띄워 보는 앱이다. 데모 letusgo2024-summer.vercel.app, 소스 github.com/letusGo-Hack/Team_6_HomP_XR_Mini.

2020.04

고려대 세종 KUDDING 프로그래밍 경진대회 수상 (1회 · 3회, 2020) · 고려대학교 세종캠퍼스

교내 KUDDING 경진대회 제1회(2020.10), 제3회(2020.12) 수상

2019.01

숭실대 고교-대학연계 심화과정 알고리즘 분석·코딩 최우수상 (2019) · 숭실대학교

숭실대 고교-대학연계 심화과정 컴퓨터과학 1 알고리즘 분석·코딩 평가 최우수상 (1위)

활동·집필·미디어

2026.01

유튜브 자취남 채널 출연 · 유튜브 자취남

직접 만든 자연어 제어 시스템(IntentCP·ShortKit)을 '자취남' 채널에서 시연

유튜브 '자취남' 채널 'IOT 천재의 집' 편 출연 (2026.01.30 게시)

2025.11

인도네시아 법인 교육 출장 · (주)스틸리언 인도네시아 법인

2025.11 인도네시아 법인 출장. iOS Premium 교육 진행, 현지 취약점 대응 포함.

iOS Premium 제품을 단독으로 현장 교육하고, 출장 중 현지 취약점 대응도 함께 처리. 만든 교육 자료는 템플릿으로 남겨 이후 국내외 교육에 재활용

2025.03 – 2025.11

KISIA S-개발자 3기 대표 멘토 · 한국정보보호산업협회 (KISIA)

KISIA 공식 프로그램 / 9개월 수행 / 멘티 팀 프로젝트 Swingft(Swift 소스코드 난독화·암호화 솔루션) 지도

2021.03 – 2021.06

제4회 KUDING 프로그래밍 경진대회 운영진 · 고려대학교 세종캠퍼스

문자열·정렬 알고리즘 문제 직접 설계·출제, 채점·운영·질의응답 담당 (2021.03~2021.06)

2019.03 – 2019.07

아두이노 지각자 체크기 제작 (퇴계원 고등학교 수업 프로젝트, 2019) · 퇴계원 고등학교

2019.03~07 퇴계원 고등학교 9인 팀, 지하철 개찰구 방식의 학급 지각자 체크기 제작

Arduino + C / 9인 팀 / 지하철 개찰구를 본떠 학급 지각자 체크기를 만든 고교 수업 프로젝트

2019.01 – 2019.02

숭실대학교 고교-대학연계 심화과정 (컴퓨터과학 1) · 숭실대학교

고교 재학 중 숭실대 고교-대학연계 심화과정(UP) 컴퓨터과학 1 수료 — Python 프로그래밍과 알고리즘 기초·분석

고교 재학 중 대학 심화과정 수강으로 Python 알고리즘 기초·분석 습득 / 이를 발판으로 알고리즘 분석·코딩 최우수상 수상

2018.03 – 2020.02

퇴계원 고등학교 프로그래밍 동아리 부장 · 퇴계원 고등학교

동아리 부원 대상 C 언어 기초 프로그래밍 및 아두이노 회로 설계 수업 진행 / 아두이노를 활용한 굴절·반사율 계측기 설계 및 제작 프로젝트 주도 / 학생동아리 융합과학 상설부스 대회에 학교 대표로 출전

경력 기술서

사내 인프라 3

2026.04 – 진행 중 · 4개월

로컬 LLM 기반 GitLab MR 코드 리뷰봇

코드를 외부로 보낼 수 없는 보안사 환경에서, 사내 TextRAG 코드 리뷰 API를 재사용해 GitLab MR 워크플로우 조립 계층을 붙인 리뷰봇입니다.

역할 사내 GitLab MR 이벤트를 받아 TextRAG의 코드 리뷰 API를 호출하고, 그 결과를 MR 코멘트로 게시하는 리뷰봇을 설계·개발·운영했습니다. 이미 있던 TextRAG 서버에 GitLab 워크플로우를 붙이는 방식으로 만들었습니다.

임팩트 TextRAG의 RAG 코드 리뷰 API를 코어 엔진으로 재사용했습니다. iOS 팀 7개 제품(Premium / Air / VPNBlock / AVSuit 등)을 대상으로 버그·보안·성능·품질·정합성 5축 리뷰를 설계하고, 실제 MR에서 자동 리뷰 코멘트가 게시되는 것을 확인했습니다. 상시 운영은 아직이며, 작동 검증까지 마친 단계입니다.

기술 `Python3` `FastAPI` `Ollama` `GitLab CI/CD`

2026.04 – 진행 중 · 4개월

TextRAG — 사내 RAG 시스템

사내 자료를 외부 LLM에 못 보내는 제약과 좁은 RAM(3모델 동시 상주 불가)을, 인덱싱·서빙 장비 분리와 메모리 예산 설계로 푼 로컬 RAG — iOS 팀에서 운영 중. 골든셋 회귀로 recall@5를 86.7%→96.7%로 개선 검증.

역할 민감한 사내 자료를 외부로 보내지 않는 보안 환경에서 로컬 RAG를 구축했습니다. 아키텍처 설계, MCP 검색 도구 구현, 좁은 RAM에 세 모델을 욱여넣는 메모리 운용, iOS 팀 도입 가이드와 교육까지 맡았습니다.

임팩트 10코퍼스·3,200여 문서·2만여 청크를 색인해 iOS 팀에서 운영 중입니다. 자체 골든셋 30문항으로 검색 품질을 측정·관리하며, 외부 소스(노션·지라)를 리모트 싱크로 붙여 약점 질의를 보강한 뒤 같은 골든셋에서 recall@5를 86.7%→96.7%·MRR 0.83으로 끌어올린 것을 측정으로 확인했습니다. 그룹 RBAC 접근제어와 쿼리 로깅 관측성을 붙였고, 같은 엔진을 사내 검색·MCP 도구·GitLab MR 리뷰봇이 함께 씁니다.

기술 `Python3` `Shell` `FastAPI` `Ollama` `vLLM`

2026.04 – 진행 중 · 4개월

사내 LLM 개발 워크플로우

사내 코드와 문서를 외부로 내보내지 않고, 추가 토큰 비용 없이 동작하는 LLM 기반 개발 워크플로우 패턴을 설계해 도입했습니다.

역할 사내 LLM 기반 개발 워크플로우(TextRAG·Claude Code·GitLab MR 리뷰봇·개인 OSS 인사이트)를 'AI 개발 워크플로우 엔지니어링' 패턴으로 구조화하고 적용을 주도했습니다. 패턴 설계 후 iOS 팀 단위로 확장했습니다.

임팩트 TextRAG를 사내에서 운영하고 GitLab MR 리뷰봇을 구축하면서, 설계·구현·리뷰·문서화 단계마다 LLM을 붙이는 패턴을 만들었습니다. 경력 DB·이력서·자소서 자동화에도 같은 구조를 썼고, book-forge(도서 원고→docx)·Glimi 등 개인 OSS 경험을 패턴으로 추출했습니다.

기술 `Python3` `FastAPI` `Ollama` `vLLM`

2026.04 – 진행 중 · 4개월

AppSuit AIModel — 온디바이스 AI 모델 보호 솔루션

온디바이스 모델 가중치가 디컴파일 시 노출되던 문제를, 고객 빌드를 변경하지 않고 AppSuit Air의 IPA 후처리 방식으로 보호 대상으로 확장했습니다.

역할 신규 제품 'AppSuit AIModel'의 초기 PoC와 기술 분석, iOS 구현, 제품소개서 작성까지 주도했습니다.

임팩트 iOS PoC에서 보호 적용 후에도 추론 결과가 동일했고 지연은 거의 없었습니다(CLIP ONNX 실기기). 내부 PoC 보고서 21장과 외부 제품소개서 12장으로 제품화됐으며 현재 고객 확보 단계입니다.

기술

`Swift` `Objective-C` `Python3` `Mach-O` `LIEF` `Dynamic Library Injection` `Code Signing` `XCFramework` `FastAPI`

2024.08 – 2024.09 · 2개월

AppSuit AVSuit — iOS 탈옥·훅킹 탐지 SDK

Premium 도입 부담을, 라이브러리 연결만으로 적용되는 경량 SDK로 줄였습니다. 탐지 반환값을 무작위로 만들어 훅 가로채기를 막았습니다.

역할 풀 기능 제품인 AppSuit Premium 도입이 부담스러운 iOS 고객을 위해 경량 탐지 SDK 'AVSuit'를 만들었습니다. 탈옥·디버깅·Frida·훅킹 4종을 잡아내는 정적 라이브러리를 설계·구현했고, 탐지 결과를 매번 다른 값으로 돌려주어 공격자가 탐지 함수를 가로채 무력화하는 우회를 막았습니다. 고객 적용 가이드와 안내 페이지까지 작성했습니다.

임팩트 출시 완료. 탈옥·디버깅·Frida·훅킹 4종 탐지 정적 라이브러리 구현. 헤더와 라이브러리만 묶어 배포해 고객은 빌드에 추가만 하면 적용. 이후 패치는 팀원에게 이관.

기술 `Swift` `Objective-C` `XCFramework` `MVVM` `Frida`

2024.06 – 2024.12 · 7개월

AppSuit Air — IPA 업로드형 SaaS 보안 솔루션

호스트 앱을 다시 빌드하거나 라이브러리를 넣을 수 없는 제약을, 서명된 IPA의 Mach-O를 후처리해 해결했습니다. 업로드만으로 보안을 입히는 SaaS 구조를 만들었고, LIEF 한계는 무파괴 재배치 구현으로 넘겼습니다.

역할 협업 프로젝트의 iOS 파트를 맡아 기획·설계·코어 개발·서버 연동·테스트·패키징 전 과정을 수행하고 후속 담당자 인수인계를 진행했습니다.

임팩트 iOS 개발, 7개월 내 제품화

- 정적 라이브러리 추가나 빌드 설정 수정 없이 IPA 업로드만으로 보안 SDK 적용 가능 → Premium 적용 범위를 도입 고객 전체로 확대
- 빌드 후처리에 의존하던 보호 기능(문자열 암호화·API 호출 은닉·리소스 복호화)을 framework 내부로 이전
- 카카오뱅크·유튜브 등 상용 앱 IPA 적용 검증 완료

기술

`Swift` `Objective-C` `Python3` `Mach-O` `LIEF` `Binary Instrumentation` `Dynamic Library Injection` `Code Signing` `Fat Binary`

2024.05 – 2024.06 · 2개월

AppSuit VPNBlock — VPN/Proxy 탐지 SDK

AOS 위주 경쟁사들이 비워 둔 iOS VPN/Proxy 탐지 영역을, VPN 앱별 탐지 매트릭스로 검증하고 Proxy 활성 탐지까지 확장해 채운 Framework

역할 LIAPP·Norton·NHN AppGuard·AppSealing 등 국내외 경쟁 솔루션을 비교 조사하고, 주요 VPN 앱들의 패키지명과 탐지 결과를 표로 정리한 탐지 매트릭스를 만들었습니다. VPN 연결 탐지에 Proxy 활성 탐지까지 확장했고, 가이드와 오픈 페이지 작성까지 맡았습니다.

임팩트 **현재 Swift 6.2로 빌드돼 안정적으로 운영 중입니다. 배포물에 SwiftUI·Swift·Objective-C 샘플이 함께 포함돼 있습니다.**

기술 `Swift` `Objective-C` `XCFramework`

2023.11 – 2023.12 · 2개월

AppSuit Premium macOS — iOS 확장

아키텍처·생명주기·패키징이 달랐던 macOS 제약을, 정적 라이브러리와 후처리 2단 구조로 x86_64용으로 다시 짜 이식했고 .pkg 재서명과 리사이너를 마무리해 Premium macOS 라인을 동일한 적용 모델로 넘겼습니다.

역할 iOS 전용이던 AppSuit Premium을 macOS로 확장하는 작업을 맡았습니다. 정적 라이브러리와 후처리 도구로 이어지는 2단 구조를 macOS 환경에 맞게 다시 설계했습니다. AppSuitSign에는 .pkg 재서명 자동화를 추가했고, 앱스토어 배포용 .pkg 리사이너는 제가 구현했습니다.

임팩트 **macOS 앱이 AppSuit Premium 보호 대상에 포함됐습니다. Big Sur부터 Sonoma까지 4개 메이저 버전을 지원합니다. AppSuitSign 릴리즈에 macOS .pkg 재서명 자동화를 추가했고, 앱스토어 배포용 .pkg 리사이너는 직접 완성했습니다.**

기술 `Python3` `Shell` `Objective-C` `Mach-O` `Code Signing`

2021.06 – 진행 중 · 5년 2개월

AppSuit Anti-Capture — 캡처·미러링 차단 SDK

iOS에서 화면 캡처·미러링을 막을 공식 API가 없어 View·Layer 계층을 직접 조작해 구현한 SDK — iPhone Mirroring 탐지는 API 없이 3개 경로를 1주 동안 비교했고, 텍스트 필드 보호 속성을 확장해 특허를 냈습니다.

역할 AppSuit Anti-Capture iOS를 공동으로 운영했습니다. iPhone Mirroring(iOS 18) 탐지를 연구했고, 원격 제어 툴 대응 검증을 맡았으며, 텍스트 필드 보호 속성을 확장한 메커니즘으로 특허를 출원했습니다.

임팩트 **'텍스트 필드 보호 속성을 이용한 스크린 캡처 및 미러링 방지 장치 및 방법' 특허 1건을 출원했습니다. iPhone Mirroring(iOS 18) 대응은 1주간 연구로 진행했으며, 관련 API가 없는 환경에서 3개 대체 경로를 비교해 권장안을 제시했습니다. 또한 원격 제어 툴을 검증하고 고객 안내 템플릿을 정립했습니다.**

기술 `Swift` `Objective-C` `XCFramework` `RxSwift` `MVVM`

2026.03 – 2026.05 · 3개월

AppSuit Premium — iOS 사용성 개편

구버전과 신버전 고객이 함께 쓰는 상황에서 하위호환을 유지하며, 흩어진 사용성 이슈 세 가지를 한 트랙으로 묶어 정비 — 스레드 네이밍 컨벤션으로 크래시 추적 시간을 1시간 이상에서 수 분으로 줄임

역할 Premium iOS 사용성 개편 세 건(2차 옵션 UX 개선, 스레드 네이밍 컨벤션 수립, 빌드 페이지 UX 안내 작성)을 본인이 주도해 설계·구현·내부 공유 문서까지 맡았습니다. 구현 방향은 팀 회의로 합의했고, 실제 실행은 팀원·타팀과 나눠 진행했습니다.

임팩트 **스레드 네이밍 공통 컨벤션을 도입해 크래시·디버그 때 기능 추적 시간을 1시간 이상에서 수 분으로 줄였습니다. 2차 옵션 사용성 정비, 탐지 정책 용어 4종 통일, 빌드 페이지 UX 개편 안내 문서까지 포함해 전체 리팩터링했고, 결과는 Premium 도입 고객 전반에 반영됐습니다.**

기술 `Swift` `Objective-C` `Python3` `XCFramework` `MVVM`

2025.11 – 2026.01 · 3개월

AppSuit Premium — 메이저 릴리즈·QA 체계 정비

6개월 주기 메이저 릴리즈를 iOS 팀과 함께 진행하며, 담당자가 바뀌어도 유지되는 QA Board와 예외처리 룰을 정비한 Premium iOS 릴리즈 트랙

역할 iOS 팀 일원으로 메이저 릴리즈의 계획, QA, 릴리즈 노트 조율에 참여하고 컴파일러 최적화 검증을 팀과 함께 수행했습니다. 릴리즈 후 QA Board 개선과 예외처리 룰 가이드를 정비했습니다.

임팩트 **메이저 릴리즈를 내면서 iOS 팀이 6개월 주기를 안정적으로 유지했습니다. 이때 정비한 QA 장치와 예외처리 룰이 다음 릴리즈 설계의 기준이 됐습니다.**

기술 `Swift` `Objective-C` `Python3` `Mach-O` `XCFramework`

2025.09 – 2025.12 · 4개월

AppSuit Premium — SSL Pinning 기능 설계

Apple 공식 API를 재구현하지 않는 조건에서, iOS 내장 SSL Pinning을 래핑해 고객이 호스트·인증서만 등록해도 MITM을 막도록 설계 — 4모듈 파이프라인 구성, 공개키 빌드 타임 고정에 따른 재빌드 필요는 안내에 명시

역할 Premium iOS의 SSL Pinning 기능을 설계 주도했습니다. iOS 내장 SSL Pinning을 래핑하는 방식을 결정하고, 정적 라이브러리(런타임 검증)·빌드 후처리·macOS 데스크톱 도구·재서명 자동화 플러그인 네 모듈이 맞물리는 구조를 잡았습니다. 구현과 가이드 일부는 팀과 함께 했습니다.

임팩트 **Premium iOS 릴리즈에 SSL Pinning 기능을 정식 포함했다. 공개키를 빌드 타임에 고정하는 방식을 써서 해외 앱 심사 체크리스트 요건을 만족했다. Charles Proxy로 MITM을 검증하는 5단계 절차를 표준화해 사내와 고객사가 같은 방식으로 확인했다. 이후 개선 회차에서 인증서 등록·재등록 흐름을 안정화했다.**

기술 `Swift` `Objective-C` `Python3` `XCFramework` `Code Signing` `Binary Instrumentation`

2025.08 – 2025.12 · 5개월

arm64e·Fat Binary 빌드 후처리 대응 — Xcode 26 Enhanced Security

Xcode 26 강화 보안으로 arm64+arm64e 통합 바이너리가 강제된 상황에서, 빌드 환경에 묶이지 않는 대응 시나리오를 지정해 고객 요청 전 선행 대응했습니다. 슬라이스 구분 없이 동일 패치 단계로 동작하도록 후처리 인프라를 확장했습니다.

역할 Xcode 26 강화 보안과 Fat Binary 대응을 선행 연구했습니다. 대응 시나리오 비교표를 만들어 방향을 정하고, Fat Binary 후처리 객체를 설계·적용했으며, Premium 6개 모듈의 호환 검증 가이드를 작성했습니다. iOS 팀 정기 회의에서 시나리오 검토를 공유했습니다.

임팩트 4개 시나리오를 비교해 하나를 미리 정해 두었고, 고객사가 강화 보안을 요청하면 1~2주 안에 대응할 수 있게 했습니다. 후처리 도구에는 Fat Binary 처리 객체를 추가해 단일 슬라이스와 통합 슬라이스를 모두 같은 패치 단계로 처리했습니다. Premium 6개 모듈의 arm64e 호환과 빌드 절차는 가이드로 정리했습니다.

기술

`Swift` `Objective-C` `Python3` `Mach-O` `LIEF` `arm64e` `Fat Binary` `Binary Instrumentation`

2025.08 – 2025.10 · 3개월

iOS 26 신규 OS 일괄 대응 — 제품 6종 일정 내 출시

Apple 9월 고정 일정 안에서 Premium·서브 제품군 6개 호환을 한 번에 확보했고, Xcode 26 Enhanced Security는 지원하지 않는다고 명시해 고객 혼선을 막은 신규 OS 대응

역할 iOS 팀 공동 대응에서 팀 명세 공유와 대응 항목 확정, 릴리즈 일정 조율을 주도했습니다. 가이드에 Enhanced Security 미지원을 명시하고, 샘플·스니펫의 NSLog를 os_log로 일괄 전환했으며, 제품별 Swift 6 호환 매트릭스를 작성했습니다.

임팩트 Premium 정규 패치를 8월 말~9월 초 예정대로 출시했고, Anti-Capture·VPNBlock·AVSuit·Keypad를 포함한 6개 제품군 업데이트를 일정 안에 완료했습니다.

기술 `Swift` `Objective-C` `XCFramework` `arm64e`

2025.02 – 2025.03 · 2개월

AppSuit Keypad — iOS 레거시 제품군 재정비

해외 금융 고객이 사용하는 레거시 Keypad Core를 수정해도 두 라인 빌드가 깨지지 않도록 호환성과 패키징을 표준화하고, 다음 담당자가 단독으로 이슈 대응이 가능하게 기준을 끌어올린 정비 작업

역할 AppSuit Keypad 제품군(Legacy Core·해외 금융 고객사 라인) 재정비를 주도했습니다. Legacy Core 프로젝트 호환화와 라이브러리 패키징 트랙을 정리해, 다음 담당자가 고객 이슈를 단독 대응할 수 있는 기반을 마련했습니다. Crypto 내장화·Swift 6 리뉴얼은 팀원이 메인으로 병행했습니다.

임팩트 iOS 팀 2025년 목표에 담당 과제로 지정됐습니다. Legacy Core를 정비하고 해외 고객 라인을 맞춰 Core 변경이 두 빌드에 안정적으로 반영되게 했습니다. 라이브러리 생성·패키징 절차를 표준화했고, 브랜치 통합을 검토해 다음 담당자가 고객 이슈를 단독으로 처리할 수 있는 수준으로 이관 기준을 맞췄습니다.

기술 `Swift` `Objective-C` `XCFramework`

2025.01 – 진행 중 · 1년 7개월

AppSuit Premium — 기능 명세·고객 가이드 표준화

제품·파일 단위로 흩어져 재사용 틀이 없던 Premium 7종과 고객 안내를, 개요·기능·검증·FAQ·릴리즈 노트 한 구조로 묶어 신규 기능도 같은 틀에 넣을 수 있게 만든 문서 표준화

역할 Premium 제품군 기능 명세·고객 가이드 문서화를 주도했습니다 — 제품별 가이드·릴리즈 노트·고객 안내 템플릿 구조를 표준화하고, 세부 작성은 팀과 협업했습니다.

임팩트 **Premium 제품군 7종 이상의 가이드 구조를 표준화하고, AppSuit Premium iOS 가이드는 릴리즈 주기에 맞춰 유지했습니다. AppSuit Air 가이드는 팀 협업으로 신규 작성했습니다. iOS 팀 제품 안내 종합본 33장의 정보구조 표준화·편성을 주도했으며, 기능 설명은 각 제품 카드에서 취합했습니다.**

2023.07 – 2023.09 · 3개월

API Hiding — ARM64 CFG 차단

Xcode·Mach-O 구조 변경으로 깨진 AppSuit Premium iOS API Hiding을 새 구조에 맞춰 3개월 안에 재작성했습니다.

역할 기존 API Hiding 로직을 분석해 바뀐 Xcode·Mach-O 구조에 맞춰 다시 작성하고, 호출 함수 수를 줄여 속도까지 끌어올렸습니다.

임팩트 **약 3개월 만에 정식 제품에 반영했고, Mach-O 바이너리의 어셈블리를 직접 읽고 고치는 Python 도구를 만들었습니다. 이후 'Dynamic API Hiding' 정식 기능으로 자리 잡았습니다.**

기술 `Objective-C` `Python3` `Mach-O` `Binary Instrumentation` `Dynamic Library Injection`

2023.03 – 2023.10 · 8개월

AppSuit Premium — Swift 바이너리 보호 3종

Swift 앱에서 문자열·클래스 이름·시스템 API 호출이 그대로 노출되던 문제를, 문자열 암호화·클래스명 난독화(2023 상반기)와 동적 API 숨김(2023 하반기)으로 막아 AppSuit Premium에 적용했습니다.

역할 Swift 앱의 문자열 암호화, 클래스 이름 난독화, 동적 API 숨김 세 보호 기능을 보안 코어 모듈(정적 라이브러리)과 빌드 후처리 단계를 묶어 설계하고 구현했습니다. 문자열 암호화와 클래스명 난독화는 2023년 상반기에, 동적 API 숨김은 그 뒤 2023년 하반기 v23.3.0에 올렸습니다. iOS 바이너리(Mach-O ARM64)를 직접 뜯어보는 분석 도구를 Python으로 만들어 결과를 검증했고, QA 자동화와 후속 인력 지도에도 같은 도구를 썼습니다.

임팩트 **Swift 보호 3종을 구현했습니다. 문자열 암호화와 클래스 이름 난독화는 2023 상반기에, 동적 API 숨김은 2023 하반기 v23.3.0에 적용했습니다. 이 기능들은 iOS 팀 제품 명세서와 고객 가이드에 정식 포함되고 검증 절차가 iOS 팀 회의에 공식 반영되었으며, AppSuit Premium 도입 고객에 배포되었습니다.**

기술 `Swift` `Objective-C` `Python3` `Mach-O` `LIEF` `Binary Instrumentation` `LC_SYMTAB`

2023.01 – 2023.03 · 3개월

Premium iOS Swift 클래스명 난독화 — Mach-O 처리에 LIEF 도입

Objective-C 클래스명만 처리하던 AppSuit Premium iOS 난독화를 Swift 클래스 심볼까지 확장하고, Mach-O 엔진을 LIEF로 바꿔 Swift 지원을 넓혔습니다.

역할 AppSuit Premium iOS의 Swift 클래스명 난독화를 맡아 Mach-O 바이너리 조작에 LIEF를 적용했습니다.

임팩트 **(__DATA_CONST, __objc_classlist)에 맹글링된 Swift 클래스 심볼을 추출해 난독화 대상으로 포함했습니다. 바이너리 write는 LIEF로 수행했습니다. LIEF 사용 시 특정 로드 커맨드 손상 문제를 찾아내고 우회했습니다.**

기술 `Python3` `LIEF` `Mach-O` `Binary Instrumentation`

2022.06 – 진행 중 · 4년 2개월

AppSuit Premium — 정적분석 방어 4종

전체 클래스 난독화가 저장 데이터 unarchive를 깨고, 문자열 암호화는 바이너리 반영을 확인할 길이 없던 제약을, 난독화 제외 규칙과 자체 Mach-O ARM64 분석 툴로 해결한 Premium 정적분석 방어 4종

역할 AppSuit Premium 보안 기능 연구를 다수 주도했습니다 — Symbol Delete(릴리즈 빌드 심볼 제거), Log Delete(디버그 로그 자동 제거), Class Name Obfuscation Exclude Rule(클래스 난독화 제외 규칙), 문자열 암호화 검증 파이프라인의 설계·연구를 단독 주도하고 구현은 팀과 협업했습니다.

임팩트 **기능별 도입 시점은 Log Delete 2022년, Framework Symbol Delete 2025년 등으로 달랐지만, 4종 모두 현행 릴리즈까지 제품에 포함돼 운영 중입니다. 릴리즈 빌드 심볼 제거로 정적 분석 난도를 높였고, 디버그 로그 자동 제거, 클래스 난독화 제외 규칙, 문자열 암호화 검증까지 직접 연구를 이끌었습니다.**

기술 `Swift` `Objective-C` `Python3` `Mach-O` `LIEF` `Binary Instrumentation` `LC_SYMTAB`

2021.10 – 진행 중 · 4년 10개월

AppSuit Premium iOS — 장기 운영

AppSuit Premium iOS 제품군 운영과 재서명·CI/CD 플러그인 유지, 고객사 이슈 대응 표준화까지 4년간 직접 주도

역할 AppSuit Premium iOS 제품군의 장기 운영과 고객사 이슈 트러블슈팅을 주도하고 있습니다. 운영 대상은 보안 코어 모듈(정적 라이브러리), 빌드 후처리 단계, 앱 재서명 도구, Xcode 빌드 파이프라인 플러그인입니다.

임팩트 **AppSuit Premium iOS를 도입한 고객 전반을 4년 이상 여러 메이저 릴리즈에서 연속 유지했습니다. 앱 재서명 도구와 빌드 자동화 플러그인도 함께 관리했습니다. 고객사 이슈는 접수부터 재현, 원인 분석, 패치, 회귀 검증, 회신까지 표준 프로세스로 묶었고, 커스텀 패치는 노션에 기록해 변경 이력을 남겼습니다.**

기술 `Objective-C` `Python3` `Mach-O`

2021.06 – 진행 중 · 5년 2개월

iOS 탈옥·훅킹 신규 변종 탐지 — Dopamine 2·RootHide·Frida

iOS 탈옥·훅킹 변종 대응 트랙에서 신규 변종 반영 우선순위와 해외 취약점 회신을 조율했습니다.

역할 iOS 탈옥·훅킹 신규 변종(Dopamine 2, Bootstrap, RootHide, TrollStore, Frida 포트 변경) 대응 트랙에서 의사결정과 고객 회신 조율을 맡았습니다. 상세 연구와 탐지 로직은 팀원이 분담했고, 저는 Premium 릴리즈 일정에 맞춰 우선순위를 조율하고 탐지 정책 설계 일부에 참여했습니다.

임팩트 **Premium 릴리즈에 Dopamine 2·RootHide 탈옥 탐지 기능을 정식 추가했습니다. 해외 고객의 취약점 회신을 조율했으며, Frida 포트 변경 우회에 대응하고 Bootstrap 탐지 연구를 등록했습니다.**

기술 `Swift` `Objective-C` `Python3` `Mach-O` `Binary Instrumentation` `Frida`

2025.01 – 진행 중 · 1년 7개월

AppSuit Toolbox — 고객 적용 지원 도구 패키지

AppSuit 적용 전 SE가 쓰던 macOS 도구를 하나로 묶자는 컨셉과 초기 UI를 제안한 AppSuit Toolbox 초기 기획 (설계·개발은 팀)

역할 AppSuit Toolbox 초기 기획에 참여했습니다. 흩어진 SE 앞단 도구를 하나의 macOS 패키지로 묶자는 컨셉과 초기 UI·기능 아이디어를 냈고, 본격 설계·개발·운영은 팀이 맡았습니다.

임팩트 **AppSuit Toolbox 제품화 초기에 분산된 SE 앞단 작업을 단일 macOS 패키지로 통합하자는 컨셉과 초기 UI·기능 아이디어를 제안**

- **이후 설계·개발·운영은 팀 담당**

기술 `Swift` `Python3` `Code Signing`

2023.11 – 진행 중 · 2년 9개월

macOS App Package Tool — .app→.pkg 패키징

메타데이터를 깨뜨리던 Python 라이브러리를 빼고, macOS 네이티브 서명 도구를 호출해 .app→.pkg 패키징·재서명을 AppSuitSign(Swift)에 넣은 macOS Premium 배포 기능

역할 Premium macOS 제품을 배포하려면 .app을 .pkg로 묶고 재서명해야 하는데, 이 단계를 AppSuitSign(Swift) 안에 직접 구현했습니다. 메타데이터를 손상시키는 Python 라이브러리 대신 macOS 네이티브 도구를 호출하는 방식으로 풀었습니다.

임팩트 **AppSuitSign GUI에서 고객 SE가 직접 패키징·재서명을 하게 됐고, 수작업이던 패키징이 표준화됐다. 2023년부터 Premium macOS 라인 도구로 운영 중이다.**

기술 `Swift` `Shell` `Code Signing`

2023.06 – 진행 중 · 3년 2개월

iOS 제품군 Jenkins CI/CD 파이프라인

제품 수가 늘면서 반복되던 수동 빌드·검증 비용과 휴먼에러를, 빌드·탐지검증·실기기 설치·로그수집을 묶은 Jenkins 파이프라인으로 줄였고 Premium·Air QA에 상시 적용했습니다 (2023~)

역할 Jenkins 기반 iOS 빌드·자동 QA 파이프라인을 2023년에 기초 구축하고, 현재 Premium·Air QA 작업에 활용하고 있습니다. iOS QA Board와 연동했습니다.

임팩트 **2023년에 Jenkins로 빌드·탐지 검증·IPA 설치·실행 로그 수집을 묶은 파이프라인을 처음 만들었습니다. libimobiledevice로 실기기 설치·로그 수집까지 자동화하고 TestFlight·내부 배포, iOS 팀 QA Board와 연동했습니다. 지금은 Premium·Air QA 작업에 실제로 쓰고 있습니다.**

기술 `Python3` `Shell` `Jenkins Pipeline`

2023.05 – 진행 중 · 3년 3개월

GitLab CI/CD 구축

2023년에 iOS 팀에서 GitLab CI/CD 도입 가능성을 검토하고 기본 파이프라인을 직접 구축했습니다. 자동 빌드와 검사는 확인했지만, 실운영에는 적용하지 않았습니다.

역할 2023년 iOS 팀에서 GitLab CI/CD 도입을 검토하며 빌드·검사 파이프라인을 직접 만들어 동작을 확인했습니다. 실사용 도입 전 탐색 단계였습니다.

임팩트 **2023년에 MR과 푸시 시점에 자동 빌드와 검사가 도는 파이프라인을 구성해 동작을 확인했습니다. Git Flow 연동과 MR 리뷰봇 연계도 검토했습니다. 실제 운영에는 도입하지 않은 탐색 단계였습니다.**

기술 `Shell` `GitLab CI/CD`

2022.09 – 2023.03 · 7개월

StealPlate — 앱스토어 심사 테스트용 iOS 앱

AppSuit 적용 앱이 App Store 심사를 통과한다는 걸 실제 심사 결과로 확인하려고 설계·운영한 테스트 앱 StealPlate — 신규 기능 검증·고객 이슈 재현 채널

역할 AppSuit를 적용한 앱이 App Store 심사를 통과한다는 걸 증명할 테스트 앱 StealPlate를 설계하고 운영하고 있습니다. Premium·Air 신규 기능은 출시 전에 여기서 먼저 심사에 올려 검증합니다.

임팩트 App Store에 등록해 실제 심사를 통과했습니다. AppSuit 적용 앱이 심사에 걸리지 않는다는 걸 사례로 보여줄 수 있게 됐습니다. Premium·Air 메이저 기능은 출시 전 이 앱으로 먼저 심사를 받아 보고, 고객 이슈가 생기면 같은 앱으로 재현해 대응합니다.

기술 `Swift` `Objective-C` `XCFramework`

2022.04 – 진행 중 · 4년 4개월

AppSuitSign — 재서명 도구

보안 적용 후 서명이 깨지고 .ipa·.pkg·.app 재서명 절차가 제각각이던 제약을, 포맷 자동 감지 단일 도구와 Developer ID 키 직접 운영으로 묶어 SE가 GUI 한 번에 끝내게 만든 AppSuitSign 입니다.

역할 2022년 4월부터 AppSuitSign의 운영과 기능 확장을 주도해 왔습니다. iOS
- macOS 통합 재서명 파이프라인을 유지보수하고, 고객 이슈에 대응하며 Toolbox 앞단과 결합하는 일을 맡았습니다.

임팩트 iOS .ipa·macOS .pkg·.app 재서명을 한 도구로 통합했습니다. Objective-C 레거시를 걷어내고 Swift·SwiftUI 로 다시 만들었으며, Developer ID Application 서명 키를 직접 운영했습니다. Toolbox 앞단과 연결해 고객 SE 가 GUI 한 번으로 재서명을 끝내게 했습니다.

기술 `Python3` `Shell` `Objective-C` `Code Signing`

2021.12 – 2022.08 · 9개월

AppSuit Premium 빌드 후처리 코어 — Python 2→3 대전환

지원 끝난 Python 2에 묶여 있던 AppSuit Premium 바이너리 후처리 코어를, 결과가 이전과 같다는 걸 확인해가며 Python 3으로 옮긴 전환 작업

역할 AppSuit Premium 빌드 산출물(Mach-O 바이너리)을 후처리하는 Python 도구를 Python 2에서 3으로 옮기는 작업을 주도했습니다.

임팩트 후처리 결과가 예전과 동일하게 나오게 맞췄다. Apple Silicon(arm64) 맥의 Python 3 환경에서 동작을 확인했다. Python 2 의존을 끊어 이후 유지보수와 최신 빌드 환경 대응이 가능해졌다.

기술 `Python3`

2021.10 – 진행 중 · 4년 10개월

CLI_XcodePlugin — 재서명·후처리 자동화 플러그인

고객사 코드 수정 없이 Xcode 빌드 과정에 끼워 넣어 서명과 재매핑을 처리하도록 만든 AppSuit Premium 후처리 자동화 플러그인 CLI_XcodePlugin (이후 iOS 팀 공동 유지)

역할 AppSuit Premium 배포의 후처리 자동화 플러그인 CLI_XcodePlugin을 처음 설계해 만들고 레거시를 정리했습니다. 이후로는 iOS 팀과 함께 유지하며 릴리즈마다 호환을 맞춥니다.

임팩트 Premium 배포 워크플로우 마지막 단계(IPA 재서명·후처리 자동화)를 맡는 플러그인을 설계·개발
- 이후 iOS 팀이 공동으로 유지·버전 대응

기술 `Python3` `Shell` `Code Signing`

PoC · 기획 2

2025.07 – 2025.12 · 6개월

MAST — 모바일 앱 취약점 점검 자동화 솔루션

AppSuit의 '처방' 앞단에서 비어 있던 '점검' 시장을, 사내 동적 탐지 자산을 재활용해 비용을 줄이며 메운 자동 진단 PoC — 금보원 19개 항목을 진단 매트릭스로 구조화, GUI·CLI 투트랙 기획

역할 신규 개념 검증(PoC) 제품 '모바일 앱 취약점 점검 자동화 솔루션(MAST)'의 구상, 기획, 경쟁 제품 분석, 일정 산정을 수행했습니다. iOS 앱 보안 운영 경험을 바탕으로 제품 제공 방식과 진단 항목 매트릭스를 설계했습니다.

임팩트 **금융보안원 평가 항목 19개에 자체 시나리오를 더해 진단 매트릭스를 구성했습니다. 경쟁 제품의 기능을 항목별로 비교하고, macOS·Windows GUI와 CLI 두 가지 방식으로 제공하도록 설계했습니다. IPA를 입력하면 진단 보고서가 생성되도록 정의했습니다. 제품화까지 약 6개월 일정을 산정해 워크시트로 정리했습니다.**

기술 `Swift` `Python3` `Frida` `libimobiledevice` `Binary Instrumentation`

2024.02 – 2024.04 · 3개월

AppSuit MacroBlock — iOS 매크로 탐지 SDK

iOS 매크로 탐지 신규 제품을 검토하다가, sandbox 제약상 실효 탐지가 스위치 제어 검증뿐이고 그것도 Premium과 겹친다는 데이터를 근거로 강행 대신 보류로 결론 낸 단독 트랙

역할 iOS 매크로 탐지 신규 제품 기획으로, 시장 조사·타사 솔루션 조사·기능 명세·개발 기간 산정을 수행하고, 'iOS OS 제약상 실효 탐지가 스위치 제어 검증 수준뿐'이라는 보류 근거를 정리해 PoC를 인계했습니다. 이후 개발 보류가 확정됐습니다.

임팩트 **경쟁사 5개사+를 매트릭스(LIAPP·NHN AppGuard·appsealing·GxShield·Norton 등)로 비교하고, iOS 매크로 부재 논거 4항목과 기능 명세·개발 기간을 정리했다. 보류 근거를 정리해 PoC를 인계하고 개발 보류로 확정했다.**

기술 `Swift` `Objective-C` `XCFramework`

팀 리딩 2

2025.07 – 진행 중 · 1년 1개월

QA 체계 구축 — QA Board·Jenkins 자동 QA

릴리즈별 QA 현황을 한 보드에서 추적하고 Jenkins 자동 검증을 붙인 iOS 팀 QA 체계

역할 iOS 팀 QA 체계를 세웠습니다. QA Board 재설계를 맡았고 Jenkins 자동 QA 파이프라인 설계와 연동을 담당했습니다.

임팩트 **QA Board를 다시 만들어 제품·차수별 QA 현황을 한 보드에서 볼 수 있게 했습니다. Jenkins로 샘플 앱 빌드·설치·탐지 검증을 자동화하고, 메이저 릴리즈 두 차례에 적용했습니다.**

기술 `Python3` `Shell` `Jenkins Pipeline`

2025.01 – 진행 중 · 1년 7개월

iOS 보안 제품 팀 운영·리딩

직급 대신 사전 과제 검증으로 배정해 1인 의존 iOS 보안 팀을 구조적으로 분산: 운영·신규 70:30, 표준 프로세스로 독립 운영 체제로 전환한 팀 리딩

역할 팀장 승격 전후로 iOS 팀 운영을 주도했습니다. 사전 과제로 역량을 보고 배정하고, 인수인계를 설계했으며 70:30 운영 체계와 개발 프로세스·회고를 세우고 본부 회의에서 팀을 대변했습니다.

임팩트 팀장이 되기 전부터 사전 과제로 팀원 기술력을 검증해 제품과 난이도를 맞춰 배정했습니다. 4개 제품군을 난이도순으로 계단식 인수인계해 1인 의존을 해소했습니다. 2025.07 승격 후에는 제품별 메인·보조·QA 배정, 운영:신규 70:30 로드맵, Git Flow·MR 템플릿·24시간 리뷰·CodeReview Guide(제품 7종 적용)를 세워 운영했습니다. 본부·리더 회의에서 iOS 팀을 대변하며 연 2~3회 메이저 릴리즈를 안정적으로 냈습니다.

개인 프로젝트

2026.04 – 진행 중

Glimi — 프레임워크 없는 멀티에이전트 런타임 커널

LangGraph·AutoGen 같은 프레임워크 없이 만든 의존성 0 멀티에이전트 런타임 커널입니다. 기억을 DB에 영속화해 모델을 바꿔도 (Claude↔로컬 Llama) 인격이 이어지고, 품질은 git에 EDD로 추적합니다.

임팩트 11주·780커밋 동안 5,000줄 모놀리스에서 의존성 0 커널을 추출했습니다. 같은 커널을 성격이 완전히 다른 두 도메인에 그대로 올려(코드 import 0) 도메인 비종속을 코드로 증명했습니다. 품질은 EDD로 git에 세대 기록해, 회귀를 숨기지 않고 PASS까지 끌어올렸습니다.

기술

`Python3` `TypeScript` `JavaScript` `Shell` `Multi-Agent` `LLM` `FastAPI` `SQLite` `WebSocket` `SSE` `Jinja2` `Cytoscape.js`

2026.06 – 진행 중

이력서 챗봇

사이트에 공개된 제 경력·프로젝트 데이터를 근거로 1인칭으로 답하는 챗봇입니다. 검색부터 답변 생성까지 로컬 서버에서 직접 돌립니다.

임팩트 외부 AI API를 쓰지 않고 로컬 서버에서 직접 돌려 운영비가 0원입니다. 2단 검색으로 관련 자료만 근거로 써, 질문과 무관한 출처가 뜨던 문제를 없앴습니다.

기술 `Python3` `TypeScript` `RAG` `LLM` `FastAPI` `ollama` `Cloudflare Tunnel` `SSE`

2026.05 – 진행 중

이력서·포트폴리오 사이트

노션 한 곳에서만 콘텐츠를 관리하고, 매시간 동기화되며 빌드·배포·PDF 생성이 자동으로 이루어지는 이력서·포트폴리오 사이트입니다.

임팩트 노션의 6개 DB에서 정적 114페이지를 자동 생성하고, 매시간 동기화합니다. 인쇄용 PDF도 동일한 데이터로 생성하며, 운영 비용은 0원입니다.

기술

`TypeScript` `Astro` `Tailwind CSS` `Notion API` `Cloudflare Workers` `GitHub Actions` `Puppeteer`

2025.12 – 2026.01

IntentCP — 말로 내린 명령을 안전하게 기기 제어로 바꾸는 중계 서버

말로 내린 명령을 서버가 곧바로 실행하지 않고, 정해진 형식(Control URL)으로 한 번 바꾼 뒤 그 형식에 정의된 동작만 실행하게 만든 오픈소스 프로젝트입니다. 자연어 해석은 폰의 Apple Intelligence와 iOS 단축어에 맡기고, 서버는 가볍게 두어 저전력 미니 서버에서 GPU 없이 상시 구동됩니다. 제어 대상은 클라우드 연동 스마트홈 기기로 켜기·끄기·밝기·상태 조회·시퀀스를 다룹니다. GitHub 22 stars (2026.05 기준).

임팩트 GitHub에 공개한 오픈소스(github.com/je-empty/IntentCP)로, 유튜브 채널 영상 노출 후 22 stars를 얻었습니다 (2026.05 기준). 지금도 자취 환경 자동화에 직접 쓰며 계속 손을 보고 있습니다.

기술 `FastAPI` `Apple Intelligence` `iOS Shortcuts` `REST API`

2025.11 – 2026.02

ShortKit — 물리 이벤트 Trigger Plane (iBeacon → Webhook)

비콘의 진입·이탈을 감지해 지정한 URL을 자동 호출하는 iOS 앱입니다. 감지와 실행을 분리해, 호출을 받을 수 있는 모든 곳과 연결할 수 있습니다.

임팩트 GitHub 공개 OSS(github.com/je-empty/ShortKit)로 2025.11.20부터 운영 중입니다. 감지 앱 ShortKit과 실행 앱 IntentCP를 함께 구성해 '비콘 감지 → 주소 호출 → 스마트홈 제어' 흐름을 구현했습니다.

기술 `iBeacon` `Webhook` `Core Location`

2019.04

데이크스트라 최단경로 C 구현

데이크스트라 알고리즘 기반 가중 그래프 최단거리 C 프로그램 (2019, 고등학교 시절)

임팩트 C, 데이크스트라 알고리즘, 가중 그래프 최단경로 (2019)

기술 `C` `Dijkstra 알고리즘`

2018.01 – 2018.06

ASUS 공유기 NAS 전환

ASUS 공유기를 NAS 서버로 전환해 Transmission, TVheadend, 웹 서버를 구축한 고등학교 시절 개인 프로젝트입니다.

임팩트 2018.01~06 개인 프로젝트

- Entware로 패키지 설치
- Transmission·TVheadend·웹 서버 3종 구축
- 만 16세에 임베디드·리눅스 환경을 학습했습니다.

기술 `Transmission` `TVHeadend`

2017.02 – 2017.11

이루요 커뮤니티 웹사이트

고등학교 때 iruyo.com 독립 도메인으로 구축한 반응형 커뮤니티 사이트 (XpressEngine, 2017)

임팩트 고등학교 재학 중 구축해 iruyo.com 독립 도메인으로 2017.02~11 약 9개월간 운영했습니다.

기술 `XpressEngine`

특허

2023.06 – 2023.12

특허 출원 — 텍스트 필드 보호 속성을 이용한 스크린 캡처 및 미러링 방지 장치 및 방법

(주)스틸리언

iOS UIView·Layer 계층과 텍스트 필드 보호 속성을 수정해 화면 캡처와 미러링을 막은 기술

출판

2026.06

비개발자 타겟 클로드 활용 도서 단독 저자

Claude 를 코딩 없이 업무에 쓰는 패턴을 비개발자용으로 정리한 단독 저서. 2026.06 출간 예정.

단독 집필 · 출판 계약 완료

병역

2022.01 – 2023.12

산업기능요원 복무 완료

재직 중 병행 복무, 2023.12 만료

---

je · empty

ai.iruyo.com

2026.07.20 12:41 (KST) 기준