AI를 이용한 개발 “잘” 하는 법

김성우

발표자 소개

2022.11~	현재 LINE+ Security R&D
2018~2022	국군정보사령부
2014	BoB 3기

주로 Agentic System을 claude code를 이용해 개발

- Agentic SAST
- Security Regulation Agent
- Secure by Design Agent

@RedCat

오늘의 주제

01 AI 할래 말래

02 AI로 개발 하는 법

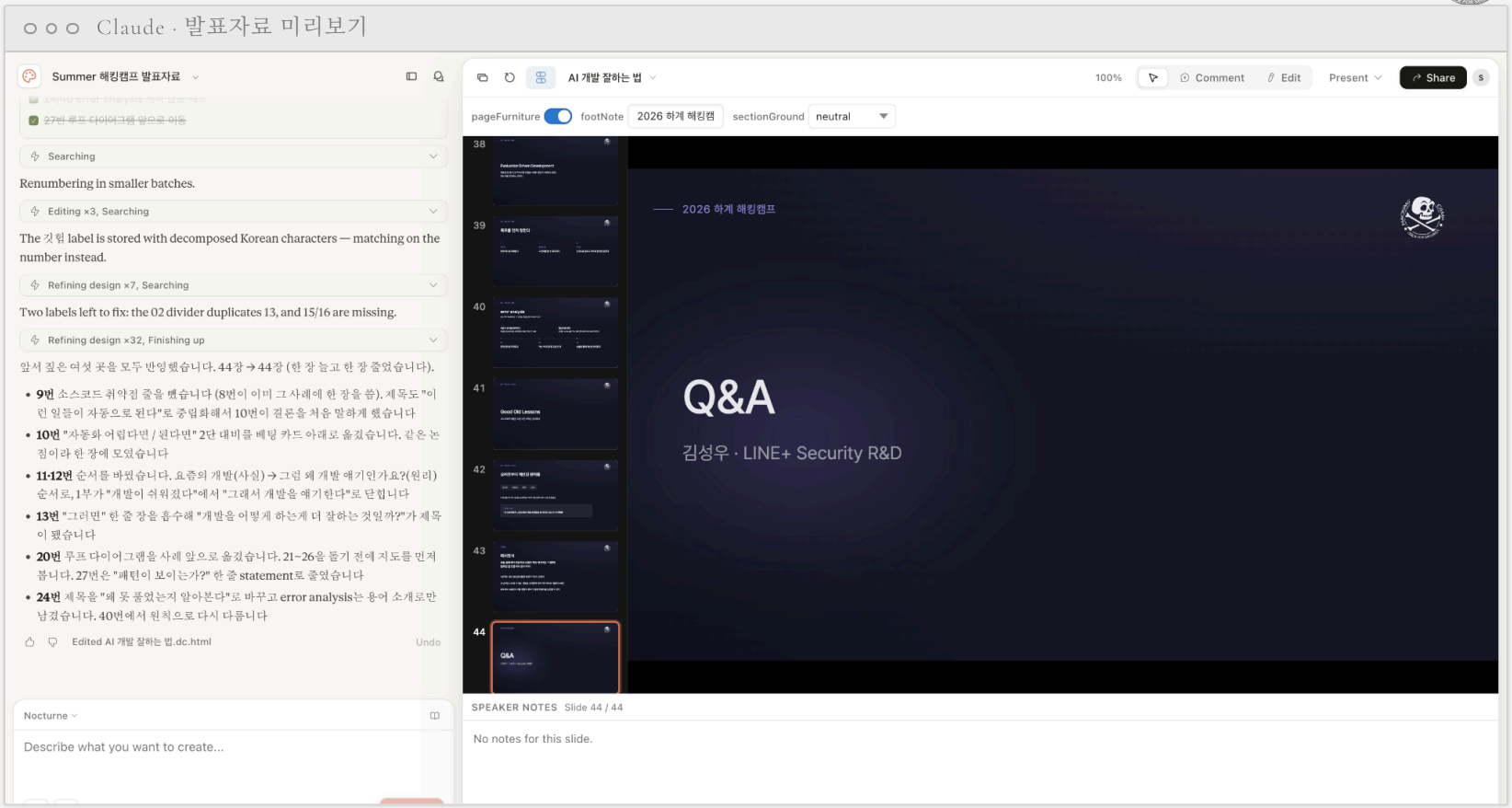
03 AI로 개발 잘 하는 법

01

AI 할래 말래


말래
말래
할래
할래
할래
말래
애매하긴해!

사실은...


Claude · 발표자료 미리보기
Summer 해킹캠프 발표자료
27번 루프 다이어그램 앞으로 이동
Searching
Renumbering in smaller batches.
Editing ×3, Searching
The 깃 힙 label is stored with decomposed Korean characters — matching on the number instead.
Refining design ×7, Searching
Two labels left to fix: the 02 divider duplicates 13, and 15/16 are missing.
Refining design ×32, Finishing up
앞서 짚은 여섯 곳을 모두 반영했습니다. 44장 → 44장 (한 장 늘고 한 장 줄었습니다).
9번 소스코드 취약점 줄을 뺐습니다 (8번이 이미 그 사례에 한 장을 씀). 제목도 "이런 일들이 자동으로 된다"로 중립화해서 10번이 결론을 처음 말하게 했습니다
10번 "자동화 어렵다면 / 된다면" 2단 대비를 베팅 카드 아래로 옮겼습니다. 같은 논점이라 한 장에 모았습니다
11·12번 순서를 바꿨습니다. 요즘의 개발(사실) → 그럼 왜 개발 얘기인가요?(원리) 순서로, 1부가 "개발이 쉬워졌다"에서 "그래서 개발을 얘기한다"로 닫힙니다
13번 "그러면" 한 줄 장을 흡수해 "개발을 어떻게 하는게 더 잘하는 것일까?"가 제목이 됐습니다
20번 루프 다이어그램을 사례 앞으로 옮겼습니다. 21~26을 돌기 전에 지도를 먼저 봅니다. 27번은 "패턴이 보이는가?" 한 줄 statement로 줄였습니다
24번 제목을 "왜 못 풀었는지 알아본다"로 바꾸고 error analysis는 용어 소개로만 남겼습니다. 40번에서 원칙으로 다시 다룹니다
Edited AI 개발 잘하는 법.dc.html
Undo
Nocturne
Describe what you want to create...
AI 개발 잘하는 법
100%
Comment
Edit
Present
Share
pageFurniture
footNote
2026 하계 해킹캠
sectionGround
neutral
2026 하계 해킹캠프
Q&A
김성우 · LINE+ Security R&D
SPEAKER NOTES
Slide 44 / 44
No notes for this slide.

우리는 왜 AI가 무서운가?

왜 AI때문에 이렇게 고민이 많이 될까?

- 기존에는 자동화하기 어려웠던 것들을 쉽게 자동화 할 수 있기 때문이다
- 자동화가 된다는 것은 그 작업에 사람이 필요하지 않다는 것
- 혹은 한 사람이 할 수 있는 일의 양이 굉장히 늘어난다는 것

우리를 무섭게 하는 것들

예시 1)

기존에는 CTF문제를 자동으로 풀게 하는 것이 어려웠다. 지금은?

CTF문제를 푸는것이 자동화가 가능하다 ⇒ CTF문제를 풀 사람이 필요할까?

예시 2)

기존에는 소스코드에서 취약점을 자동으로 찾게 하는 것이 어려웠다. 지금은?

소스코드에서 취약점을 찾는 것이 자동화가 가능하다 ⇒ 취약점 찾을 사람이 필요할까?

당장 지금은 어떨까?

- 푼다는데 못푸는것도 같고
- 다하는데 다하진 못한다는것도 같고
- 사람이 할게 없는 것 같은데 있는것도 있는 것 같은

애매한 시대

할래?

면접관	이 부분은 어떻게 처리하셨나요?
나	claude code한테 적절하게 잘 처리해달라고 했습니다.

너가 만든걸 설명도 못하고, 어떻게 했는지도
말을 못할거면 claude code를 뽑지 너를 왜뽑니?

말래?

친구	이번에 Agent를 이용해서 CVE를 25개 얻었어. CTF에서는 문제를 15개 풀었단다.
나	나는 AI없이 혼자힘으로 한문제도 풀지 못했어. CVE도 없어

AI같은걸 이용해서 아무것도 모르고 CVE를 25개 얻은 저 멍청한자를 뽑지말고 CVE가 하나도 없는 나를 뽑아라

애매하긴해

어쩌라는 거냐면요

- 지금은 어쩔 수 없이 두 가지를 동시에 준비해야 한다.
- 아니 한가지도 빡센데 두가지를?
- 아니다 여러분은 이미 고등학생때 10과목을 한번에 공부했고, 대학교에서도 7과목을 한번에 공부하고 있다. 두가지정도는 할 수 있다

어떤 두가지?

01

기본기를 탄탄하게 다지며 보안이
근본적으로 뭘 하려는 건지 고민한다

02

Agent를 이용해 아이디어를 구현한다

기본기

01 여러분이 잘 아는 CS기본기

OS, 컴파일러, 자료구조, 알고리즘, 데이터베이스, 네트워크, 프로그래밍언어, ...

02 해킹의 기본기

웹 해킹, 시스템 해킹, 리버싱, ...

End to End Red Teaming(Recon to Impact)

03 보안의 기본기

예방, 탐지, 차단, 추적

04 개발의 기본기

개발 방법론, 인프라

해킹/보안의 근본적 의미

나에 대해

- 내가 해킹을 할 줄 안다고 해서 왜 돈을 받지?
- 나한테 월급 주는 사람들은 내가 어떤 능력을 갖고 어떻게 활용하길 바랄까?

보안에 대해

- 보안이 왜 필요하지?
- 왜 이렇게 많은 사람을 필요로 하지?
- 어떤 사고사례들이 있었지?

기본기 공부 할때는 써?

문제집 풀 때 어떻게 했나요?
학원에서 어떻게 했나요?

- 먼저 혼자 해보고 안되면 물어보나요? 아니면 문제를 보는 순간 선생님한테 다 물어보나요?
- 보통 개념을 혼자 힘으로 이해 할 수 있는지 머리싸매보고
- 문제를 혼자 힘으로 풀 수 있는지 고민해본 후에 이해가 안되면 물어봅니다

똑같이 하면 됩니다.

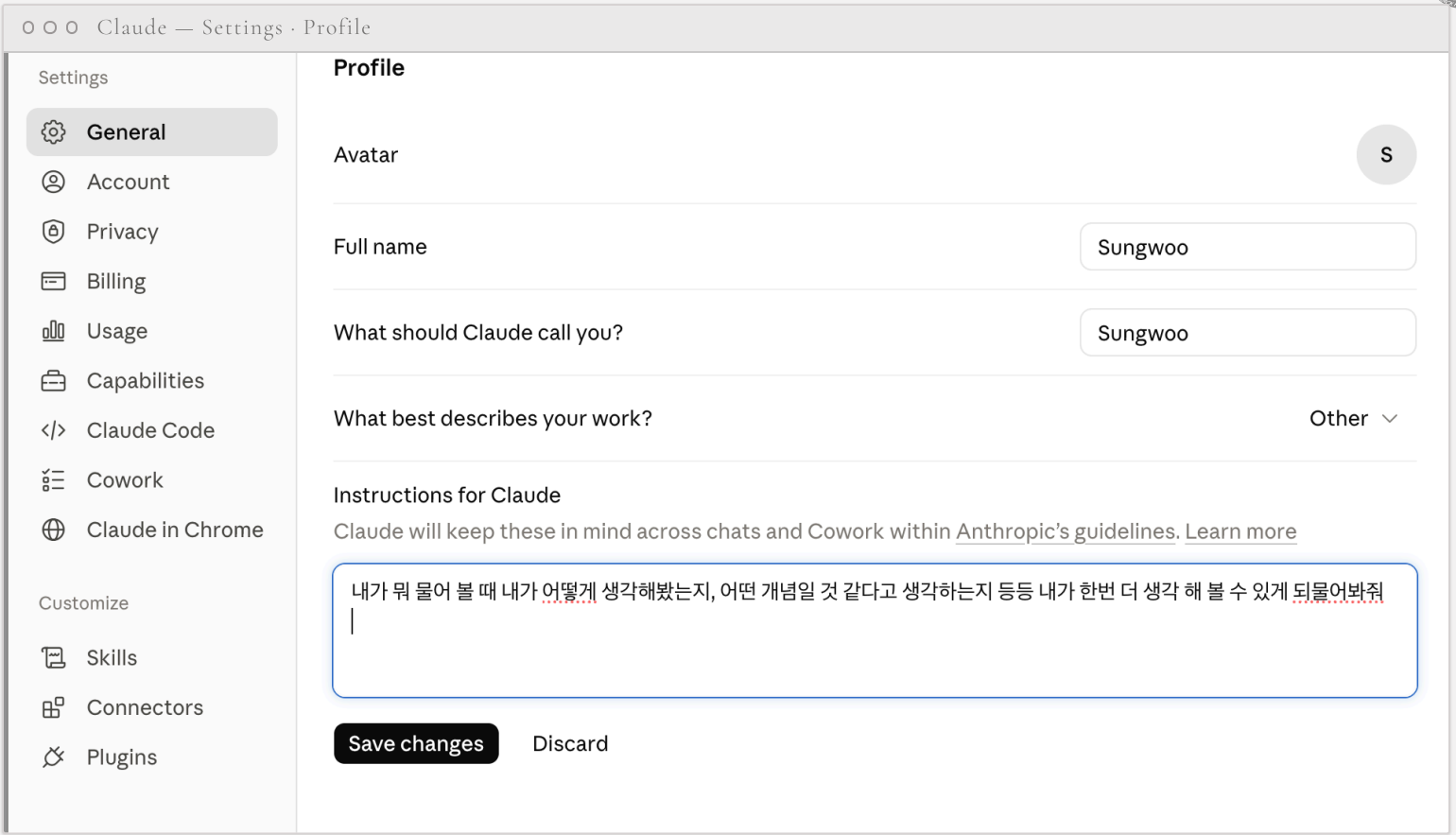
옛날 CTF 공부할 때 좋았던 방법

01 CTF 참가를 안하고 writeup을 본다고 해서 공부가 되진 않습니다

02 CTF를 참가해서 문제 하나를 48시간 잡았는데도 못 풀었을 때

03 그 다음에 writeup을 보면 훨씬 효과가 좋습니다

셀프 족쇄


Claude — Settings · Profile
Settings
General
Account
Privacy
Billing
Usage
Capabilities
Claude Code
Cowork
Claude in Chrome
Customize
Skills
Connectors
Plugins
Profile
Avatar
S
Full name
Sungwoo
What should Claude call you?
Sungwoo
What best describes your work?
Other
Instructions for Claude
Claude will keep these in mind across chats and Cowork within Anthropic's guidelines. Learn more
내가 뭐 물어 볼 때 내가 어떻게 생각해봤는지, 어떤 개념일 것 같다고 생각하는지 등등 내가 한번 더 생각 해 볼 수 있게 되물어봐줘
Save changes
Discard

02

AI로 개발 하는 법

자 그러면 원래 주제로...

AI를 이용해 어떻게 개발을 할까요?
생각보다 쉽습니다 (돈만 있다면)

01 claude plan을 결제한다(max5 이상 추천)

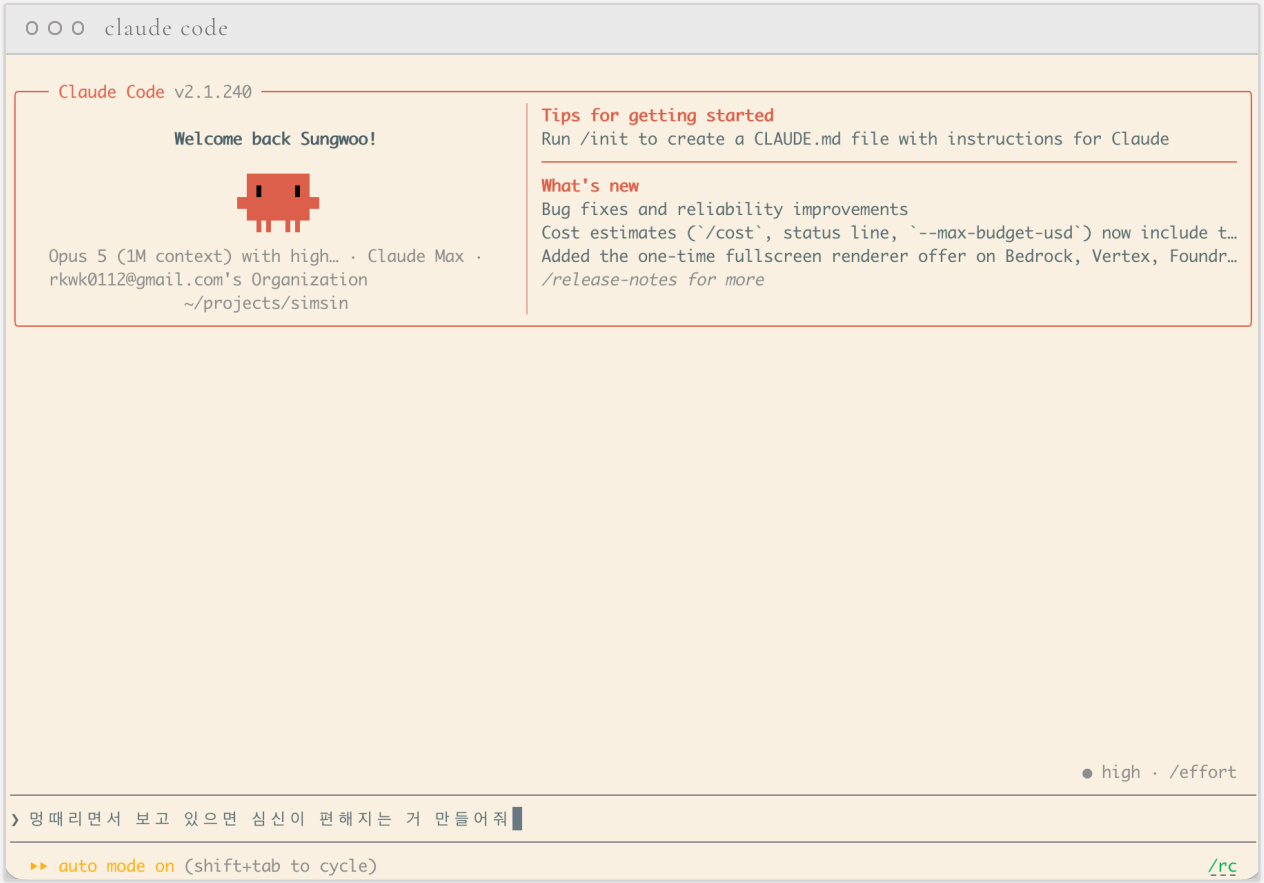
02 claude code를 설치한다

03 디렉토리를 연다

04 claude code를 실행한다

05 <만들고 싶은 것> 만들어줘 라고 한다.

해볼까요?


claude code
Claude Code v2.1.240
Welcome back Sungwoo!
Opus 5 (1M context) with high… · Claude Max ·
rkwk0112@gmail.com's Organization
~/projects/simsin
Tips for getting started
Run /init to create a CLAUDE.md file with instructions for Claude
What's new
Bug fixes and reliability improvements
Cost estimates (`/cost`, status line, `--max-budget-usd`) now include t…
Added the one-time fullscreen renderer offer on Bedrock, Vertex, Foundr…
/release-notes for more
● high · /effort
❯ 멍때리면서 보고 있으면 심신이 편해지는 거 만들어줘
▸▸ auto mode on (shift+tab to cycle)
/rc

만들어지는중...


claude code
Welcome back Sungwoo!
Opus 5 (1M context) with high… · Claude Max ·
rkwk0112@gmail.com's Organization
~/projects/simsin
Run /init to create a CLAUDE.md file with instructions for Claude
What's new
Bug fixes and reliability improvements
Cost estimates (`/cost`, status line, `--max-budget-usd`) now include t…
Added the one-time fullscreen renderer offer on Bedrock, Vertex, Foundr…
/release-notes for more
❯ 멍때리면서 보고 있으면 심신이 편해지는 거 만들어줘
Listed 1 directory
● Skill(artifact-design)
└ Successfully loaded skill
● 프로젝트가 비어 있으니 새로 만들겠습니다. 멍때리기용 앰비언트 화면 – 5개 장면이 천천히 순환하는 풀스크린 캔버스로 설계했습니다.
디자인 플랜
- Color: 단일 다크 월드로 커밋 (뷰어 테마와 무관하게 밤 화면). 바탕 #06070A, 물빛 #A4D6E5, 반딧불 호박색 #FFC46B, 새벽 남색 #16324A, 텍스트 #E8EDF2 / 흐린 글씨 #8C97A6
- Type: 고운바탕(Gowun Batang) – 장면 이름과 문구용 부드러운 명조. IBM Plex Mono Light – 시계·라벨용. Inter/Space Grotesk 회피.
- Layout: 전면 캔버스 + 크롬 없음에 가까운 오버레이. 좌하단 세로 장면 목록, 우상단 경과 시간, 4초 무동작 시 UI와 커서가 사라짐. 장면 전환은 5초 크로스페이드(캔버스 2겹).
● Running 1 shell command…
✶ Computing… (4m 41s · ↓ 19.4k tokens)
❯
▸▸ auto mode on (shift+tab to cycle) · esc to interrupt · ← for agents
/rc

완성!

○ ○ ○ claude code

● **Artifact**(/Users/cd80/projects/simsin/muyeom.html)
  ⎿ Published https://claude.ai/code/artifact/4dd04cb2-f715-4fc6-8ea6-f353299f90a7
  ⎿ Allowed by auto mode classifier

무념무상 → https://claude.ai/code/artifact/4dd04cb2-f715-4fc6-8ea6-f353299f90a7

전면 캔버스 하나에 장면 다섯 개가 3분 30초마다 5초 크로스페이드로 넘어갑니다. 아무것도 안 해도 계속 굴러가고, 4초간 손을 떼면 UI와 커서가 사라져서 화면만 남습니다.

장면
- **잔물결** — 빗방울 파문이 타원으로 퍼지며 서로를 지나감. 달빛 반사 물결 포함
- **반딧불** — 값노이즈 흐름장을 따라 떠도는 불빛, 각자 다른 주기로 깜빡임
- **번짐** — 어두운 물에 풀리는 흰 먹. 가산 합성으로 연무처럼 쌓였다 사라짐
- **진자** — 주기가 하나씩 다른 진자 30개. 58초마다 다시 한 줄로 정렬됨
- **조수** — 15초 호흡으로 부풀었다 가라앉는 수평선, 7겹 파도와 달빛 길

조작 — 화면을 누르면 그 자리에 파문/먹이 떨어지고, 반딧불은 커서를 피해 흩어집니다. SPACE 다음 장면, 1~5 직접 선택, F 전체화면, S 소리.

소리는 기본 꺼짐. 켜면 WebAudio로 브라운 노이즈를 저역 필터에 통과시켜 파도처럼 천천히 열고 닫고, 10~28초마다 5음계 종소리가 하나씩 울립니다. 외부 오디오 파일 없이 전부 생성음이라 CSP 걸릴 일 없습니다.

prefers-reduced-motion이 켜져 있으면 속도를 42%로 낮추고 자동 순환과 잔상을 끕니다. 파일은 /Users/cd80/projects/simsin/muyeom.html에 있습니다.

* Baked for 7m 30s

›

Demo

https://cd80.github.io/simsin/

좀 더 제대로 된거 없어요?


claude code
Claude Code v2.1.240
Welcome back Sungwoo!
Opus 5 (1M context) with high… · Claude Max ·
rkwk0112@gmail.com's Organization
~/projects/vulnhunter
Tips for getting started
Ask Claude to create a new app or clone a repository
What's new
Bug fixes and reliability improvements
Cost estimates (`/cost`, status line, `--max-budget-usd`) now include t…
Added the one-time fullscreen renderer offer on Bedrock, Vertex, Foundr…
/release-notes for more
❯ 취약점 분석 도구 만들어줘
▸▸ bypass permissions on (shift+tab to cycle)
/rc

그럴듯하다


claude code
Opus 5 (1M context) with high… · Claude Max ·
rkwk0112@gmail.com's Organization
~/projects/vulnhunter
What's new
Bug fixes and reliability improvements
Cost estimates (`/cost`, status line, `--max-budget-usd`) now include t…
Added the one-time fullscreen renderer offer on Bedrock, Vertex, Foundr…
/release-notes for more
❯ 취약점 분석 도구 만들어줘
Ran 1 shell command
● 빈 프로젝트네요. 방향이 크게 갈려서 두 가지만 확인할게요.
← ☒ 도구 유형 ☐ 스택 ✔ Submit →
어떤 종류의 취약점 분석 도구인가요?
1. 소스코드 정적분석 (SAST)
소스코드를 AST/패턴 기반으로 파싱해서 SQL 인젝션, 커맨드 인젝션, 하드코딩된 시크릿, 경로 조작 등 위험 패턴을 탐지. 로컬 코드베이스 대상, 외부 네트워크 불필요.
2. 의존성 취약점 스캐너 (SCA)
requirements.txt / package.json / go.mod 등을 파싱해 OSV·GitHub Advisory DB와 대조하여 알려진 CVE를 리포트. 네트워크 API 조회 필요.
3. 웹 애플리케이션 스캐너 (DAST)
실행 중인 웹앱에 요청을 보내 XSS, SQLi, 보안 헤더 누락 등을 능동 탐지. 본인 소유/승인된 대상에만 사용하는 전제.
4. 통합 프레임워크
플러그인 구조로 SAST + SCA + 시크릿 스캔을 한 CLI에 묶고, SARIF/JSON/HTML 리포트를 출력. 범위가 가장 크지만 확장 가능.
❯ 5. 요즘 AI가 취약점을 잘 찾는다던데 ✔
6. Chat about this
Enter to select · Tab/Arrow keys to navigate · ctrl+g to edit in VS Code · Esc to cancel

난 몰라 해줘


claude code
Opus 5 (1M context) with high… · Claude Max ·
rkwk0112@gmail.com's Organization
~/projects/vulnhunter
What's new
Bug fixes and reliability improvements
Cost estimates (`/cost`, status line, `--max-budget-usd`) now include t…
Added the one-time fullscreen renderer offer on Bedrock, Vertex, Foundr…
/release-notes for more
취약점 분석 도구 만들어줘
Ran 1 shell command
빈 프로젝트네요. 방향이 크게 갈려서 두 가지만 확인할게요.
← ☒ 도구 유형 □ 스택 ✔ Submit →
구현 언어와 인터페이스는?
1. Python CLI
파이썬 + Rich/argparse 기반 CLI. AST 모듈, 라이브러리 생태계가 풍부해 분석 도구에 가장 적합.
2. Go CLI
단일 바이너리로 배포, 빠른 병렬 스캔. 배포 편의성이 중요할 때.
3. TypeScript/Node CLI
JS/TS 코드베이스 분석이나 웹 대상 작업에 유리.
4. Python + 웹 UI
CLI 코어 + FastAPI 대시보드로 결과를 브라우저에서 확인.
5. 뭐가 좋을까
6. Chat about this
Enter to select · Tab/Arrow keys to navigate · ctrl+g to edit in VS Code · Esc to cancel

아...

claude code

● User answered Claude's questions:
  ⎿ · 어떤 종류의 취약점 분석 도구인가요? → 요즘 AI가 취약점을 잘 찾는다던데
    · 구현 언어와 인터페이스는? → 뭐가 좋을까

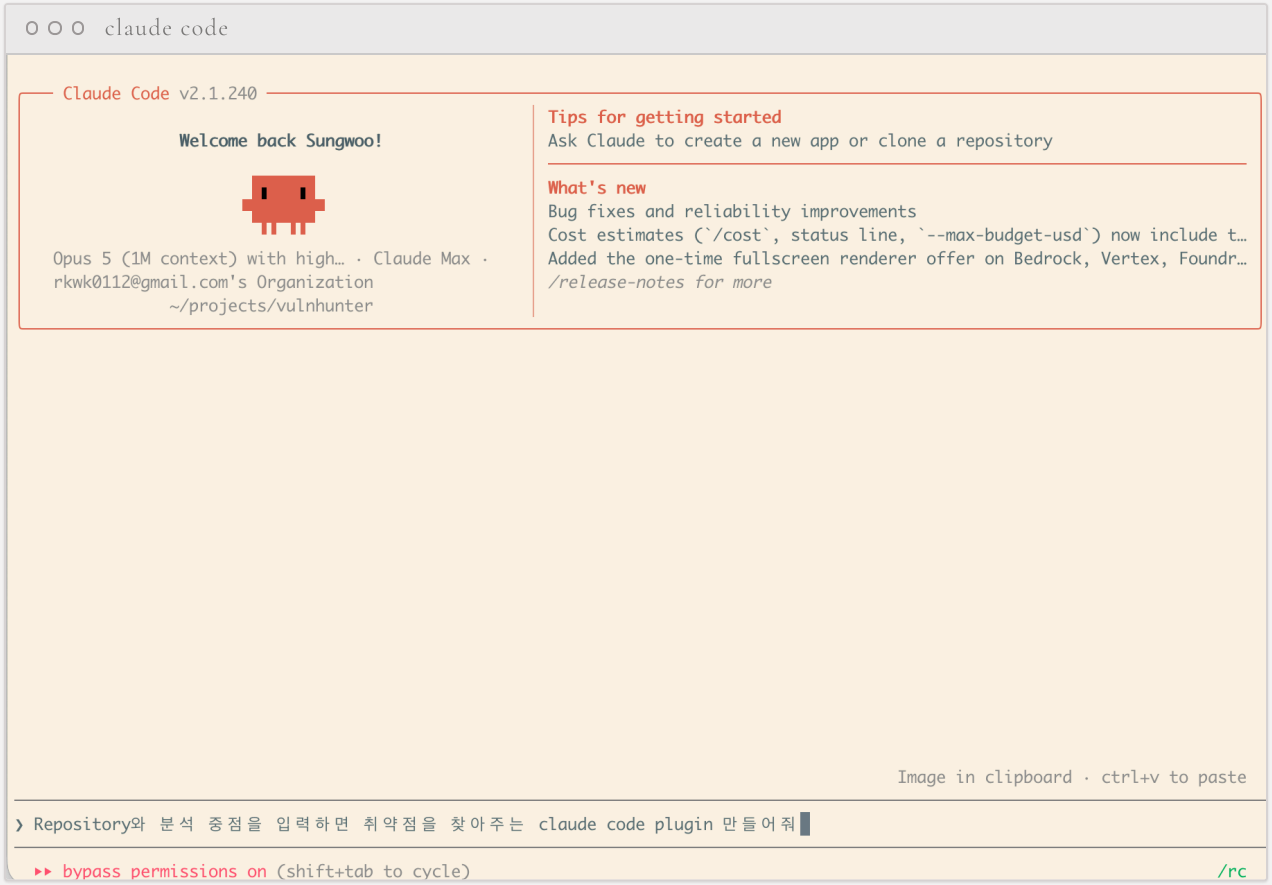
● 빈 프로젝트고, 두 답변 모두 "추천해달라"는 쪽이라 제가 정하고 만들겠습니다.

결정: Python CLI + "정적분석 → LLM 검증" 하이브리드

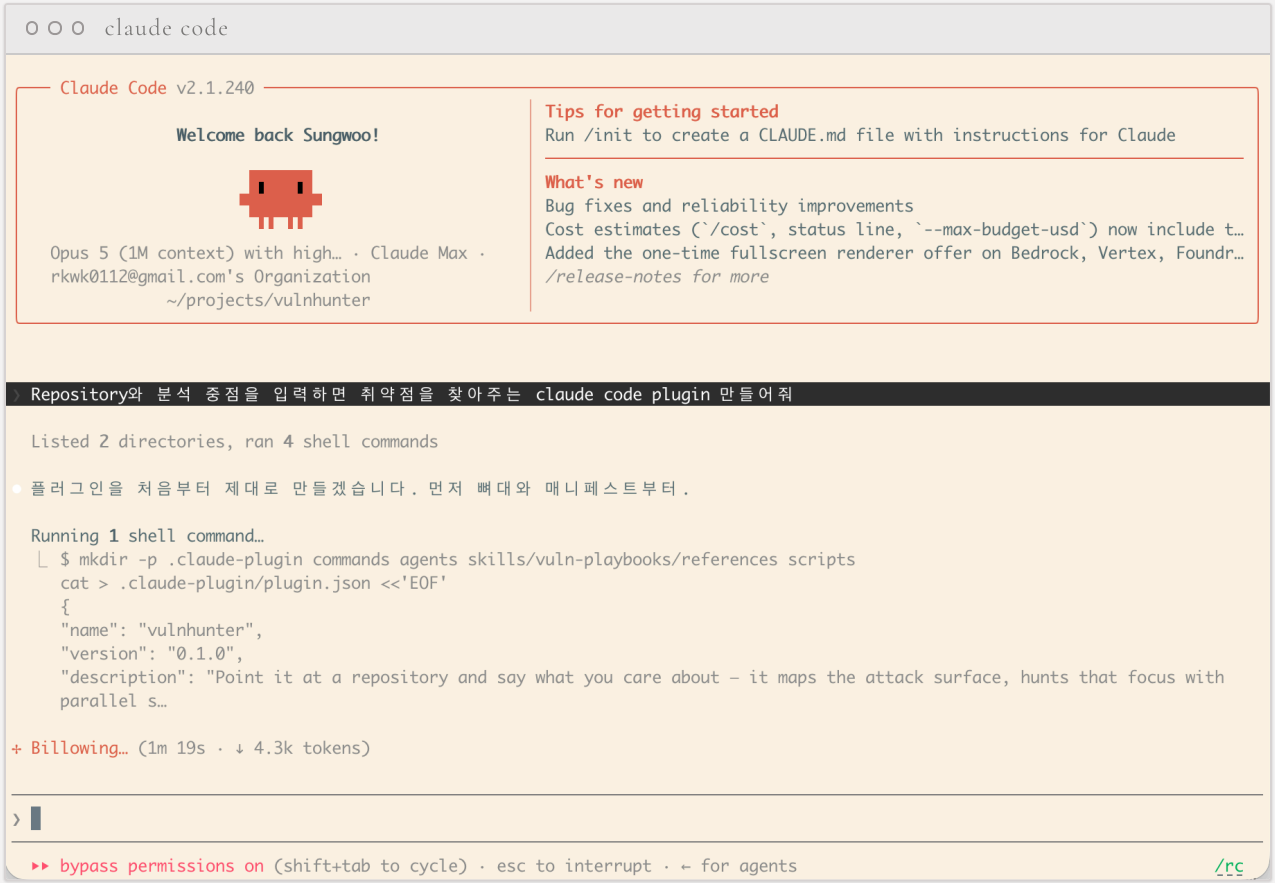
이유를 짧게: LLM에 레포 전체를 던지면 환각과 비용이 폭발합니다. 실제로 잘 동작하는 구조는 ①결정론적 분석기가 taint(오염) 추적으로 후보를 좁히고 → ②각 후보의 코드 컨텍스트만 잘라서 Claude가 "실제 도달 가능한가/새니타이즈 되는가"를 판정해 오탐을 죽이고 → ③룰이 못 잡는 인증우회·IDOR 같은 논리 취약점은 별도 LLM 심층 감사로 훑는 방식입니다. 세 층 다 넣겠습니다.

Ran 2 shell commands
  ⎿ Interrupted · What should Claude do instead?

다시


claude code
Claude Code v2.1.240
Welcome back Sungwoo!
Opus 5 (1M context) with high… · Claude Max ·
rkwk0112@gmail.com's Organization
~/projects/vulnhunter
Tips for getting started
Ask Claude to create a new app or clone a repository
What's new
Bug fixes and reliability improvements
Cost estimates (`/cost`, status line, `--max-budget-usd`) now include t…
Added the one-time fullscreen renderer offer on Bedrock, Vertex, Foundr…
/release-notes for more
Image in clipboard · ctrl+v to paste
❯ Repository와 분석 중점을 입력하면 취약점을 찾아주는 claude code plugin 만들어줘
▸▸ bypass permissions on (shift+tab to cycle)
/rc

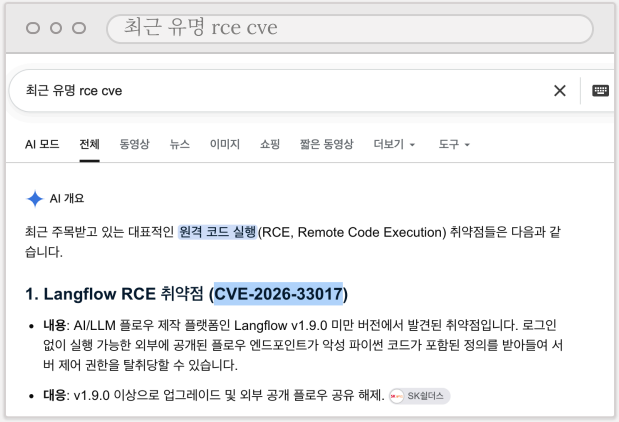
ㄱㄱ


claude code
Claude Code v2.1.240
Welcome back Sungwoo!
Opus 5 (1M context) with high… · Claude Max ·
rkwk0112@gmail.com's Organization
~/projects/vulnhunter
Tips for getting started
Run /init to create a CLAUDE.md file with instructions for Claude
What's new
Bug fixes and reliability improvements
Cost estimates (`/cost`, status line, `--max-budget-usd`) now include t…
Added the one-time fullscreen renderer offer on Bedrock, Vertex, Foundr…
/release-notes for more
› Repository와 분석 중점을 입력하면 취약점을 찾아주는 claude code plugin 만들어줘
Listed 2 directories, ran 4 shell commands
플러그인을 처음부터 제대로 만들겠습니다. 먼저 뼈대와 매니페스트부터.
Running 1 shell command…
$ mkdir -p .claude-plugin commands agents skills/vuln-playbooks/references scripts
cat > .claude-plugin/plugin.json <<'EOF'
{
"name": "vulnhunter",
"version": "0.1.0",
"description": "Point it at a repository and say what you care about — it maps the attack surface, hunts that focus with
parallel s…
✢ Billowing… (1m 19s · ↓ 4.3k tokens)
›
▸▸ bypass permissions on (shift+tab to cycle) · esc to interrupt · ← for agents
/rc


claude code
Claude Code v2.1.240
Welcome back Sungwoo!
Opus 5 (1M context) with high… · Claude Max ·
rkwk0112@gmail.com's Organization
~/projects/vulnhunter
Tips for getting started
Run /init to create a CLAUDE.md file with instructions for Claude
What's new
Bug fixes and reliability improvements
Cost estimates (`/cost`, status line, `--max-budget-usd`) now include t…
Added the one-time fullscreen renderer offer on Bedrock, Vertex, Foundr…
/release-notes for more
❯ Repository와 분석 중점을 입력하면 취약점을 찾아주는 claude code plugin 만들어줘
Listed 2 directories, ran 4 shell commands
플러그인을 처음부터 제대로 만들겠습니다. 먼저 뼈대와 매니페스트부터.
Read 1 file, ran 5 shell commands
이제 에이전트(전문가 프롬프트)를 작성합니다.
Running 1 shell command…
✢ Billowing… (4m 51s · ↓ 24.1k tokens)
Image in clipboard · ctrl+v to paste
❯
▸▸ bypass permissions on (shift+tab to cycle) · esc to interrupt · ← for agents
/rc


claude code
완성했습니다. /Users/cd80/projects/vulnhunter에 플러그인 전체가 들어있고, claude plugin validate . 통과했습니다.
.claude-plugin/plugin.json, marketplace.json
commands/ vulnhunt.md vulnhunt-verify.md vulnhunt-report.md vulnhunt-patch.md
agents/ vuln-recon.md vuln-hunter.md vuln-verifier.md vuln-patcher.md
skills/vulnerability-playbooks/ SKILL.md + references/ (10개 클래스별 플레이북)
scripts/ prepare_target.sh surface_scan.py findings.py render_report.py
README.md
사용법
/vulnhunt ~/projects/shop 결제 모듈에서 권한 우회랑 금액 조작
/vulnhunt https://github.com/org/repo SSRF
/vulnhunt . # 중점 생략 시 스윕 결과로 선택지 제시(AskUserQuestion)
핵심 설계 – 오탐을 코드로 거른다
- 3단계: 표면 스윕(결정론적 regex) → vuln-recon이 중점을 헌트 레인으로 번역 → 레인별 vuln-hunter 병렬 추적 → 후보마다 vuln-verifier 3명이 서로 다른 렌즈(도달성/방어수단/영향도)로 반증 시도.
- 판정은 모델이 안 합니다. findings.py tally가 표를 세서 confirmed(TP 2표 이상, 반대 우세 아님) / rejected / contested로 확정하고, 오케스트레이터는 이를 뒤집지 못하게 명시했습니다.
- 발견 저장은 스키마 검증을 통과해야 합니다 – attacker_control, file:line 단위 source→sink 경로가 없으면 거부(테스트로 확인). "위험해 보이는 함수"는 리포트에 못 들어갑니다.
- 플레이북 각 파일에 "흔히 틀리는 findings" 섹션을 넣어 헌터/검증자가 프레임워크 기본 방어(ORM 파라미터화, 템플릿 자동 이스케이프, 전역 미들웨어)를 오탐으로 올리지 않게 했습니다.
동작 검증: 취약 샘플 앱으로 스윕 → SQLi/command/SSRF/pickle/path 5개 레인 + 시크릿 전부 탐지, add→중복제거→투표→집계→리포트 렌더링 전 경로 스모크 테스트 통과. 에이전트 실사 파이프라인은 실제 /vulnhunt 실행에서 처음 돌아갑니다(제 쪽에서 서브에이전트는 띄우지 않았습니다).
Jump to bottom (click) ↓
›
▸▸ bypass permissions on (shift+tab to cycle) · ← for agents
/rc

테스트


최근 유명 rce cve
최근 유명 rce cve
AI 모드
전체
동영상
뉴스
이미지
쇼핑
짧은 동영상
더보기
도구
AI 개요
최근 주목받고 있는 대표적인 원격 코드 실행(RCE, Remote Code Execution) 취약점들은 다음과 같습니다.
1. Langflow RCE 취약점 (CVE-2026-33017)
내용: AI/LLM 플로우 제작 플랫폼인 Langflow v1.9.0 미만 버전에서 발견된 취약점입니다. 로그인 없이 실행 가능한 외부에 공개된 플로우 엔드포인트가 악성 파이썬 코드가 포함된 정의를 받아들여 서버 제어 권한을 탈취당할 수 있습니다.
대응: v1.9.0 이상으로 업그레이드 및 외부 공개 플로우 공유 해제.
SK쉴더스


github.com — fix: prevent RCE via data parameter in build_public_tmp endpoint #12160
fix: prevent RCE via data parameter in build_public_tmp endpoint #12160
Merged
Jkavia merged 4 commits into main from security/fix-build-public-tmp-rce on Mar 13
Conversation 15
Commits 4
Checks 99
Files changed 2
Jkavia commented on Mar 12 · edited by coderabbitai Bot
Collaborator
Security Fix: Unauthenticated RCE Prevention
Vulnerability
The POST /api/v1/build_public_tmp/{flow_id}/flow endpoint accepted attacker-controlled flow data via the data parameter, allowing unauthenticated remote code execution (RCE) through arbitrary Python code injection.
RCE: GHSA-vwmf-pq79-vjvx
Root Cause
Reviewers
coderabbitai[bot]
ogabrielluiz
Adam-Aghili
Assignees
No one assigned
Labels
bug
lgtm

대상 repository 셋업

terminal

```
https://github.com/langflow-ai/langflow/pull/12160 이 PR merge되기 전 repository 상태로 clone해줘

Ran 6 shell commands

완료했습니다.

클론 위치: /Users/cd80/projects/langflow

- PR #12160 (fix: prevent RCE via data parameter in build_public_tmp endpoint)은 2026-03-13에 squash merge 되어 머
  73b6612입니다.
- 그 부모 커밋 1fdbbfa (docs: uv included in 1.8.1 images (#12135)) 이 PR 머지 직전 상태이며, 여기에 pre-12160 브랜
  체크아웃해 두었습니다.

PR이 건드린 파일 (현재 워킹트리에는 수정 전 상태로 존재):
- src/backend/base/langflow/api/v1/chat.py
- src/backend/tests/unit/test_chat_endpoint.py (PR에서 새로 추가된 테스트라 지금은 해당 테스트 없음)

원본 main으로 돌아가려면 git checkout main, 수정 내용을 보려면 git show 73b6612 하시면 됩니다.

✻ Sautéed for 1m 55s
                                                    Image in clipboard · ctrl-
```

○ ○ ○ vuln-hunter

› 설치하고, /Users/cd80/projects/langflow 에서 POST /api/v1/build_public_tmp/{flow_id}/flow endpoint에 취약점 있는지 봐줘

Ran 4 shell commands

● 설치 완료 (skills 5, agents 4). 이제 파이프라인을 돌립니다.

Ran 4 shell commands

● **vulnhunter:vuln-recon**(Recon the public build endpoint)

Ran 4 shell commands

Ran 1 shell command

● 설치 완료했고 (vulnhunter@vulnhunter-marketplace, agents 4 / skills 5), langflow 대상 런을 시작했습니다. 표면 스윕 끝났고 vuln-recon이 헌트 플랜을 짜는 중입니다 – 끝나면 레인별 헌터를 붙입니다.


vuln-hunter
Ran 1 shell command
recon 노트가 완성됐습니다 (exec 체인까지 추적됨). 헌터를 레인별로 붙입니다.
vulnhunter:vuln-hunter(Hunt RCE via data param)
vulnhunter:vuln-hunter(Hunt principal confusion and secrets)
vulnhunter:vuln-hunter(Hunt session id cross-tenant access)
vulnhunter:vuln-hunter(Hunt files param and SSRF)
vulnhunter:vuln-hunter(Hunt public flow disclosure and DoS)
5개 레인 헌터가 병렬로 돌고 있습니다. 완료되는 대로 후보를 모아 검증 단계로 넘깁니다.
5 hunter agents scanning flow endpoint in parallel
Jump to bottom (click) ↓
· 9m 27s
bypass permissions on (shift+tab to cycle) · esc to interrupt · ← for agents · ↓ to manage
/rc
main
recon ROOT=/Users/cd80/projects/langflow RUN_DIR=/Users/... idle
lane1-rce ROOT=/Users/cd80/projects/langflow RUN_DIR=/Users/... idle
lane2-authz ROOT=/Users/cd80/projects/langflow RUN_DIR=/Users/... 10m 26s · ↓ 122.9k tokens
lane3-session ROOT=/Users/cd80/projects/langflow RUN_DIR=/Users/... idle
1 idle agent
↓ 1 more

검증중..

○ ○ ○ vuln-verifier

24개 후보가 나왔습니다. 이제 반증 검증 – 핵심 경로 후보부터 3개 렌즈로 돌립니다.

- **vulnhunter:vuln-verifier**(Verify VH-006 reachability)
- **vulnhunter:vuln-verifier**(Verify VH-006 defenses)
- **vulnhunter:vuln-verifier**(Verify VH-006 impact)
- **vulnhunter:vuln-verifier**(Verify VH-018 reachability)
- **vulnhunter:vuln-verifier**(Verify VH-018 defenses)
- **vulnhunter:vuln-verifier**(Verify VH-019 reachability)

짜잔

ㅇㅇㅇ 결과 리포트

› [file] ~/.vulnhunter/runs/langflow/20260822T153643Z/report.md (282.7KB)

결론: 있습니다. 그것도 심각하게. 후보 24건 → 검증 통과 16건 (critical 4), 반증으로 3건 기각, 5건은 포커스 밖이라 미검증.

근본 원인 하나

build_public_tmp는 **미인증 호출자의 data(그래프 전체)를 받아서** DB에 저장된 flow 대신 그걸로 그래프를 만들고, **flow 소유자 권한으로** 실행합니다. access_type=PUBLIC 검사는 *어떤 flow id를 부를 수 있는지*만 막고 *무엇이 실행되는지*는 전혀 막지 않습니다 (chat.py:633 → build.py:298 → Graph.from_payload).

Critical 4건

ID	내용
VH-006	**미인증 RCE.** 노드의 template.code가 그대로 exec() (custom/validate.py:396,442). AST 제한·import 제한·샌드박스 전부 없음. 검증자가 실제 lfx 코드를 venv에 설치해 payload로 **재현 확인.** ?event_delivery=direct면 결과가 같은 응답으로 반환
VH-018	**소유자 자격증명 탈취.** 템플릿 필드에 load_from_db:true + 원하는 변수명 → 서버가 owner의 CREDENTIAL 변수를 **복호화**해서 넘겨줌. 코드 실행 없이도 성립 (param_handler.py:193)
VH-019/VH-002	**인스턴스 전체 채팅 로그 덤프.** aget_messages에 owner 스코핑이 없어 session_id 필터만 제거하면 모든 사용자·모든 flow의 메시지가 한 요청에 (memory.py:29)
VH-012	**SQL Database 컴포넌트의 database_url이 자유 문자열** → langflow 자기 DB(user 테이블 password/is_superuser)나 내부 DB에 임의 SQL

03

AI로 개발 “잘” 하는 법

쉬워진 개발

예전이었다면 공부도, 구현도
오래걸렸을 도구를 한두시간이면
만들 수 있게 됐습니다

그러면 이 도구를 AI를 쓰지
않고 개발 할 수 있어야 할까요?

아니면 AI를 쓰고 개발하는
것으로 충분할까요?

초점의 이동

예전에는

개발을 하기 위해서 프로그래밍 언어, 알고리즘, 도메인 지식을 모두 갖추고도 개발에 시간이 오래걸렸기 때문에, “뭔가를 개발 할 수 있다” 자체가 중요했음

요즘은

개발이 너무 쉬워졌기 때문에 “좋은 결과물”을 만든다 가 더 중요해짐

경쟁력

- CTF Agent를 똑같이 만들더라도 누군가는 더 빠르고, 더 싸고, 더 잘 풀 수 있다
- Agentic SAST를 다같이 만들어도 누군가는 크롬 제로데이를 찾고, 누군가는 워게임 Easy문제도 못푼다

AI를 이용한 개발을 어떻게 “잘” 할 수 있을까?

원칙들

01 Never reinvent the wheel

이미 있는지 철저히 조사하고, 있다면 발명보다 개선에 중점

02 Build Fast, Drop Fast

길어봤자 일주일 만들어보고, 써보고 느끼고, 삭제하고 다시

03 Evaluation Driven Development

어떤 결과가 나와야 하는지를 먼저 정의하고 만든다

04 Error Analysis

못 찾았을 때, 근본 원인을 파악해서 개선시킨다

Never reinvent the wheel

"바퀴를 재발명 하지 마라"
⇒ 바퀴가 이 세상에 없다고
착각하지 마라

- 만들고자 하는 것이 이미 있지는 않은지 조사
- 있는데 만들고 싶다면? ⇒ 만들면 된다. 그 자체로 좋은 공부가 될 수 있다
- 이 원칙은 "좋은 개발" 을 하기 위한 원칙 ⇒ 내가 만든 결과물이 좋다는 것을 보장하는 방법
- 이미 있는지 철저히 조사하고, 있다면 발명/실험보다는 개선에 중점
- 직접 만든 것과 비교할 대상을 찾는 의미도 있음

Build Fast, Drop Fast

- 만드는 게 쉬워진 만큼 갈아엎고 다시 만드는 것도 쉽다
- 한번도 만들어보지 않고 상상하고 계획만 하는 것 보다
- 뭐라도 만들어보고 보면서 개선하는게 낫다
- 만들지 않고 이런거 있으면 좋겠다고 상상만 하지 말자

길어봤자 일주일 만들어보고, 써보고 느끼고,
삭제하고 다시 만들자

Evaluation Driven Development

내가 만드는 도구가 이러이러한
작업을 수행하면 이런 결과가
나와야 된단걸 먼저 정의하는 방법

- 예시) Agentic SAST를 만들면 최근 CVE 하나를 소스코드만 보고 찾을 수 있어야 한다
- 명확한 검증 기준 없이 그냥 만들기만 하면?
- 그냥 냅다 크롬에 돌리고 리눅스에 돌려서 수주의 시간, 수천만원을 쓰고나서야 잘못 만들었단 걸 알게 된다

Error Analysis

- Agentic SAST를 만들었는데, 찾아야 한다고 생각한 취약점을 못 찾는다면?
- 다행히 우리는 실전에 돌리기 전에 테스트 할 수 있는 정답지가 많이 있다
- 예시) POST /api/v1/build_public_tmp/{flow_id}/flow에 Unauthenticated RCE가 존재한다. 이 취약점은 이러이러해서 발생하고, 이것은 패치 커밋이다.
- 이런 정답지를 claude code에게 주면서 error analysis를 시켜 볼 수 있다.

---

claude code에게

"너가 지금 이러이러한 소스코드를 분석했는데, 여기서 이런 취약점이 나왔어야 했어. 근데 찾지 못했어. 취약점을 심층 분석하고, 우리 프레임워크의 구조를 분석 한 후, 왜 취약점을 찾지 못했는지 근본적인 원인을 파악해서 개선해줘"

마인드

- 나한테 필요한걸 만들자
- 나한테 쓸모 있는걸 만들자
- 사람들이 어떤걸 좋게 평가해줄까 가 중요한게 아니라, 내가 어떤걸 필요로 하냐부터
- 유용할 확률이 조금 높아짐

너도 안쓰는걸 우리보고 쓰란거냐

어떤 것을 “잘” 한다는 것

뭔가를 잘 하기 위해서는

반복

도전적 목표 설정

꺾이지 않는 마음

이 중요하다.

Q&A

들어주셔서 감사합니다

rkwk0112@gmail.com