AI for Security for AI

AI 에이전트로 AI 에이전트 해킹하기

Hacking AI agents with AI agents

류석준 (@SEORYO) · **정태영 (@ttzero25)**

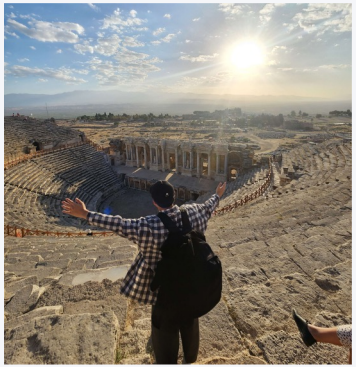
숭실대학교 ASC · Agent-Zero

33rd Hacking Camp · 2026

WHO WE ARE

발표자 소개

AGENT ZERO
ZERO-DAY RESEARCH TEAM



류석준

@SEORYO

Agent-Zero LEAD

- 숭실대학교 전자정보공학부
- 화이트햇스쿨 3기 수료
- HSPACE PhysicalLab
- AI Safety Center 학부연구생
- 29회 해킹캠프 참여

정태영

@ttzero25

Agent-Zero

- 숭실대학교 AI소프트웨어학부
- Communications System Security Lab 학부연구생
- 2025 개인정보 불법유통 대응 대학생 모니터링단
- 분야 : Physical AI · ROS · 자율주행 보안
- 31회 해킹캠프 참여

MOTIVATION

그래서 이걸 왜 하게 됐을까요?

민첩한 하루를 보내는 닌자

플젝 주제 ㅈㅈ 받습니다

오전 2:14

오전 2:14 CVE 쌀먹하고 싶어요

민첩한 하루를 보내는 닌자

그럼 오픈소스 하나 ㄱㄱ

오전 2:20

MOTIVATION · FOR REAL

그래서 이걸 왜 하게 됐을까요?

01

프롬프트 → 하네스

프롬프트로 하던 AI 버그헌팅을 에이전트 하네스 기반 자동화로 발전

02

오픈소스를 타겟

실제 프로덕션 오픈소스에서 재현 가능한 취약점을 찾는다

03

CVE로 증명

설계한 하네스를 CVE·GHSA 실적으로 증명

TIMELINE

Agent-Zero 상반기 Timeline

3–4월

팀 빌딩 ·
기초 지식 공부

5월

에이전트
설계

6–7월

에이전트 보완
본격적인 취약점 제보 시작

8월

에이전트
보완

AGENDA

목차

BACKGROUND

Background

01

BACKGROUND · 01

0-day와 1-day

취약점 존재 → 연구자·공격자 발견 → 벤더 통보 → 패치 공개 → 사용자 적용 완료

0-DAY · 패치가 존재하지 않음

1-DAY / N-DAY · 패치는 있으나 미적용

0-DAY — 패치가 없는 취약점

벤더도 모르고 방어 시그니처도 없어 탐지·차단이 어렵다.

1-DAY / N-DAY — 패치는 나왔다

공격 코드를 diff로 역산할 수 있어 공개 직후가 가장 위험

BACKGROUND · 01

1-day를 읽는 법 : CVE · CWE · CVSS

CVE

사건 번호

발견된 개별 취약점 하나에
붙는 고유 식별자

CWE

결함의 유형

취약점의 근본 원인을
분류하는 약점 목록

CVSS

심각도 점수

얼마나 위험한지를
0.1–10 으로 표준화

BACKGROUND · 01

심각도 측정, CVSS와 '정직한 점수'

Qualitative Severity Ratings

CVSS v2.0 Ratings		CVSS v3.x Ratings		CVSS v4.0 Ratings	
Severity	**Severity Score Range**	**Severity**	**Severity Score Range**	**Severity**	**Severity Score Range**
		None*	0.0	None*	0.0
Low	0.0-3.9	Low	0.1-3.9	Low	0.1-3.9
Medium	4.0-6.9	Medium	4.0-6.9	Medium	4.0-6.9
High	7.0-10.0	High	7.0-8.9	High	7.0-8.9
		Critical	9.0-10.0	Critical	9.0-10.0

CVSS 예시 — 7.7 HIGH

AV:N 네트워크 **AC:L** 낮음 **AT:P** 요구조건

PR:L 낮은권한 **UI:N** 상호작용無

C / I / A : **High**

Severity

High 7.7 / 10

CVSS v4 base metrics

Exploitability Metrics	
Attack Vector	Network
Attack Complexity	Low
Attack Requirements	Present
Privileges Required	Low
User interaction	None
Vulnerable System Impact Metrics	
Confidentiality	High
Integrity	High
Availability	High
Subsequent System Impact Metrics	
Confidentiality	None
Integrity	None
Availability	None

Learn more about base metrics

CVSS:4.0/AV:N/AC:L/AT:P/PR:L/UI:N/VC:H/VI:H/VA:H/SC:N/SI:N/SA:N

BACKGROUND · 01

GHSA — GitHub 취약점 공시 체계

① **비공개 제출**
Security 탭

→

② **Draft Advisory**
초안 생성·검토

→

③ **비공개 협의·패치**
private fork

→

④ **CVE 요청**
GitHub = CNA

→

⑤ **Publish**
패치 후 공개

→

⑥ **전파**
OSV·Dependabot

GHSA ID

GHSA-xxxx-xxxx-xxxx · GitHub 고유 식별자

CVE 없이 GHSA만 발급될 수도 있다.

일반 이슈로 올리는 순간 = 이미 공개

조율된 공개(coordinated disclosure)가 원칙

공개 시점은 유지보수자가 결정

02

AGENT ARCHITECTURE

에이전트 구조 분석

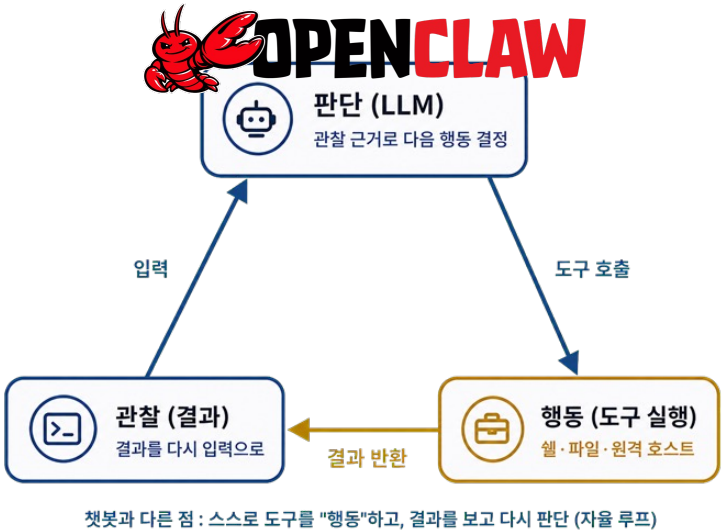
ARCHITECTURE · 02

에이전트란 무엇인가?

AI 에이전트 (Google Cloud 정의)

AI를 사용해 사용자를 대신하여 목표를 추구하고 태스크를 완료하는 소프트웨어 시스템

추론 · 계획 · 기억이 가능하며 일정 수준의 **자율성**으로 의사결정·학습·조정을 처리한다.



챗봇과 다른 점 : 스스로 도구를 "행동"하고, 결과를 보고 다시 판단 (자율 루프)

ARCHITECTURE · 02

대표적인 에이전트 : OpenClaw

셀프호스트 개인 AI 비서

사용자 기기 / 서버에서 직접 운영

메신저 내 대화형 동작

WhatsApp · Discord 등에서 상호작용

자체 런타임

외부 프레임워크 없이 자체 에이전트 루프

실제 행동

파일 · 셸 · 브라우저 등 도구를 직접 호출

ARCHITECTURE · 02

챗봇과 에이전트의 결정적 차이는 “권한”

CHATBOT

텍스트를 생성해 응답한다.

행동할 수 없어 공격 표면이 좁다.

AGENT

도구를 호출해 **shell 실행 · 파일 쓰기 · 원격 명령**이 가능.

→ 에이전트에만 존재하는 핵심 공격 표면

= **권한 처리**

ARCHITECTURE · 02

권한 중심 구조 분석

별도 프로세스 · 신뢰 경계

사용자

Auto-reply
발신자가 누구인지 판정

→

Gateway
모든 권한(인가)을 검사

→

Node-host
원격 호스트에서 OS 명령 실행

확장 서브시스템

Skills	MCP 브릿지	Sandbox	Plugin SDK
Gateway가 조율	Gateway가 조율	Gateway가 조율	Gateway가 조율

ARCHITECTURE · 02

신뢰 경계와 권한 상승

실증 관찰 (OpenClaw 코퍼스) — **모든 계단(권한 승격)마다 버그가 있을 가능성 산재**

ARCHITECTURE · 02

OpenClaw 인가의 두 가지 기준

① 스코프 — '무엇을'

`operator.read`	조회만 (상태·목록·로그)
`operator.write`	메시지 전송·툴 호출·node 릴레이
`operator.admin`	설정·업데이트·고위험 승인

② 발신자 — '누가'

아무 발신자	기본 = 신뢰 안 함
authorized	명령 실행은 허용된 대상
owner	소유자 전용 툴 권한

두 기준이 곱해지므로, 한 기준의 검사를 빠뜨린 경로가 생기면 낮은 권한이 높은 동작에 도달한다

AGENT ARCHITECTURE

에이전트 구조 분석

03

1-DAY · 03

에이전트에서 주로 발생하는 취약점은?

통념

'AI 취약점' 하면 흔히 떠올리는 것

Prompt Injection · Hallucination · Jailbreak

실제

오픈소스 에이전트 소스의 취약점은 대부분

레거시 보안 취약점

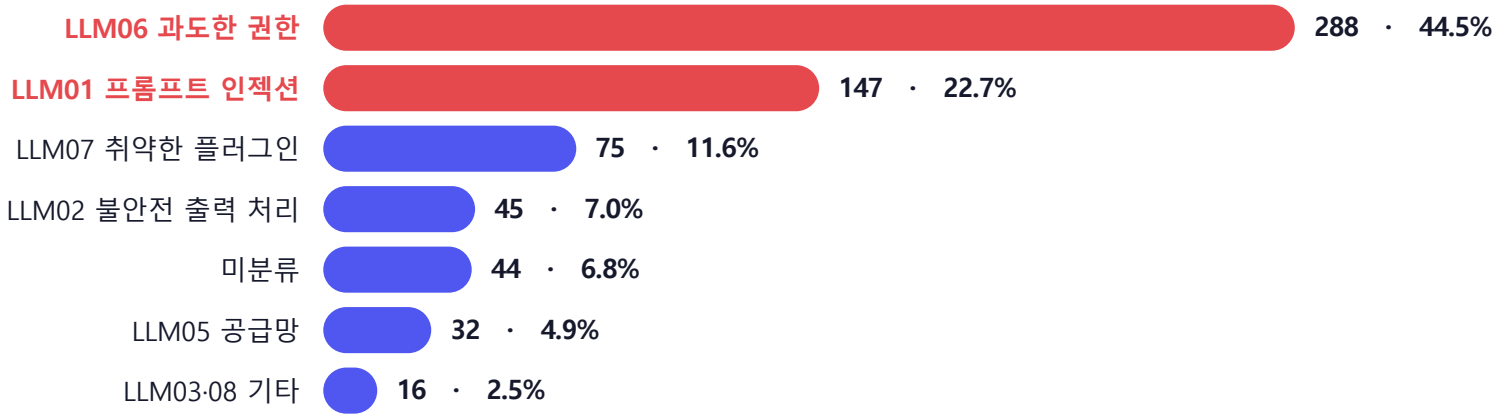
DoS · SSRF · Auth Bypass · RCE · LPE …

1-DAY · 03

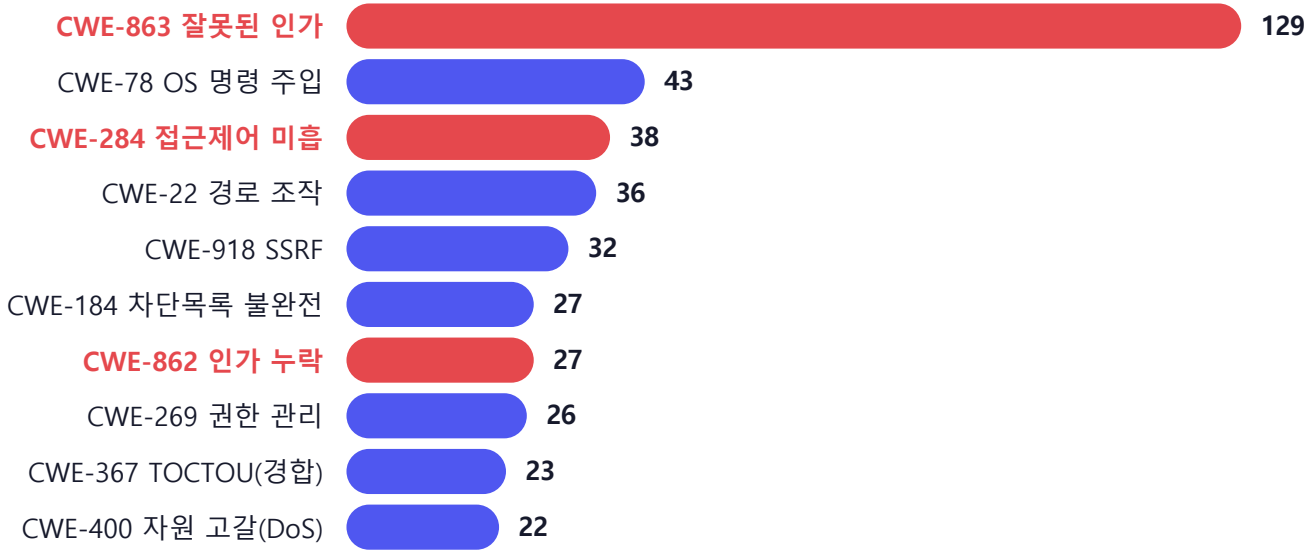
OWASP LLM Top 10 으로 본 취약점 유형

OpenClaw · 총 647건 · 5개월 간 OpenClaw GHSA 공개 취약점 (2026-02~06)

14 Critical	219 High	350 Medium	64 Low



CWE로 보면 — 인가가 대부분의 원인

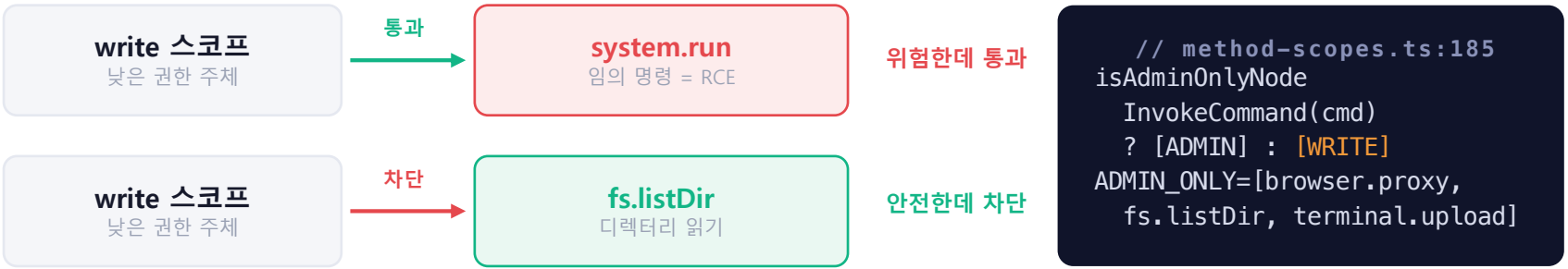


Authorization 계열(빨강)이 전체 근본 원인의 약 1/3 — 잘못된 권한 제어가 가장 큰 원인.

1-DAY · 03

OpenClaw 임의 코드 실행 취약점

권한 검사가 거꾸로 뒤집혀 있다



영향 write 스코프로 이미 페어링된 임의 호스트에서 OS 명령 실행 — 전 paired 머신 RCE.

수정 system.run 계열을 admin-only에 추가 · node-host에서 caller 스코프 재검증.

OpenClaw Flow : 메시지가 호스트 명령이 되기까지

1 발신자 인가	owner인가 / 명령 실행 허용 대상인가
2 툴 표면 결정	비-owner면 위험한 owner-only 툴 제거
3 스코프 검사	system.run ⇒ write / 나머지 ⇒ admin
4 승인 바인딩	실제 승인 레코드에 다시 묶어 위조 방지
5 Node exec 정책	deny / allowlist / 승인 최종 판단
system.run 호스트 명령 실행	**해당 표면에 도달할 경우 → RCE**

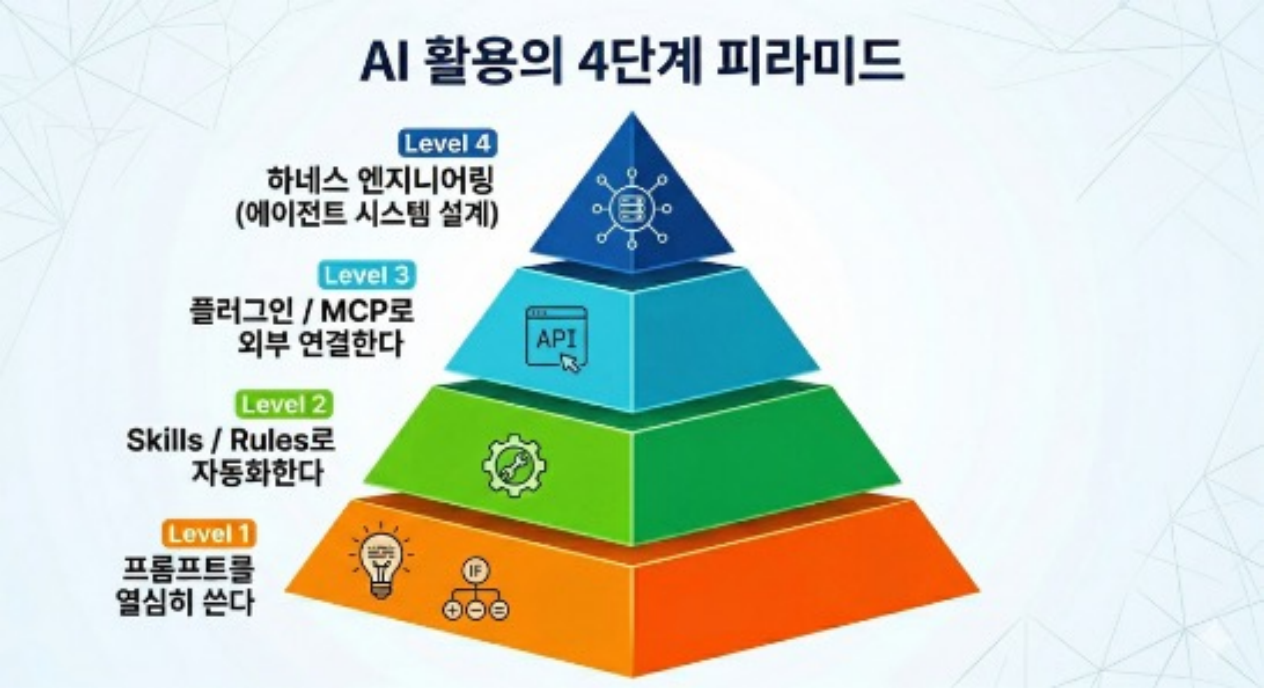
5개 관문을 순서대로 통과해야 명령이 실행된다.
한 Gate를 뚫어도 다음이 막아야 정상 — 취약점은 대개 **Gate 관리 문제**.

HARNESS DESIGN

에이전트 하네스 설계

04

에이전트를 활용하는 방법?



HARNESS · 04

에이전트 하네스란 무엇인가

모델 (LLM)

텍스트 in → 텍스트 out

호출 사이 기억 없음

스스로 도는 루프 없음

도구는 '의도'만 표현

\+

하네스로 감싸면

= 에이전트 : 하네스가 더하는 것

① 스스로 도는 루프

② 도구 실행

③ 기억·컨텍스트 관리

④ 가드레일 (안전·정책)

⑤ 트레이싱 (관찰·기록)

\+ 시스템 프롬프트·파싱

핵심 **하네스는 상수가 아니라 '변수'다 — 같은 모델도 하네스에 따라 성능·결과가 크게 달라진다. (METR)**

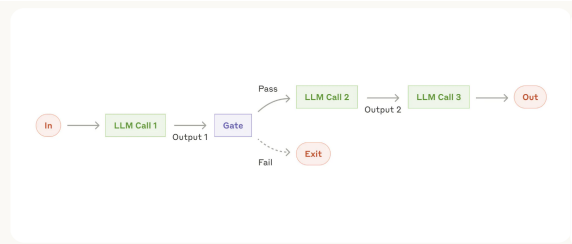
HARNESS · 04

하네스 엔지니어링 : 어떻게 설계하는가

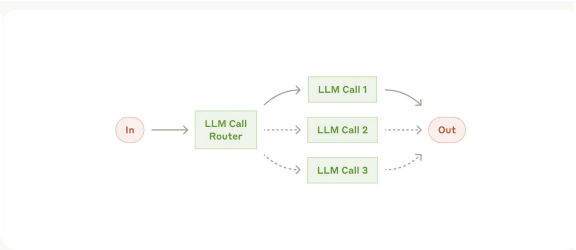
1	2	3	4	5
Prompt chaining	**Routing**	**Parallelization**	**Orchestrator–workers**	**Evaluator–optimizer**
작업을 순차 단계로 분해	입력을 분류해 전문 경로로	독립 하위작업 병렬 (분할·투표)	중앙 LLM이 하위작업을 위임	한 LLM 생성, 다른 LLM이 피드백

원칙 (Anthropic) — 가능한 단순하게. 오케스트레이션 = 단일 에이전트 루프 ↔ 멀티 에이전트.

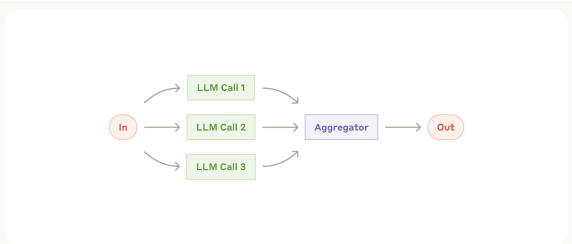
하네스 엔지니어링 : 설계 방법론



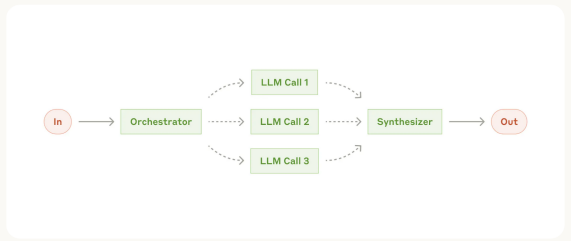
The prompt chaining workflow



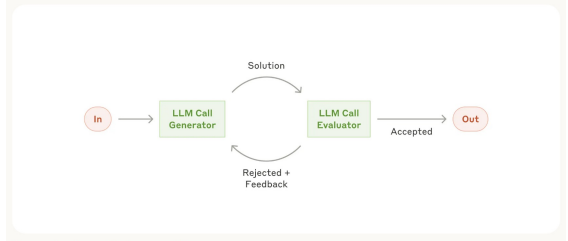
The routing workflow



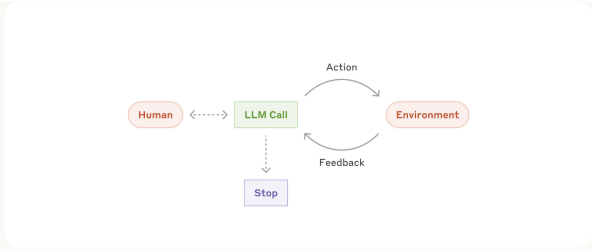
The parallelization workflow



The orchestrator-workers workflow



The evaluator-optimizer workflow



Autonomous agent

https://www.anthropic.com/engineering/building-effective-agents

HARNESS · 04

CrayFisher & DeepSealer

CrayFisher

0-day 헌팅 특화 하네스

공격자·방어자·독립 심판 3역할의 판정 구조

DeepSealer

GitHub URL → GHSA 리포트

5개 에이전트 릴레이 + 단계별 검증 게이트

HARNESS · 04

0-day 헌팅 특화 하네스 : CrayFisher

Orchestrator–workers	Orchestrator가 Recon·Defender·Judgment를 순차 호출·조율
Evaluator–optimizer	Recon 후보 → Defender 반박 → Judgment 최종 판정 (오탐 제거)
Routing	detect_stack이 유형 판별 → '에이전트' 또는 '웹앱' 분석 분기
Augmented LLM	LLM + 도구(Python) + 메모리(AZ-okf) + 검색(GHSA 코퍼스)
Guardrails	주장마다 file:line 인용 필수 · FP-check 통과 · 반박 시 폐기
Ground-truth loop	정적 분석에 그치지 않고 PoC를 실제 실행해 재현 확인

HARNESS · 04

CrayFisher : 세 가지 역할

Recon 공격자	코드를 읽고 실제 익스플로잇 가능한 후보를 구조적 증거와 함께 표면화
Defender 트리아저	각 후보에 정책 파일의 '보고 불가' 조건을 코드로 확인 - REBUTTED / PARTIAL / CONFIRMED
Judgment 독립 심판	공격·반박 양측 인용을 직접 읽고 승부 결정 - REBUTTED는 절대적, 의심되면 보류

HARNESS · 04

0-day 헌팅 특화 하네스 작동 flow

데이터 수집 도구 · JSON만 방출, 판단 없음

DETECT_STACK

ARCHITECTURE_MAP

AGENT_TRUST_GRAPH

SEMGREP_RUN

INCOMPLETE_FIX_SCAN

ENV_DENYLIST_FUZZ

HARNESS · 04

0-day 헌팅 특화 하네스 : DeepSealer

5-AGENT RELAY

Recon 후보 → **Analyzer** 흐름 추적 → **Defender** 반박 → **Judgement** 독립 판정 → **PoC-Validator** 재현

핵심은 버그 발견보다 **재현 가능성에 집중** - 앞 단계의 확신이 뒷 단계를 오염시키지 않도록 컨텍스트를 분리했다.

HARNESS · 03

DeepSealer 설계 철학 & 게이트

선행공개 게이트
OSV · GHSA · huntr · 웹 — 4소스 선행공개 확인

기본설정 게이트
기본(default) 설정에서 재현되는 경로만

크리티컬 체인
핵심 익스플로잇 체인에 도달하는지 검증

적대적 오탐 검사
반박(adversarial) 검사를 통과해야 채택

진실성 5기준
모든 finding에 PoC 실증 5기준 적용

HARNESS · 03

동적 검증 & 자가학습

버그 발견보다 재현 가능성에 집중 → 다음 헌팅을 스스로 학습한다

01

PoC 티어

DYNAMIC · HARNESS ·
STATIC_ONLY · REFUTED

02

Variant Sweep

확정 취약점의 형제(sibling)를
같은 근본원인으로 자동 스윕

03

자가학습

확정 → 취약점 패턴·가드 비대칭
반려 → 재발 오탐 사전 필터

05

RESULTS

그래서 얼마나 찾았을까?

RESULTS · 05

Agent 결과

37 유효 취약점 발견

10+ CVE 발급

20+ 버그헌팅 타겟

OpenClaw · Qbee Agent · n8n 등에서 발견

CVE-2026-35626 CVE-2026-62202 CVE-2026-65016

FUTURE WORK · 05

앞으로 뭐하지?

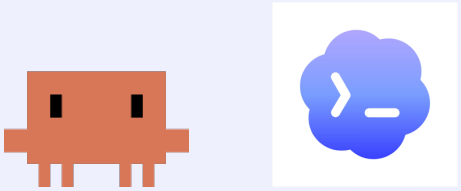
FUTURE WORK · 05

앞으로 뭐하지? - 버그바운티로 수익화

Codex ✓ In scope

$100000 | P1 $500 – $1500 | P2 $250 – $500 | P3 $100 – $250 | P4 $50 – $100

Asset	Low Avg. bounty $190 13.93% submissions	Medium Avg. bounty $1,104 43.18% submissions	High Avg. bounty $3,563 41.78% submissions	Critical Avg. bounty $8,000 1.11% submissions
Non-Core Assets	$100 – $250	$250 – $750	$1,000 – $2,500	$3,000 – $5,000
Core Assets	$250 – $750	$1,000 – $2,500	$3,000 – $5,000	$7,500 – $10,000

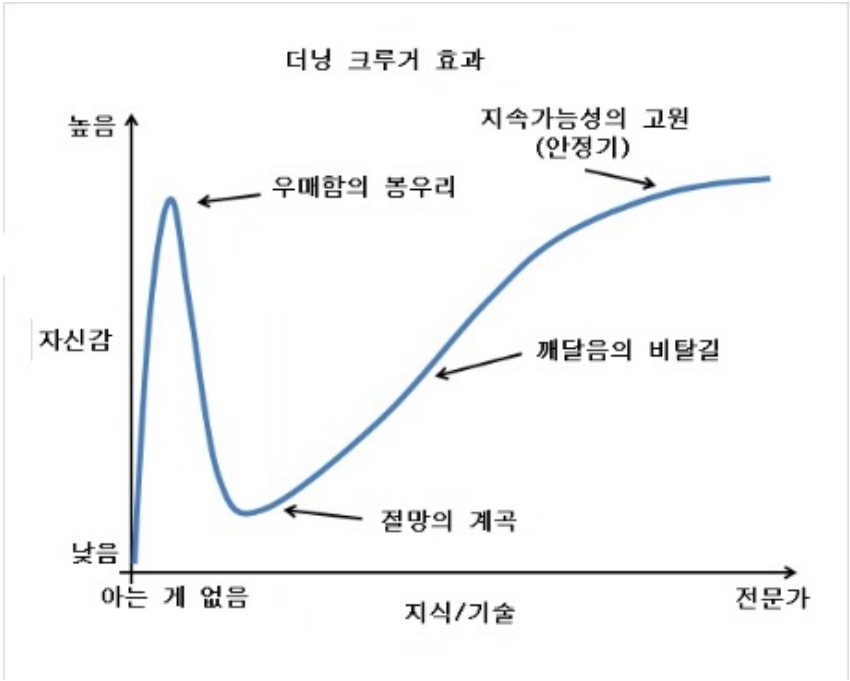


두 하네스를 바운티 프로그램에

연결해 지속 가능한 수익 구조로

CONCLUSION · 05

저희가 CVE 말고 얻은 건...



CONCLUSION · 05

저희가 CVE 말고 얻은 건...

01

모두가 딸깍에 도전하는 시대

누구나 AI로 취약점을 찾는 시대,
차별화는 기초 지식과 하네스
설계에서 갈린다

02

Duplicated와의 싸움

신규성은 결국 시간 싸움:
선행 제보와 9시간·3.5시간 차이

03

내 AI로 더 좋은 결과를

같은 모델도 하네스가 다르면
결과물이 완전히 달라진다

감사합니다 👋

류석준

EMAIL seory0@outlook.kr

DISCORD @SEORY0

INSTAGRAM @r.sjun

정태영

EMAIL teo0studio@gmail.com

DISCORD @ttzero_

INSTAGRAM @tt__zero_

Q & A

질문은 여기로!

slido.com · #2634 673