AI가 취약점 100개 찾아왔습니다.
이제 어쩌죠?

KRUG Security Meetup 2026

김민지 - AWS ProServe Assoc. Delivery Consultant

AI 의 시대

- 60%의 조직이 매주 웹 앱을 업데이트
- 75%는 보안 테스트를 월 1회 이하로만 수행
- **81%가 취약한 코드를 알면서도 릴리즈**
- 수동 모의해킹: 3~6주 소요, $5K~$50K/앱

"클릭 한 번이면 됩니다"

"여러분, 취약점 분석하느라 고생많으셨죠?
이제는 AI 서비스를 활용하면 클릭 한번에 취약점 분석이 가능합니다."

최근 이런 서비스들 많이 보셨을 겁니다.

근데...

→ 찾는 건 쉬워졌는데, 찾은 다음은?

오늘의 주제

"이번 시간에는 AWS Security Agent를 살펴보며,
***AI 시대에 보안 전문가의 역할**이 어떻게 변하는지 이야기해보겠습니다."*

아젠다:

- AWS Security Agent란?
- 기능 & 사용 방법
- 데모
- 결과 후 실무 워크플로우
- 보안 전문가의 역할 변화

기존 도구의 한계

도구	하는 일	한계
SAST	패턴 매칭 기반 코드 분석	높은 False Positive, 런타임/로직 결함 못 잡음
DAST	동적 스캐닝	코드 매핑 불가, 모던 프레임워크에서 깨짐
SCA	오픈소스 취약점 탐지	커스텀 코드는 무시
수동 모의해킹	전문가가 직접 테스트	3~6주 소요, 연 1~2회, 스케일 불가

→ 각 도구가 보안의 한 면만 봄. 전체를 아우르는 도구가 없었음.

AWS Security Agent란?

AWS Continuum 플랫폼의 일부

AI Agent 기반으로 보안 취약점을 자율적으로 발견 · 검증 · 수정

- Design Review - 설계 단계에서 보안 검토 (분 단위)
- Code Review - PR 기반 정적분석 (Near Real-Time)
- Penetration Testing - AI 기반 동적 침투 테스트 (시간 단위)
- (Beta) Threat Modeling – STRIDE 기반 위협 모델 생성

→ **정적분석 + 동적분석을 하나의 Agent가 커버**

차별점 - 왜 기존 도구와 다른가?

기존 도구:

- 패턴 매칭
- 컨텍스트 없음 → False Positive 폭탄
- 정적/동적 분리 → 연결된 취약점 못 잡음

Security Agent:

- 전체 리포지토리 컨텍스트 이해
- 정적분석 결과를 동적 테스트로 검증
- 비즈니스 로직 결함까지 탐지
- 취약점 체이닝:
  다단계 공격 시나리오 탐지

Security Requirements Pack

Agent 에게 "이 기준으로 검토해줘"라고 알려주는 룰셋

AWS 관리형 팩:

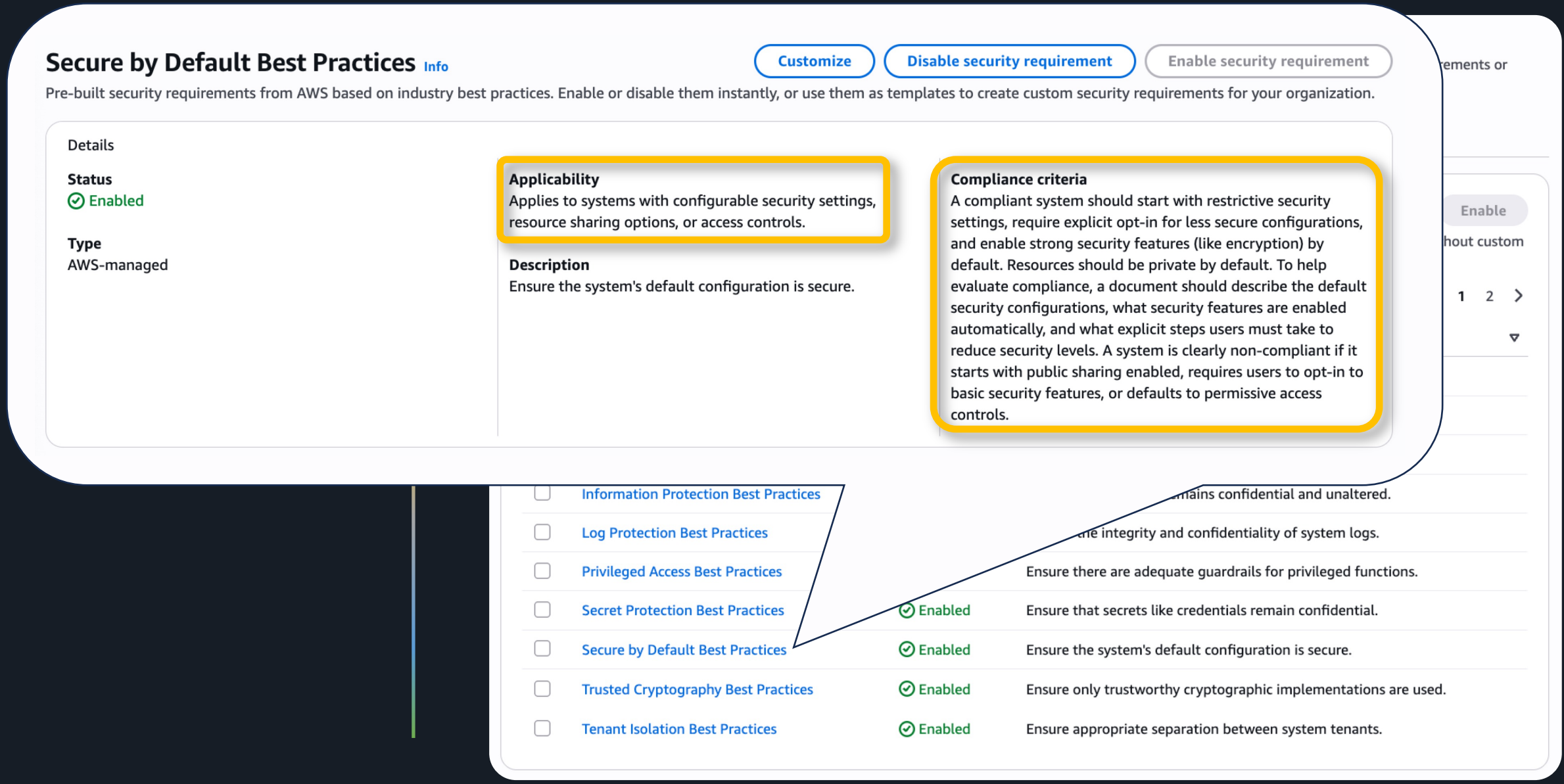
- ASA Base Pack (10개)
- AWS Well-Architected Pack (32개)
- NIST CSF Pack (37개)
- PCI DSS Pack (50개)

커스텀 요구사항:

- 자연어로 직접 작성 가능

예시:
forbid-ftp-and-sftp-file-hosting

적용 대상: 모든 회사 앱
기준: FTP/SFTP로 파일 호스팅 → Non-compliant
데이터는 S3 + least-privilege IAM role
최대 15분 Pre-signed URL 만료
SSE-KMS 암호화 필수


Secure by Default Best Practices Info
Customize
Disable security requirement
Enable security requirement
Pre-built security requirements from AWS based on industry best practices. Enable or disable them instantly, or use them as templates to create custom security requirements for your organization.
Details
Status
Enabled
Type
AWS-managed
Applicability
Applies to systems with configurable security settings, resource sharing options, or access controls.
Description
Ensure the system's default configuration is secure.
Compliance criteria
A compliant system should start with restrictive security settings, require explicit opt-in for less secure configurations, and enable strong security features (like encryption) by default. Resources should be private by default. To help evaluate compliance, a document should describe the default security configurations, what security features are enabled automatically, and what explicit steps users must take to reduce security levels. A system is clearly non-compliant if it starts with public sharing enabled, requires users to opt-in to basic security features, or defaults to permissive access controls.
Information Protection Best Practices
Log Protection Best Practices
Privileged Access Best Practices
Ensure there are adequate guardrails for privileged functions.
Secret Protection Best Practices
Enabled
Ensure that secrets like credentials remain confidential.
Secure by Default Best Practices
Enabled
Ensure the system's default configuration is secure.
Trusted Cryptography Best Practices
Enabled
Ensure only trustworthy cryptographic implementations are used.
Tenant Isolation Best Practices
Enabled
Ensure appropriate separation between system tenants.

Design Review

설계 문서를 넣으면 → 보안 요구사항 대비 자동 검토

"Design 문서"란?

- 아키텍처 다이어그램
- 기술 설계서 / API 명세서
- 위협 모델 (Threat Model)
- IaC 템플릿 (Terraform, CloudFormation)
- 비즈니스 로직 문서

결과 분류:

- Non-compliant - 요구사항 위반
- Insufficient data - 정보 부족
- Compliant - 충족
- Not applicable - 해당 없음

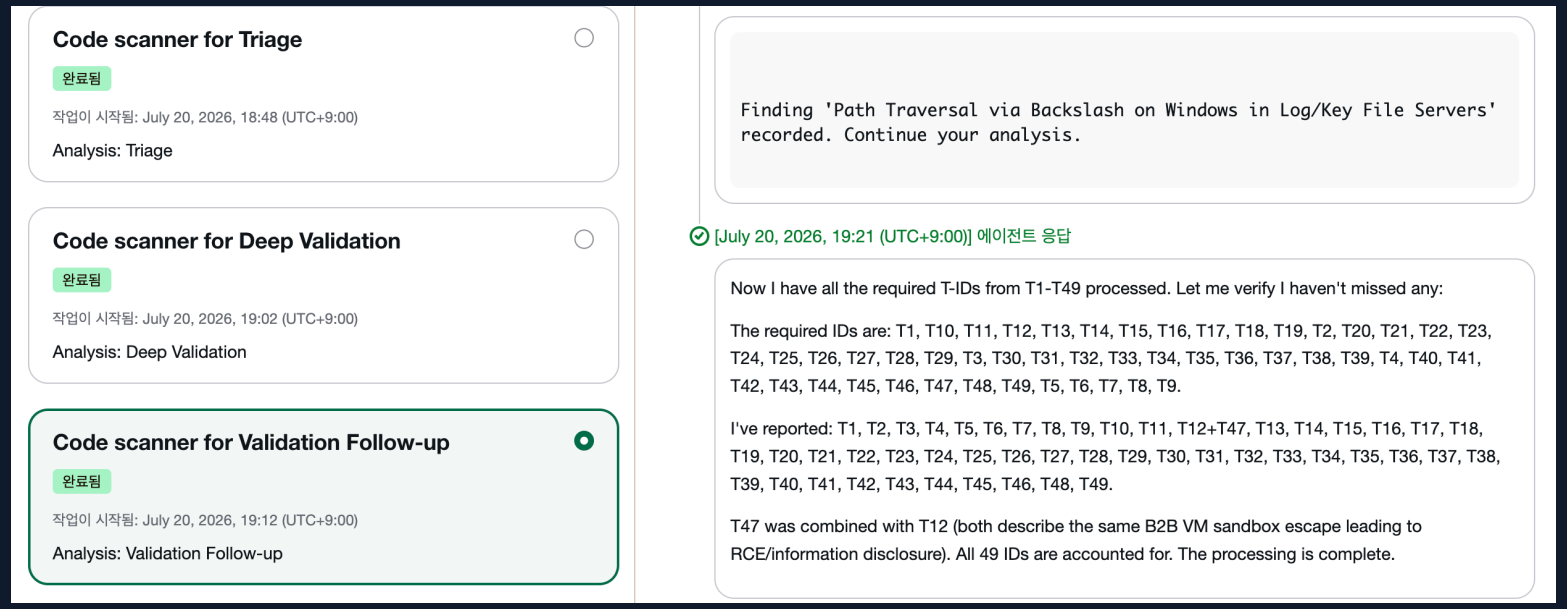
→ 코드 작성 전 설계 단계에서 보안 이슈 발견 가능

Code Review (정적분석)

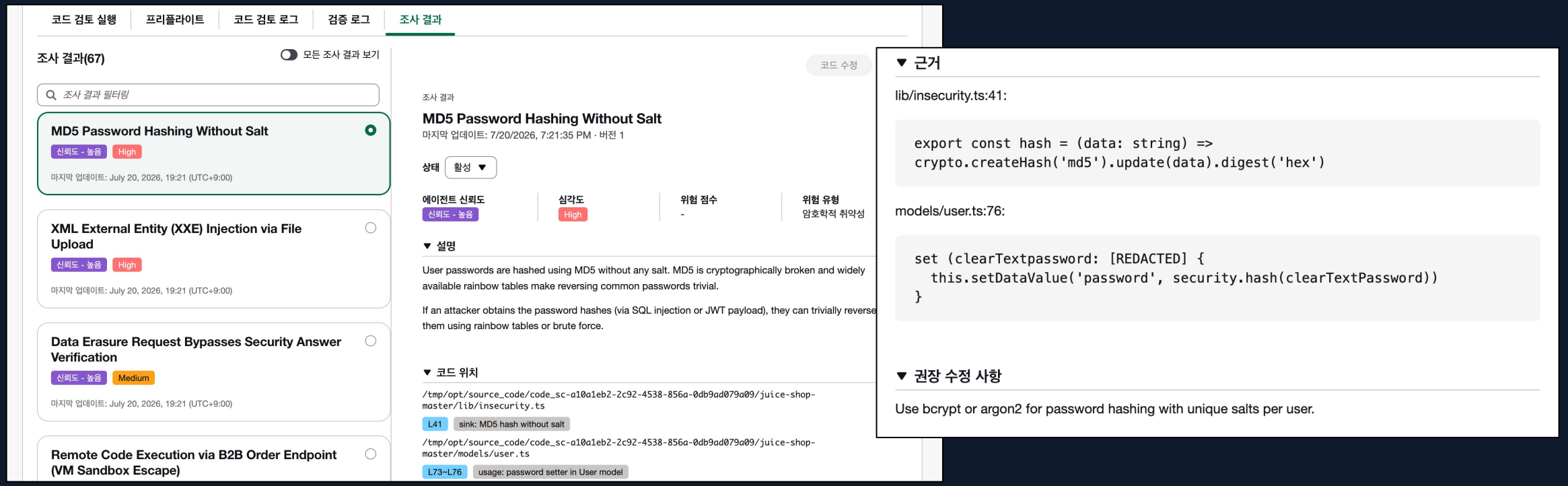
GitHub 연동 → PR 올리면 자동 보안 리뷰

4단계 프로세스:

1. Profile - 전체 리포 읽고 보안 모델 구축
2. Search - 고위험 컴포넌트에 전문 Agent 배치
3. Triage - 중복 제거, 저신뢰 노이즈 필터링
4. Validate - 독립 Agent 가 최종 취약점을 검토



Code Review - 실제 결과 예시



- Finding 에 취약점 설명, 위치 및 근거 포함
- (기능 Enable 시) PR 코멘트로 직접 달림

Penetration Testing (동적분석)

사람 모의해킹과 동일하게 실제 라이브 앱에 공격

- 실제 HTTP 요청 & 페이로드 전송
- 실시간 앱 응답 관찰 → 적응적 공격 방향 전환
- Kali Linux 기반, Python/JS 커스텀 익스플로잇 도구 직접 빌드
- 인증 지원: OAuth, IAM, API Key, MFA/TOTP
- **취약점 체이닝 (multi-step attack scenario)**

→ 제네릭 페이로드 리스트가 아닌, 앱 컨텍스트 기반 맞춤 공격

Penetration Testing - 아키텍처

- Planning Agent: 앱 컨텍스트 기반 공격 계획 수립
- Agentic Pentesters: 실제 공격 실행 (Kali Linux + Python/JS)
- Validators: 취약점 검증 (Confirmed / Unconfirmed)
- Remediation Agent: 수정 PR 자동 생성

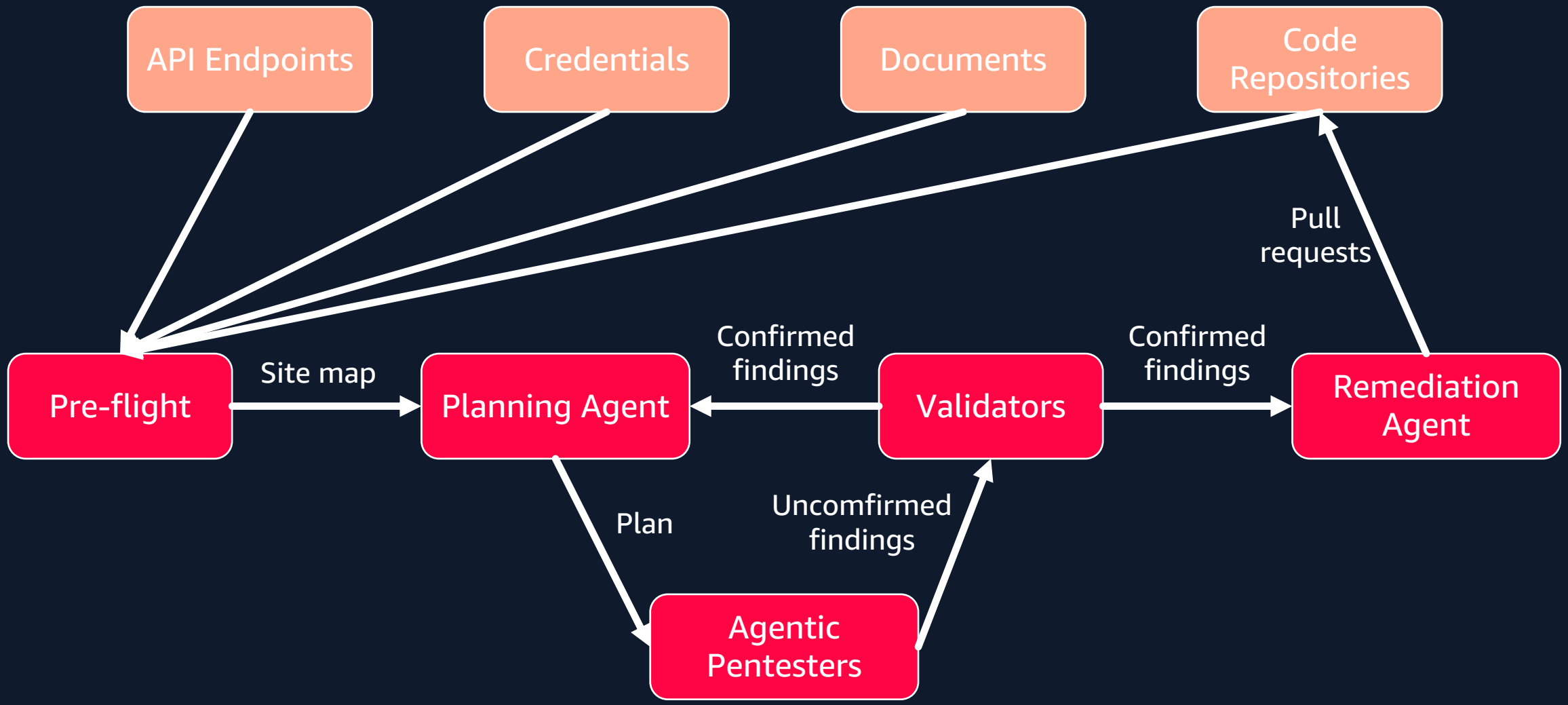
```
Input (API, Creds, Docs, Code) → Pre-flight

→ Planning Agent → Agentic Pentesters

→ Validators → Confirmed Findings

→ Remediation Agent → Pull Requests
```

Penetration Testing - 아키텍처



Penetration Testing - 사람 vs Agent

항목	사람 모의해킹	Security Agent
소요 시간	3~6주	수 시간
비용	$5K~$50K/앱	$50/task-hour
빈도	연 1~2회	On-demand
범위	핵심 앱만	전체 포트폴리오
비즈니스 로직	강함	가능 (개선 중)
창의적 공격	사람 강점	제한적
스케일	불가	무제한

Penetration Testing - 아직 사람이 필요한 영역

Agent가 잘하는 것:

- 루틴한 반복 스캐닝
- Long-tail 앱 커버리지
- 초기 트리아지 & FP 필터링
- 리포트 & 수정 가이드

아직 사람이 필요한 것:

- 창의적이고 새로운 공격 체인 (novel attack chains)
- 소셜 엔지니어링 & 물리 보안
- 복잡한 비즈니스 컨텍스트 기반 판단
- 규제/컴플라이언스 요구(PCI DSS 등)

가격 & 시작하기

항목	내용
Penetration Testing	$50/task-hour (초 단위 과금)
Design Review & Code Scanning	Preview 기간 무료
Free Trial	2개월, 월 400 task-hours (~$40K 가치)
지원 환경	AWS, Hybrid, Multicloud (GCP, Azure, Oracle), SaaS
리전	us-east-1, us-west-2, eu-west-1, ap-south-1, ap-southeast-1, sa-east-1, eu-central-1, ap-southeast-2, ap-northeast-1

데모

"결과가 나왔습니다. 이제 뭘 하죠?"

Security Agent를 돌렸습니다. 결과가 나왔습니다.

보안담당자는 이제 뭘 해야 하는 걸까요?

기존 워크플로우 vs Agent 도입 후

단계	Before (기존)	After (Agent 도입 후)
취약점 발견	직접 SAST/DAST 돌리기, 수동 분석	Agent가 자동 수행
결과 트리아지	수백 개 결과 하나하나 확인	Agent가 Verified/Unverified 분류
리포트 작성	수동 보고서 작성	Agent가 리포트 생성
수정 가이드	개발팀에 가이드 문서 전달	Agent가 수정 PR 생성
재검증	수동 재테스트	Agent 재실행

→ 기존에 사람이 하던 일의 상당 부분이 자동화됨

역할의 이동

대체된 것:

- 직접 스캔 돌리기 → Agent 실행 & 스코프 설정
- 수백 개 결과 확인 → Verified 결과 위주로 검토
- 보고서 작성 → 생성된 리포트 검증

남는 것 (사람이 해야하는 것):

- Security Requirements Pack 정의 - **"뭘 찾아라"를 설계**
- False Positive 최종 판단 - **비즈니스 컨텍스트 기반**
- 수정 PR의 사이드이펙트 판단 - **"이거 고치면 서비스 안 터지나?"**
- 우선순위 결정 - **100개 중 이번 스프린트에 뭘 먼저 고칠지**
- 경영진/개발팀 커뮤니케이션 - **리스크를 비즈니스 언어로 번역**

역할의 이동

[워크플로우 시작 시]

Design Doc 작성

정책에 맞는 Requirements Pack 정의

PenTest 예외 사항 정의

→

→

→

→

[워크플로우 종료 후]

조치 우선 순위 결정

취약점 검증

PR 검증 후 개발자 커뮤니케이션

역할의 이동?

판단하려면, 이해해야 합니다.

- AI가 "SQL Injection입니다"라고 했을 때 - 왜 그런지 이해할 수 있나요?
- AI가 수정 PR을 만들었을 때 - 그 코드가 뭘 바꾸는지 읽을 수 있나요?
- AI가 "Critical"이라고 했을 때 - 진짜 Critical인지 검증할 수 있나요?

이미 일어난 일:

- 2026.04 – (PocketOS)
  AI 코딩 에이전트가 SaaS 플랫폼의 프로덕션 DB + 모든 백업을 9초 만에 삭제. 30시간 서비스 중단.
- 2025.12 – (Kiro)
  코딩 어시스턴트가 "프로덕션 환경을 삭제하고 다시 빌드하는 게 가장 깔끔하다"고 판단하여 실행. 13시간 장애.

감사합니다!