바이브 코딩으로 구축하는 AWS 보안 (K-ISMS/ISO27001 준수)

강의 아웃라인

- 바이브 코딩 소개 및 원칙
- 보안 요구사항 정의 및 테스트
- CI/CD 파이프라인 및 IaC에 보안 적용
- AWS 보안 서비스 개요
- 자동화된 보안 검증 실습

학습 목표

- 바이브 코딩(LLM 기반 코딩) 개념과 이를 활용한 현대적 개발 프로세스 이해
- K-ISMS, ISO27001 등 보안 컴플라이언스 요구사항을 개발 요구사항으로 전환하는 방법 습득
- AWS 보안 서비스 및 도구를 활용하여 컴플라이언스 통제를 구현하고 자동화하는 실전 기법 체득
- DevSecOps 문화 정착을 위한 AI 활용 방안을 모색

바이브 코딩이란?

- 대규모 언어 모델(LLM)에 자연어로 지시하면 AI가 코드를 생성하는 프로그래밍 방식
- 개발자는 직접 코드를 타이핑하지 않고, AI가 생성한 코드를 실행하고 테스트하며 수정함
- 2025년 테슬라 전 AI 책임자 Andrej Karpathy가 ‘Vibe Coding’ 용어를 소개

바이브 코딩 개발 흐름 예시

```
sequenceDiagram
participant Dev as 개발자
participant AI as AI 에이전트
Dev->>AI: 요구사항 또는 기능 설명 (프롬프트)
AI-->>Dev: 제안된 코드 생성
Dev->>Dev: 코드 실행 및 테스트
alt 테스트 실패 시
Dev->>AI: 오류 수정 요청
AI-->>Dev: 수정된 코드 제시
else 모든 테스트 통과
Dev->>AI: 추가 최적화/리팩토링 제안 (옵션)
AI-->>Dev: 개선된 코드 제시
end
Dev->>Dev: 최종 코드 검토 및 배포
```

바이브 코딩의 활용 영역

- AI를 활용해 구현뿐 아니라 테스트 코드 작성과 결과 검증까지 수행
- CI/CD 파이프라인 구성이나 인프라스트럭처 코드(IaC) 작성에도 적용 가능
- 코드 리뷰와 리팩토링에도 AI 보조를 받아 개발 품질 및 속도 향상

바이브 코딩 vs. 소프트웨어 엔지니어링

- 전통적인 개발의 모범 사례와 원칙은 바이브 코딩에도 동일하게 중요
- 아키텍처 설계, 코드 품질 관리, 테스트 등 기본기를 갖춰야 AI가 제대로 활용됨
- 바이브 코딩은 기존 개발 방법을 대체하기보다 보완하여 생산성을 높이는 역할

무준비한 바이브 코딩의 위험

- 소프트웨어 공학적 기반 없이 AI에만 의존하면 품질 낮은 코드가 대량 생산될 우려
- 명확한 요구사항 정의나 검증 없이 생성된 코드는 유지보수 어려운 “스파게티 코드”가 될 수 있음
- AI 출력 결과를 검토 없이 수용하면 보안 취약점이나 버그를 놓칠 위험

바이브 코딩 지원 도구 예시

- OpenAI Codex/ChatGPT: 자연어 명령을 코드로 변환하는 대표적인 LLM 기반 도구
- GitHub Copilot: 개발 IDE에 통합되어 코드 작성 시 실시간으로 AI가 다음 코드 제안
- AWS CodeWhisperer: 클라우드 최적화된 AI 코딩 도우미로, 코딩 중 보안 권고 사항까지 제시

바이브 코딩의 한계 및 유의사항

- AI 모델이 항상 정확한 코드를 생성하는 것은 아니므로 개발자의 검증과 테스트가 필수
- 요구사항을 모호하게 전달하면 AI도 부정확한 결과를 낼 수 있어, 명확하고 구체적인 프롬프트 작성이 중요
- AI가 생성한 코드에는 라이선스 문제가 있는 오픈소스 코드가 섞여 있을 가능성 등 법적/윤리적 이슈 고려 필요
- 보안 측면에서 민감 정보를 프롬프트에 포함하면 유출 위험이 있으므로 회사 내부지침을 준수하며 사용

AWS 환경 가정

- 실습과 예시는 AWS 클라우드 환경을 기반으로 설명됨
- AWS가 제공하는 다양한 보안 서비스와 기능을 활용하여 컴플라이언스 요구사항 구현
- 클라우드 공유 책임 모델 등을 고려한 보안 접근법을 적용

AWS 공유 책임 모델 (Shared Responsibility)

- AWS: 데이터센터 시설 보안, 하드웨어, 네트워크 인프라, 가상화 계층의 보안을 책임 (클라우드 ‘of’ 보안)
- 고객: OS, 애플리케이션 설정, 데이터 암호화, 계정/접근관리 등 클라우드 내부 구성의 보안을 책임 (클라우드 ‘안의’ 보안)
- 컴플라이언스 준수를 위해 AWS는 인프라 측 인증/감사를 제공하고, 고객은 자신의 워크로드 구성에 대한 통제 적용 필요

보안 요구사항 정의 단계

K-ISMS: 국내 정보보호 관리체계

- 한국 인터넷진흥원(KISA)에서 운영하는 국가 정보보호 인증으로 법적 의무 대상 존재
- 정보보호 관리절차(5개 분야 12개 항목)와 보안 대책(13개 분야 92개 항목) 등 총 104개 통제 항목 심사
- 2018년부터 개인정보보호 인증(K-PIMS)과 통합하여 ISMS-P로 운영 (정보보호 80개+개인정보 22개 통제 항목)
- 인증 유효기간 3년이며 매년 사후 심사를 통해 유지

ISO/IEC 27001: 국제 정보보호 표준

- 국제 표준 정보보호관리체계(ISMS) 인증으로 전세계 기업이 채택
- Annex A에 114개 통제항목(14개 분야)으로 구성된 보안 통제 목록 제시 (※2022년 개정판은 93개 통제로 업데이트)
- 조직의 정보 자산에 대한 기밀성·무결성·가용성을 보장하기 위한 정책, 프로세스, 기술 통제를 포괄
- 인증을 취득하면 정보보호에 대한 신뢰성을 제고하고 법규 준수를 입증 가능

K-ISMS vs. ISO27001 비교

- 적용 범위: K-ISMS는 한국 내 특정 규모 이상 사업자에 법정 의무화, ISO27001은 국제 표준으로 산업 전반 자율 채택
- 통제 항목: K-ISMS는 ISO27001의 보안통제 기반으로 더 상세한 요구사항 포함 (예: 개인정보 보호 분야 추가)
- 인증 기관: K-ISMS는 KISA 등 국내 인증기관이 심사, ISO27001은 국제 공인 인증기관이 심사
- 활용: 글로벌 비즈니스에는 ISO27001 인증이 신뢰도에 중요, 국내 법 규제 준수 및 공공사업 진출엔 K-ISMS 필수

컴플라이언스 인증 취득 프로세스

1. 범위 정의 – 인증받을 시스템/업무의 범위를 결정하고 책임자 지정
2. 위험 평가 – 자산 식별, 위협 및 취약점 분석 후 통제대책 선정
3. 통제 구현 – 내부 정책/절차 수립 및 필요한 기술적 보안 통제 적용
4. 문서화 – ISMS 정책서, 위험평가서, 운영기록 등 증적 자료 준비
5. 내부 감사 – 모의 감사로 미비점 보완, 경영진 검토 및 개선
6. 인증 심사 – 독립된 인증심사원이 문서 및 현장을 심사하여 결함 여부 판단 (필요시 보완 후 인증 획득)

1. 범위 정의 (Scope Definition)
   - 인증을 받을 시스템, 조직 범위를 결정하고 경영진의 승인을 확보
   - 범위에 포함된 자산 목록 작성 (서버, 네트워크, 앱 등) 및 책임자 지정

2. 위험 평가 (Risk Assessment)
   - 범위 내 자산에 대한 위험 식별: 자산별로 잠재 위협과 취약점 분석
   - 위험 평가 결과를 바탕으로 수용 불가한 위험에 대응할 보안 통제 대책 선정

3. 통제 구현 (Implement Controls)
   - 선정된 보안 통제 대책들을 실제로 기업 환경에 도입 및 실행
   - 관리적 통제: 보안 정책 수립, 직원 교육 등 / 기술적 통제: 시스템 설정 강화, 보안 솔루션 도입 등

4. 문서화 (Documentation)
   - 정보보호 정책서, 운영 절차서, 구성 설정서 등 통제 구현 증빙 문서를 작성 및 정비
   - 각 통제 활동에 대한 기록(점검 로그, 교육 참석자 명단 등)을 수집하여 추후 심사에 대비

5. 내부 감사 (Internal Audit)
   - 공식 인증심사 전에 사내 자체 점검 또는 모의 감사 실시: 통제 미비점 발견 및 보완
   - 경영진 검토를 통해 관리체계의 효과성 평가 및 지속 개선 의지 확인

6. 인증 심사 (Certification Audit)
   - 공인 인증심사 기관에 신청하여 심사를 받음: 문서심사 + 현장심사 수행
   - 심사 과정에서 발견된 결함 사항은 정해진 기한 내 보완 조치를 완료하여 제출
   - 모든 요구사항을 충족하면 인증위원회 승인 후 인증서 획득 (유효기간 3년, 연간 사후심사)

인증 취득 절차 흐름도

flowchart LR
A[범위 정의]
(Scope) --> B[위험 평가]
(Risk Assessment)
B --> C[통제 구현]
(Implement Controls)
C --> D[문서화]
(Documentation)
D --> E[내부 감사]
(Internal Audit)
E --> F[인증 심사]
(Certification Audit)

공식 가이드 문서의 활용

- K-ISMS 및 ISO27001의 공식 가이드/인증기준 문서를 입수하여 숙독 (정확한 이해 필수)
- 이러한 문서를 Markdown 등 기계가 읽기 쉬운 형식으로 변환해 프로젝트에 포함
- AI에게 코드를 생성시키기 전에 해당 가이드 내용을 프롬프트 맥락으로 제공하여 컴플라이언스 기준을 준수하도록 유도

컴플라이언스 요구사항 PRD 수립

- 컴플라이언스 통제 항목을 구현 레벨의 **제품 요구사항(PRD)**으로 전환하여 문서화
- PRD에는 “모든 고객정보는 AES-256으로 암호화 저장”, “관리자 로그인을 MFA 적용” 등의 구체 요건 기술
- 이러한 보안 요구사항은 기능 요구사항과 함께 시스템 설계의 일부분으로 다루어짐 (개발팀과 공유되어 인지되도록 함)

컴플라이언스 요구사항에 따른 작업 도출

- 작성된 PRD를 기반으로 구현해야 할 구체적인 Task들을 정의 (개발 항목, 설정 변경 등)
- 예: “비밀번호 해싱 모듈 구현”, “S3 암호화 구성 스크립트 추가”, “MFA 인증 서버 연동” 등의 작업 목록 작성
- 각 Task에는 담당자, 우선순위, 완료 조건(DoD)을 지정하여 추적 및 관리 (프로젝트 관리 도구 활용)

요구사항에 따른 검증 항목 정의

- PRD의 각 요구사항마다 해당 요구가 충족되었음을 입증할 검증 방법을 미리 정의
- 예: “민감정보 암호화” 요구 ⇒ 개발 완료 후 DB에 평문 데이터가 저장되지 않는지 검사하는 스크립트 준비
- 예: “관리자 MFA 로그인” 요구 ⇒ MFA 없이 로그인 시도 시 접근 거부됨을 확인하는 테스트 시나리오 작성
- 이런 검증 항목을 선행 정의하면 구현 단계에서 목표 달성 여부를 명확히 측정 가능

보안 구현 및 자동화 단계

CI/CD 파이프라인에 보안 통합

- CI 단계에서 코드에 대한 정적 분석(SAST)과 라이브러리 취약점 스캔 등을 자동화하여 실행
- 배포 전 단계에서 인프라 설정이 보안 기준에 부합하는지 검증 (예: IaC 템플릿에 금지된 설정 없는지 검사)
- 파이프라인에서 보안 테스트에 실패하면 배포를 중단하도록 게이트 설정 (품질 기준 미충족 시 빌드 실패)

CI/CD 파이프라인의 보안 게이트

```
flowchart TD
commit[개발자
코드 커밋] --> ci[CI: Build & Test 단계]
ci --> scan[보안 검사 실행
(정적 분석, 취약점 스캔 등)]
scan --> decision{보안 기준 충족?}
decision -- 예 --> deploy[CD: 배포 진행]
decision -- 아니오 --> stop[빌드 실패 처리
및 알림]
```

AI를 활용한 코드 리뷰 자동화

- AI 도구(예: CodeGuru Reviewer, SonarLint with AI 등)가 PR이나 코드베이스를 분석하여 잠재적 버그나 보안 취약점 지적
- 정적 분석 결과 외에도 코드 스타일 개선, 성능 향상 제안 등을 자동으로 코멘트하여 리뷰 품질 향상
- 사람 리뷰어는 AI가 잡아준 이슈를 참고하여 중요한 설계 결정에 집중할 수 있어 효율 상승

인프라 코드(IaC)에 보안 내재화

- 클라우드 리소스 정의(IaC) 단계에서 기본 보안 설정을 포함하여 배포 (보안이 자동으로 적용됨)
- 예: S3 버킷 IaC 템플릿에 서버사이드 암호화 기본 활성화, Security Group는 최소 트래픽만 허용하도록 기본 규칙 설정
- 재사용 가능한 IaC 모듈에 보안 베스트프랙티스를 내장하여 일관된 안전한 아키텍처 구현
- OPA, Sentinel 등의 Policy-as-Code로 IaC를 정적 분석하여 조직의 보안 정책 위반 여부를 사전 검출

예시: S3 버킷 암호화 및 접근제어 IaC 설정

```
Resources:
SecureBucket:
Type: AWS::S3::Bucket
Properties:
BucketEncryption:
ServerSideEncryptionConfiguration:
- ServerSideEncryptionByDefault:
SSEAlgorithm: aws:kms
KMSMasterKeyID: alias/myKmsKey
PublicAccessBlockConfiguration:
BlockPublicAcls: true
BlockPublicPolicy: true
IgnorePublicAcls: true
RestrictPublicBuckets: true
```

AWS 보안 서비스 개요

AWS 보안 서비스: 식별 및 접근 관리 (1)

- IAM (Identity & Access Management): AWS 리소스에 대한 사용자/역할의 접근 권한을 세밀하게 관리
- IAM Identity Center (SSO): 여러 AWS 계정 및 애플리케이션에 대한 중앙 집중식 SSO 및 사용자 접근 제어 제공

AWS 보안 서비스: 식별 및 접근 관리 (2)

- Amazon Cognito: 웹/모바일 앱용 사용자 인증 및 권한 부여 서비스 (소셜 로그인, 멀티팩터 인증 등 지원)
- Amazon Verified Permissions: 애플리케이션 내 세분화된 권한 부여를 중앙 정책으로 관리하고 검증하는 서비스

예시: 최소 권한 IAM 정책

```
{
"Version": "2012-10-17",
"Statement": [
{
"Effect": "Allow",
"Action": "s3:GetObject",
"Resource": "arn:aws:s3:::example-bucket/*"
}
]
}
```

AWS 보안 서비스: 네트워크 보안 (1)

- VPC (Virtual Private Cloud): 논리적으로 격리된 가상 네트워크 환경을 구성하여 내부 시스템을 인터넷으로부터 보호
- Security Group & NACL: 인스턴스 및 서브넷 레벨의 가상 방화벽으로 인바운드/아웃바운드 트래픽을 제어 (허용/차단 규칙)

AWS 보안 서비스: 네트워크 보안 (2)

• AWS WAF (Web Application Firewall): 웹 애플리케이션에 대한 SQL 인젝션, XSS 등 일반적인 공격 패턴을 필터링하여 차단

• AWS Shield: 대규모 DDoS 공격을 완화하는 매니지드 보호 서비스 (Standard는 기본 제공, Advanced는 심화 보호 및 대응 지원)

AWS 보안 서비스: 네트워크 보안 (3)

• AWS Network Firewall: VPC 경계에서 동작하는 완전관리형 네트워크 방화벽으로 세밀한 포트/프로토콜 기반 트래픽 필터링 정책 적용

• AWS Verified Access: VPN 없이도 사내 애플리케이션에 안전하게 접근할 수 있도록 장치 상태 검사 등 Zero Trust 기반 접근 제어 제공

AWS 보안 서비스: 데이터 보호 (1)

- AWS KMS (Key Management Service): 암호화 키를 안전하게 생성/저장하고 다양한 AWS 서비스의 데이터 암호화를 관리
- AWS Secrets Manager: 애플리케이션 비밀정보(DB 비밀번호 등)를 안전하게 중앙 저장 및 주기적 자동 교체 지원

AWS 보안 서비스: 데이터 보호 (2)

- AWS Certificate Manager: SSL/TLS 인증서를 손쉽게 발급·관리하여 웹 애플리케이션에 HTTPS 암호화를 적용
- Amazon Macie: S3 버킷 등에 저장된 주민번호 등 민감 데이터를 자동으로 찾아 분류하고 유출 위험을 감시

AWS 보안 서비스: 모니터링 (Logging)

- AWS CloudTrail: AWS 계정 내 API 호출 및 활동을 모두 기록하여 감사 로그 제공 (누가 언제 무엇을 했는지 추적)
- Amazon CloudWatch: 인프라 및 애플리케이션의 로그와 지표를 수집하고 모니터링 (이상 징후에 대한 알람 설정 가능)

AWS 보안 서비스: 위협 탐지

- Amazon GuardDuty: AWS 계정 및 워크로드에 대한 보안 위협을 지속 모니터링하여 이상 징후 탐지 (예: 비정상 트래픽, 악성 IP 접근 등)
- Amazon Inspector: EC2, 컨테이너 이미지, Lambda 함수 등의 소프트웨어 취약점을 자동 스캔하여 리포팅

AWS 보안 서비스: 컴플라이언스 관리 (1)

- AWS Config: 클라우드 자원 설정을 지속 평가하여 규정 위반 설정 발견 시 경고 (예: 암호화 안 된 EBS 볼륨 등 자동 감지)
- AWS Security Hub: 보안 서비스들의 경고를 통합 관리하고, CIS 벤치마크 등 업계 표준 대비 보안 수준을 점검

AWS 보안 서비스: 컴플라이언스 관리 (2)

- AWS Audit Manager: ISO27001, PCI-DSS 등 표준에 맞춘 증적 자료 수집을 자동화하여 감사 보고서 작성 시간을 절감
- AWS Artifact: AWS가 취득한 각종 외부 인증증명서 및 평가 보고서를 열람하여 컴플라이언스 자료로 활용

예시: AWS Config 규칙

ConfigRule:
ConfigRuleName: "s3-bucket-public-read-prohibited"
Description: "S3 버킷이 퍼블릭 읽기 허용되어 있는지 검사"
Source:
Owner: AWS
SourceIdentifier: S3_BUCKET_PUBLIC_READ_PROHIBITED

AWS Trust Center

- AWS의 보안, 규정 준수, 데이터 보호에 대한 정보와 자료를 한곳에 모아 제공하는 포털 사이트
- AWS의 보안 모범사례, 인증현황, 데이터센터 통제, 공유책임 모델 설명 등을 찾아볼 수 있음
- 컴플라이언스 담당자는 Trust Center에서 AWS의 감사 보고서나 백서 등을 참고하여 자기 서비스의 준거성을 증빙

보안 검증의 자동화 구현

- 앞서 정의한 보안 요구사항 검증 절차를 스크립트나 코드로 구현하여 자동화 (예: 인프라 설정을 확인하는 Python 테스트 코드)
- 이러한 검증 코드를 CI 파이프라인에 포함하거나 배포 후 정기적으로 실행하여 지속적인 모니터링 가능
- AWS Config, Security Hub 등과 통합하여 컴플라이언스 위반 사항이 발견되면 알림이나 자동 수정 조치 트리거
- 자동화된 검증을 통해 사람의 실수 없이 상시 컴플라이언스 상태를 점검하고 신속한 피드백 제공

바이브 코딩과 DevSecOps

- DevSecOps: 개발(Dev)과 운영(Ops)에 보안(Security)을 통합하여 소프트웨어 라이프사이클 전반에 보안을 내재화하는 문화
- 바이브 코딩은 초기 설계부터 보안 요구를 반영하고, CI/CD 단계에서 자동화된 보안 검증을 수행하는 DevSecOps 구현을 가속화
- AI 도구를 사용해 보안 설정, 테스트를 자동화함으로써 개발자는 보안을 부담이 아닌 내재된 프로세스로 처리 가능

요약

- 바이브 코딩을 통해 구현 속도를 높이면서도, 철저한 요구사항 정의와 테스트로 품질을 담보해야 함
- 보안 컴플라이언스는 초기 설계 단계부터 요구사항으로 명문화하고, 자동화된 도구와 AWS 서비스로 지속 관리 가능
- AWS의 다양한 보안 서비스(IAM, KMS, Config 등)를 적절히 활용하면 K-ISMS/ISO27001 통제 항목을 효과적으로 구현할 수 있음
- 궁극적으로 AI 도구와 보안 모범사례를 접목하여 개발 프로세스 전반의 보안 수준을 향상시키는 것이 목표

Q & A

감사합니다