IRMS : 보안사고 대응 관리 시스템
(Incident Response Management System)

발표자 : 18팀 임승연 (20210808)

Intro.

주제

IRMS : 침해사고 대응 관리 시스템

설명

침해사고 발생 시 보안 문서를 보고하거나 전달해야 하는 상황에서
위.변조 또는 유출의 가능성이 존재

이를 방지하기 위해 보안 문서를 전자봉투로 패키징하여 전달할 수 있는
웹 기반의 보안 관리 시스템을 구현하고자 함.

의의

기업 간 어떻게 하면 민감한 데이터들을 안전하게 보관할 수 있을지
Security Engineer의 입장에서 생각

가상 시나리오

1. 발생 배경

   2025년 5월, 국내 보안 기업 B는 자사에서 운영 중인 Threat Intelligence 플랫폼을 통해 국제적으로 보고된 APT 공격 기법과 유사한 패턴으로 보이는 네트워크 활동 감지
   → 내부 위협 정보가 포함된 위협 인텔리전스 보고서를 생성

   B회사는 민감한 정보가 담긴 보고서를 국가 주요 기반시설을 운영하는 에너지 공공기관인 A에 전달할 필요가 있다고 판단

   * 이메일이나 클라우드 전송 불가 → 위변조 또는 중간자 공격의 위험
   * 외부 메일의 파일 첨부 불가 → A 기관은 내부망이 인터넷과 분리된 망분리 환경

   ⇒ 전자봉투 기반의 전송 시스템 사용

가상 시나리오

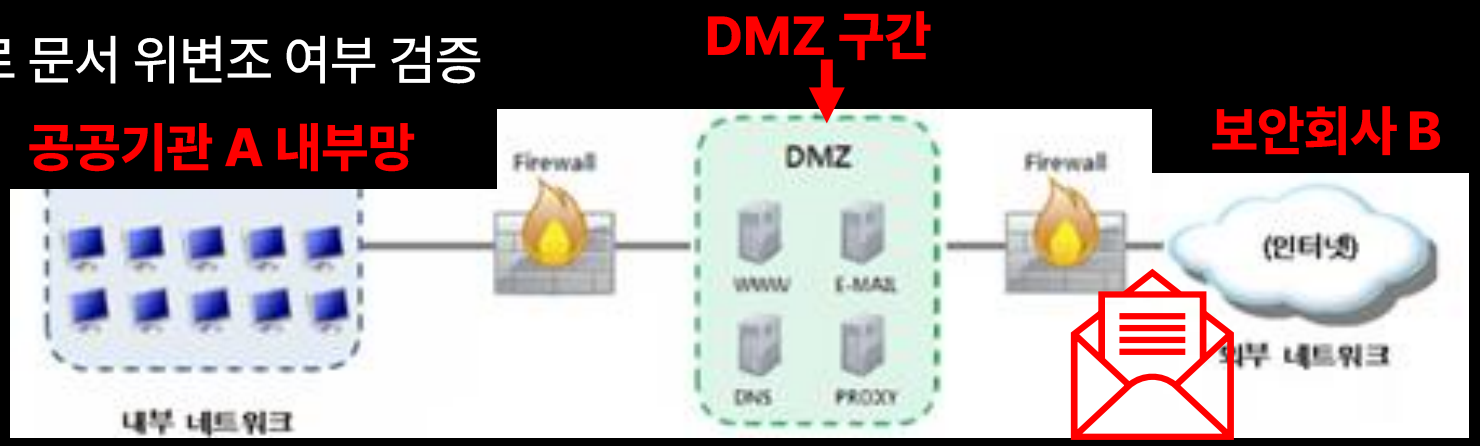
2. 발생 단계
   - 국내 보안 회사 B의 userb는 보고서 파일을 생성
   - AES-256 알고리즘을 사용하여, 대칭키로 문서를 암호화
   - 대칭키는 에너지 공공기관인 A의 공개키를 이용해 RSA 방식으로 암호화
   - 암호화된 대칭키와 암호화된 문서 파일은 전자봉투 형태로 패키징
   - 봉투 자체에 SHA-256 해시 기반의 서명을 부여 → 무결성 검증

HOW?

- 공개키를 안전하게 교환하는 방식 ?
- 중간자 공격(MitM) : 네트워크, 웹 또는 브라우저 기반 보안 프로토콜의 취약성을 악용하여 합법적인 트래픽을 우회하고 피해자의 정보를 훔치는 방식 ex) 이메일 하이재킹, 세션 하이재킹, SSL 하이재킹
- 공인인증기관(CA) → PEM 형식(Base64로 인코딩한 텍스트형식의 파일)의 공개키 인증서
- VPN 기반의 SFTP 보안 채널

가상 시나리오

3. 전자봉투의 전달 과정
   - 보안 회사 B에서 생성된 전자봉투는 외부망을 통해 전달
   - 에너지 공공기관인 A의 DMZ 구간에 위치한 보안 중계 시스템을 거쳐
   - 내부망으로 안전하게 전송
4. 전자봉투 검증 과정
   - A 회사의 보안팀은 자체 보안 시스템을 통해 봉투를 검증 및 복호화
     - PEM 인증서 기반의 RSA 개인키로 대칭키 복호화
     - 복호화된 AES 대칭키로 보고서 파일의 본문을 복호화
     - SHA-256 해시 + 디지털 서명으로 문서 위변조 여부 검증



시연 영상

실제 구현 로직

1. 키 쌍 생성 (RSA)
   - 전자봉투 생성/전달 → 키 쌍 생성이 우선
   - RSA 2048 비트의 키 쌍 생성 → PEM형식으로 DB에 저장
2. 전자봉투 생성 (AES + RSA + 서명)
   - AES-256 대칭키를 생성 → 문서 암호화
   - 해당 AES 키는 RSA 공개키로 암호화
   - 전체 암호문은 SHA-256 기반의 RSA 개인키로 서명

   ⇒ DB에 저장
3. 전자봉투 전달
   - 기존 수신자의 AES 대칭키를 업로더의 개인키로 복호화
   - 이를 수신자의 공개키로 다시 암호화해 전달

실제 구현 로직

4. 복호화 및 검증
    - RSA 개인키로 대칭키를 복호화
    - AES 키로 문서 복호화
    - 업로드의 공개키로 전자서명 검증
5. 회원가입 시 보안 로직
    - 비밀번호 복잡도 체크 (8자 이상, 대/소문자/숫자/특수문자 포함)
    - 이메일 형식 정규식 검사
    - 비밀번호 해싱 → Bcrypt 사용(내부에서 salt 적용)
6. 로그인 시 보안 로직
    - JWT 토큰 생성 : secret key로 HMAC SHA-256 서명
    - JWT 서명 검증과 만료 확인 → 위조 방지 및 유효기간 체크

실제 구현 로직

< 파일에 저장된 문서에 대해 전자봉투를 생성하는 기능 >

기능 설명	클래스	메서드
사용자가 업로드한 파일을 AES로 암호화한 뒤, AES 키는 RSA 공개키로 암호화하고 전자서명을 추가하여 저장	EnvelopeService	saveEncryptedEnvelope()
AES 키 생성	AESUtil	generateKey()
파일 내용 AES 암호화	AESUtil	encrypt()
AES 키 Base64 인코딩	AESUtil	encodeKeyToBase64()
AES 키를 RSA 공개키로 암호화	RSAUtil	encryptWithPublicKey()
전자서명 생성 (SHA256withRSA)	RSAUtil	signData()

< 해당 문서의 전자봉투를 검증하는 기능 >

기능 설명	클래스	메서드
전자봉투에 저장된 암호문을 복호화하고 서명을 검증하여 무결성 확인	EnvelopeService	verifyAndDecryptEnvelope()
암호화된 AES 키를 RSA 개인키로 복호화	RSAUtil	decryptWithPrivateKey()
복호화된 Base64 AES 키 디코딩	AESUtil	decodeKeyFromBase64()
AES 복호화	AESUtil	decrypt()
업로더의 공개키로 서명 검증	RSAUtil	verifySignature()

실제 구현 로직

< 전자서명에 필요한 비대칭키를 생성/저장하는 기능 >

기능 설명	클래스	메서드
RSA 키쌍 생성	RSAUtil	generateKeyPair()
키쌍을 PEM 포맷으로 인코딩 후 저장	KeyService	generateKeyPairForUser() + encodeToPem()

< 대칭암호화에 필요한 비밀키를 생성/저장하는 기능 >

기능 설명	클래스	메서드
AES 대칭키 생성	AESUtil	generateKey()
AES 키를 문자열로 인코딩	AESUtil	encodeKeyToBase64()
복호화를 위한 문자열 AES 키 디코딩	AESUtil	decodeKeyFromBase64()

코드 리뷰

52번 규칙 Avoid in-band error indicators (인밴드 오류 지표를 삼가라)

- KeyController.resolveToken()이 null을 반환하게 되면, getUsernameFromToken(token)에서 NullPointerException이 발생할 수 있기 때문에, 토큰이 null일 때 예외 처리

```
@PostMapping("/generate")
public ResponseEntity<?> generateKeys(HttpServletRequest request) {
    String token = jwtTokenProvider.resolveToken(request);
    String username = jwtTokenProvider.getUsernameFromToken(token);
```

```
@PostMapping("/generate")
public ResponseEntity<?> generateKeys(HttpServletRequest request) {
    String token = jwtTokenProvider.resolveToken(request);
    if(token == null) {
        throw new RuntimeException("인증 토큰이 필요합니다.");
    }
    String username = jwtTokenProvider.getUsernameFromToken(token);
```

REFERENCE

- https://www.fortinet.com/kr/resources/cyberglossary/man-in-the-middle-attack
- https://www.entrust.com/ko/resources/learn/what-is-pki
- https://raptor-hw.net/xe/know/147875

- Java Coding Guidelines 75 Recommendations for Reliable and Secure Programs

감사합니다.