한국산학기술학회논문지
Vol. 11, No. 3, pp. 926-930, 2010

컴포넌트기반 보안개발방법의 프레임워크

홍진근[1*]
[1]백석대학교 정보통신학부

Framework of Security Development Method based on Component

Jin-Keun Hong[1*]
[1]**Division of Information Communication, Baekseok University**

요 약 본 논문은 컴포넌트 기반의 보안 시스템 개발을 위해 요구되는 개발 프레임워크에 관한 것이다. 정보시스템의 개발방법론 적용과 함께, 정보보호 제품의 개발방법론 적용이 현시점에 요구되고 있다. 본 논문에서는 개발방법의 NIST 요구규격, SDLC 단계별 요구기준, 위험평가 주요 보안 가이드 라인을 살펴보았다. 또한 SDLC 주요 보안 요소를 살펴보았고, 컴포넌트 기반 보안 프레임워크 수립에 대한 이해를 돕기 위해, 위협트리 STRIDE 기반의 외부 엔티티에 대한 스푸핑 측면에서 보안설계와 DFD 관계를 분석 제시하였다.

Abstract This paper is about a development framework, which is required to develop of security system is based on component. With applying of SDLC(system development life cycle) of information system, the application of information security products DLC is required at this point of time. In this paper, we review NIST requirement specification of development method, requirement criteria of SDLC in each stage, and major security guidelines of risk assessment. Also we are reviewed major security element of SDLC, and to aid understanding of security framework based on component, present the relationship fo security design and DFD in respect of spoofing for the outside entity based on threat tree STRIDE,

Key Words : SDLC, Testing

1. 서론

정보보호 시스템, 컴포넌트 기반의 정보보호 시스템의 개발방법론에 대한 고려는 크게 부각되지 않았으나, 일반 정보시스템의 소프트웨어를 기반으로 하는 개발방법론의 진화와 보안의 중요성이 높아짐에 따라, 정보보호 제품에서 개발방법론 적용에 대한 요구 점차 강조되고 있다. 이러한 환경에 웹 기반의 보안 평가 도구가 개발되어 적용되어 오고 있으며[1], 오픈소스에 대한 취약성에 대한 연구가 요구되어 왔다[2].

Miller 등[3]은 어플리케이션 신뢰성의 퍼지 테스팅에 대한 연구를 수행한 바 있고, G. McGraw 등[4]은 소프트웨어 보안 테스팅을 주제로 연구한 바 있다. NIST는 소프트웨어 보안 개발 방법론에 대한 규격을 정의하여 제시하고 있으며[5], 어플리케이션 코딩에 대한 가이드라인을 M. G. Graff 등의 자료로부터 도움을 받을 수 있다[6].

기존 연구에서, 소프트웨어 버퍼 오버플로우에 대한 연구로 자동화된 탐지 및 예방 기법으로 stackGuard 기법에 대한 연구나, 힙 기반의 버퍼 오버플로우 방지 방안 wrapper, 메모리 관리 정보의 오염으로 힙 기반의 오버플로우 문제를 보호하는 방안, 런타임 랜덤화에 대한 연구가 있었다.

동적 메모리 오류에 대한 기존 연구에서는 메모리 노출이나 접근 오류의 바른 검출을 주제로 하거나, 포인터나 어레이 접근 오류의 효율적인 검출, C 프로그래밍 환경에서 어레이와 포인터를 백워드 호환가능한 바운드 체크방안이나, 동적 버퍼 오버플로우 검출기에 대한 연구가 있었다.

이외에도 디버깅과 런타임 유형 체크나 C의 안전한 표현 방안, 스마트 카드 기반에 라이센스 관리, 버퍼 오버

*교신저자 : 홍진근(jkhong@bu.ac.kr)
접수일 09년 12월 12일 수정일 10년 01월 20일 게재확정일 10년 03월 18일

플로우 공격에 대응한 프로세스 구조나, 하드웨어 스택 보호에 대한 연구가 있었는가 하면, 메모리 모델에 제어 데이터 공격 예방을 주제로 하거나, 신뢰성과 보안 지원을 제공하는 구조적인 프레임워크에 대한 연구, 탬퍼 방지에 대한 프로세싱을 위한 구조 연구가 있었다.

또한 보안 개발방법론에 대한 추가적인 연구를 위해, 어플리케이션 보안과 개발 체크리스트(미국방성), MS SWL 서버 DB 보안 체크리스트, 네트워크/데스크탑 어플리케이션/네트워크 인프라/디렉토리 서비스 등의 체크리스트에 대한 연구가 요구된다.

본 논문에서는 컴포넌트 기반의 보안 개발방법에 대한 프레임워크를 접근함에 있어서, NIST(National Institute of Standards and Technology)에서 정의하는 단계별 표준을 중심으로 NIST 규격, SDLC 단계별 요구기준, 위험평가 주요 보안 가이드라인을 기술하였다. 또한 SDLC 주요 보안 요소를 보안기술 고려사항과 특히 임베디드 보안시스템에서 요구된 보안요소에 대한 고려사항을 살펴보았다. 본 논문에서는 보안 프레임워크 수립에 대한 이해를 돕기 위해 위협트리 STRIDE 기반으로 분석 사례를 제시하였다.

이는 SDLC 기반으로 개발되는 컴포넌트 보안시스템에서 STRIDE (spoofing, tempering, repudiation, information disclosure, DoS, elevation of privilege)와 DFD(Data Flow Diagram)[7] 관계성이 구체적으로 적합하게 정의되어야 하며, 분류항목에 따른 적합한 보안 설계와 테스팅 방안이 마련되어야 하기 때문이다.

본 논문의 구성은 2장에서 SDLC를 위한 정보보호 단계별 표준과 위험평가 가이드라인을 기술하였고, 3장에서 SDLC 주요 보안요소 기술을, 4장에서 위협트리 분석 그리고 5장에서 결론을 맺었다.

2. SDLC를 위한 정보보호 단계별 표준

2.1 NIST 규격

NIST가 제시하는 SDLC 관련 표준 문서에는 정보기술 시스템을 위한 위험관리 가이드(SP 800-300), 연방PKI 인프라 및 공개키 기술 개요(SP800-32), 정보기술을 위한 기술적 모델(SP 800-33), 정보기술시스템을 위한 업무연속계획(SP 800-34), 방화벽과 방화벽 정책을 개요(SP 800-41), 네트워크 보안 테스트를 위한 가이드라인(SP 800-42), 무선 네트워크 보안(802.11, 블루투스, 핸드헬드 디바이스, SP 800-48), 정보기술 보안인식과 훈련프로그램(SP 800-50), 연방정보시스템을 위한 보안통제(SP 800-53) 등이다.

2.2 SDLC 단계별 기준

초기계획단계에서는 업무파트너 계약(800-35, 800-27), 기업 구조의 문서화(800-47), 정책/법의 식별 및 명시(800-14), 기밀성/무결성/가용성 지원(FIPS 199, 800-60), 정보와 정보시스템의 보호 카테고리(FIPS 199), 제품 사양 도출(800-36), 사전 위험 평가(800-30)로 구성된다. 획득과 개발단계에서는 위험평가(800-30), 보안 통제의 베이스라인 설계(800-53, 800-36, 800-23), 비용 분석 및 보고(800-64, 800-36), 보안계획 수립(800-55), 보안 테스트와 평가(CC, FIPS 140-2)를 실시한다.

구현과 평가단계에서는 제품 및 컴포넌트 조사 및 수용(800-64, 800-51, 800-61, 800-36, 800-35, 800-56, 800-57), 시스템 ST&E 계획(800-55), 보안 인증(800-37, 800-53A), 잔여위험(800-37), 보안 인가(800-37)가 요구된다.

철회단계에서는 제품의 전환계획(800-64), 컴포넌트 철회(800-35), 매체 소거(800-36), 정보 아카이빙(800-14, 800-12)단계로 구성된다.

2.3 SDLC 위험평가 가이드라인

제품 위험평가는 공격에 대한 시스템 취약성의 수준을 결정하는데 주요 요소이며, 기반 OS 성격, 패스워드, ACL 설정 권한문제, 디렉토리 서비스의 스키마 문제, 방화벽 정책, 스크립트 코드의 기능과 사용언어, 어플리케이션 차원에서 보안 특성, 네트워크 트래픽, 어플리케이션 파일 분석과 침투테스팅 등의 문제가 검토 되어야 한다.

위협모델링은 적용하고자 하는 목표 시스템의 핵심 위협 시나리오(클라이언트, 서버, 관리자)를 정하고 환경(OS, 버전, 개발환경)을 리스트업한다. 보안 전체 조건 정의에서는 암호키가 올바른지, 퍼미션이 문제가 없는지, 관련 정책에 대한 정의가 올바른지, 연결 스트링이나 암호키를 보호하는 수단이 제공되는지, DB 서버, ACL의 올바른 강제화된 인증이 이루어지는지에 대한 조건을 정하는 것이 필요하다. 어플리케이션을 모델화하는 데 있어 DFD를 작성할 때는 STRIDE (ID 스푸핑, 탬퍼링, 부인방지, 정보노출, 서비스 거부, 권한 높이기)에 기반한 위협 식별 분류방법을 적용할 수 있다.

위협 등급분류(1-4등급; 위협 없음(0), 미수용(1), OK(2), Good(3), Excellent(4))에 대해서는 개발사에 따라 적용하되 분류에 따라 위험정도(접근성, 인증성, 교환, 노출, 거부공격에 따른 서비스 지속성 등)를 평가하고, MS

사의 경우 DREAD (Damage potential, Reproducibility, Exploitability, Affected users, Discover ability) 방식을 적용하고 있다.

평가를 위해 수행되는 테스트는 화이트와 블랙 테스트 방식을 적용하되, 퍼지 테스팅, 침투 테스팅, 런 타임 검증, 위협 모델 검토와 업데이트, 실시간 공격 테스트, 표면 공격 테스트 등이 이루어진다.

구조체 확인 테스트에는 인터페이스(소켓, RPC, Named pipe, NetBios) 테스트, 레지스터리 테스트, 액티브 디렉토리, HTTP 데이터(헤더, 폼, 개체, 질의 문자열, 페리로드, SOAP 데이터와 헤더), 명령 라인 매개변수(C/C++의 argv[] 데이터, c#의 args 어레이) 등의 테스트를 실시한다.

스푸핑 위협 테스트에 대해서는 인증프로토콜에 대한 시험, 저장장치나 네트워크에서 신용정보를 확인할 수 있는지 여부, 보안토큰의 인증없이 실행되는 경우, 사용자 자격 증명에 대한 전수 공격 등을 테스트한다.

데이터 변조위협 문제를 해결하기 위해 어플리케이션에 권한부여, 접근통제가 적용되는지에 대한 테스트가 이루어져야 한다.

서비스 거부 위협에 대한 로그 기록 거부나 이벤트 로그의 적합성을 검사해야 한다.

정보노출에 대해서는 권한에 따른 데이터 접근 가능성, 데이터가 노출됨에 따라 어플리케이션 실패 여부가 확인되어야 하고 프로세스의 정상 종료나 디스크 클린이 실행되는지, 디스크에서 민간함 데이터가 탐지되는지를 테스트한다.

프로세스를 넘치게 하는 데이터양에 대한 요청 응답을 멈추게 하는지, 오류 데이터로 프로세스가 멈추거나 디스크 공간, 메모리가 부족한지, 자원이 제한되어 오류가 일어나는지 등을 테스트한다.

3. SDLC 보안요소 기술

3.1 보안기술 고려사항[8]

SDLC에서 보안 요소기술에는 암호문제, 키관리에서 키사용, 기밀정보보호, 입력정보, 정규데이터의 표현, 최소권한의 원칙, 무결성이나 가용성, 식별 및 인증, 보안원칙이나 안전한 코딩에 대한 문제가 대두된다.

암호문제는 사용환경에 따른 난수에 대한 고려가 중요한데, .NET이나 COM 기반의 웹 서에 사용되는 랜덤함수 사용문제에 대한 검토나, C/C++에서 CryptoAPI, VB 스크립트나 자바 스크립트의 CAPICOM 등에 대한 고려가 존재한다. 사용되는 코드는 기본적으로 AES 알고리즘에 128비트 이상의 키를 요구하고, RSA/DH 알고리즘에 2048비트 이상의 사양을 요구하고 있다. ECC의 경우 256비트 이상 조건을 요구하고 SHA256/384/512나, MAC 128 비트 이상의 키 길이를 사용해야 한다. 개인 키나 민감한 정보에 대해 DAPI를 적용해야 하고 암호 키를 유도하기 위해 안전한 KDF(CAPI에 CryptDeriveKey, .NET용 PasswrodDeriveBytes, RFC2898 Derive Bytes)를 사용하는 것이 요구되고 있다. C 런타임 함수 rand( )나 시스템 함수 GetTickCount 같은 랜덤함수는 코드 내에서 사용해서는 안되며, CryptGenRandom(C/C++ 코드), Rand_S (CryptGenRandom 호출하는 새로운 C 런타임 라이브러리 함수), .NET 코드용 RNG Crypto ServiceProvider, 스크립트 언어용 CAPICOM의 GetRandom이 사용된다.

키관리에서는 장시간 키와 단시간 키로 분류되고 장시간 사용되는 키는 주로 인증이나 무결성, DB 및 파일 저장에 지속적인 데이터 보호용도로 사용된다. 단시간 사용되는 키는 IPSec, SSL/ TLS, RPC/DCOM 등의 네트워크 프로토콜 보호용도로 사용된다. 멀티쓰레드 환경에서 어플리케이션 프로그램을 설계한다면 평문과 암호문에 각 버퍼를 사용하고 함수 호출에 암호문 버퍼의 초기화 작업은 반드시 요구된다.

기밀정보의 경우 메모리 관리에 유의해야 하는데, 최대한 빠른 시간 내에 메모리에 저장된 암호정보는 버퍼 덮어쓰기를 수행하는 것이 필요하며, 장시간 사용하지 않는 메모리에 존재하는 민감한 정보는 암호화시키거나 잠금함수를 사용하여 처리하는 것이 필요하다.

입력 정보에 대한 검증은 반드시 필요하며 이후 코드 실행이 이루어져야 한다. 컴포넌트 즉 DLL 파일, Active X 컨트롤, 재사용 클래스 라이브러리는 반드시 검증하고, 외부 호출코드에 입력되는 함수, 변수, 메소드, 등록정보도 검증이 필요하다. 정규데이터 표현에 있어서 웹에서 정규화 버그 문제는 규칙에 맞도록 입력을 제한하는 것이 바람직하다.

공격 영역을 감소하기 위해 개방된 소켓 수, 파이프 수, RPC 엔드 포인트 수, 서비스 수, 디폴트로 실행되는 서비스 수, 높은 권한으로 실행되는 서비스 수, 필터나 어플리케이션 수, 동적인 웹 페이지 수, 관리자 그룹에 추가된 계정 수, 약한 접근 통제를 사용하는 파일 또는 폴더, 레지스트리 키 수를 최소화하는 것이 필요하다.

3.2 임베디드 시스템의 보안요소 고려사항

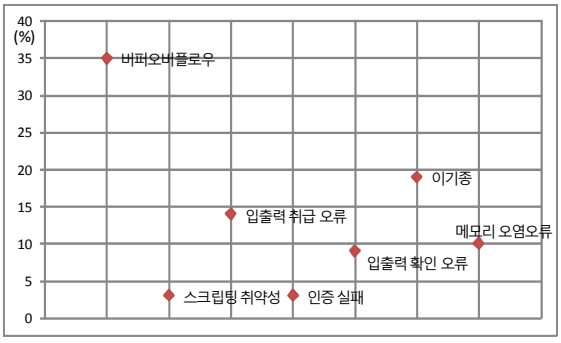
SDLC 보안 통합은 보안 취약성 문제나 잘못된 환경설정에 있어서 사전식별 및 완화가 가능하고, 개발일정과 비용측면에서 컴포넌트 재사용과 공유 서비스 식별, 위험

관리를 통해 효과를 증진할 수 있다.

시스템을 설계하는 과정에서 프로세싱 갭(시스템의 능력이나 보안 기능 처리 과정에 실제 계산측면과 차이) 존재 여부, 배터리 갭(시스템에서 보안처리 과정에 요구되는 에너지 요구규격과 실제 배터리 용량 사이의 차이) 존재 여부, 탬퍼 저항(소프트웨어나 물리적, 부채널 공격 등으로부터 시스템의 보호) 문제, 보증 갭(시스템의 복잡도, 신뢰성이 증가함에 따라 보증에서 차이) 문제, 전체 시스템의 비용과 보안에서 요구하는 비용 사이의 균형을 맞춘 설계 문제가 핵심적으로 고려되어야 한다.

실제 탬퍼저항이나 프로세싱 갭 문제는 시스템에서 소프트웨어 안전한 실행이라는 이슈가 제기되는데, 임의 공격에서 실행되는 소프트웨어의 보호를 어떻게 할 것인가이다. 일반적으로 발생하는 소프트웨어 오류 분포는 다음 표에서 제시하였다.

알려진 보안 공격에는 스택 스매싱 공격이 있는데 프로그램 상의 민감한 데이터가 악의적으로 행동하는 행동자에 의해 오염되는 공격으로 스택 내의 리턴 주소가 오염된다. 힙 오버플로우 공격은 널리 알려진 공격으로 대부분 C 동적 메모리 관리 malloc 함수나 free 라이브러리 함수 등을 사용할 때, 할당된 버퍼 크기를 초과하여 데이터 기록이나 저장 데이터, 함수 주소를 변경하여 임의 코드를 실행하여 공격한다. 오버플로우 공격에 대한 대책으로 메모리 방화벽이 적용되고 있다.



[그림 1] 소프트웨어 오류 분포

4. 위협트리 모델 분석

4.1 STRIDE 유형과 DFD 관계

SDLC에서 위협모델[9]에서 일반적으로 적용되는 위협 트리와 DFD(Data Flow Diagram) 관계를 살펴보면 다음 표1에서와 같다.

[표 1] RFID시스템의 STRIDE유형과 DFD관계

DFD 요소타입	S	T	R	I	D	E
외부 실체	X		X			
데이터 흐름		X		X	X	
데이터 저장소		X	+	X	X	
프로세스	X	X	X	X	X	X

표1에서 +는 로깅 또는 감사 데이터를 기록하는 데이터 저장의 특정한 형태를 나타낸다.

4.2 외부 실체의 스푸핑 위협 분석과 테스팅

클라이언트, 리더, 태그로 연결된 RFID 접속환경에서 스푸핑 위협을 살펴보면 다음 표2와 같다. 각 위협 트리는 유관 테이블을 따른 적합한 설계와 테스팅 개념을 고려하는 것이 요구된다. 표2에서 제시된 외부 실체에 대한 스푸핑 위협은 위협분석 가운데 하나로 스푸핑이 발생 가능한 분류에 따른 설계 및 테스팅을 분석한 것이다.

일반적으로 보안 설계 환경에서는 DFD 요소별(예, 외부 실체, 데이터 플로우, 데이터 저장, 프로세스 측면)로 스푸핑, 템퍼공격, 부인방지, 정보 노출, 서비스 거부, 권한 높이기에 대한 관계로부터, 각 세부 항목별로(요소에 따른 위협) 보안 설계와 그에 따른 테스팅이 이루어져야 한다.

[표 2] 위협에 대한 RFID 보안설계와 테스트항목(DFD의 외부실체 프로세스에서 스푸핑위협)

요소에 따른 위협 분류	설계	테스팅
취약한 인증서 변경 관리	인증서가 복잡한 정책은, 적합하지 않은 인증서에 대한 타임아웃은	고갈된 키 공간 공격을 사용하는 것 없이 테스트하는 것은 어렵다.
널 인증서	어플리케이션이 널 인증서나 패스워드 없는 계정 지원은	익명의 접속이나 패스워드 없는 접속을 시도한다.
인증 품질 저하	어플리케이션이 오래되거나 약한 인증을 지원하는가, 그러면 디폴트로 가능한 기법은	오래된 프로토콜과 협상 가능한 클라이언트와 서버는 빌드하고, 품질저하된 기법을 사용하여 위협 트리 스푸핑을 재시도한다.

[표 2] RFID 위협에 대한 보안 설계와 테스트항목 (외부 실체나 프로세스 스푸핑) (계속)

요소에 따른 위협 분류	설계	테스팅
예측 가능한 인증서	인증서가 제공하는 랜덤성은(쿠키인증은 예측가능 인증서)	종단에서 인증서를 생성할 때, 기존 인증정보를 조사하고, 그때 다른 컴퓨터와 접속하며, 인증서를 조사하고, 인증서에 패턴을 볼 수 있는지 확인
취약한 서버 인증서 저장	서버에 인증서가 있나, 어떻게 보호하나?	• 인증서가 잘 보호되어 있나(인증서가 어디에 보관되어 있나) • 암호화 방식은 • 암호키가 어디에 저장되나 • 알려진 인증서를 정하고 인증서를 위한 저장소를 삭제하라
취약한 클라이언트 인증서 저장	클라이언트에 인증서가 있는가, 어떻게 보호하나	상동
취약한 키 분배 센터 저장	키 분배 센터가 있는가, 어떻게 보호되나	상동
취약한 RFID 리더인증서 저장	리더에 인증서가 있는가, 어떻게 보호되나	상동
취약한 RFID 태그 인증서 저장	태그에 인증서있는가, 어떻게 보호되나	상동
취약한 인증서 전송	두 종단 간에 인증서가 어떻게 전송되는가	알려진 인증서 업데이트/정하라. 인증서에 대한 유선으로 청취하라
무 인증	인증 시스템 없는 것 필요한가	인증 없이 접속하여 수집 가능한 데이터양을 정하고, 사용자가 접속할 수 있는 자산 정도를 정하라.

5. 결론

본 논문은 컴포넌트 기반의 보안 시스템 개발을 위해 요구되는 개발 프레임워크에 관해 접근하였다. 논문에서는 정보보호 제품의 개발방법론 적용이 요구됨에 따라, 보안시스템 개발방법에 대한 NIST 요구규격, SDLC 단계별 요구기준, 위험평가 주요 보안 가이드라인, DLC 주요 보안 요소를 살펴보았다. 컴포넌트 기반의 보안 시스템 개발을 위해 반드시 요구되는 위협트리 STRIDE 측면을 강조하고, 외부개체의 스푸핑 측면에서 보안설계와 DFD 관계를 사례로 제시하였다.

참고문헌

[1] Yourdon, Ed. Just Enough Structured Analysis project. Chapter9, "Data Flow Diagrams, "http://www.yourdon.com/strucanalysis/chapters /ch9.html.

[2] Open Source Vulnerability Database. Symlink Vulnerabilites, http://www. osvdb.org/searchdb.php?vuln_title = symlink, last updated January 31, 2006.

[3] Miller, Barton P., "Fuzz Testing of Application Reliability," http://www .cs.wisc.edu/~bart/fuzz/fuzz.html, Dec. 2005.

[4] KeyLength.com, "Cryptographic Key Length Recommendation," http:// www.keylength.com.

[5] Curphey, Araujo, “Web Application Security Assessment Tools”, IEEE Security and Privacy archive, Volume 4 , Issue 4, July 2006.

[6] G. McGraw, B. Potter, “Software Security Testing”, IEEE Security & Privacy, May 2004.

[7] NIST, “Security Considerations in the Information SDLC”, SP 800-64 Rev. 1, 2004.

[8] Swiderski, Frank, and Window Snyder, Threat Modeling, Redmond, WA: Microsoft Pess, 2004.

홍 진 근(Jin-Keun Hong) **[정회원]**

• 2008년 12월 ~ 현재 : 백석대학교 정보통신학부 교수

<관심분야>
전술통신, 센서넷, RFID, 무선랜 보안