

Software 개발 보안

소프트웨어 개발보안

보안 상식 퀴즈! OX Quiz 1

사람의 실수로 인한 버그는
보안 취약점이 맞다.

(O, X)

OX Quiz 1 : 정답 O

보안 취약점은
실수로 인한 버그도 포함한다.

물론 사이버 어택도, 보안 취약점이다.

보안 상식 퀴즈! OX Quiz 2

보안 활동은

해킹으로 인한 침투를 당하지 않도록 하는 것을 뜻하며,
시스템이 정지되지 않게 하는 활동은 아니다.

(O, X)

OX Quiz 2 정답 : X

보안 활동은,
시스템 해킹으로 손실 뿐만 아니라,
시스템 정지로 인한, 서비스 중단이 일어나지 않도록 해야한다.

시큐어 코딩은
안전한 소프트웨어가 되도록 하는 노력이다.

시큐어코딩은?

- 1. 공격으로 부터 안전하게
- 2. 버그로 인한 에러가 없도록 안정적인 운영
- 3. 시스템 정지되는 일이 없도록 안정적인 운영
- 을 위해서 하는 것이 시큐어 코딩

S/W Secure Weakness (보안취약점)

- S/W 설계, 구조, 구현, 코딩 에 내포된
버그 (결함, 오류), Mistake 등을 뜻한다.

출처 : cwe.mitre.org

보안취약점은
사이버 공격 취약점만을 나타내는 용어가 아니다.

S/W 보안 취약점으로 인해서 발생하는 일

1. 시스템 정지로, 서비스 중단
2. 시스템의 중요 정보가 노출 (해킹)
3. 시스템 내부에 접근 / 제어 할 수 있는 Root 권한 탈취

시스템 오작동 / 중단 역시,
보안 취약점으로 인해 발생하는 것이다.

보안 취약점을 해석하는 방법

“안전하게, 안정적으로”

프로그램이 운영될 수 있도록,

없애야 하는 포인트 → 보안취약점

Software 개발보안이란?

- 소프트웨어 개발과정에서 개발자의 실수, 논리적 오류 등으로 인해 발생할 수 있는 **보안 취약점**, 보안약점들을 최소화하여 사이버 보안 위협에 대응할 수 있는 **안전한 소프트웨어**를 개발하기 위한 **일련의 보안 활동**을 의미한다

정리

Secure (보안)

- 위험, 손실 및 범죄가 발생하지 않도록 방지하는 것.

S/W 보안 취약점

- 소프트웨어 버그, 실수 등을 뜻함
- 침투 공격에 대한 취약점만을 나타내는 용어가 아님

Software 개발보안

- SW 보안 취약점을 최소화 하여, 안전한 소프트웨어 개발하기 위한 보안 활동을 뜻함

Secure Coding 개념

소프트웨어 개발보안

Secure Coding 이란?

Secure Coding

= 안전한 코딩

= 보안 취약점을 S/W 개발 초기 단계에서 부터 제거하려는 노력

시큐어코딩 법률

국내 "시큐어코딩 의무화 법안" 발표

1. 감리대상 전 사업에 대해 적용
 - 정보시스템 구축사업, 5억원 이상인 경우 감리 대상
2. 모든 공공기관은 시큐어 코딩 준수사항을 이행해야 함



Secure Coding vs S/W 개발 보안

S/W 개발 보안과 Secure Coding 을 같은 의미로 표현
→ KISA

S/W 개발 보안과 Secure Coding 을 다른 의미로 표현

- S/W 개발보안 > Secure Coding
 - S/W 개발 보안 : 모든 단계에서, 안전한 개발을 위한 보안 활동
 - Secure Coding : 구현 단계에서, 안전하게 코딩하는 보안 활동
- Oracle, MS

용어 표준화가 안되어있다.
문맥에 따라 해석하자.

국내 해킹사례

국내는 대부분 웹 Application 취약점으로 인한 개인정보 유출 사고가 발생한다. (by KISA)

국내 해킹 사례

뽕뿌 해킹사례

- 190만 개인정보 유출
- SQL Injection, XSS 공격취약
 - <https://namu.wiki/w/%EB%BD%90%EB%BF%8C%20%EA%B0%9C%EC%9D%B8%EC%A0%95%EB%B3%B4%20%ED%95%B4%ED%82%B9%20%EC%82%AC%EA%B1%B4#s-2.5>

대기업에서는

대한민국 국민 5,167만명

- 2021년 9월 기준

국내 주요 개인정보 유출 사례

시기	기업	개인정보(만명)
2008년 2월	옥션	1,800
4월	하나로텔레콤	600
9월	GS칼텍스	1,125
2011년 4월	현대캐피탈	175
7월	SK컴즈	3,500
11월	넥슨	1,320
2012년 3월	SK텔레콤·KT	20
5월	EBS	400
7월	KT	870

KT 870 + 1200 만명

≡ 매일경제

뉴스 오피니언 프리미엄 연예 스포츠 증권 부동산

경제 기업 사회 국제 부동산 증권 정치 IT·과학 문화 기획·연재 Special Ed

www.kised.or.kr

stP 창업진흥원

국민과 함께 창업의 미래를 여는
혁신 창업 파트너

KT도 870만명 개인정보 줄줄 섰다

휴대폰 정보 5개월간 빼내 텔레마케팅에 활용...해커 2명 구속

황지혜, 배미정, 조진형 기자 | 입력 : 2012.07.29 18:34:36 수정 : 2012.07.31 10:42:15

0

뉴스룸 | 최신기사

KT 홈페이지 해킹...1천200만명 개인정보 털렸다

송고시간 | 2014-03-06 15:09 中文

강종구 기자

기자 페이지



LG U+, SKT의 1230만건

LG유플러스·SKT 등에서 고객정보 1230만건 또 유출

머니투데이 | 박소연 기자

VIEW 17,703 | 2014.03.11 11:06

   의견 남기기

부산 남부경찰서는 중국의 개인정보 유통업자 A씨 등
으로부터 입수한 1230만건의 개인정보를 유통한 혐
 의(개인정보보호법 위반)로 문모씨(44)를 구속했다고
 11일 밝혔다.

죽어야 끊는 담배..7일만에 "금연 비법" 밝...

경찰은 또 이를 구매해 통신판매업과 대출권유, 업체
홍보 등에 이용한 혐의로 권모씨(31) 등 17명을 불구
속 입건했다.



식사가 금상속

인터파크 1000만건

단독

인터파크 고객정보 1000여만건 유출...경찰, 해킹 혐의 수사 착수

구교형 기자 입력 : 2016.07.25 15:46 | 수정 : 2016.07.25 17:22



- 해외 IP 경유해 DB 서버 장악 시도
- 금품 요구 협박까지...‘2차 피해’ 우려

경찰이 인터넷 종합쇼핑몰 인터파크가 외부세력으로부터 해킹당해 1000여만건의 개인정보가 유출된 정황을 포착하고 수사에 나섰다.

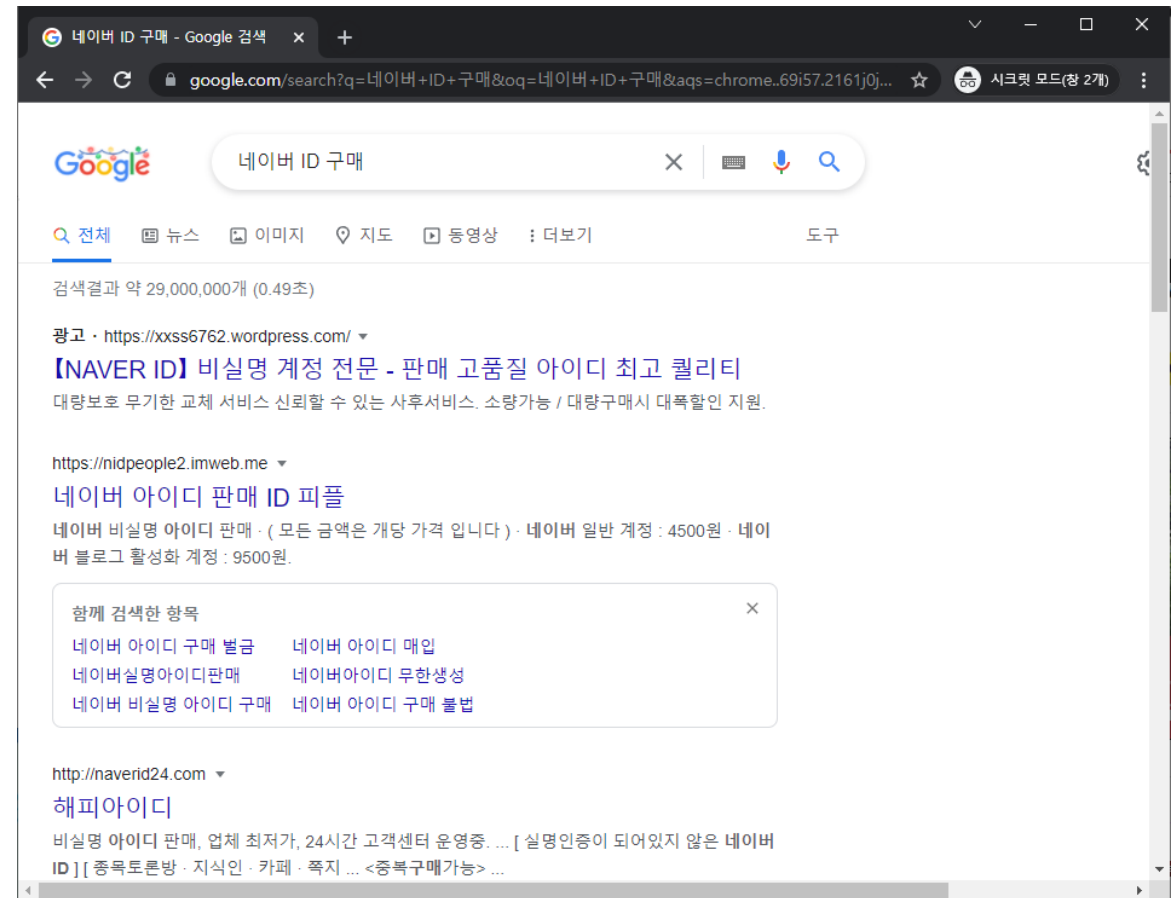
회사 직원 PC가 악성코드에 감염돼 데이터베이스(DB) 서버에서 정보가 새어나간 것으로 추정된다. 해킹을 한 일당은 회사 측을 상대로 폭로를 미끼로 금품까지 요구한 상황이어서 2차 피해가 우려된다.



네이버

2019 년 해킹

- 2200 여명
- 아이디는 1,000개 단위로 구매 가능 (?)



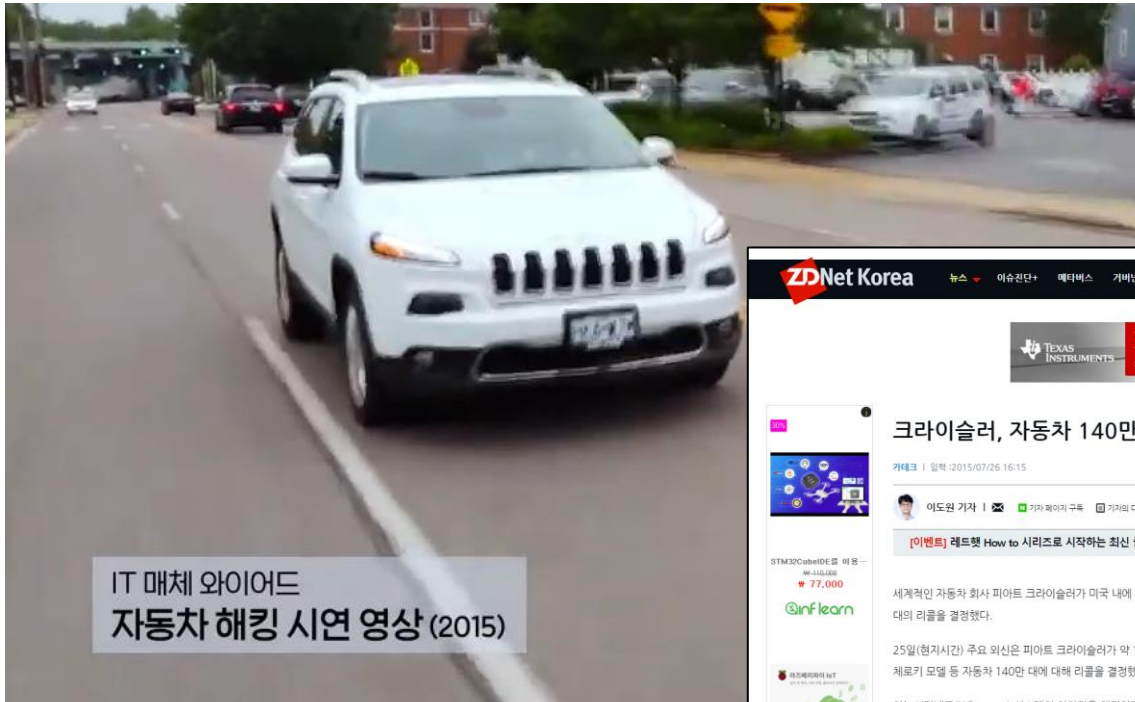
임베디드 해킹 사례 / 보안

테슬라 / 크라이슬러 해킹사례, 임베디드 보안 움직임

자동차 해킹 사례

2015년 BlackHat 컨퍼런스 : Jeep 해킹

- <https://youtu.be/dS2royTlqdQ?t=165>



‘테슬라의 굴욕’…中 텐센트 자회사 해킹팀, ‘모델S’ 해킹 성공

입력 2016-09-21 15:16 | 수정 2016-09-21 17:10

김나은 기자 [구독하기](#)

김정웅 기자 [구독하기](#)

자율주행 시스템 우려 더욱 커질 전망

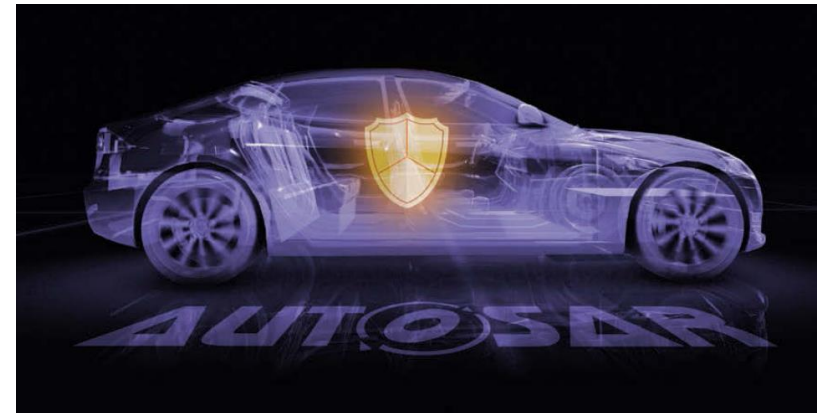
중국 텐센트 자회사의 해킹팀이 테슬라모터스의 전기차 '모델S'의 시스템을 해킹해 차량을 원격으로 조종하는 데 성공했다고 20일(현지시간) 워싱턴포스트(WP)가 보도했다. 이번 해킹 성공으로 테슬라의 자율주행 시스템 보안에 대한 우려가 커질 것으로 보인다.

보도에 따르면 중국 보안업체 킨시큐리티랩(Keen Security Lab-이하 킨랩)의 해킹팀은 전날 테슬라의 모델S 자율주행 시스템을 해킹해 원격 조종하는데 성공했다고 밝혔다. 킨랩은 중국 인터넷 기업 텐센트의 자회사다. 이 회사의 해킹팀은 이번 해킹으로 주행 중인 **차량의 브레이크를 제어하는 것은 물론** 사이드미러 와이퍼를 움직이고 트렁크 문을 여는 등 원격조종에 성공했다고 설명했다. 여기에 섀시프, 라이트, 차량 문 잠금도 원격 제어가 가능했다고 덧붙였다. 킨랩은 이번 테스트는 모델S에 국한돼 진행했으나 테슬라의 다른 차종에도 같은 조작이 적용될 수 있을 것이라고 설명했다.

AUTOSAR 보안

차량의 표준 프레임워크에 포함되어 있는 보안

- H/W, S/W Lib 모듈로 일관된 암호화 서비스
- TLS, IPSec 통신
- 보안 이벤트를 안전한 메모리에 보관
- 인가된 Application 리소스에 접근 가능
- Firewall / IDS



임베디드 시스템의 어려운 점 1

PC와 달리 IDS (침입탐지시스템) / Firewall 설치가 어렵다.

- 필요할 때 설치할 수 있는 시스템이 아니다.

일반적인 임베디드 시스템은..

- 보안을 위해 사용자 행동을 로깅하지 않는다. (Non-OS)
- 내부 메모리를 주기적으로 감시하지 않는다.
- 서버로 리포트 하지 않는다.
- 내부 메모리에 침입탐지 리포트 하지 않는다.

임베디드 시스템의 어려운 점 2

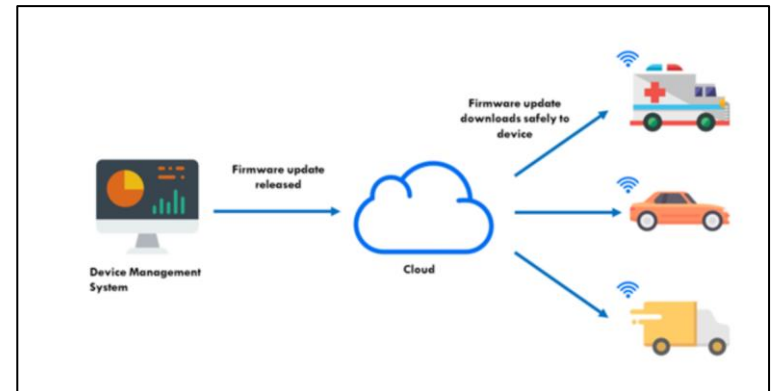
Zeroday Attack 대비 어려움

- 취약점 패치되기 전, 취약점 공격(Exploit)

→ 긴급 S/W 패치, Firmware Update 필요

→ FOTA (Firmware Over-The-Air programming)

- 몇 시간 이내 보안 패치를 전송하여, 자동으로 다운로드 설치
- 차량 소유주들이 센터에 들리지 않기에 필요



임베디드 시스템의 어려운 점 3

HW 모듈, SW 공급사 등이 따로 존재

→ 자체 제작은 보안 관리가 되지만,
외주 제작은 보안 심사의 어려움

임베디드 보안 대책

1. 암호화 / 인증 모듈
2. Firewall / IDS / 패킷분석 시스템 내장
3. SW 업데이트 대책 마련
4. 자체 Firmware 대신, 표준 플랫폼 OS 사용 (상용 RTOS, Linux 개발)
5. 고속 처리를 위한 보안 H/W 모듈 사용 (HSM : H/W Security Module)
6. 공급사 보안 인증 관리
7. 시큐어코딩

.. 등등

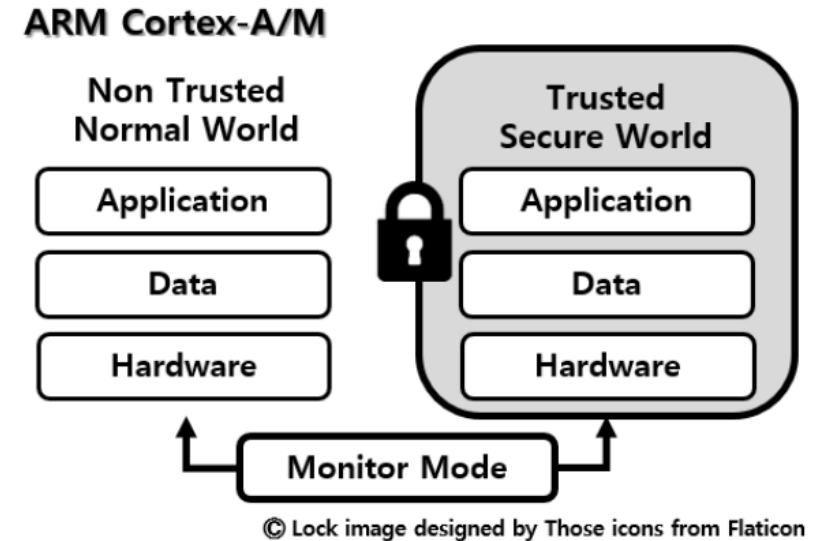
ARM의 TrustZone

Normal World

- Secure World 접근 불가
- 일반적인 Kernel / App

Secure World

- 인증된 Application 만 접근 가능
- Normal World 접근 가능
- 부팅시 해당 World로 부팅(PBL) → Normal World Bootloader



TPM (Trusted Platform Module)

TCG의 H/W 보안 표준

1. 암호화에 사용되는 키를 저장하는 NVRAM
2. 고속 Hash Engine 포함
3. Random 생성기 포함

Windows 11 설치 조건 : PC 내, TPM Chipset 필수

TPM-FAIL 취약점

TPM Data 탈취 / 디지털 서명 위조 가능한 취약점 발견

Intel / ST사 칩셋에서 발견

- CVE-2019-11090
- CVE-2019-16863



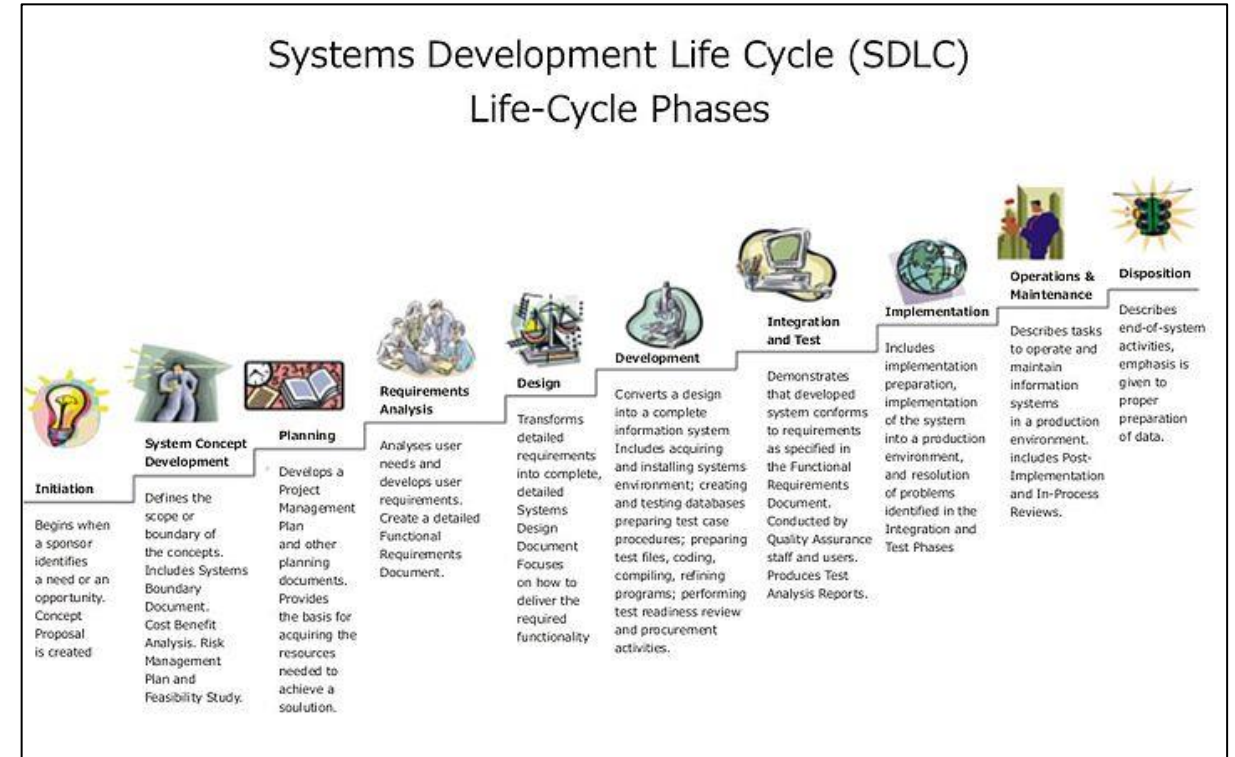
CVE : 알려진 보안 취약점을 뜻함 / 번호를 붙여 취약점 List를 미국 비영리 기관 MITRE에서 관리

Secure SDLC

SDLC

S/W 생애주기

- 분석→설계→개발→테스트→운영

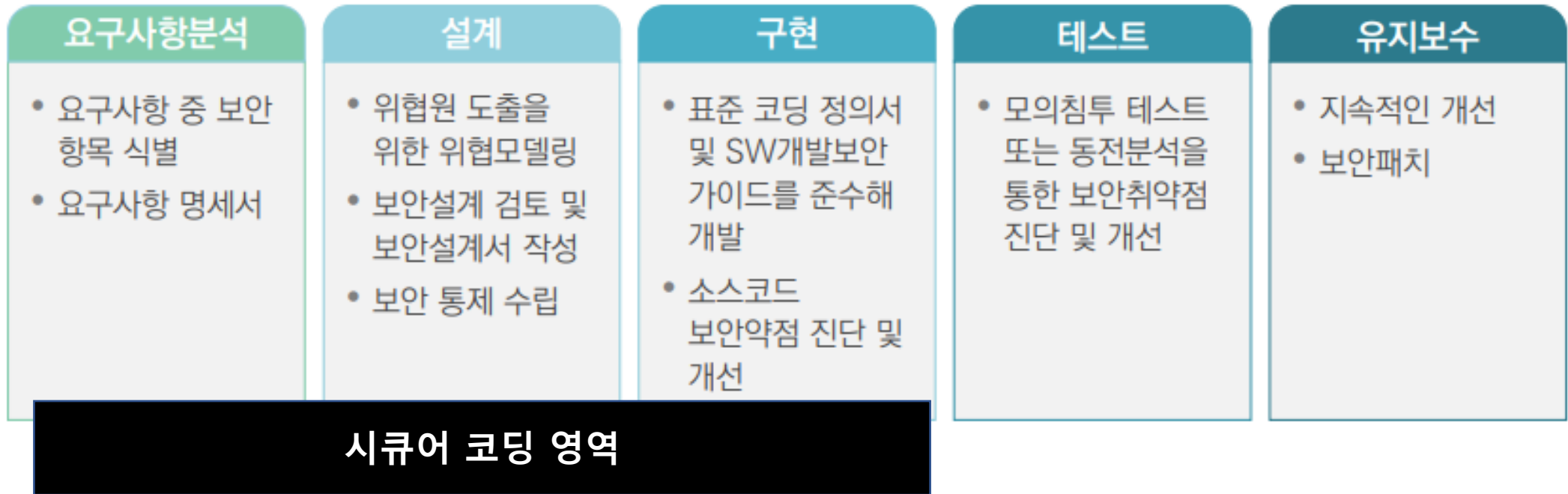


미국 법무부 SDLC 10단계

KISA의 Secure SDLC

SW개발생명주기에 보안 항목을 추가

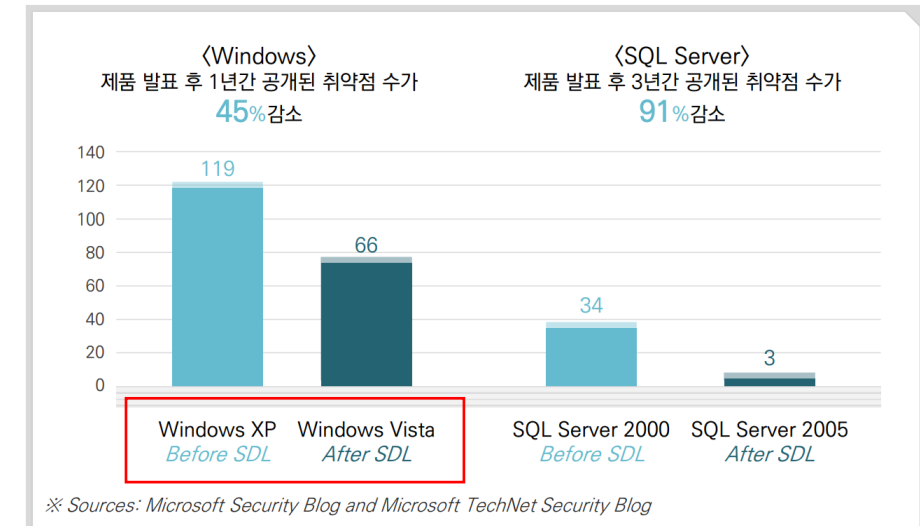
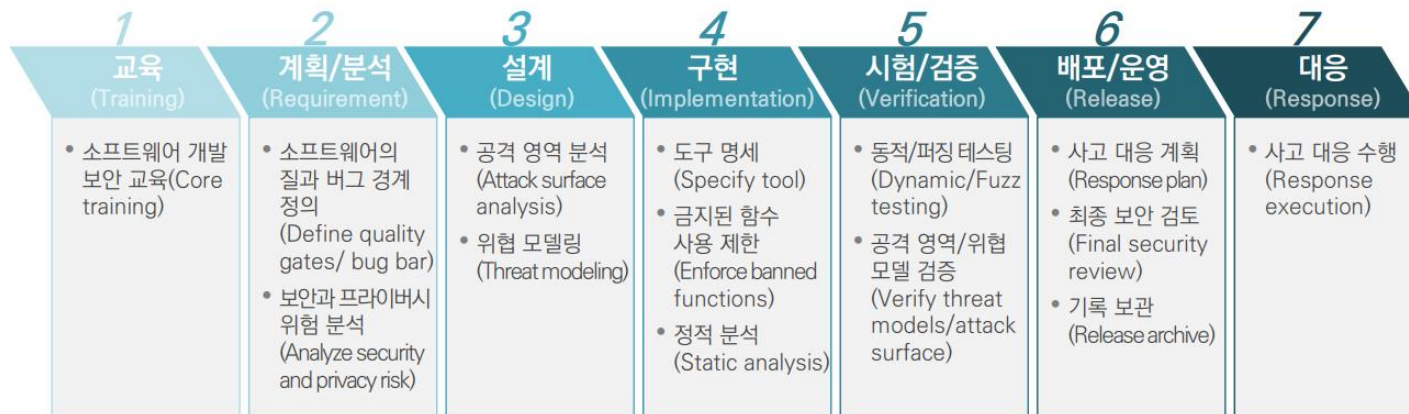
- 요구사항 분석 부터 보안활동을 해야 함을 강조



MS-SDL

MS의 SDLC (Secure Development Life Cycle)

- MS에서 안전한 S/W를 개발하기 위한 개발 프로세스
- SDL 적용 대비 50% 이상 취약점이 감소되었다고 발표함



[참고] 취약점 뉴스 보고 / 리서치

삼성전자 취약점 뉴스 기사

- 타이젠 2017, <https://www.bloter.net/newsView/blt201704060005>
- Smarthings 2018, <https://www.itworld.co.kr/news/110257>
- 갤럭시 2019, <https://www.dailysecu.com/news/articleView.html?idxno=70111>

해외 기업의 취약점 수

- 2020년 기준

Name	Rank 2020	Rank 2019	Count 2020	Count 2019
Microsoft Corporation	1 	9	1566	940
Oracle Corporation	2	2	1336	1515
Red Hat	3 	5	1327	1407
Google	4 	8	1254	1237
SUSE	5 	1	1214	1688
IBM Corporation	6 	4	1094	1429
Software in the Public Interest, Inc.	7 	3	1056	1455
Canonical Ltd.	8 	7	894	1280
Dell	9 	6	746	1380
Cisco Systems	10	10	692	624

단계별 보안 취약점 제거 필요

요구사항 분석 / 설계 단계

- 시큐어코딩

구현 단계

- 시큐어 코딩

테스트 단계

- 바이너리 취약점 분석 (외부 Firmware / Library)
- 보안 정적 분석도구 활용
- 모의 해킹

KISA Secure Coding

분석/설계 단계에서 SW 강화 활동

관리 정보를 명시, 관리 방법 설계

우측 표는

관리해야 하는 정보들에 대한 등급 예시

- KISA 기준

개인정보 별
암호화 등 안전성 확보에 필요한 조치를
설계가 필요하다.

- 일방향 암호화 (Hash)
- 보다 안전한 알고리즘 암호화
- SSL 인증서 등

등급	설명	자산가치	분류	개인정보 종류
1등급	그 자체로 개인 식별이 가능하거나, 민감한 개인정보 또는 관련 법령에 따라 처리가 엄격히 제한된 개인정보, 유출 시 범죄에 직접적으로 이용 가능한 정보 등	5	고유식별정보	• 취약한 API 사용 통제 • 주민번호, 여권번호, 운전면허번호, • 외국인등록번호
			민감정보	• 사상·신념, 노동조합·정당의 가입·탈퇴, 정치적 견해, 병력 (病歷), 신체적·정신적 장애, 성적(性的)취향, 유전자 검사정보, 범죄경력정보 등 사생활을 현저하게 침해할 수 있는 정보
			인증정보	• 비밀번호, 바이오정보(지문, 홍채, 정맥 등)
			신용정보/ 금융정보	• 신용정보, 신용카드번호, 계좌번호 등
			의료정보	• 건강상태, 진료기록 등
			위치정보	• 개인 위치정보 등
			기타 중요정보	• 해당 사업의 특성에 따라 별도 정의
2등급	조합되면 명확히 개인의 식별이 가능한 개인정보, 유출 시 법적 책임 부담 가능한 정보	3	개인식별정보	• 이름, 주소, 전화번호, 핸드폰번호, 이메일 주소, 생년월일, 성별 등
			개인관련정보	• 학력, 직업, 키, 몸무게, 혼인 여부, 가족상황, 취미 등
			기타 개인정보	• 해당 사업의 특성에 따라 별도 정의
3등급	개인정보와 결합하여 부가적인 정보 제공 가능 정보, 제한적인 분야에서 불법적 이용 가능 정보	1	자동생성정보	• IP정보, MAC주소, 사이트 방문 기록, 쿠키(cookie) 등
			가공정보	• 통계성 정보, 가입자 성향 등
			제한적 본인 식별 정보	• 회원번호, 사번, 내부용 개인식별정보 등
			기타 간접 개인정보	• 해당 사업의 특성에 따라 별도 정의

입력데이터 검증 및 표현

사용자, 프로그램 입력 데이터에 대한
유효성 검증 체계를 갖추고,
실패시 처리 방법을 설계

번호	설계항목	설명	비고
1	DBMS 조회 및 결과 검증	• DBMS 조회시 질의문(SQL) 내 입력값과 그 조회결과에 대한 유효성 검증방법(필터링 등) 설계 및 유효하지 않은 값에 대한 처리방법 설계	입출력 검증
2	XML조회 및 결과 검증	• XML 조회시 질의문(XPath, XQuery 등) 내 입력값과 그 조회결과에 대한 유효성 검증방법(필터링 등) 설계 및 유효하지 않은 값에 대한 처리방법 설계	
3	디렉토리 서비스 조회 및 결과 검증	• 디렉토리 서비스(LDAP 등)를 조회할 때 입력값과 그 조회결과에 대한 유효성 검증방법 설계 및 유효하지 않은 값에 대한 처리방법 설계	
4	시스템 자원 접근 및 명령어 수행 입력값 검증	• 시스템 자원접근 및 명령어를 수행할 때 입력값에 대한 유효성 검증 방법 설계 및 유효하지 않은 값에 대한 처리방법 설계	
5	웹 서비스 요청 및 결과 검증	• 웹 서비스(게시판 등) 요청(스크립트 게시 등)과 응답결과 (스크립트를 포함한 웹 페이지)에 대한 유효성 검증방법 설계 및 유효하지 않은 값에 대한 처리방법 설계	
6	웹 기반 중요 기능 수행 요청 유효성 검증	• 비밀번호 변경, 결제 등 사용자 권한 확인이 필요한 중요기능을 수행할 때 웹 서비스 요청에 대한 유효성 검증방법 설계 및 유효하지 않은 값에 대한 처리방법 설계	
7	HTTP 프로토콜 유효성 검증	• 비정상적인 HTTP 헤더, 자동연결 URL 링크 등 사용자가 원하지 않은 결과를 생성하는 HTTP 헤더·응답결과에 대한 유효성 검증방법 설계 및 유효하지 않은 값에 대한 처리방법 설계	
8	허용된 범위내 메모리 접근	• 해당 프로세스에 허용된 범위의 메모리 버퍼에만 접근하여 읽기 또는 쓰기 기능을 하도록 검증방법 설계 및 메모리 접근 요청이 허용범위를 벗어났을 때 처리방법 설계	
9	보안기능 입력값 검증	• 보안기능(인증, 권한부여 등) 입력 값과 함수(또는 메소드)의 외부입력 값 및 수 행결과에 대한 유효성 검증방법 설계 및 유효하지 않은 값에 대한 처리방법 설계	
10	업로드·다운로드 파일 검증	• 업로드·다운로드 파일의 무결성, 실행권한 등에 관한 유효성 검증방법 설계 및 부적합한 파일에 대한 처리방법 설계	파일 검증

입력데이터 검증시 신경써야할 보안약점

구분	설계단계	구현단계
입력 데이터 검증 및 표현 (10개)	DBMS 조회 및 결과 검증	<ul style="list-style-type: none"> • SQL 삽입
	XML 조회 및 결과 검증	<ul style="list-style-type: none"> • XML 삽입 • 부적절한 XML 외부개체 참조
	디렉토리 서비스 조회 및 결과 검증	<ul style="list-style-type: none"> • LDAP 삽입
	시스템 자원 접근 및 명령어 수행 입력값 검증	<ul style="list-style-type: none"> • 코드 삽입 • 경로조작 및 자원삽입 • 서버사이드 요청 위조 • 운영체제 명령어 삽입
	웹 서비스 요청 및 결과 검증	<ul style="list-style-type: none"> • 크로스사이트 스크립트
	웹 기반 중요 기능 수행 요청 유효성 검증	<ul style="list-style-type: none"> • 크로스사이트 요청 위조
	HTTP 프로토콜 유효성 검증	<ul style="list-style-type: none"> • 신뢰되지 않은 URL 주소로 자동접속 연결 • HTTP 응답분할
	허용된 범위내 메모리 접근	<ul style="list-style-type: none"> • 포맷스트링 삽입 • 메모리 버퍼 오버플로우
	보안기능 입력값 검증	<ul style="list-style-type: none"> • 보안기능 결정에 사용되는 부적절한 입력값 • 정수형 오버플로우 • Null Pointer 역참조
	업로드 · 다운로드 파일검증	<ul style="list-style-type: none"> • 위험한 형식 파일 업로드 • 부적절한 전자서명 확인 • 무결성 검사 없는 코드 다운로드

보안기능

인증, 접근통제,
권한관리, 비밀번호 등
정책이 적절히 반영
될수있도록 설계

번호	설계항목	설명	비고
1	인증 대상 및 방식	• 중요정보·기능의 특성에 따라 인증방식을 정의하고 정의된 인증방식을 우회하지 못하게 설계	인증 관리
2	인증 수행 제한	• 반복된 인증 시도를 제한하고 인증 실패한 이력을 추적하도록 설계	
3	비밀번호 관리	• 생성규칙, 저장방법, 변경주기 등 비밀번호 관리정책별 안전한 적용 방법 설계	
4	중요자원 접근통제	• 중요자원(프로그램 설정, 민감한 사용자 데이터 등)을 정의하고, 정의된 중요자원에 대한 신뢰할 수 있는 접근통제 방법(권한관리 포함) 설계 및 접근통제 실패 시 처리방법 설계	접근 권한 관리
5	암호키 관리	• 암호키 생성, 분배, 접근, 파기 등 암호키 생명주기별 암호키 관리 방법을 안전하게 설계	암호 관리
6	암호연산	• 국제표준 또는 검증필 암호모듈로 등재된 안전한 암호 알고리즘을 선정하고 충분한 암호키 길이, 솔트, 충분한 난수값을 적용한 안전한 암호연산 수행방법 설계	
7	중요정보 저장	• 중요정보(비밀번호, 개인정보 등)를 저장·보관하는 방법이 안전하도록 설계	중요 정보 처리
8	중요정보 전송	• 중요정보(비밀번호, 개인정보, 쿠키 등)를 전송하는 방법이 안전하도록 설계	

보안기능 설계시, 신경써야할 보안약점

보안 기능 (8개)	인증 대상 및 방식	<ul style="list-style-type: none">• 서버사이드 요청 위조• 적절한 인증 없는 중요기능 허용• 부적절한 인증서 유효성 검증• DNS lookup에 의존한 보안결정
	인증 수행 제한	<ul style="list-style-type: none">• 반복된 인증시도 제한기능 부재
	비밀번호 관리	<ul style="list-style-type: none">• 하드코드된 중요정보• 취약한 비밀번호 허용
	중요자원 접근통제	<ul style="list-style-type: none">• 부적절한 인가• 중요한 자원에 대한 잘못된 권한 설정
	암호키 관리	<ul style="list-style-type: none">• 하드코드된 중요정보• 주석문 안에 포함된 시스템 주요 정보
	암호연산	<ul style="list-style-type: none">• 취약한 암호화 알고리즘 사용• 충분하지 않은 키 길이 사용• 적절하지 않은 난수 값 사용• 부적절한 인증서 유효성 검증• 솔트 없이 일방향 해쉬함수 사용
	중요정보 저장	<ul style="list-style-type: none">• 암호화 되지 않은 중요정보• 사용자 하드디스크에 저장되는 쿠키를 통한 정보 노출
	중요정보 전송	<ul style="list-style-type: none">• 암호화 되지 않은 중요정보

에러처리 / 세션통제

- 예외처리가 모두 처리되게끔 설계
- HTTP을 통해 연결 유지시, 세션 정보 노출, 하이재킹 사고 발생하지 않도록 설계

번호	설계항목	설명	비고
1	예외처리	<ul style="list-style-type: none"> 오류메시지에 중요정보(개인정보, 시스템 정보, 민감 정보 등)가 노출되거나, 부적절한 에러·오류 처리로 의도치 않은 상황이 발생하지 않도록 설계 	에러 처리

번호	설계항목	설명	비고
1	세션통제	<ul style="list-style-type: none"> 다른 세션 간 데이터 공유금지, 세션 ID 노출금지, (재)로그인시 세션 ID 변경, 세션종료(비활성화, 유효기간 등) 처리 등 세션을 안전하게 관리할 수 있는 방안 설계 	세션 통제

에러 처리 (1개)	예외처리	<ul style="list-style-type: none"> 오류 메시지 정보노출
세션 통제 (1개)	세션통제	<ul style="list-style-type: none"> 잘못된 세션에 의한 데이터 정보 노출

구현단계 보안약점

- 설계단계에서 부터 고려되어야 하는 것들을 색으로 표현됨

	입력데이터 검증 및 표현	보안기능		에러처리	세션통제	
입력 데이터 검증 및 표현 (17개)	SQL 삽입	코드 삽입	경로 조작 및 자원 삽입	크로스사이트 스크립트	운영체제 명령어 삽입	위험한 형식 파일 업로드
	신뢰되지 않은 URL 주소로 자동 접속 연결	부적절한 XML 외부개체 참조	XML 삽입	LDAP 삽입	크로스사이트 요청 위조	서버사이드 요청 위조
	HTTP 응답 분할	정수형 오버플로우	보안기능 결정에 사용되는 부적절한 입력값	메모리 버퍼 오버플로우	포맷 스트링 삽입	
보안 기능 (16개)	적절한 인증없는 중요 기능 허용	부적절한 인가	중요한 자원에 대한 잘못된 권한 설정	취약한 암호화 알고리즘 사용	암호화되지 않은 중요정보	하드코딩된 중요정보
	충분하지 않은 키 길이 사용	적절하지 않은 난수값 사용	취약한 비밀번호 허용	부적절한 전자서명 확인	부적절한 인증서 인증서 유효성 검증	사용자 하드 디스크에 저장되는 쿠키를 통한 정보 노출
	주석문 안에 포함된 시스템 주요정보	솔트 없이 일방향 해쉬 함수 사용	무결성 검사 없는 코드 다운로드	반복된 인증시도 제한 기능 부재		
시간 및 상태 (2개)	경쟁조건: 검사 시점과 사용 시점(TOCTOU)	종료되지 않은 반복문 재귀함수				
에러처리 (3개)	오류메시지 정보 노출	오류 상황 대응 부재	부적절한 예외 처리			
코드오류 (5개)	Null Pointer 역참조	부적절한 자원해제	해제된 자원 사용	초기화되지 않은 변수 사용	신뢰할 수 없는 데이터의 역직렬화	
캡슐화 (4개)	잘못된 세션에 의한 데이터 정보노출	제거되지 않고 남은 디버그 코드	Public 메소드 부터 반환된 Private 배열	private 배열에 public 데이터 할당		
API 오용 (2개)	DNS Lookup에 의존한 보안 결정	취약한 API 사용				

취약성 생기지 않도록 설계하는 방법

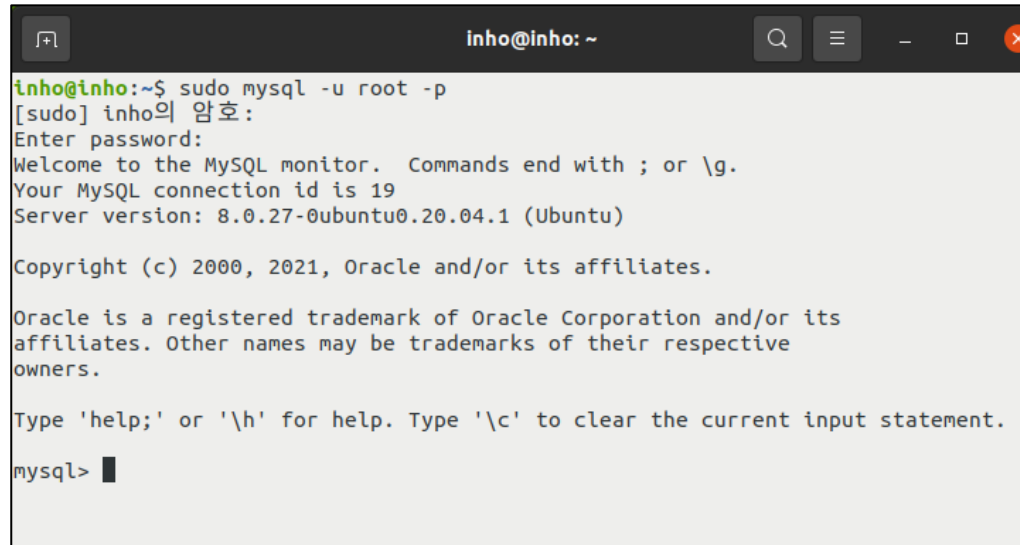
- 제작 초기부터 안전하게 설계된 미들웨어 프레임워크를 도입
- 공통 라이브러리 구축하는 방법
 - 예시
 - 자동차 보안 모듈을 가진 AUTOSAR
 - 표준화된 통합 프레임워크
 - Spring
 - 안전한 웹 프레임워크

SQL Injection을 위한 개발 환경 세팅

개발환경 세팅, mysql을 root로 접속하기

VirtualBox + Ubuntu 18.04 LTS

- `sudo apt install mysql-server`
- `sudo mysql -u root -p`
- 비번은 우분투 Password 입력하기

A terminal window titled 'inho@inho: ~' with standard Ubuntu window controls. The terminal shows the command 'sudo mysql -u root -p' being executed. It prompts for a password, then displays a welcome message for the MySQL monitor, including the connection ID (19) and server version (8.0.27-0ubuntu0.20.04.1). It also shows copyright and trademark information for Oracle. The prompt changes from 'mysql>' to 'mysql> ' with a cursor.

```
inho@inho:~$ sudo mysql -u root -p
[sudo] inho의 암호:
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 19
Server version: 8.0.27-0ubuntu0.20.04.1 (Ubuntu)

Copyright (c) 2000, 2021, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> █
```

작업 DB 만들기

create database **testdb**;

- utf-8 인코딩으로 생성된다.

show databases;

```
mysql> create database testdb
-> ;
Query OK, 1 row affected (0.02 sec)

mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| performance_schema |
| sys |
| testdb |
+-----+
5 rows in set (0.00 sec)
```

유저 만들기

```
mysql>
mysql> create user inho@localhost identified by '1';
Query OK, 0 rows affected (0.01 sec)

mysql> grant all privileges on testdb.* to inho@localhost;
Query OK, 0 rows affected (0.01 sec)

mysql>
```

create user **inho**@localhost identified by '**1**';

- 유저 이름 : inho
- 접근 허용 위치 : localhost (해당 PC에서만)
- 비밀번호 : 1

grant all privileges on **testdb.*** to **inho@localhost**;

- 권한 부여 DB : testdb
- 권한 부여 ID : inho@localhost

유저 잘 만들어졌는지 확인하기

use mysql;

- 읽을 DB 선택

select user, host, from user;

- user Table에서
user와 host 필드 확인

```
mysql> use mysql
Database changed
mysql>
mysql> select user, host from user;
+-----+-----+
| user          | host      |
+-----+-----+
| debian-sys-maint | localhost |
| inho           | localhost |
| mysql.infoschema | localhost |
| mysql.session   | localhost |
| mysql.sys       | localhost |
| root           | localhost |
+-----+-----+
6 rows in set (0.00 sec)

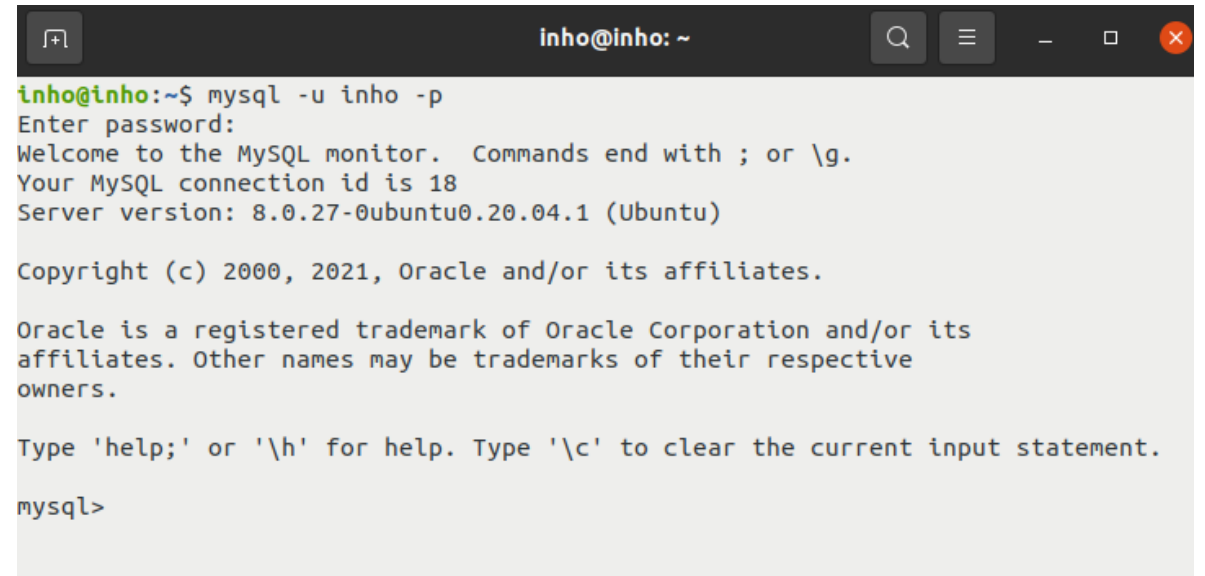
mysql>
```

접속 테스트

root로 접속된 mysql을 종료

mysql 접속하기

- `mysql -u inho -p`
- 비번 입력

A terminal window titled 'inho@inho: ~' with standard window controls. It shows the execution of the command 'mysql -u inho -p'. The prompt 'Enter password:' is shown, followed by a series of welcome messages from the MySQL monitor, including the connection ID (18) and server version (8.0.27-0ubuntu0.20.04.1). The prompt 'mysql>' is visible at the bottom.

```
inho@inho:~$ mysql -u inho -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 18
Server version: 8.0.27-0ubuntu0.20.04.1 (Ubuntu)

Copyright (c) 2000, 2021, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

user Table 추가하기

DB 사용하기

- use testdb;

Table 생성하기

- create table **user** (
 no int not null auto_increment,
 id varchar(20),
 pass varchar(20),
 address varchar(50),
 primary key(**no**)
);

결과 확인

- desc user

```
mysql> desc user;
```

Field	Type	Null	Key	Default	Extra
no	int	NO	PRI	NULL	auto_increment
id	varchar(20)	YES		NULL	
pass	varchar(20)	YES		NULL	
address	varchar(50)	YES		NULL	

```
4 rows in set (0.00 sec)
```

```
mysql>
```

Sample data 추가하기

insert into 문, 3명 정보 넣기

- insert into user(id, pass, address) values('admin', '1234', 'seoul');
- insert into user(id, pass, address) values('who', '12', 'busan');
- insert into user(id, pass, address) values('are', '34', 'gangwon');

결과 확인

- select * from user;

```
mysql>
mysql> select * from user;
+----+-----+-----+-----+
| no | id    | pass | address |
+----+-----+-----+-----+
| 1  | admin | 1234 | seoul   |
| 2  | who   | 12   | busan   |
| 3  | are   | 34   | gangwon |
+----+-----+-----+-----+
3 rows in set (0.00 sec)

mysql> █
```

작업 디렉토리

작업디렉토리 이름 : sql

- mkdir ~/sql
- cd sql

C++ 개발세팅

C / C++ 개발 세팅

sudo apt install vim make gcc g++ -y

- vim 에디터 설치
- make build system 설치
- gcc / g++ 설치
- -y 옵션 : 설치 확인 메시지 안띄움

vi 대신 gedit를 써도 무방

C++과 MySQL의 커넥터

mysql 커넥터 설치

- `sudo apt-get install libmysqlcppconn-dev`

공식 개발 문서

- <https://dev.mysql.com/doc/connector-cpp/8.0/en/connector-cpp-introduction.html>

C++ 기본 코드

접속 후

Table 정보 가져오는 코드

[C++_SQLInjection1] 접속코드.cpp

```
1 #include <iostream>
2 #include <cppconn/driver.h>
3 #include <cppconn/resultset.h>
4 #include <cppconn/statement.h>
5 #include <cppconn/exception.h>
6
7 using std::cout;
8 using std::cin;
9
10 int main() {
11
12     try{
13         //1. connection
14         sql::Driver *driver = get_driver_instance();
15         sql::Connection *con;
16         con = driver->connect("localhost", "incho", "1");
17         con->setSchema("testdb");
18
19         sql::Statement *state;
20         sql::ResultSet *result;
21
22         //2. sql command
23         state = con->createStatement();
24         result = state->executeQuery("select * from user");
25
26         //3. print result
27         while (result->next()) {
28             cout << result->getString("id") << " ";
29             cout << result->getString("pass") << " ";
30             cout << result->getString("address") << " ";
31             cout << "\n";
32         }
33
34         delete result;
35         delete state;
36         delete con;
37
38     } catch (sql::SQLException &e) {
39
40         cout << "====SQL EXCEPTION====\n";
41         cout << "FUNCTION : " << __FUNCTION__ << "\n";
42         cout << "LINE : " << __LINE__ << "\n";
43         cout << "ERR : " << e.what() << "\n";
44         cout << "Error Code : " << e.getErrorCode() << "\n";
45         cout << "SQL State : " << e.getSQLState() << "\n";
46     }
47
48     return 0;
49 }
```

출처 : 공식 reference

<https://dev.mysql.com/doc/connector-cpp/1.1/en/connector-cpp-examples-complete-example-1.html>

C++ Makefile

빌드 후 실행 코드

실행방법

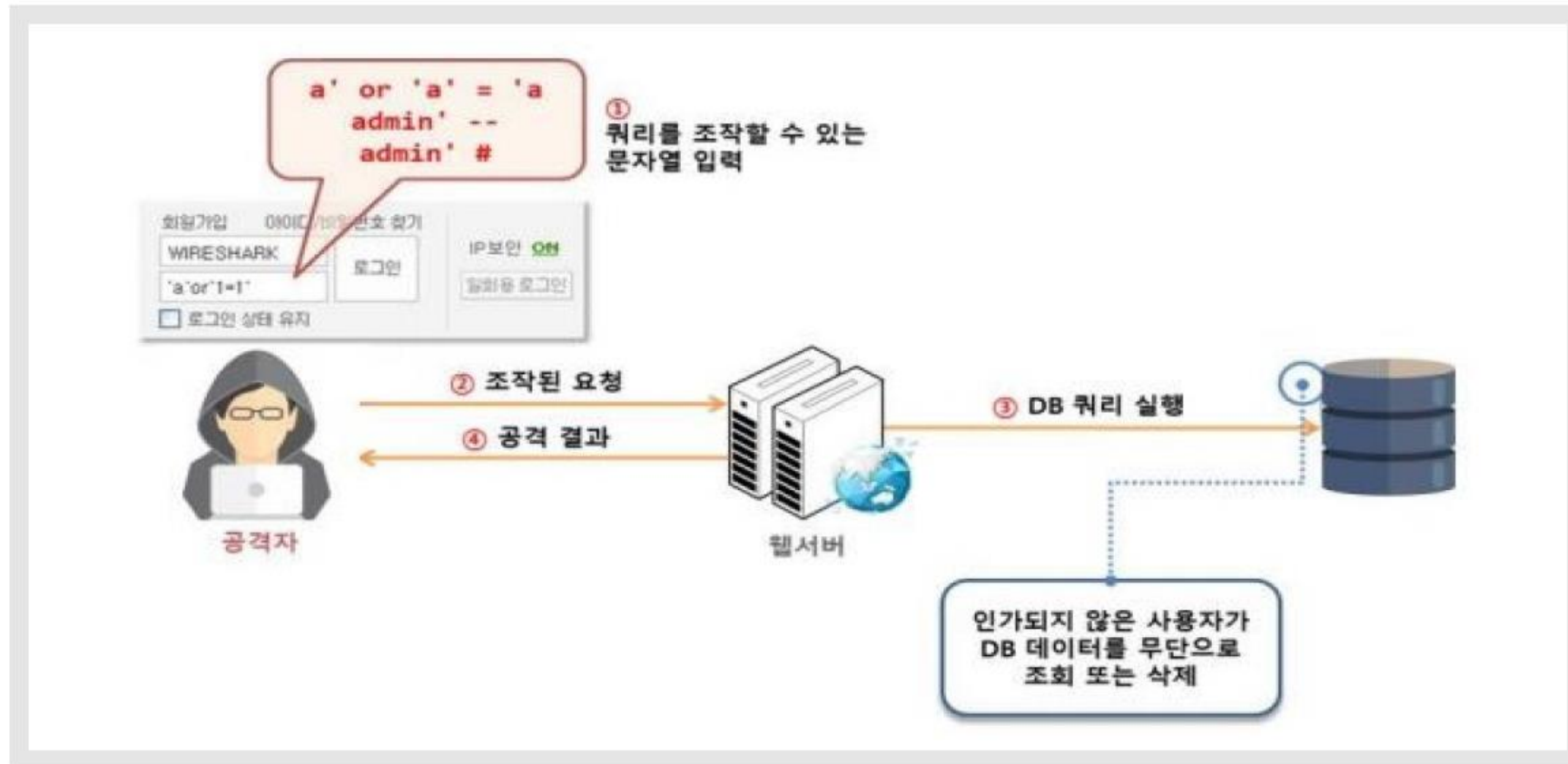
- make

```
inho@inho:~/sql
1 cpp:
2     g++ -I/usr/include/cppconn -o test test.cpp -lmysqlcppconn && ./test
3
~
~
~
```

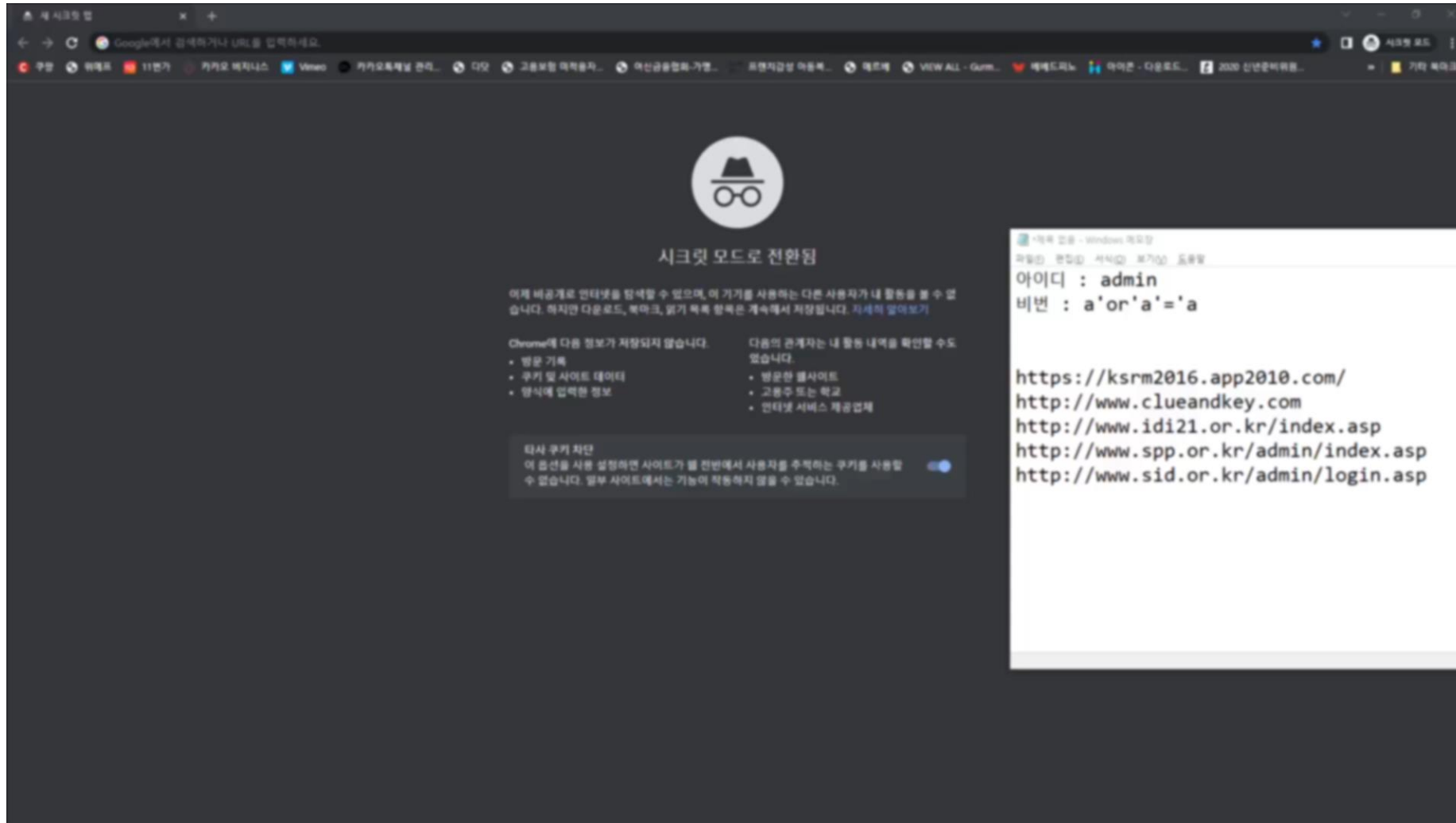
SQL Injection 원리

입력데이터에 대한 유효성 검증 안할 경우

Query 조작 명령어 삽입



SQL Injection 예시



랜섬웨어 예시 (몸값요구)

비트코인 입금 요구 메시지

- SQL Injection 으로 가능

```
mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| WARNING |
+-----+
2 rows in set (0.00 sec)

mysql> use WARNING;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> show tables;
+-----+
| Tables_in_WARNING |
+-----+
| Readme |
+-----+
1 row in set (0.00 sec)

mysql> select * from Readme;
+-----+
| id | eMail | BitCoin | Exchange | Description |
+-----+
| 1 | cru3lty@safe-mail.net | 1G5tfypKqHGds8WsYe1HR5JxiwffRzUUas | https://localbitcoins.com | Your Database is downloaded and backed up on our secured servers. To recover your lost data: Send 0.2 BTC to our Bitcoin Address and Contact us by eMail with your MySQL server IP Address and a Proof of Payment. Any eMail without your MySQL server IP Address and a Proof of Payment together will be ignored. You are welcome. |
+-----+
```

[C++] ID Password 검사 코드로 수정

```
//1. connection
sql::Driver *driver = get_driver_instance();
sql::Connection *con;
con = driver->connect("localhost", "incho", "1");
con->setSchema("testdb");
```

```
sql::Statement *state;
sql::ResultSet *result;
```

```
string id;
string pass;
```

```
cout << "ID : ";
cin >> id;
```

```
cout << "PASS : ";
cin >> pass;
```

```
string sql;
sql += "select * from user where id = ";
sql += id;
sql += "' ";
sql += "and pass = ";
sql += pass;
sql += "'";
```

```
//cout << sql << "\n";
```

```
//2. sql command
state = con->createStatement();
result = state->executeQuery(sql.c_str());
```

```
//3. print result
```

```
bool isLogin = false;
string getID, getPass, getAddress;
```

```
if (result->next()) {
    getID = result->getString("id");
    getPass = result->getString("pass");
    getAddress = result->getString("address");
```

```
    isLogin = true;
```

```
}
```

```
if (isLogin == true) {
    cout << "LOG IN SUCCEESSD\n";
```

```
}
```

```
else {
    cout << "LOG IN FAIL\n";
```

```
}
```

```
delete result;
```

```
delete state;
```

```
delete con;
```

동작 테스트

password

- a'or'a'='a 로 로그인 성공
- admin'or'a'='a'--
 - 이후 주석처리 코드,
 - 로그인 성공

```
inho@inho:~/sql$ make cpp
g++ -I/usr/include/cppconn -o test test.cpp -lmysqlcppconn && ./test
ID : admin
PASS : 12
LOG IN FAIL
inho@inho:~/sql$
inho@inho:~/sql$ make cpp
g++ -I/usr/include/cppconn -o test test.cpp -lmysqlcppconn && ./test
ID : admin
PASS : 1234
LOG IN SUCCEESSD
inho@inho:~/sql$
inho@inho:~/sql$ make cpp
g++ -I/usr/include/cppconn -o test test.cpp -lmysqlcppconn && ./test
ID : admin
PASS : a'or'a'='a
LOG IN SUCCEESSD
inho@inho:~/sql$
inho@inho:~/sql$ make cpp
g++ -I/usr/include/cppconn -o test test.cpp -lmysqlcppconn && ./test
ID : admin'or'a'='a'--
PASS : 1
LOG IN SUCCEESSD
```

SQL Injection 두 가지 기본 형태

a'or'**a**'=**a**

- 뒷 부분 코드, 모두 참

```
ID : admin  
PASS : a'or'a'='a  
select * from user where id = 'admin' and pass = 'a'or'a'='a';
```

admin'or'a'='a'--

- 모두 참, 이후 뒷부분 모두 주석

```
ID : admin'or'a'='a--  
PASS : 1  
select * from user where id = 'admin'or'a'='a--' and pass = '1';  
LOG IN SUCCEESSD _
```

SQL Injection 방어 방법

유효성 검사

- 문자열에 특수 문자 검사
- 허용된 글자수인지 검사

안전한 Statement class 사용

- 안전하지 않은 `Statement` class 대신
C++ : `prepareStatement` class 사용



[C++] 안전한 코드

state → pState 로 변경

```
1 #include <iostream>
2 #include <string>
3 #include <cppconn/driver.h>
4 #include <cppconn/resultset.h>
5 #include <cppconn/statement.h>
6 #include <cppconn/exception.h>
7 #include <cppconn/prepared_statement.h>
8
9 using std::cout;
10 using std::cin;
11 using std::string;
12
13 int main() {
14
15     try{
16         //1. connection
17         sql::Driver *driver = get_driver_instance();
18         sql::Connection *con;
19         con = driver->connect("localhost", "incho", "1");
20         con->setSchema("testdb");
21
22         sql::PreparedStatement *pState;
23         sql::ResultSet *result;
24
25         string id;
26         string pass;
27     }
```

```
//2. sql command
pState = con->prepareStatement("select * from user where id = ? and pass = ?");
pState->setString(1, id);
pState->setString(2, pass);
result = pState->executeQuery();

//3. print result
bool isLogin = false;
string getID, getPass, getAddress;

if (result->next()) {
    getID = result->getString("id");
    getPass = result->getString("pass");
    getAddress = result->getString("address");

    isLogin = true;
}

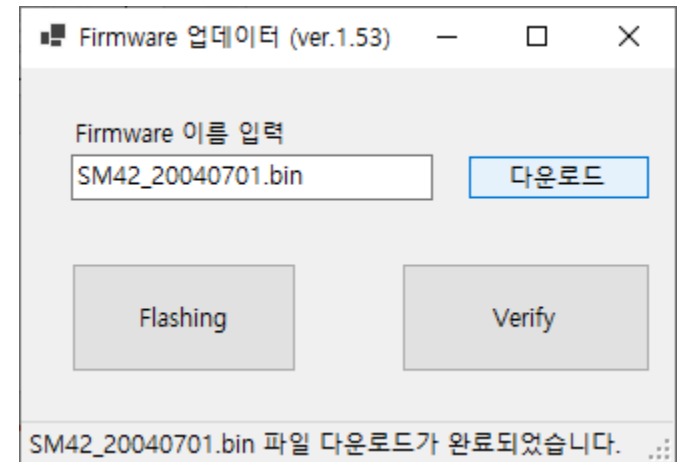
if (isLogin == true) {
    cout << "LOG IN SUCCEESSD\n";
}
else {
    cout << "LOG IN FAIL\n";
}
```

경로조작

예시. Firmware 다운로드 Tool

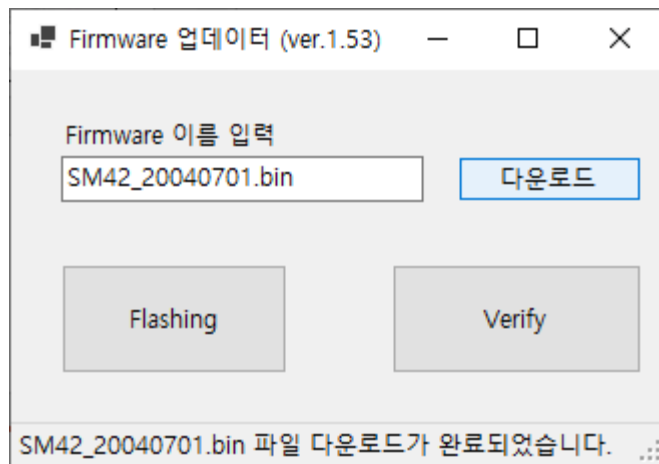
서버에 저장된 Firmware 이름을 입력하면,
해당 Firmware를 다운로드 & Flashing을 해주는 도구

- 파일명 입력시
→ 배포서버에서 다운로드 기능

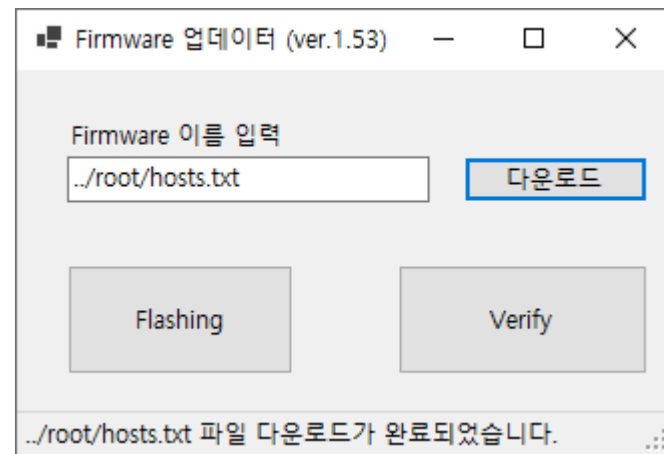


경로조작 이슈

파일명에 경로까지 포함하여 입력 시
Server에 있는 파일들을 다운로드 할 수 있음



정상적인 사용 예시



비정상적인 사용 예시

이름 저장 프로그램

1. 이름변경 → NEW 이름 입력
2. Save → 파일명 입력
3. Load → 파일명 입력

```
=====
NAME : NO NAME
=====

1. Change Name
2. Save Name To File
3. Load Name To File

CMD :
```



```
=====
NAME : NO NAME
=====

1. Change Name
2. Save Name To File
3. Load Name To File

CMD : 1
NEW NAME : INHO
```



```
=====
NAME : INHO
=====

1. Change Name
2. Save Name To File
3. Load Name To File

CMD : █
```

안전하지 않은 소스코드

<https://gist.github.com/mincoding1/b7495c779fd0e5ab3775a9a13d343a>

문제 발생

- Save Name 으로 내부 파일이 덮어쓰기 가능 → 시스템 망가짐
- Load Name 으로 내부 파일 읽을 수 있음 → 정보 탈취

```
=====
NAME : NO NAME
=====

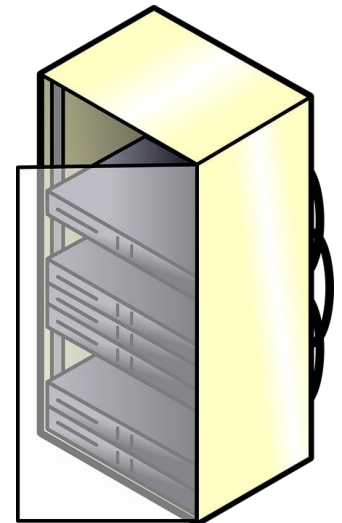
1. Change Name
2. Save Name To File
3. Load Name To File

CMD : 3
LOAD FILE NAME : ../nth.c
```

```
=====
NAME : #include <pthread.h>
=====

1. Change Name
2. Save Name To File
3. Load Name To File

CMD :
```



[도전] 위험한 코드, C++

경로조작을 막도록
필터링 코드 추가

- ../ 또는 ./ 코드 제거

```
#include <iostream>
#include <fstream>
#include <istream>
#include <string>

using std::cout;
using std::cin;
using std::string;
using std::ifstream;
using std::ofstream;

string name = "NO NAME";
string fileName;

void printStatus() {
    cout << "\n\n";
    cout << "===== \n";
    cout << "    NAME : " << name << "\n";
    cout << "===== \n";
    cout << "\n";
    cout << "1. Change Name\n";
    cout << "2. Save Name To File\n";
    cout << "3. Load Name To File\n";
    cout << "\n";
}

void saveName() {
    ofstream fout(fileName);
    fout << name;
    fout.close();
}
```

```
void loadName() {
    ifstream fin(fileName);
    if (fin.fail()) {
        cout << "ERROR :: THERE's NO FILE\n";
        return;
    }

    getline(fin, name);
    fin.close();
}

int main()
{
    while(1) {
        printStatus();

        int n;
        cout << "CMD : ";
        cin >> n;
        cin.ignore();

        if (n == 1) {
            cout << "NEW NAME : ";
            getline(cin, name);
        }
        else if (n == 2) {
            cout << "SAVE FILE NAME : ";
            cin >> fileName;
            saveName();
        }
        else if (n == 3) {
            cout << "LOAD FILE NAME : ";
            cin >> fileName;
            loadName();
        }
    }

    return 0;
}
```

필터 추가 코드

유효성 검사 함수 추가

```
bool isValid(string tar)
{
    if (tar.find("./") == -1) return true;

    cout << "ERROR :: INVALID FILE NAME\n";
    return false;
}
```

```
static boolean isValid(String tar) {
    if (tar.indexOf("./") == -1) return true;

    System.out.println("ERROR :: INVALID FILE NAME");
    return false;
}
```

```
int main()
{
    while(1) {
        printStatus();

        int n;
        cout << "CMD : ";
        cin >> n;
        cin.ignore();

        if (n == 1) {
            cout << "NEW NAME : ";
            getline(cin, name);
        }
        else if (n == 2) {
            cout << "SAVE FILE NAME : ";
            cin >> fileName;
            if (isValid(fileName)) saveName();
        }
        else if (n == 3) {
            cout << "LOAD FILE NAME : ";
            cin >> fileName;
            if (isValid(fileName)) loadName();
        }
    }

    return 0;
}
```

SQL Injection / 경로제어 방어 대책

외부의 입력을 믿지 않는다.

→ User의 입력은 유효성 검사 (필터링) 필수

Script Upload & Run

Script

해커가 제작한 Script / 실행파일을

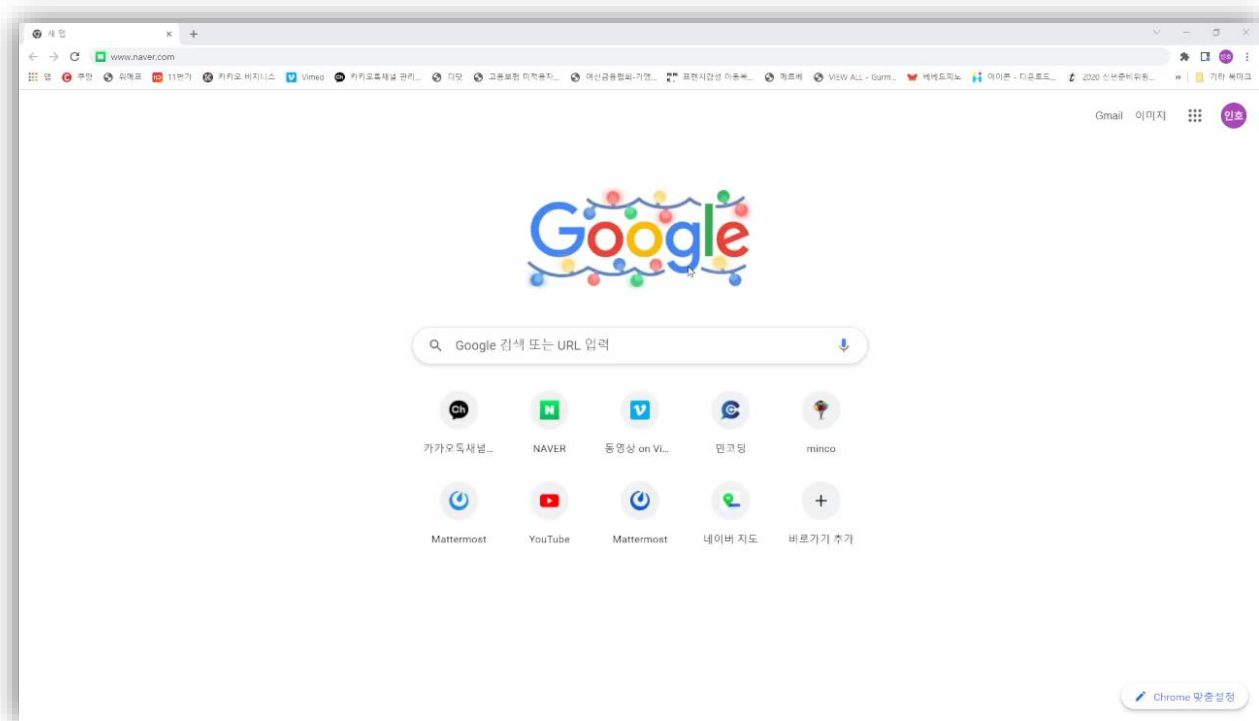
1. 서버 측, 업로드
2. Run

위 두 가지 조건이 성립된다면,
모든 동작이 가능하다.



[취약점 1] 위험한 형식 파일 업로드

1. **서버측 업로드** : 자료실에 php / asp / jsp 등 파일을 업로드
2. **Run** : 업로드된 링크로 접근하여 스크립트를 수행시킴



[취약점 2] OS 명령어 삽입

Argument에 특정 경로의 실행파일을 OS 권한으로 실행하는 코드

1. **서버측 업로드** : X (다른 해킹 기법으로 업로드)
2. **Run** : 원하는 경로의 Script / 실행파일 수행

안전하지 않은 코드의 예 JAVA

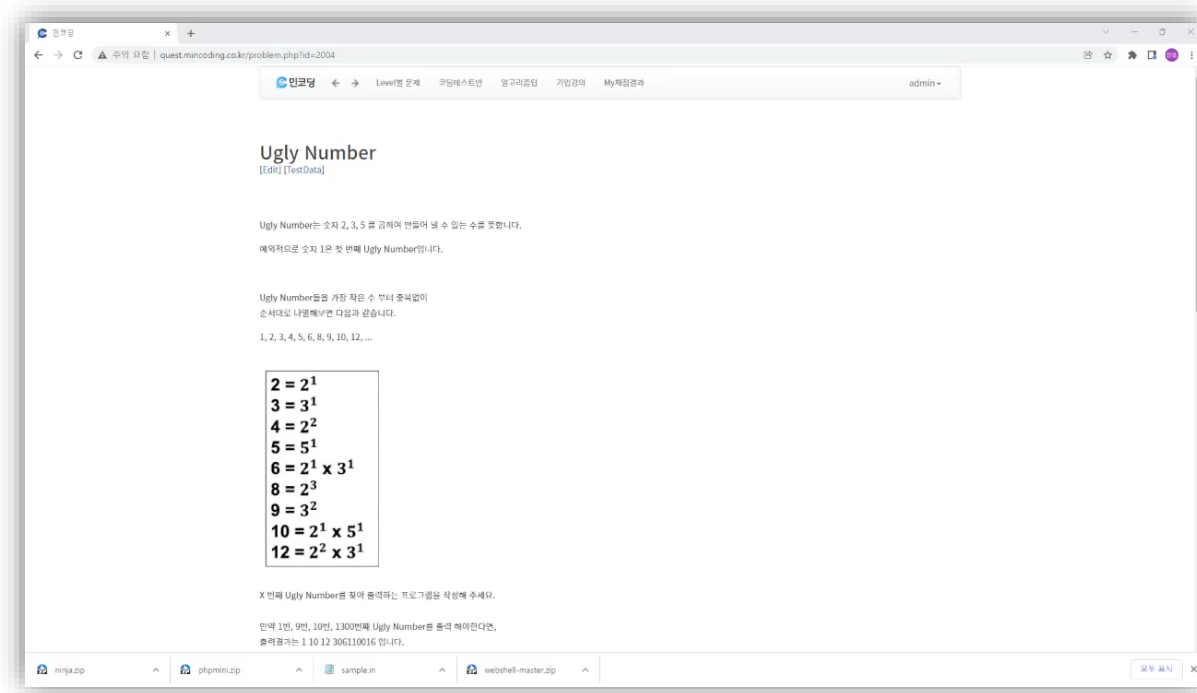
```
public static void main(String args[]) throws IOException {  
    // 해당 프로그램에서 실행할 프로그램을 제한하고 있지 않아 파라미터로 전달되는 모든 프로그램이  
    실행될 수 있다.  
    String cmd = args[0];  
    Process ps = null;  
    try {  
        ps = Runtime.getRuntime().exec(cmd);  
        ...  
    }  
}
```

Client PC / Server PC 에 Runner가 설치된다.

[취약점 3] XSS (Cross-Site Scripting)

웹에서 JavaScript로 하는 간단한 공격

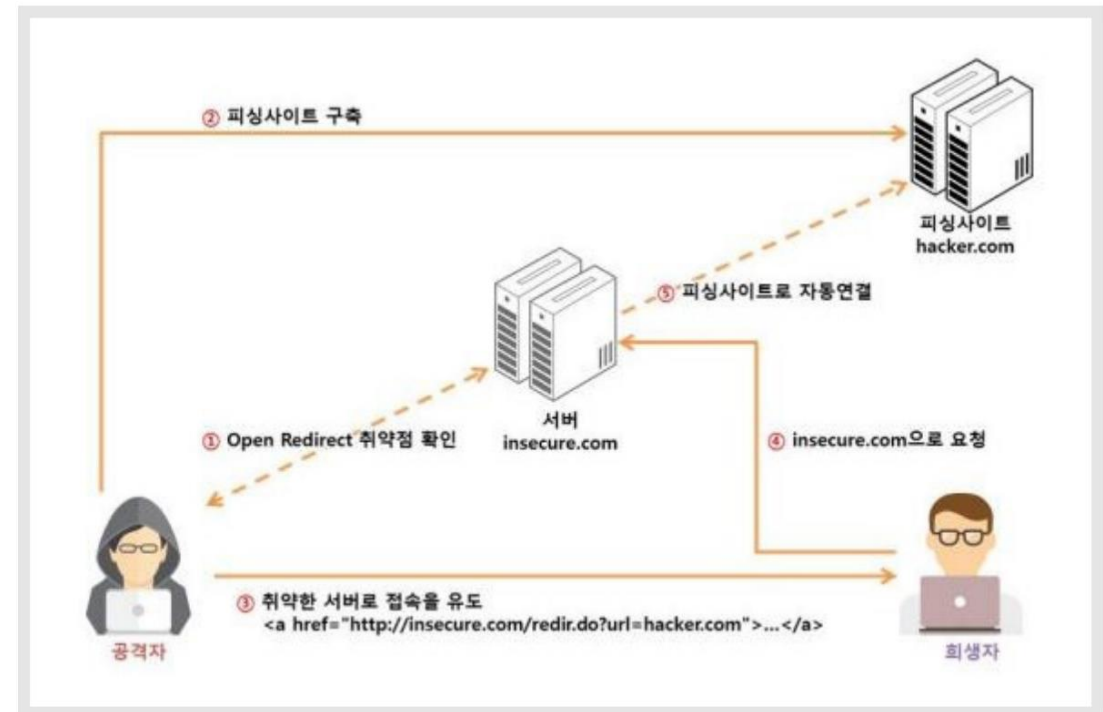
1. **서버측 업로드** : 사용자 게시판에 스크립트 삽입
2. **RUN** : 웹 페이지 접속시 즉시 스크립트 실행됨



XSS 응용

신뢰되지 않는 URL 주소로 자동접속 연결

- 사이트 해킹을 통한, Redirect 명령어 삽입
 1. Redirect 취약점 확인
 2. phishing 사이트 구축
 3. 로그인 낚시 시작



Script Upload & Run 대응 필요

1. Script 다운로드 받는 행동 사전 방어 필요
2. Script를 수행하는 코드에 적절한 유효성 검사 기능 필요
3. 공격활동 감지, Report 체계 수립

OWASP A09. 보안 로깅 및 모니터링 실패

보안을 위한 로깅, 모니터링을 해야한다.

- 행동을 로깅하고, 침투를 감지할 수 있는 시스템이어야 한다.
- 의심스러운 활동을 모니터링한다.
- 로그를 로컬이 아닌, 서버에 저장한다.
- 실시간으로 공격을 감지하며, 공격자에게 경고해야 한다.
- 데이터의 무결성이 유지되는지, 검사하는 코드가 있어야한다.



생각해보자. 예외코드의 빈도수 감지, 경고 대책

한 해커가, 우리 시스템에 다양한 Injection을 시도한다고 가정한다.

이로 인해 **유효성 검사 탈락율**, **예외 코드 수행율**이 비정상적으로 많아지게 된다.

의심스러운 활동을 로깅, 감지, 경고 할 수 있도록 대책을 생각해보자.

아래 세 가지 항목에 대한 대책을 Text로 작성하여 제출한다.

1. 로깅할 수 있는 방법 (무엇을, 어떻게, 얼마나 자주, 어디에 저장)
2. 모니터링을 위한 리포트 방법
3. 공격자에게 경고 방법

Secure C언어 구현

Secure 코딩을 위한, 취약점 분석 및 대응

Type Mismatch

버그의 원인을 찾고 수정하자.

```
#include <stdio.h>
#include <stdlib.h>

size_t find(char *buf, char ch)
{
    for (int i = 0; buf[i]; i++) {
        if (buf[i] == ch) return i;
    }

    return -1;
}

int main()
{
    char *str = "SHOW ME THE MONEY";
    char tar = 'Z';

    int ret = find(str, tar);
    printf("%d", ret);

    return 0;
}
```

출력결과 : -1 (정상)

```
#include <stdio.h>
#include <stdlib.h>

size_t find(char *buf, char ch)
{
    for (int i = 0; buf[i]; i++) {
        if (buf[i] == ch) return i;
    }

    return -1;
}

int main()
{
    char *str = "SHOW ME THE MONEY";
    char tar = 'Z';

    if (find(str, tar) >= 0) {
        printf("FIND!!\n");
    }
    else {
        printf("THERE'S NO Char(%c)\n", tar);
    }

    return 0;
}
```

출력결과 : FIND (비정상)

Null Pointer Dereference

시스템이 종단을 초래하는 대표적인 에러

- 다음 코드의 문제점을 찾고, 해결하자.

```
#include <stdio.h>

int main()
{
    int *p = 0;
    *p = 1;

    return 0;
}
```

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main()
{
    char *buf = (char *)malloc(10);
    strcpy(buf, "HI");
    printf("%s\n", buf);

    return 0;
}
```

```
#include <string.h>
#include <stdio.h>
#include <stdlib.h>

int main()
{
    char *buf = "SHOW ME THE MONEY";
    char *p = strstr(buf, "ME");

    printf("%s\n", p);

    return 0;
}
```

표준이 없는 호출 순서

순서가 필요한 코드로 수정시, 버그 발생

```
#include <stdio.h>
#include <stdlib.h>

int cnt = 0;
int abc() {
    cnt *= 3;
    return cnt;
}

int run() {
    cnt += 5;
    return cnt;
}

int main()
{
    if (abc() == run()) {
        printf("OH");
    }
    printf("cnt = %d\n", cnt);

    return 0;
}
```

출력결과 : 15

```
#include <stdio.h>
#include <stdlib.h>

int cnt = 0;
int abc() {
    cnt *= 3;
    return cnt;
}

int run() {
    cnt += 5;
    return cnt;
}

int go(int a, int b) {
    return a + b;
}

int main()
{
    printf("%d", go(abc(), run()));

    return 0;
}
```

표준에 명시되어 있지 않은 함수 호출 순서

scanf_s

C11 부터 추가된, C언어 표준 Library

```
#include <stdio.h>

int main()
{
    int n;
    scanf_s("%d", &n);
    printf("n = %d\n", n);

    char ch;
    scanf_s(" %c", &ch, sizeof(ch));
    printf("ch = %c\n", ch);

    char buf[10];
    scanf_s("%s", buf, sizeof(buf));
    printf("STR = %s\n", buf);

    return 0;
}
```

cppreference.com

Create account

Search

Page

Discussion

View

Edit

History

C

File input/output

scanf, fscanf, sscanf, scanf_s, fscanf_s, sscanf_s

Defined in header <stdio.h>

int scanf(const char	*format, ...);	(1)	(until C99)
int scanf(const char *restrict	format, ...);		(since C99)
int fscanf(FILE	*stream, const char	*format, ...);	(2) (until C99)
int fscanf(FILE *restrict	stream, const char *restrict	format, ...);	(since C99)
int sscanf(const char	*buffer, const char	*format, ...);	(3) (until C99)
int sscanf(const char *restrict	buffer, const char *restrict	format, ...);	(since C99)
int scanf_s(const char *restrict	format, ...);	(4)	(since C11)
int fscanf_s(FILE *restrict	stream, const char *restrict	format, ...);	(5) (since C11)
int sscanf_s(const char *restrict	buffer, const char *restrict	format, ...);	(6) (since C11)

scanf_s 표준에 대해서

C11 부터 표준으로 등재

- 그러나 GCC, Keil, DS-5, IAR 에서는 scanf_s 추가 안함
- Visual Studio의 MSVC는 scanf_s 존재

비표준이 아닌 표준이나,
GCC 등 다양한 컴파일러에 등재가 안되어있음

strcpy_s

문자열 복사시, 메모리 범위 검사를 위한 C 표준 함수 (C11)

- 복사할 범위를 지정해야 한다.
- 복사할 곳에 Null 이 없거나, 범위에 벗어나면 예외가 발생

```
char str[10] = "ABCD";  
char buf[10];  
  
strcpy_s(buf, sizeof(str), str);  
  
printf("%s\n", buf);
```

[도전1] 코드 수정하기

오류를 없애고,
안전한 코드로 변경하기

<https://gist.github.com/d8490e21b4d003fb63ae82196a7d2254.git>

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main()
{
    //입력 버퍼 만들기 (100글자 까지 허용)
    char* buf = (char*)malloc(100);
    scanf("%s", buf);

    //대소문자 답을 공간 만들기
    char* bigChar = (char*)malloc(strlen(buf));
    char* smallChar = (char*)malloc(strlen(buf));

    int bt = 0;
    int st = 0;
    for (int i = 0; buf[i]; i++) {
        if (buf[i] <= 'Z') {
            bigChar[bt++] = buf[i];
        }
        else {
            smallChar[st++] = buf[i];
        }
    }

    //소문자를 답을 공간 만들기

    //결과를 저장할 배열
    char result[10];
    strcpy(result, bigChar);
    strcat(result, smallChar);
    printf("%s\n", result);

    return 0;
}
```

[도전2] 코드 수정하기

1. 시큐어코딩 하기
2. 해킹시도 모니터링 & 경고

<https://gist.github.com/22827e7ad4e07936917be92b000f12f7.git>

```
int main(int argc, char* argv[])
{
    int mappingTable[MAXSIZE][MAXSIZE];
    map = mappingTable;
    fillDummyRandData();

    int dy = atoi(argv[1]);
    int dx = atoi(argv[2]);
    float ms_delay = atof(argv[3]);
    int start = clock();

    for (int i = 0; i < 1000; i++) {
        read(dy, dx);
        msleep(ms_delay);
    }

    int latency = clock() - start;
    if (latency > 5000) {
        printf("ERROR REPORT :: LATENCY OVER\n");
        return 0;
    }

    printf("%d ms", latency);
    return 0;
}
```


SEI CERT Secure Coding

SEI CERT 시큐어코딩

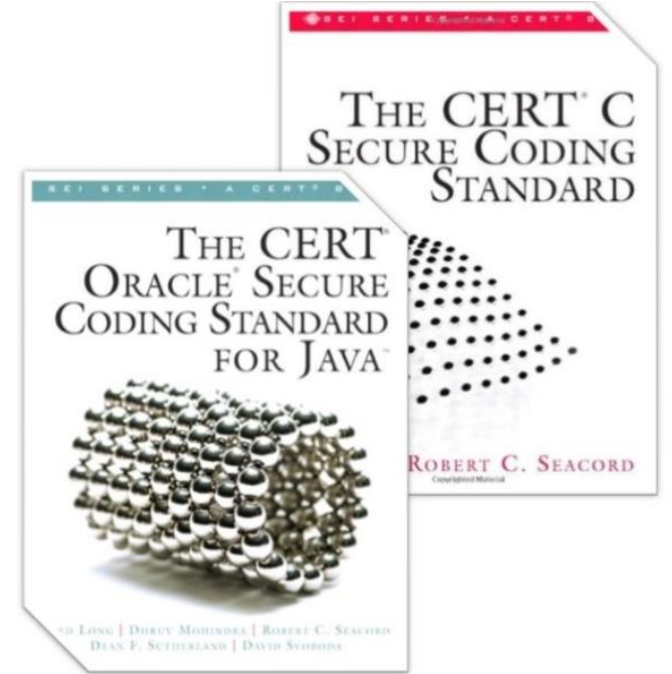
코딩표준

Secure Coding을 위한 코딩표준

- SEI CERT C
- SEI CERT-C++
- SEI CERT Oracle for JAVA

품질을 위한 코딩표준

- 시큐어 보다는, 품질 중점을 둠
- 종류
 - MISRA C : 자동차에서 시작, 임베디드 품질을 위함
 - AUTOSAR C++ : 자동차 코딩 표준



SEI CERT Secure Coding

SEI란 ?

- Carnegie-Mellon University의 Software Engineering Institute
- 미국 국방부와 계약으로 시작된, 소프트웨어 공학회, 협회
- 소프트웨어 개발 프로세스를 개선하기 위한 목적
- 보안 / AI / 정부기술지원 등 조직으로 구성

SEI **CERT** Secure Coding

CERT란?

- Computer Emergency Response Team
- 컴퓨터 비상 대응팀, 보안 사고에 대한 대처를 한다.
- 세계 여러 CERT가 존재하며,
SEI CERT가 Head이다.

심각도

심각

- 특정 코드를 원격으로 수행한다
- Level 1 (Priority 12 ~ 27)

중간

- 의도하지 않은 정보를 공개한다.
- Level 2 (Priority 6 ~ 9)

낮음

- 서비스 거부 공격
- Level 3 (Priority 1 ~ 4)

분류

이 표준은, 두 가지 단어로 의무 성격을 나타낸다.

CERT C Secure Coding Standard Rule & Recommendation

- Rule: 코드 작성 시 개발자가 반드시 지켜야 하는 규칙
- Recommendation: 코드의 품질 향상을 위한 제안사항으로, Recommendation의 위반이 코드 결함을 나타내지는 않음

CERT C Secure Coding Standard 카테고리

- | | |
|--|---|
| <ul style="list-style-type: none">• Rule 01. 전처리기 (PRE)• Rule 02. 선언과 초기화 (DCL)• Rule 03. 표현식 (EXP)• Rule 04. 정수 (INT)• Rule 05. 부동소수점 (FLP)• Rule 06. 배열 (ARR)• Rule 07. 문자와 문자열 (STR)• Rule 08. 메모리 관리(MEM)• Rule 09. 입력과 출력 (FIO) | <ul style="list-style-type: none">• Rule 10. 환경(ENV)• Rule 11. 시그널(SIG)• Rule 12. 에러 관리 (ERR)• Rule 13. 애플리케이션 개발 인터페이스(API)• Rule 14. 동시 실행 (CON)• Rule 48. 기타(MSC)• Rule 50. 부록 POSIX(POS)• Rule 51. Microsoft Windows (WIN) |
|--|---|

CERT C 공식사이트

- <https://wiki.sei.cmu.edu/confluence/display/c>

