WhiteHat School 1기

Oracle DB 취약점 Write-up

Team: 과부화

Writer: 정진교

취약점

1. Oracle Server Scanning

1-1. 포트 스캐닝:

- 공격자가 오라클 데이터베이스 서버의 포트를 스캔하여 열려 있는 포트를 식별하는 행위다. 공격자는 열려 있는 포트를 통해 서버에 접근하려고 할 수 있다.

- TNS (Transparent Network Substrate) Listener는 Oracle 데이터베이스와 클라이언트 간의 통신을 관리하는 역할을 하는 Oracle의 네트워크 컴포넌트다. TNS는 Oracle 데이터베이스가 클라이언트의 연결 요청을 받아들이고, 이를 데이터베이스 인스턴스로 라우팅하여 효율적인 통신을 도와준다. 여러 개의 데이터베이스 인스턴스가 동작하는 경우, TNS Listener는 클라이언트가 요청하는 데이터베이스로 연결을 중계한다.

- TNS Listener는 특정 포트에서 대기하며 클라이언트의 연결 요청을 수신한다. 기본적으로 Oracle의 TNS Listener는 TCP/IP 포트 1521을 사용하며, 이는 클라이언트가 Oracle 데이터베이스에 연결하기 위한 표준 포트다. 오라클 DB서버에서 가장 많이 해킹 공격이 이루어지는 곳들 중 하나는 TNS Listener 포트(TCP/1521)이다.

- TNS Listener는 데이터베이스가 클라이언트의 연결을 수신할 수 있는 상태로 유지된다. 이를 리스닝 상태라고 한다.

- TNS Listener는 다양한 데이터베이스 서비스에 대한 요청을 처리한다. 각 데이터베이스 인스턴스는 Listener에 서비스 이름, 호스트 주소, 포트 번호 등을 등록한다. 클라이언트는 이 정보를 사용하여 데이터베이스에 연결한다.

- 리스너의 구성은 listener.ora 파일에서 정의된다. 이 파일에는 리스너의 동작과 관련된 다양한 설정이 포함되어 있다.

1-2. 서비스 식별 및 버전 정보 노출:

- 서버에 대한 서비스 및 버전 정보를 획득하려고 시도하는 것이다.
  서비스 및 버전 정보를 통해 공격자는 특정 취약성을 탐지하고 이를 이용할 수 있다.

1-3. Oracle PL/SQL 관련 비인가 접근 시도, SQL Injection 공격

- Oracle DB는 PL/SQL(Procedural Language/Structured Query Language)을 사용하여 프로시저, 트리거, 함수, 패키지 등을 개발하고 실행할 수 있는 프로그래밍 언어를 제공한다.

- PL/SQL 언어를 이용해 공격자가 서버에 대한 비인가 접근을 시도하는 행위다. 약한 인증 및 권한 관리 설정 등으로 인해 발생할 수 있는 취약점을 이용하려는 시도가 있을 수 있다.

SQL 인젝션 공격:

- 공격자가 보안상의 취약점을 이용하여, 임의의 SQL문을 주입하고 실행되게 하여 데이터베이스가 비정상적인 동작을 하도록 조작하는 행위. 데이터베이스와 연동된 웹 애플리케이션에서 공격자가 입력이 가능한 폼에 조작된 질의문을 삽입하여 웹 서비스의 데이터베이스 정보를 열람 또는 조작할 수 있는 취약점.

PL 인젝션 공격:

- PL(Procedural Language) 인젝션은 주로 데이터베이스에서 사용되는 프로시저, 트리거, 함수 등의 프로시저얼 언어에 대한 인젝션 공격

- Stored Procedure에서의 PL Injection 예시

```
CREATE OR REPLACE PROCEDURE login_procedure(username IN VARCHAR2, password IN VARCHAR2) AS
BEGIN
  EXECUTE IMMEDIATE 'SELECT * FROM users WHERE username = ''' || username || ''' AND password = ''' || password || '''';
END login_procedure;
```

이 코드에서는 사용자 입력을 직접 문자열에 추가하고 실행한다.

'admin' OR '1'='1' --' 입력을 하면, 쿼리는 'SELECT * FROM users WHERE username = 'admin' OR '1'='1' --' AND password = ''' 으로
항상 참이 되는 조건이 된다.

- Function에서의 PL Injection 예시

```
1 CREATE OR REPLACE FUNCTION get_user_data(user_id IN NUMBER) RETURN VARCHAR2 AS
2   result VARCHAR2(100);
3 BEGIN
4   EXECUTE IMMEDIATE 'SELECT username FROM users WHERE id = ' || user_id INTO result;
5   RETURN result;
6 END get_user_data;
7 demo
```

사용자 입력을 직접 문자열에 추가하고 실행하고 있다.

'1; DROP TABLE users --' 와 같은 입력을 하면, 두 번째 문장이 실행되어 users 테이블이 삭제될 수 있다.

SQL Injection 취약점이 존재하기에 Function Injection 또한 존재하고 있다.

Function Injection:

- 함수 인젝션은 주로 데이터베이스나 다른 시스템에서 사용되는 함수 호출에 대한 공격이다. SQL 함수, 프로시저, 사용자 정의 함수 등이 대상이 될 수 있다. 공격자는 함수 호출에 악의적인 인자를 전달하여 원하지 않는 동작을 유도하거나 시스템의 무단 접근을 시도한다.

예시)

```
1 EXEC my_function('input_data');
```

만약 사용자가 입력으로 '); DROP TABLE users; --' 와 같은 값을 제공하면 쿼리는 다음과 같이 해석될 수 있다.

```
1 EXEC my_function(''); DROP TABLE users; --');
```

입력 값으로 전달된 함수 호출이 이후의 코드를 주석 처리하고 새로운 쿼리가 실행되게 된다.

SQL Injection과 Function Injection 간의 차이점:

타겟 객체 : SQL 인젝션은 주로 SQL 쿼리를 타겟으로 하고
함수 인젝션은 주로 데이터베이스나 시스템에서 사용되는 함수 호출을 타겟으로 한다.

공격 방법 : SQL 인젝션은 주로 문자열 연결을 이용하여 SQL 코드를 주입하고
함수 인젝션은 함수 호출의 인자를 악의적으로 조작하여 공격한다.

SQL 인젝션은 주로 WHERE 절에서 사용자 입력을 이용한 조건식에 공격을 시도하고
함수 인젝션은 주로 데이터베이스나 시스템 함수 호출에서 인자를 조작하여
공격을 시도한다.

1-4. 그 외 다양한 오라클 서버 관련 스캔 방법들

1-4-1. nmap을 활용한 스캔(bash):

- 'nmap' 은 다양한 서비스 및 포트에 대한 스캔을 수행하는 데 사용된다.
  Oracle 서버의 포트를 확인하고 서비스 버전 정보를 획득할 수 있다.

```
nmap -p 1521,1526,1529,1530,2483,2484,2485,2486,2487,2488,2489,2490,2482,2481,80,443,5500,7777,8888,5555,7890 target_ip
```

1-4-2. Oracle TNS Listner 스캔(bash, 1-1 참고):

- Oracle 데이터베이스는 TNS (Transparent Network Substrate) Listener를 통해 클라이언트와 통신한다. TNS Listener가 열려 있는 포트(기본 1521)를 확인하고 버전 정보 등을 수집할 수 있다.

```
tnscmd10g version -h target_ip
```

1-4-3. SQL*Plus와 TNS 스캔(bash):

SQL*Plus와 TNS를 이용하여 서비스에 대한 정보를 수집할 수 있다.

```
sqlplus scott/tiger@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=target_ip)(PORT=1521))(CONNECT_DATA=(SID=orcl)))
```

1-4-4. Oracle 데이터베이스 버전 확인(sql):

데이터베이스에 연결하여 버전 정보를 확인하는 SQL 쿼리를 실행할 수 있다.

```
SELECT * FROM v$version;
```

1-4-5. Oracle TNS 명명 규칙(Brute-Force):

TNS 명명 규칙을 통해 다양한 데이터베이스 및 서비스에 대한 정보를 찾을 수 있다.

```
1  tnscmd10g status -h target_ip
```

1-4-6. Oracle 익명 로그인 시도:

기본적으로 익명 로그인이 허용되어 있는지 확인하기 위해 익명 사용자로 로그인 시도를 할 수 있다.

```
1 CONNECT / AS SYSDBA;
```

보안대책

1-1. 포트 스캐너 보안 대책

암호화 및 보안 : 데이터베이스 통신을 보호하기 위해 SSL(암호화된 연결)을 구성할 수 있다.

필터링 및 접근 제어 : 특정 IP 주소에서의 연결만 허용하거나, 특정 클라이언트로부터의 연결을 차단.

리스너 로그: 리스너는 로그 파일을 생성하여 클라이언트 연결 및 이벤트를 기록한다. 이 로그를 검토하여 잠재적인 문제를 식별할 수 있습니다.

동적 서비스 등록 관리: 동적 서비스 등록을 허용하면 데이터베이스가 리스너에 자동으로 등록될 수 있다.

1-2. 서비스 식별 및 보안 정보 노출 관련 보안 대책

서버 배너 숨기기:

- 웹 서버, 애플리케이션 서버, 데이터베이스 등에서 출력되는 배너를 숨기거나 수정하여 실제 서버 및 버전 정보를 숨기게 되면 공격자가 취약성을 알아내기 어려워진다.

기본 값 및 예측 가능한 값 변경:

- 서버의 기본 값 및 예측 가능한 값은 공격자가 시스템을 탐지하고 공격을 시도하기 용이하게 만든다. 값을 변경하여 예측성을 줄이고 보안을 강화할 수 있다.

사용자 정의 에러 페이지 설정:

- 에러 메시지나 예외 상황이 발생했을 때 사용자에게 제공되는 에러 페이지를 사용자 정의하여 일반적인 서버 및 버전 정보가 노출되지 않도록 한다.

보안 패치 및 업데이트 적용:

- 운영 중인 소프트웨어 및 서버 컴포넌트에 대한 최신 보안 패치 및 업데이트를 적용하여 이미 알려진 취약점을 보완한다.

정보 노출을 최소화하는 웹 서버 구성:

- 웹 서버의 설정을 조정하여 디렉터리 목록을 비활성화하고, 불필요한 파일을 숨기는 등의 조치를 취하여 정보 노출을 최소화할 수 있다.

웹 애플리케이션 방화벽 사용:

- 웹 애플리케이션 방화벽을 사용하여 서비스 식별 및 버전 정보가 노출되는 시나리오를 탐지하고 차단할 수 있다.

보안 헤더 사용:

- 보안 헤더를 사용하여 서버 및 버전 정보를 숨기거나 최소화할 수 있다. 예로, HTTP 응답 헤더에 ServerTokens 및 ServerSignature와 관련된 설정을 조정할 수 있다.

보안 감사 및 로깅 설정:

- 시스템에서 발생하는 로그를 확인하여 서비스 식별 정보가 노출되는 경우를 감지하고 대응할 수 있는 감사 및 로깅 설정을 수행한다.

1-3. 프로시저얼 언어에 대한 인젝션 공격 보안 대책

파라미터화된 쿼리 사용 : 사용자 입력을 직접 문자열에 추가하는 대신 파라미터를 사용하여 동적 SQL을 생성한다.

입력 유효성 검증 : 사용자 입력을 검증하여 예상치 못한 값이나 문자를 걸러내고, 필요에 따라 거부한다.

접근 권한 제한 : 사용자가 실행할 수 있는 프로시저나 함수에 대한 권한을 제한하여 최소한의 권한 원칙을 따른다.